Detection of thin, curvilinear structures : Advances, Algorihms and Implementations

Petr Dokládal

Université Paris-Est

Mémoire en vue d'obtention de

l'Habilitation à Diriger des Recherches

Spécialité : Mathématiques et STIC - traitement du signal et des images

Detection of thin, curvilinear structures : Advances, Algorihms and Implementations

Présenté par **Petr Dokládal**

soutenue publiquement le 1er juillet 2013

devant le jury composé de:

M. Olivier Déforges (rapporteur)	professeur, INSA Rennes
M. Philippe Salembier (rapporteur)	professeur, Universitat Polytècnica de Catalunya, Espagne
M. Jocelyn Chanussot (rapporteur)	professeur, INP de Grenoble
M. Mohammed Akil (examinateur)	professeur, ESIEE, LIGM
Mme. Isabelle Bloch (examinateur)	professeur, Telecom-ParisTech
M. Michel Couprie (examinateur)	professeur, ESIEE, LIGM
M. Marc Van Drogenbroeck (examinateur)	professeur, Université de Liège, Belgique
M. Dominique Jeulin (directeur)	professeur, Mines-ParisTech

Abstract

The habilitation is an opportunity to stop to make an appraisal of the past and think of the future of one's career. Following this idea, this report is a retrospective of the last twelve years of my career, years that I have spent with the Centre of Mathematical Morphology of Mines-ParisTech, where I have been developing my research.

I have compiled in this text a synthesis of my work, structured around two principal axis.

The first axis, a methodological one, lists methodological advances regarding the detection of thin objects. The second axis – an algorithmic one, lists a few original algorithms, and an efficient implementation, always presented in the context of the detection of thin structures.

In the first axis one can find three principal contributions:

- a spatially-variant mathematical morphology approach, steered by a local analysis of structures – referenced hereafter as the morpho-hessian approach.

- parsimonious path opening – an approach to detect thin, not necessarily straight, structures. The parsimonious variant that we propose is obtained by decoupling the research of paths and the filtering. It not only allows to define new operators, but also decrease in a significant manner the complexity and, consequently, the computation time.

- attribute thinnings based on an original attribute - the geometric diameter, allowing very efficient extraction of thin structures.

In the second axis one can find the following contributions :

- a 1-D dilation algorithm and two different algorithms of 1-D morphological opening. These three algorithms have interesting properties allowing an efficient implementation.

In the Implementations part one can find a few real implementations for image processing applications running under heavy real-time constraints:

- several FPGA implementations, and

- one GPU implementation.

Based on these implementations we could validate the computation efficiency of these algorithms. Precisely, we have proposed the first implementation of morphological neighbourhood processor using an arbitrarily large neighbourhood. The size of the neighbourhood (even a large size) does not bring any implementation difficulties, nor have a negative impact on the computation efficiency.

A whole chapter is dedicated to technology transfer to the industrial sector. The principal section is devoted to industrial application from the domain of material science, and particularly, those linked to the inspection or control. The following section comes from the medical and biomedical domain. Finally, the third section, a shorter one, is dedicated to special or embedded applications.

This report is completed by a selection of my principal publications allowing to find the most important bibliographic references of the text.

Résumé

L'habilitation à diriger des recherches est une occasion de s'arrêter et prendre un moment pour faire le point sur le passé et de réfléchir sur l'avenir de sa carrière. Dans cette optique, ce mémoire est une rétrospective des douze dernières année de ma carrière, des années que j'ai passées au Centre de Morphologie Mathématique de Mines-ParisTech, où j'ai mené mes travaux de recherche.

Dans ce mémoire, j'ai répertorié ces travaux autour de deux axes principaux. Le premier axe, méthodologique, fait état des avancées méthodologique de la détection d'objets fins. Le second axe - algorithmique - répertorie des algorithmes originaux, et mises en pratiques efficaces, toujours présentés dans le contexte de détection d'objets fins.

Dans le cadre du premier axe, on répertorie trois contributions principales :
- une approche morphologique variant dans l'espace, contrôlée par une analyse locale des structures, nommée approche morpho-hessienne.
- ouverture parcimonieuses par chemins - est une variante parcimonieuse des ouvertures par chemin, obtenue en découplant la recherche des chemins et leur filtrage. Cette approche permet non seulement de définir des opérateurs nouveaux, mais également de baisser de manière significative la complexité et, par conséquent, le temps de calcul.
- amincissement par attributs - basés sur un attribut original - le diamètre géométrique, permettent l'extraction d'éléments fins de manière très efficace.

Dans le cadre du second axe - algorithmique - on répertorie des algorithmes originaux, et mises en pratiques efficaces, toujours présentés dans le contexte de détection d'objets fins. Dans ce volet, nous retrouvons:
- un algorithme de dilatation 1-D et - deux algorithmes différents d'ouverture morphologique 1-D. Ces trois algorithmes présentes des propriétés intéressantes pour une mise en œvre efficace.

Dans le volet de mises en pratique efficaces nous retrouvons des réalisations pour des applications de traitement d'images travaillant sous fortes contraintes temps réel:
- plusieurs réalisation matérielle (FPGA), et
- une réalisation GPU, ont permis de valider l'efficacité calculatoire de ces algorithmes. Entre autre, nous avons pu proposer une première réalisation de processeur morphologique à taille de voisinage arbitrairement large. La taille du voisinage (même très grande) n'introduit ni de difficulté de réalisation, ni d'impact négatif sur l'efficacité de calcul.

Un chapitre entier est consacré à la partie applicative, faisant état des collaborations industrielles. La section principale est consacrée aux applications industrielles, du domaine de sciences de matériaux, et plus particulièrement l'inspection ou le contrôle. Le deuxième section vient du domaine médical et biomédical. Enfin, une troisième section, plus courte, est consacrée aux applications spéciales ou embarquées.

Ce mémoire est annexé par une sélection de mes principales publications scientifiques permettant de retrouver aisément les références bibliographiques les plus importantes de ce mémoire.

Acknowledgements

First of all, I would like to express my gratitude to the members of the board for having honored me by their presence in the board, and particularly the reviewers for having accepted to evaluate this manuscript. I also want to thank Dominique Jeulin who has accepted to play the role of the advisor of this habilitation. He has provided number of valuable suggestions and completions to this text.

I ought to mention Gilles Bertrand, my PhD advisor who opened me the doors of discrete geometry in the image processing framework. When I joined the Centre of Mathematical Morphology (CMM), Jean-Claude Klein was my first close collaborator after my arrival to CMM, and I appreciated "doing electronics" with him. I also wish to mention Fernand Meyer, the head of the CMM, who creates a constructive, motivating and pleasant working environment. I will not forget *individually* all other colleagues from the Center of Mathematical Morphology for their availability for interesting discussions, and generally, for their kindness, and ability to create a pleasant working environment.

I wish to thank the PhD students who have accepted to participate in these projects. Everything was much easier thank to your willingness, volunteering and skills. I loved discussing with you after lunch – a cup of tea in hand – an extremely fuzzy idea, and discover to my surprise by the evening the very same day that you have implemented it – and that it works.

Lastly but of outmost importance, I'd like to mention the people most special to me, my family, my parents, children Maxim and Hugo, and my wife Eva who has always been of an incomparable support and source of encouragement to me.

... to Maxim, Hugo and Eva

Contents

I Curriculum Vitæ, Research&Teaching activity, Publications **9**

1 Curriculum Vitæ **11**

2 Research&Teaching activity **14**
2.1 Teaching Activity 14
2.2 Journals' Reviewer Service 15
2.3 Member in Research Groups 15

3 PhD Candidates Supervision **16**

4 Academic Collaboration **18**

5 List of Publications **19**

II Detection of thin objects **23**
5.1 Introduction 25

6 Morpho-Hessian Approach **26**

7 Parsimonious Path Openings **28**

8 Attribute Thinnings **31**

III Algorithms **35**

9 Morphological algorithms **37**
9.1 Efficient 1-D Dilation Algorithm 37
9.2 Efficient 1-D Opening: Algorithm 1 38
9.3 Efficient 1-D Opening: Algorithm 2 39
9.4 Massive Marching: A Massively Parallel Watershed Algorithm 40

10 Implementations **41**
10.1 Curve-Evolution PDEs 41
10.2 Spatially-Variant Morphology 41
10.3 Parallel implementation of serial morphological filters 44
10.4 Massively parallel implementation 45
10.5 Conclusion 46

IV Contractual Research and Applications **49**
10.6 Material Science . . . 51
10.6.1 The Tocata project . . . 51
10.6.2 The Colas project . . . 51
10.6.3 The Aedinca project . . . 53
10.6.4 The Cocascope project . . . 55
10.7 Medical&Biomedical Science . . . 56
10.7.1 The L'Oréal partnership . . . 56
10.8 Other research . . . 56
10.8.1 The Freia project . . . 56
10.8.2 The Cosmeca project . . . 57

V Perspectives **59**

11 Perspectives **60**

VI Annexe : Selected Papers **71**

Part I

Curriculum Vitæ, Research&Teaching activity, Publications

Chapter 1

Curriculum Vitæ

This chapter aims at presenting briefly my career, including the education and the appointments that I held. I start by giving the principal dates and milestones completed below by a few details and retrospective appraisal on my professional route.

Petr Dokládal

Research Engineer

Centre of Mathematical Morphology (CMM)
Department of Mathematics and Systems
MINES ParisTech
35, rue Saint-Honoré
77305 Fontainebleau, FRANCE
Tel. : + 33 (0) 1 64 69 47 98
Fax. : + 33 (0) 1 64 69 47 07
E-mail : petr.dokladal@mines-paristech.fr

PERSONAL DATA

First name: **Petr**, *Family name*: **Dokládal**
Date and Place of Birth: **August 31st 1971**, **Brno, Czech Republic**
Nationality: **Czech Republic**
Languages:
- **Czech** - mother's tongue
- **French** - fluent
- **English** - fluent
- **Spanish** - basics

Marital status: **Married, two children**

EDUCATION

1996 - 2000	**PhD** from **Marne la Vallée University**, France, *Title*: *Segmentation of Grey-Scaled Images : A Topological Approach*, *Hosting laboratory*: A^{2}SI, ESIEE, *Advisor*: Gilles Bertrand *Defended*: January 31st 2000, *Defense board*: J.-M. Chassery and J. Jan (reviewers), I. Bloch, D. Arquès, J. Smékal and G. Bertrand
1994 - 1996	**PhD cursus** - 2 year equivalent of the french DEA, option Signal Processing
1989 - 1994	**Engineer in Telecommunications**, Faculty of Electrotechnical Engineering, **Technical University Brno**, Czech Republic

---

APPOINTMENTS

since 2000 on	**Research Engineer**, CMM (Centre of Mathematical Morphology), Mines-ParisTech
2000 (Feb - Nov)	**Post Doctoral researcher**, TSI (Traitement du Signal et Image), Telecom-ParisTech, Paris RNRT project : *Study of radio-electric field impact on the human cerebral activity, construction of an individual head atlas from a MRI acquisition*
1997 - 1998	**Military service** (Sep - Sep), Czech Republic
1994 - 1996	**Assistant Professor**, Department of Telecommunications, Faculty of Electrotechnical Engineering, Technical University Brno, Czech Republic

---

My educational background is that of a telecommunications engineer. I have followed the undergraduate training in acoustic signal processing, with focus on filtering, speech recognition, and implementation of filters. Obviously, signals in telecommunications are processed in real-time, with possibly as low latency as possible. I have really appreciated – and still do – this domain.

I have completed my engineer education by a 2-year postgraduate course, that – at that time – made a part of PhD. It consisted of a series of lectures equivalent to the former french DEA, or the present European *master of research* course. It was – and still is – usual that during that time the PhD candidates have a quarter- to half-time lecturer's appointment to give lectures, ensure lab sessions and supervise diploma projects. The rest of the time is to be devoted to research. It is expected that the candidates defend the PhD after one additional year. Given the volume of the lectures, and the teaching appointment, this is unreal, and nobody actually manages to complete the PhD in only three years.

I have followed another way. After completing the 2-year postgraduate course I have – essentially because of personal interest – diverted to image processing. I have applied to – and have been selected for – a co-supervised PhD scholarship from the french embassy in Prague. This scholarship covered three 6-month periods in France to prepare a co-supervised PhD under the joint guidance of two advisers.

I have prepared my PhD at the A^{2}SI lab, at ESIEE Paris, under the guidance of Gilles Bertrand. Before I completed I had to accomplish my military service, at that time still compulsory, not only in the Czech Republic, but also in few other European countries. One of the results of my PhD thesis was topologically controlled image segmentation based on the *simple point* concept. This approach is interesting wherever the correct topology of the result is known and should be respected.

I have used this approach during my Post Doc internship, where I was in charge of the image segmentation task of the RNRT project Comobio, aiming at evaluation of harmful effects of an excessive exposure to the electromagnetic field of mobile phones. It was modeled by computing the energy dissipation in various internal organs of the head, located close to the mobile antenna. The

local energy dissipation was computed by application of the Maxwell equations on the segmented 3-D image of the head. In such an application the topological correctness of the result is of utmost importance.

At the end of 2000, I have joined the Center of Mathematical Morphology at Mines-Paristech (formerly School of Mines of Paris). I had since then the occasion to work on a number of topics. Immediately after my arrival at CMM, my engineer's background was was very useful when I was working on topics far from the image processing:

- modeling and system identification for the diabetes regulation (Cosmeca project)
- electronics and embedded system programming (Cosmeca project)

Despite this brief "return to origins", my interests have always been revolving around the image processing, and algorithms.

Now, when I look back trying to set up this retrospective balance sheet - and the Habilitation is an excellent occasion to do so – I find that my engineer's education has influenced my career and choices to a non-negligible extent. This is actually the reason why efficient algorithms and implementation would always remain in my center of interest.

One example can be found in the efficient 1-D dilation and opening algorithms that – seen as the sup-convolution and an analogy of low-pass filter – can also be implemented in a way usual in the signal processing, the FIR filter implementation.

I have progressively focused to detection of thin objects – a frequent topic in image processing – present in numerous domains, namely biomedical or medical or material image processing. Thin objects are indeed a model that can be used to represent a variety of objects. Their size is often very small compared to the inspected area. This brings an additional difficulty, an overwhelming amount of data to process. For these reasons I have also proposed fast algorithms applicable to the extraction of thin objects. I detail my principal contributions to this domain in this report.

Chapter 2

Research&Teaching activity

2.1 Teaching Activity

The teaching activity of researchers at Armines/Mines-Paristech is generally quite limited. I admit to have had less occasion to teach than I would wish since I consider that marrying teaching and research activity is beneficial for both sides, the students and the lecturer.

Having an active and recent experience in research allows bringing live applicative examples from various domains, making the lecture attractive and motivating for the students. Motivating because it allows to show a close contact of the matter with the real world, attractive because of examples or exercises from real applications. Conversely, a live contact with students, gives a researcher an opportunity to stay in contact with students, find and enroll valuable trainees. Lastly - but this remains a personal appreciation - I find satisfactory sharing acquired knowledge with other people. Consequently, in the years to come I wish to enlarge my implication in teaching.

To sum up, I have opened and/or participated in existing courses by giving the following lectures:

• During my post-gradual courses, from 1994 to 1996, at Technical University Brno, I have given a number of labs, practical sessions and supervised diploma projects, from the discrete signal processing domain.

• During my PhD, from 1996 to 1999 - I have provided a number of lecture and lab sessions at ESIEE, Paris, essentially in the C/C++ programming, either in the engineer cycle, or in the 'formation continue' cycle.

Since 2003, I have regularly provided the following lectures.

• ENSTA Paris - Partial Differential Equations in Image Processing. I have proposed and opened a lecture on image processing techniques using partial differential equations. In the beginning, it recalls basic terms from the geometry. Then it introduces image segmentation by using active contours, image filtering, and continuous mathematical morphology. The lecture is supported by a lab session in Matlab.

• ENSTA Paris - Real Time Image Processing on FPGA. I have proposed and opened a lecture on real time implementation of image processing technique on FPGA. The lecture introduces fundamental concepts of pressing data in a flow, parallel execution, and synchronization. The lecture is greatly supported by labs, where the students implement a contour detection application by using the zero-crossing of the laplacian. The application is connected to a webcam, and the result is projected on a VGA screen in the real-time.

• Université Paris Descartes/ParisTech - I have partially taken over, and remodeled, an existing series of lectures in the Image Processing track of the Biomedical Engineering Master. The lectures provide an introductory insight into image processing and segmentation techniques, before focusing more specifically to mathematical morphology tools. The sessions are open to students from two domains, engineering and medical, with different prerequisites. The lectures are supported by labs and projects.

• ESIEE Paris - Image processing for mobile systems. This lecture, together with lab sessions, introduces image processing and filtering fundamentals. It is supported by lab sessions, where the students are to develop a simple application supposed to run in real time, processing the webcam image stream. The students become acquainted with OpenCV, color conversion and motion detection.

• ESIEE Paris - Optimization methods in image processing. This lecture focuses on algorithmic aspects and optimal implementation of image processing techniques. It explains the necessity of good understanding the code, data dependence, memory management to achieve efficient implementation of applications. Using the Deriche contour detection, and the Hough transform, it shows a funny application example of lane departure detection for car drivers.

2.2 Journals' Reviewer Service

I regularly (usually oscillating from three to four times a year) provide a review service to several journals in the domain of image processing an object recognition. This list is not exhaustive (only the most recent, since 2009).

- Image Analysis and Stereology
- Signal Processing
- IMAVIS
- TCAS
- Signal Processing: Image Communication
- Signal, Image and Video Processing
- IET, Image Processing
- Journal of Mathematical Imaging and Vision

2.3 Member in Research Groups

- IAPR Technical committee #15 - Graph-based Representation
- GDR 720, ISIS - Groupe de Recherche, Information Signal Image Vision

Chapter 3

PhD Candidates Supervision

During my stay at CMM, since the end of 2000, I had the occasion to participate in the direction of several PhD students, listed below. The list is divided into two groups according to my implication: i) a co-direction properly speaking, where my role was an official co-director, and ii) an implication in the guidance, during a temporary stay of the PhD candidate, during which he has been working under my guidance.

Co-direction (in inverse chronological order) :

1. *Jan Bartovský* [defended 2012] - co-directed with M. Akil, E. Dokládalová (ESIEE) and V. Georgiev (Univ. West Bohemia). Jan's main contributions include the proposition of an efficient algorithm for 1-D morphological opening, a two-level parallelism allowing efficient implementation of sequential morphological filters. Jan has developed on an FPGA the first morphological processor with arbitrarily large neighborhood, allowing to obtain previously unachieved performances.
2. *Vincent Morard* [defended 2012] - co-directed with E. Decencière. Vincent has contributed to the detection of thin curvilinear objects in combination with statistical learning. He has contributed to path filtering, adaptive structuring elements, attribute thinnings, parsimonious statistical analysis (AdaCOS), and has proposed an efficient algorithm for morphological opening. Application to material imaging, industrial control.
3. *Olena Tankyevych* [defended 2010] - co-directed with H. Talbot. The objective of Olena's work is the enhancement and detection of thin objects in presence of noise. Morpho-hessian - a contribution to spatially-variant morphological filtering. Application to enhancement and detection of thin objects in medical images : detection of vessel systems (MRI of the head), detection of catheter (computer assisted cardio surgery).
4. *Raffi Enficiaud* [defended 2007] - I have directed Raffi's PhD work from 2003 to 2004, when he was affected to the Cosmeca project, and particularly to the car drivers' drowsiness detection. I have stopped advising Raffi's PhD after 2004 when his topic has diverted to object oriented programming applied to image processing.
5. *Eva Dejnožková* [defended 2004] - co-directed from 2000 to 2004 with J.-C. Klein. Eva has proposed Massive Marching - a first massively parallel algorithm for computing the propagation of a front. It can be used for solving the eikonal equation, or as the first fully parallel algorithm for computation of the watershed.

Implication in scientific guidance.

Hugo Hedberg and Pavel Karas have done - during their PhD preparation - a several months stay in France, at CMM and at ESIEE, respectively, during which we had the occasion to work together. They could produce, under my guidance, an interesting scientific output.

- *Pavel Karas* [to be defended 2013] - direction Michal Kozubek, Faculty of Informatics, Masaryk University, Brno, Czech Republic. Under my guidance, in the framework of his PhD, oriented towards efficient implementations of morphological operators on various platforms, Pavel Karas has evaluated several 1-D opening algorithms in the view of their efficient implementation on a massively parallel platform, a Tesla GPU. This evaluation has allowed choosing the Morard opening algorithm as the best candidate to implement. Pavel has adapted the Morard algorithm to the HW specificities, and to working under different orientations to compute the pattern spectrum. His implementation allows obtaining a previously unachieved throughput for high-end industrial applications.
- *Hugo Hedberg* [defended 2008] - direction Viktor Öwall, Head of the Department of Electrical and Information Technology, Lund University, Sweden. During his internship at CMM, Hugo has proposed an efficient algorithm for spatially varying binary morphological dilation, and its implementation of FPGA.

Master project supervision:

I had the occasion to advise one Paris-Est University master student, *Petr Šebesta*, during his internship in CMM. Petr has developed and implemented on a GPU an interactive viewer application for large 3-D medical data using a progressive resolution ray-tracing rendering.

Chapter 4

Academic Collaboration

1. Department of Electrical Engineering, University of Lund - efficient FPGA implementation of spatially-variant binary morphological operators for application in intelligent-camera systems, hand-held devices, etc. I have provided a scientific guidance to a PhD candidate Hugo Hedberg, who has spent a four-month internship, working at CMM. The results of this collaboration were published in [41].

2. ESIEE Paris - co-direction of two PhD students:

   • Ollena Tankyevych - a PhD thesis co-supervised with Hugues Talbot, associate professor at ESIEE Paris.

   • Jan Bartovský - PhD thesis co-directed by Mohammed Akil, E. Dokládalová, professor and associate professor at ESIEE Paris and Vjačeslav Georgiev, associate professor at University of West Bohemia, Pilsen, Czech Republic. We have received Jan at CMM for a one-year internship during his PhD. Under my guidance, Jan has developed and implemented on a FPGA a collection of morphological operators in a very efficient way. The results were published [3–5]. Some results were also used in [6].

3. Institute of Experimental and Applied Physics (Czech Technical University of Prague). Detection and classification of particle tracks recorded by the Medipix and Timepix devices. These devices are matrix detectors able to record energetic deposits freed during the impact of a charged particle (alpha, gamma or electron). These particles leave a different shape tracks that can be used for various purposes: analysis of radioactive fluxes, high-contract material or biological radiography, low-dosage scintigraphy, etc.

   I have provided a scientific guidance to two student master projects. Previous results have been published in [6], other results are to be published.

4. Laboratoire d'Optique et Biosciences, Ecole Polytechnique - detection and analysis of orientation of collagen fibres in microscope images. This tool is used for two applications: i) as aid to the development of the image acquisition by ellipsometry, by providing a ground truth data for the orientation measurement [78], and ii) for diverse biological studies, e.g. evaluation of the answer of the skin under mechanical stress. This work is still being in progress, the results are to be published.

Chapter 5

List of Publications

International journals (peer reviewed)

1. E. Decencière, E. Tancrède-Bohin, P. Dokládal, S. Koudoro, A.-M. Pena and T. Baldeweck, Automatic 3D segmentation of multiphoton images: a key step for the quantification of human skin. Skin Research and Technology, DOI: 10.1111/srt.12019

2. J. Bartovský, P. Dokládal, E. Dokládalová, M. Bilodeau, M. Akil. Real-Time Implementation of Morphological Filters with Polygonal Structuring Elements. J. Real-Time Image Processing, 2012, DOI 10.1007/s11554-012-0271-8

3. O. Tankyevych, A. Dufour, B. Naegel, H. Talbot, C. Ronse, J. Baruthio, P. Dokládal and N. Passat. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Medical Image Analysis, (*to appear*)

4. V. Morard, E. Decencière, P. Dokládal. Efficient geodesic attribute thinnings based on the barycentric diameter. J. Mathematical Imaging and Vision, 2012, DOI 10.1007/s10851-012-0374-7

5. V. Morard, P. Dokládal and E. Decencière, One-dimensional openings, granulometries and component trees in O(1) per pixel. J. of Selected Topics in Signal Processing, DOI: 10.1109/JSTSP.2012.2201694

6. P. Karas, V. Morard, J. Bartovský, T. Grandpierre, E. Dokládalová, P. Matula and P. Dokládal, GPU Implementation of Linear Morphological Openings with Arbitrary Angle, J. Real-Time Image Processing, DOI: 10.1007/s11554-012-0248-7

7. J. Bartovský, P. Dokládal, E. Dokládalová and V. Georgiev. Parallel Implementation of Sequential Morphological Filters, J. Real-Time Image Processing, 2011, DOI: 10.1007/s11554-011-0226-5.

8. P. Dokládal and E. Dokládalová. Computationally Efficient, One-Pass Algorithm for Morphological Filters, J. Vis. Commun. Image R., 22 , 2011, 411–420.

9. H. Hedberg, P. Dokládal and V. Öwall. Binary Morphology with Spatially Variant Structuring Elements: Algorithm and Architecture. IEEE Transactions on Image Processing, vol. 18, no. 3, march 2009, 562-572.

10. E. Dejnožková and P. Dokládal. Embedded real-time architecture for level-set-based active contours. EURASIP - Journal of Applied Signal Processing, 17, 2005,1-16.

11. E. Dejnožková and P. Dokládal. A parallel architecture for curve-evolution PDEs. International Journal of Image Analysis and Stereology, 22, 2003.

12. P. Dokládal, I. Bloch, M. Couprie, D. Ruijters, R. Urtasun and L. Garnero. Topologically controlled segmentation of 3D magnetic resonance images of the head by using morphological operators. Pattern Recognition, 36:2463–2478, 2003.

13. C. Choleau, P. Dokládal, J-C. Klein, W. K. Ward, G. S. Wilson and G. Reach. Prevention of hypoglycaemia using risk assessment with a continuous glucose monitoring system. Diabetes, 51:3263–3273, November 2002.

Conferences (with lecture committee and proceedings)

1. P. Matula, E. Tancrède-Bohin, P. Dokládal, S. Koudoro, A.-M. Pena, E. Decencière, T. Baldeweck. Automatic segmentation of epidermal layers through in vivo skin multiphoton microscopy images, Focus on microscopy, Maastricht, Netherlands, 2013

2. J. Bartovský, P. Dokládal, E. Dokládalová, M. Bilodeau. One-scan algorithm for arbitrarily oriented 1-D morphological opening and slope pattern spectrum, ICIP, 2012

3. J. Bartovský, E. Dokládalová, P. Dokládal, M. Akil. Efficient FPGA architecture for oriented 1-D opening and pattern spectrum, ICIP, 2012

4. T. Baldeweck, E. Tancrède, P. Dokládal, S. Koudoro, V. Morard, F. Meyer, E. Decencière, A. -M. Pena. In vivo multiphoton microscopy associated to 3D image processing for human skin characterization. Photonics West, 2012

5. N. Ortega-Quijano, B. H. Haj Ibrahim, S. Bancelin, M.-C. Schanne-Klein, A. Nazac, P. Dokládal, E. Decencière, J. L. Arce-Diego, A. De Martino. Orientational characterization of fibrillar collagen in histopathological samples by SHG microscopy and Mueller polarimetry, Photonics West, 2012

6. V. Morard, E. Decencière, P. Dokládal. Region growing structuring elements and new operators based on their shape, The 13th IASTED International Conference on Signal and Image Processing (SIP), December, 2011.

7. J. Bartovský, D. Schneider, E. Dokládalová, P. Dokládal, V. Georgiev, and M. Akil, Morphological Classification of Particles Recorded by the Timepix Detector , International Symposium on Image and Signal Processing and Analysis, ISPA 2011.

8. J. Bartovský, P. Dokládal, E. Dokládalová, Fast Streaming Algorithm for 1-D Morphological Opening and Closing on 2-D Support, ISMM 2011

9. V. Morard, E. Decencière, P. Dokládal, Geodesic attributes thinnings and thickenings, ISMM 2011

10. V. Morard, P. Dokládal, E. Decencière, Linear Openings in Arbitrary Orientation in O(1) per Pixel, ICASSP 2011

11. J. Bartovský, P. Dokládal, E. Dokládalová, and V. Georgiev , Stream Implementation of Serial Morphological Filters with Approximated Polygons, ICECS 2010

12. J. Bartovsky, Eva Dokládalová, Petr Dokládal and V. Georgiev, Pipeline Architecture for Compound Morphological Operators , ICIP 2010

13. O. Tankyevych, H. Talbot, P. Dokládal, N. Passat, Direction- adaptive grey-level morphology: Application to 3D vascular brain imaging, ICIP 2009.

14. O. Tankyevych, H. Talbot, P. Dokládal and N. Passat. Spatially Variant Morpho-Hessian Filter: Efficient Implementation and Applications. ISMM 2009,

15. P. Dokládal and D. Jeulin. Extraction of fibres from X-microtomographic images of fiber-reinforced composite materials. ISMM 2009,

16. P. Dokládal and E. Dokládalová. Grey-scale Morphology with Spatially-Variant Rectangles in Linear Time. Acivs, October 2008,

17. P. Dokládal and E. Dokládalová. Grey-scale 1-D dilations with spatially-variant structuring elements in linear time. Eusipco, August 2008.

18. O. Tankyevych, H. Talbot and P. Dokládal. Curvilinear Morpho-Hessian Filter. ISBI, May 2008,

19. O. Tankyevych, L. Marak, H. Talbot and P. Dokládal. Segmentation of 3D nano-scale polystyrene balls. ISMM, October 2007.

20. S. Marusic, P. Dokládal, A. Erginay, J-C. Klein and M. Palaniswami. Frequency Selective Height-from-Stereo Reconstruction : Application to Retinal Oedemas Detection. Eusipco. September 2007.

21. P. Guermeur, P. Dokládal, E. Dokládalová and A. Manzanera. FPGA Lab Session in a General-Purpose Image Processing Course. RCE. Avril 2007.

22. P. Dokládal. Fast implementation of variational, contour-based object tracking. EUSIPCO. September 2005.

23. E. Dejnožková and P. Dokládal. Modelling of Overlapping Circular Objects Based on Level Set Approach. ICIAR 2004, volume I., pages 416–423. Springer, September/October 2004.

24. E. Dejnožková and P. Dokládal. Asynchronous multi-core architecture for level set method. IEEE ICASSP 2004.

25. P. Dokládal, R. Enficiaud, and E. Dejnožková. Contour-based object tracking with gradientbased attraction field. IEEE ICASSP 2004.

26. E. Dejnožková and P. Dokládal. A parallel algorithm for solving the Eikonal equation. IEEE ICASSP, Hong Kong, April 2003.

27. E. Dejnožková and P. Dokládal. A multiprocessor architecture for PDE-based applications. VIE 2003, Guildford UK, July 2003.

28. E. Dejnožková, P. Dokládal, and J.C. Klein. A parallel computation of distance function. In Recent Trends in Multimedia Information Processing, IWSSIP, Manchester UK, November 2002.

29. P. Dokládal, R. Urtasun, I. Bloch and L. Garnero. Segmentation Of 3D Head MR Images Using Morphological Reconstruction Under Constraints And Automatic Selection Of Markers. IEEE ICIP, 2001.

30. G. Reach, C. Choleau, P. Dokládal, and J.-C. Klein. Prevention and management of hypoglycaemia using the concept of finite state machine. In 21st workshop of the study group on artificial insulin delivery, pancreas and islet transplantation, Acta Diabetologica, volume 38, Igls, Austria, January 2001.

31. P. Dokládal, C. Lohou, L. Perroton, and G. Bertrand. Liver blood vessels extraction by a 3-D topological approach. In MICCAI '99, Cambrigde, Great Britain, September 1999.

32. P. Dokládal, C. Lohou, L. Perroton, and Gilles Bertrand. A new thinning algorithm and its application to extraction of blood vessels. In BioMedSim '99, April 1999, France.

33. P. Dokládal and G. Bertrand. Adjacency graphs and image segmentation: Study of application. In Biosignal '98, Université Technique de Brno, République Tchèque, 1998.

34. P. Dokládal. Topology operators for image segmentation. In ECSAP '97, June 1997.

35. P. Dokládal. Greyscale 2D-image topology using graphs. In Telekomunikace '96, 1996.

Patents

1. French patent "Procédé pour caractériser l'épiderme et le derme à partir d'images multiphoton tridimensionnelles in vivo de la peau", N^o OA11508

Thematic journals

J.-C. Klein, P. Dokládal, G. Ségarra, A. Caduff, and G. Slama. Surveillance en direct: Prévention des hypoglycémie sur la route. In Equilibre, Association Française des Diabétiques, N°241, October, 2003.

In preparation

Journal articles:

1. V. Morard, P. Dokládal and E. Decencière, Parsimonious path openings, Transactions of Image Processing, (**submitted**)

2. V. Morard, P. Dokládal and E. Decencière, Adaptive Coefficient Shrinkage.

3. V. Morard, E. Decencière, P. Dokládal, T. Baldeweck, A. M. Pena, E. Tancrède, C. Cornillon, Characterization of the Dermis for Photoaging Evaluation.

Patents:

1. Procédé de contrôle non-destructif de pièces métalliques

Part II

Detection of thin objects

5.1 Introduction

Detection of thin structures is a ubiquitous but difficult task in image processing, used in a great range of applications going from the biomedical field to the industrial domain. One can cite a handful of examples, blood vessels extraction from eye fundus images [114, 118], road detection from remote sensing images [50, 107] or automated cracks detection from metallic parts for non-destructive testing Morard *et al.* [71, 72]. The difficulty to detect these objects comes from their thinness, and from the possible presence of noise. Such objects easily suffer from disconnections due to noise or image acquisition artifacts such as partial volume effect.

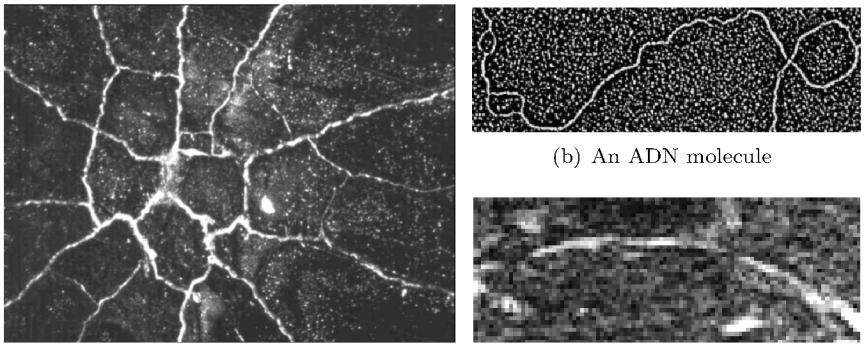
In mathematical morphology, the typical approach to enhance thin structures is to compute the supremum of openings (or the infimum of closings) with linear structuring elements in many orientations [55, 97]. The same strategy can be used with a bank of directional Gabor filters or Difference-of-Gaussians filters [53, 63]. However, this approach encounters limitations with highly tortuous curvilinear structures.

We have proposed tools from the domain of adaptive mathematical morphology to levy the limitation on maximal tortuosity. In Tankyevych *et al.* [103] and later [102], a scale space approach based on the Hessian matrix is used to detect the local orientation. In Morard *et al.* [72], the structuring elements adapt their shapes to enhance very thin cracks of any tortuosity. In Morard *et al.* [113], the so-called area opening is a connected filter, generalized by Breen and Jones [12].

Using non-increasing criteria to build attribute thinnings yields very efficient filters. The inertia of the connected component (CC) weighted by its area gives an interesting shape descriptor for elongated structures [106, 116]. Then, thinnings based on geodesic attributes Morard *et al.* [71] are efficient to filter any CC using geodesic length, geodesic tortuosity or geodesic elongation criteria. Besides, more constraints can be introduced to these connected filters. The so-called *path openings* (PO) proposed by Heijmans *et al.* [42, 43] use underlying oriented acyclic graphs to restrict the set of possible paths. This filter efficiently removes from an image all the paths which are shorter than a given threshold.

The path openings, in the original version, suffer from several drawbacks. They are not invariant under rotation. The length of diagonal paths is overestimated. An improvement of this bias is proposed in Luengo Hendriks [44]. Another problem is that in the presence of noise, the detection of long, thin, curvilinear structures is unreliable since longer structures are more likely to be corrupted by the noise. Talbot and Appleton [100] solve this problem by proposing *incomplete path openings*, able to deal with gaps in the paths. In result, the computation complexity is high, prohibitive for many applications.

In Morard *et al.* [73] we have proposed a parsimonious version of path openings. It detects curvilinear, thin structures using complete or incomplete paths with a linear complexity algorithm. Additionally, it decreases the anisotropy. The accuracy is improved and the computation times are reduced by several orders of magnitude compared with path openings.



(a) Cracks in a metallic part

(b) An ADN molecule

(c) A brain vessel (MRI)

Figure 5.1: Thin objects in automated image processing applications.

Chapter 6

Morpho-Hessian Approach

Again, the motivation of this work is enhancement and detection of elongated, curvilinear objects.

The classical morphological method to reconnect noise-corrupted thin objects is to use a spatially variant closing by a segment, aligned with the structure to restore. The local orientation can be detected in various ways. However, this needs to be done with an extreme caution, since artifacts can be created wherever two thin objects approach or in junctions. We propose a logical solution to this issue by detecting the local orientation and size and then using a spatially-variant closing steered by a local analysis of the structures to restore.

We propose a hybrid second-order-derivative analysis and morphological filtering method within the framework of scale-space theory. We detect the local orientation by using the regularized, partial, second derivatives of the 3D input image, see Frangi [35]. These derivatives are obtained by convolving the image with partial second derivatives of a gaussian kernel at some scale σ, [34,52]. Collected and written in a matrix notation, the derivatives form the Hessian matrix. The eigen-decomposition of this matrix can be used to detect the local shape, see e.g. Sato *et al.* [35,87]. For example, a thin, curvilinear shape gives $\lambda_1 \ll \lambda_2, \lambda_3$ and the eigenvectors e_2, e_3 define the plane orthogonal to the local orientation vector.

Based on these observations, Frangi *et al.* [35] have developed a discriminant function allowing enhancing tubular structures and attenuating other morphologies. Applied at different scales σ, the vesselness allows to detect structures of different size, each at its appropriate scale.

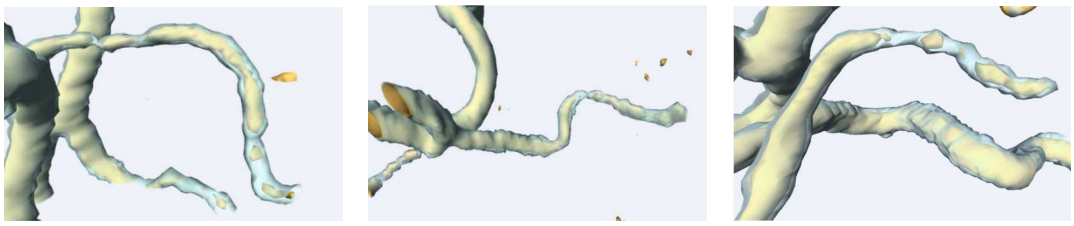
The vesselness function gives a probability for a pixel to belong to a tubular object. It offers a possibility to segment tubular objects by thresholding the vesselness η above some threshold. Whereas the orientation field is quite reliable inside the objects it gives a sparse orientation information outside. Hence, the orientation field is propagated from inside to the vicinity of the object by dilation of the vector orientation field.

The reconnection of discontinued objects is then done by using a spatially-variant morphological closing by a segment oriented at every voxel in the direction of the local orientation. We compose the closing as $\varphi = \varepsilon_B \delta_{\widehat{B}}$. The dilation by the reflected SE $\widehat{B}$ acts by propagating values from the position of the SE's origin to the other voxels covered by the SE. The reason is that the orientation field is reliable inside the objects and random outside. Fig. 6.1 gives an example of the results.

This hybrid morpho-hessian filter has been explored in the PhD thesis of Ollena Tankyevych [101], principally motivated by medical applications: by the diagnosis, treatment planning and follow-up of vascular diseases.

i) The first application, the assessment of arteriovenous malformations (AVM) of cerebral vasculature. The small size and the complexity of the vascular structures, coupled to noise, image acquisition artifacts, and blood signal heterogeneity make the analysis of such data a challenging task.

ii) The second application concerns the processing of low dose X-ray images used in interventional radiology therapies requiring the insertion of guide-wires in the vascular system of patients. Such procedures are used in aneurysm treatment, tumor embolization and other clinical procedures. Due to low signal-to-noise ratio of such data, guide-wire detection is needed for their



(a) (b) (c)

Figure 6.1: Morpho-Hessian filtering results (cyan) on a brain vessel image (yellow).

visualization and reconstruction. The results have been published in [102].

Chapter 7

Parsimonious Path Openings

The Heijmans' path openings and closings [42,43] have been proposed to extract thin, long and not necessarily straight structures. Short structures, supposedly introduced by noise are eliminated, whereas the long ones remain unchanged. These structures are modeled by connected paths allowing a limited local curvature. The opening explores all paths from a defined connectivity class, and filters them against a length criterion.

The Heijmans' path openings admit a dual – the path closings. Whereas the path openings select and filter bright structures, the path closings (after inversion of the image) select and filter the dark structures in an image.

The original path openings by Heijmans are not invariant under rotation. Indeed, the length of diagonal paths is overestimated. An improvement of this bias has been proposed by Hendriks [44]. The Hendrick's solution consists of partially constraining the connectivity graph. The connectivity of the graph globally retains its flexibility, but local oscillations of the paths are attenuated.

Path opening is a costly operator since the number of paths passing through a pixel is combinatorial. Optimizations have soon been proposed by Appleton *et al.* [1,100], allowing to decrease the complexity to logarithmic. A recent contribution in Cokelaer [20] has proposed an efficient implementation of incomplete paths in 3-D.

Despite a number of optimizations, the path openings – and especially the incomplete path openings – still remain prohibitive for time-critical industrial applications. Knowing that processing all paths makes the process generally slow and that a great majority of these paths bring redundant information we introduce *parsimonious* path openings based on an incomplete scan of the image, see Morard *et al.* [73]. Compared to the original version, there are two major modifications: i) a relevant subset of paths is selected according to the content of the image and, ii) the research of paths is decoupled from the operator applied alongside these paths.

We propose a parsimonious scan of the image support with a parameter allowing to prefer either local or global search. Whereas the local search offers better accuracy, the global search offers better robustness to noise.

The parsimonious path openings bring three major improvements: i) they extract preferentially relevant structures in the image, ii) they prevent the overestimation of the length of diagonal paths, and iii) finally, they can advantageously be implemented with the efficient, 1-D opening algorithm discussed in Sec. 9.2. Together with the incomplete scan of the support, and the algorithmic efficiency for any data accuracy (integer or real data), we have accelerated the timings by several orders of magnitude.

Consider for some image f a family of paths $\pi_i \in \Pi_f$. The signal $f(x)$, $x \in [x_1, \ldots, x_n] = \pi_i$ is a 1-D signal. Let $\gamma_L^{\pi_i}(f)$ denote the PPO opening of size L on f alongside π_i. Then we also have

$$\varphi_L^{\pi_i}(f) = -\gamma_L^{\pi_i}(-f) \tag{7.1}$$

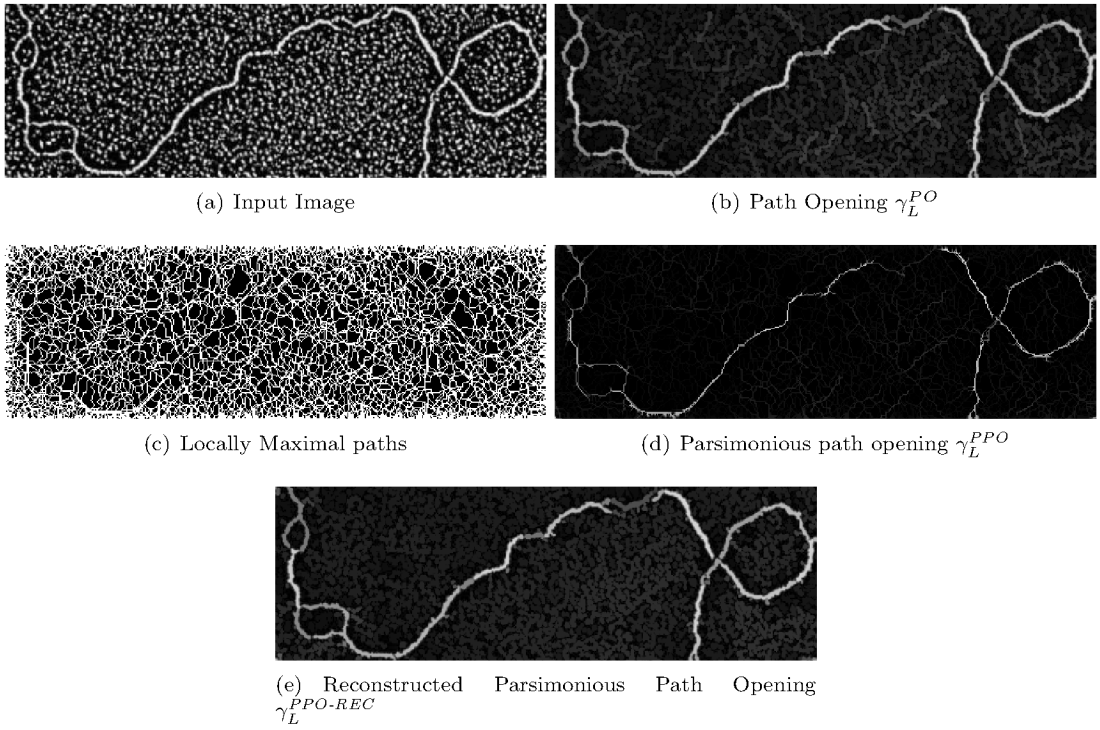
the closing of size L on f alongside π_i. Such a path closing possesses all the properties of closings (increasingness, extensivity and idempotence). The opening and closing γ^π and φ^π admit the composition of sequential filters provided they are applied alongside the same path π.

A *parsimonious incomplete-path opening* can be computed by first reconnecting the paths by a small closing on each path π_i, followed by an opening of size L, alongside the same path π_i. A parsimonious incomplete-path opening $\gamma_{L,\epsilon}^{PIPO}$ is given by:

$$\gamma_{L,\epsilon}^{PIPO}(f) = f \wedge \bigvee_{\pi_i \in \Pi_f} \gamma_L^{\pi_i} \varphi_\epsilon^{\pi_i}(f) \tag{7.2}$$

The infimum between the input image and the result is to preserve the required anti-extensivity of an opening. The computation complexity remains unchanged.

The parsimonious path openings yield a thin result. A result equivalent to path openings can be obtained by reconstruction under the input image. Fig. 7.1 illustrates an example of DNA observed in a scanning electron microscope. The molecule has been detected by using morphological reconstruction of parsimonious incomplete-path openings under the input image.



(a) Input Image

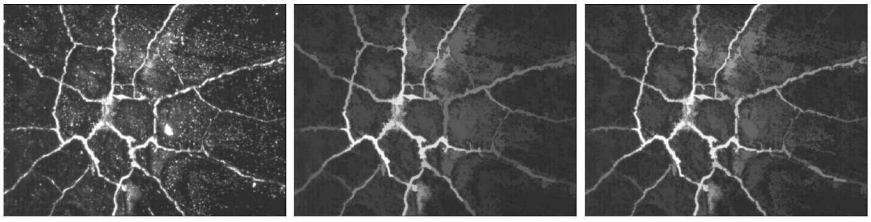
(b) Path Opening γ_L^{PO}

(c) Locally Maximal paths

(d) Parsimonious path opening γ_L^{PPO}

(e) Reconstructed Parsimonious Path Opening $\gamma_L^{PPO\text{-}REC}$

Figure 7.1: Parsimonious path openings with $L = 40$.



(a) Cracks in a metallic part (b) $\gamma_{L,k}^{PPO-REC}$ (c) $\gamma_{L,k,\delta}^{PPO-REC}$

Figure 7.2: Detection of cracks : (a) input image, (b) parsimonious path openings of size $L = 80$ pixels with $k = 10$ and (c) parsimonious incomplete path openings of size $L = 80$ pixels with $k = 10$ and with a tolerance of $\delta = 5$ pixels.

Chapter 8

Attribute Thinnings

The path openings relax the local curvature limit of supremum of openings by a rigid segment to detect thin, curvilinear structures. As said above, instead of using rigid linear structuring elements, they use flexible paths inferred from an underlying connectivity graph. The paths are kept if, and only if, their length is longer than a given constant. Nonetheless, randomly tortuous, or coiled structures cannot be entirely detected.

In Morard et al. [75] we relax any existing constraint on the tortuosity. We wish to keep or delete objects according to intuitive *attributes* like length, tortuosity, elongation or circularity. Given some connected component X_i, these attributes will be respectively denoted $L(X_i)$, $T(X_i)$, $E(X_i)$ and $C(X_i)$. These attributes allow to define *criteria* like "longer than or equal to" $(L(.) \geq \lambda)$, or "less tortuous than" $(T(.) < \lambda)$, with some $\lambda \in \mathcal{R}^+$. Formally, a criterion χ is a function mapping the set of connected components of D into $\{0, 1\}$, where 0 can be interpreted as *false* and 1 as *true*.

In mathematical morphology, an *attribute opening* is an idempotent, anti-extensive and increasing operator, which filters image connected components which do not fulfill a given criterion. However, the above-listed attributes do not necessarily form increasing criteria. Without the increasingness, we obtain the – more general – *attribute thinnings*.

Whereas binary attribute openings naturally extend to gray-scale images through the thresholding, the extension of the non-increasing attribute thinnings is not straightforward. Some filtering rules are reported in the literature to construct this extension [12, 86, 106]. The choice of the rule depends on the application. However, it is shown in [105, 106], that the subtractive rule is preferable. This is the only rule that fulfills two intuitive requirements: (i) after filtering, all the structures that do not meet the criterion are removed; and (ii) the difference image $f - \rho_\chi(f)$ contains only the structures that do not meet the criterion. The crack detection application requires the separation of cracks from the background and the subtractive rule is perfectly suitable. Therefore, we are using [75] the *subtractive rule*, to extend thinnings to gray scale images.

Additionally, the above-mentioned attributes, based on the objects' length, usually use the costly geodesic diameter, initially proposed by Lantuéjoul and Beucher [56]. The direct approach to compute the geodesic diameter of an object X consists of computing, for each point of the boundary, the geodesic distance within X to all other points of X. The maximum of the supremum of the distances is the geodesic diameter. The complexity of this algorithm is high (depends on the area and the perimeter). Schmitt [88] showed that using a subset of the boundary points is sufficient. However, despite the important speed-up thus achieved, still too many propagations remain to compute. Maisonneuve and Lantuéjoul designed an efficient parallel algorithm to compute the geodesic diameter in a hexagonal grid [57]. Using a particular propagation in the hexagonal grid, starting from the object's boundary, the algorithm gives the geodesic diameter in a single propagation. However, the object needs to be simply connected, otherwise the algorithm never ends. This limitation is too restrictive, since a group of cracks may represents a non-simply connected object (see below Fig. 8.3(a)).

In Morard *et al.* [75] we propose a new measure, the barycentric diameter that only uses two

distance calculations, and evaluate its accuracy. Secondly, we propose an original algorithm to efficiently compute an attribute thinning on gray-scale images using the subtractive rule, for any criterion based on the barycentric diameter. The algorithm uses a queue, and simulates the relief emerging process. We start with the relief completely submerged by water, and let the water progressively sink. As soon as appears the first (global) maximum, its connected component is progressively reconstructed and tested on the criterion, see Fig. 8.1 component 1. Other local maxima progressively appear, but are not yet process and we reconstruct all connected component at lower threshold sets that are supersets of the global maxima. Finally, other local maxima are processed in the same way but the aggregation stops when a CC is already processed (Fig. 8.1 components 2 and 3).

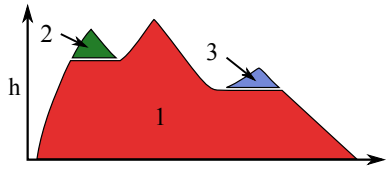


Figure 8.1: Illustration of the algorithm principle for a one-dimensional signal with three local maxima. The component 1 is analyzed first, followed by the components 2 and 3.

Its worst-case complexity is $\mathcal{O}(n^2(log(m))$, with n the number of pixels in the image, and m the mean number of pixels in the queue. In practice though the computation time is far from this worst-case bound and is rather linear with the number of pixels.

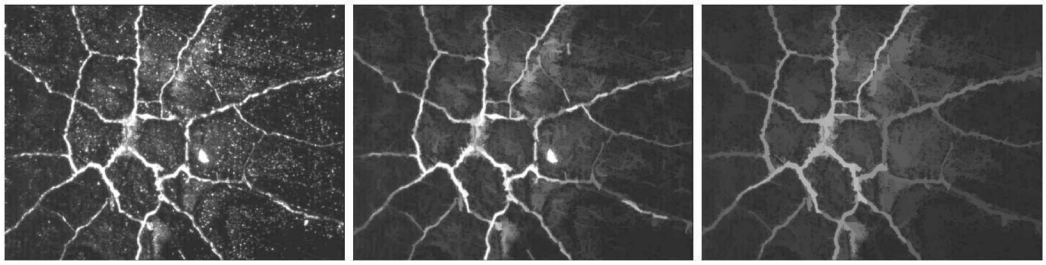
To conclude, the barycentric-diameter attribute thinnings have been developed to detect thin, tortuous objects, see examples in Figs 8.2 and 8.3. Albeit developed to detect cracks, they can be used to detect other kinds of similar fibrous structures. We illustrate the use of these thinnings in time-critical industrial applications such as automated non-destructive surface inspection. They use the subtractive rule that allows to decompose the image on components verifying/not-verifying the criterion. Compared to path openings [43] the attribute thinnings relax any constraint on the tortuosity or the local curvature, and have proven to be efficient, and faster even than the efficient implementation proposed in [44, 100].

There are two possible and interesting future generalizations of the proposed algorithm to efficiently compute other operators based on geodesic attribute thinnings. On the one hand, the morphological pattern spectra [61] estimate the size and shape distribution of the searched structures. On the other hand, ultimate openings [7, 45] extract structures with the highest contrast. Typically, these operators require the computation of a family of thinnings of increasing size. Using this algorithm, we can compute pattern spectra and ultimate thinnings within only one “relief emerging” process.

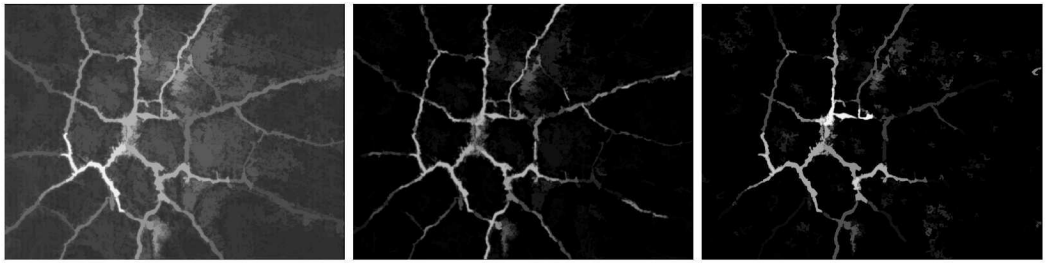
(a) Input (b) Opening with rotating segment of length 60 pixels (c) Path openings of size 160 pixels

(d) Barycentric diameter thinning ($L \geq 600$) (e) Geodesic elongation thinning ($E \geq 50$) (f) Geodesic tortuosity thinning ($T \geq 2.5$)

Figure 8.2: A DNA molecule: five different methods to detect this structure.



(a) Input (b) Opening with rotating segments of length 21 pixels (c) Path openings of size 220 pixels



(d) Barycentric diameter thinning ($L \geq 220$) (e) Geodesic elongation thinning ($E \geq 4$) (f) Geodesic tortuosity thinning ($T \geq 1.1$)

Figure 8.3: Crack detection: to detect these thin structures, we use five different methods. Geodesic attribute thinnings yield the best enhancement.

Part III

Algorithms

Chapter 9

Morphological algorithms

9.1 Efficient 1-D Dilation Algorithm

We have proposed an efficient algorithm for 1-D functional, morphological dilation with interesting properties. It processes data from the left to the right, with zero latency, very low memory consumption, a constant complexity per pixel $\mathcal{O}(1)$ and the faculty to process any scalar data of arbitrary accuracy. The combination of these properties allows an extremely efficient implementation of the fundamental morphological operation that is the dilation. We show how to compute in one pass an Alternate Sequential Filter (ASF^n) regardless the number of stages n.

It is the first algorithm combining these properties compared to the state of the art. Indeed, the first to propose a constant complexity algorithm is van Herk [109], followed by Gil and Werman [39], and later by Lemonnier and Klein [59]. On the other hand, they all require a forward and backward scans. Indeed, the first to propose a streaming algorithm with constant complexity is Lemire [58]. However, an intermediate storage of local maxima results in a random access to the input data.

The combination of the properties of zero latency and processing sequentially from the left to the right inherits under concatenation, and transposes to compound operations. This finds its importance at two practical levels:

• At the first hand, concatenation of dilations and erosions gives birth to sequential morphological filters, starting by opening and closings, up to ASF of arbitrary size. Another application are granulometries, that are sequences of openings with increasing size.

• At the second hand, concatenation of the same operation using orthogonal linear segments allows extension to higher dimensions. For example, a rectangular structuring element R decomposes as $R = H \oplus V$ where H and V are horizontal and vertical segments and $\oplus$ is the Minkowski addition. Then the dilation by a rectangle R can be computed by

$$\delta_R = \delta_V(\delta_H) \tag{9.1}$$

When non-orthogonal segments are used, one can obtain polygons. For example, the dilation by a $2n$-top ($n \in \mathbb{N}$) regular polygon SE P_{2n} decomposes into a set of n dilations by inclined linear segments L_{α_i}

$$\delta_{P_{2n}}(f) = \underbrace{\delta_{L\alpha_1}(\ \ldots\ \delta_{L\alpha_n}}_{n \text{ times}}(f)) \tag{9.2}$$

This separability is known since years, see Matheron [64], or later Serra [90]. However, the properties of the 1-D dilation algorithm make that this decomposition can be seen under a new light. Indeed, the sequential access to data at all levels, zero latency, low memory and constant complexity $\mathcal{O}(1)$, are inherited under concatenation, and make the computation of Eqs. 9.1-9.2 sequential, and very efficient.

Figure 9.1 illustrates the propagation of real image data through an ASF^4 after having read approximately one third of the input image. The SE is a square of size s+1 at the s-th stage. The

individual operators, with sequential data dependence, are running simultaneously. There is no intermediate data storage between the stages; the intermediate results are pipelined.

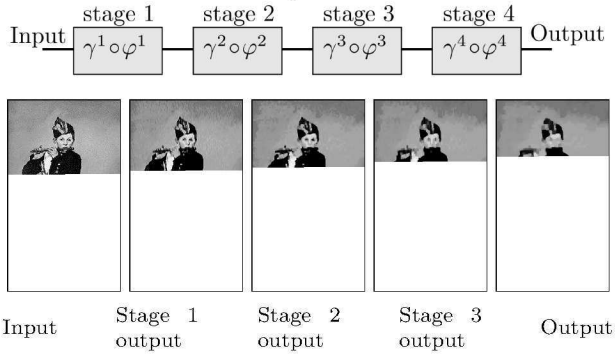


Figure 9.1: Computation of ASF^4. After having read approximately one third of the input image, the output of ASF^4 is available with only several lines of delay. See Sec. 10.3 for details regarding the implementation.

Several efficient implementations on various platforms by several PhD students (Vincent Morard, Jan Bartovský and Pavel Karas) have confirmed the expectations of performance, see Part 10 below.

This original dilation algorithm has been extended to openings and granulometries by two students (Vincent Morard, Jan Bartovský)

9.2 Efficient 1-D Opening: Algorithm 1

With Vincent Morard, we have proposed an efficient 1-D opening algorithm using the detection of a signal into a collection of cords.

Consider the decomposition of a 1-D signal into a collection of maximal cords $\mathcal{C}$, see Fig. 9.2. Given the collection of cords $\mathcal{C}$ one can directly compute attribute openings by length, the size distribution and component tree.

Consider a 1-D signal $f : \mathbb{Z} \to \mathbb{R}$. Let $X^h = \{x \mid f(x) \geq h\}$ denote the threshold of f at level h, and $\{X_i^h\}$ the set of connected components of X^h. Several threshold levels can extract the same component. The maximum k allowing to extract the component X_j^h is $k = \min_{x \in X_j^h} f(x)$. Then, we call a maximal *cord* a couple $c = (X_j, k)$, and we call k the altitude of the cord. Fig. 9.2 illustrates the decomposition of a 1-D signal into its cords.

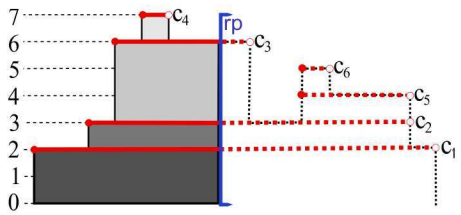


Figure 9.2: A 1-D signal and the associated maximal cords.

The component X_j corresponding to c_j is an interval of $\mathbb{Z}$, let $L(c_j)$ denote its length. The length satisfies the inclusion property, since for any two cords c_i and c_j we have $X_i \subset X_j$, if $k_i > k_j$. We say that c_j is an *ancestor* of c_i and c_i is a *descendant* of c_j. An immediate descendant is a *child*. If c_i is the child of c_j, then c_j is the *parent* of c_i. The parent-child relationship gives a tree called component tree, or max-tree. Additionally, $\mathcal{C}$ allows direct computing of the attribute

openings by length $\gamma_\lambda(\mathcal{C}) = \{c_j | L(c_j) \geq \lambda\}$ and the pattern spectrum of $\xi(\mathcal{C})(\lambda) = \sum_{L(c_i)=\lambda} V(c_i)$. Where the volume of cord c_i is given by $V(c_i) = (k_i - k_j)L(c_i)$, where c_j is the parent of c_i.

The reconstruction of a signal f from its set of cords $\mathcal{C} = \{(X_j, k_j)\}$ is straightforward:

$$f(x) = \max_{(X_j,k_j)\in C\,:\,x\in X_j} k_j$$

Hence, an efficient decomposition of a function into its set of cords allows an efficient computation of openings, pattern spectra and component trees by using only logical or arithmetic operations. We have proposed an efficient algorithm in Morard *et al.* [74]. Its efficiency stems from several facts: the signal is read sequentially; every cord is visited once, and only once, in the order child-parent; the complexity is $\mathcal{O}(1)$ per pixel. This algorithm only uses comparisons, and can then process arbitrary data types that form an ordered group.

The proposed algorithm is applied to 2-D images in several ways: i) the classical linear openings for oriented filtering and/or enhancing of linear structures, ii) the collection of size distributions for all orientations gives the oriented pattern spectrum and iii) provided the image support is endowed with a path-connectivity graph it can be advantageously used for efficient computation of Heijmans path openings [43]. See Sec. 7 above for details.

9.3 Efficient 1-D Opening: Algorithm 2

With Jan Bartovský we have proposed another algorithm for 1-D morphological openings. Following the principle of opening that erases peaks narrower than the width of the SE, it is based on recursive elimination of peaks. Conceptually, its principle follows this idea. Take narrow-vision-making glasses that limit the width of your vision to the width of the SE. Look over these glasses at the signal from the left to the right, and erase all peaks that you can see. Because of these glasses, you will not see peaks wider than the SE. All other peaks will be erased.

This principle has two main advantages:

i) the locality (you only can see a limited portion of data) reduces the memory requirements since only a limited portion of data is stored, and increases the efficiency of the memory cache.

ii) processing in stream, from the left to the right. We have discussed in Dokládal [29], also cf. Sec. 9.1, that sequential access to data is beneficial at all levels. The sequential access to data is used in streaming architectures, it eases the implementation on FPGA, but is also useful for usual, e.g. PC or GPU architectures.

The algorithm processes the signal from the left to the right. The data covered by the SE are stored in a FIFO queue. The elimination of peaks is based on a set of tested configurations. The algorithm has low - further irreducible - latency and low memory. For details see Bartovský *et al.* [3]. This algorithm accumulates - during the elimination of the peaks - the length and height of the slices to extract the pattern spectrum of the residue $f - \gamma_\lambda f$.

We have extended the 1-D opening algorithm to process 2-D data with the same principles of locality and processing in stream. Given that the input image is read in the raster scan order, that is line by line from the left to the right, an using a specific decomposition of the data stream we have sequential access to input and output data whatever is the orientation of the SE.

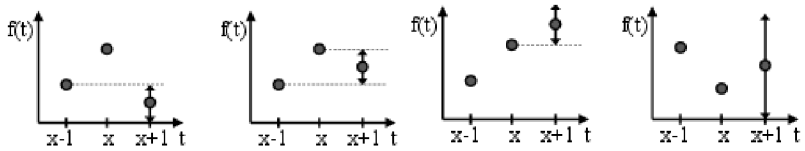


Figure 9.3: Four configurations for peak identification: (a) and (b) are a peak, whereas (c) and (d) are not.

We have obtained efficient implementation of 1-D morphological operations with rotated SE regardless the orientation of the SE. It can easily and efficiently be implemented in HW because

of the sequential access to data at all levels. We have low memory requirements, useful for large images.

Regarding the implementation on GPU. We have compared in Karas *et al.* [49] the implementation of several 1-D opening algorithms to conclude that the Bartovský algorithm effectively allows obtaining best performances, regardless the orientation of the SE.

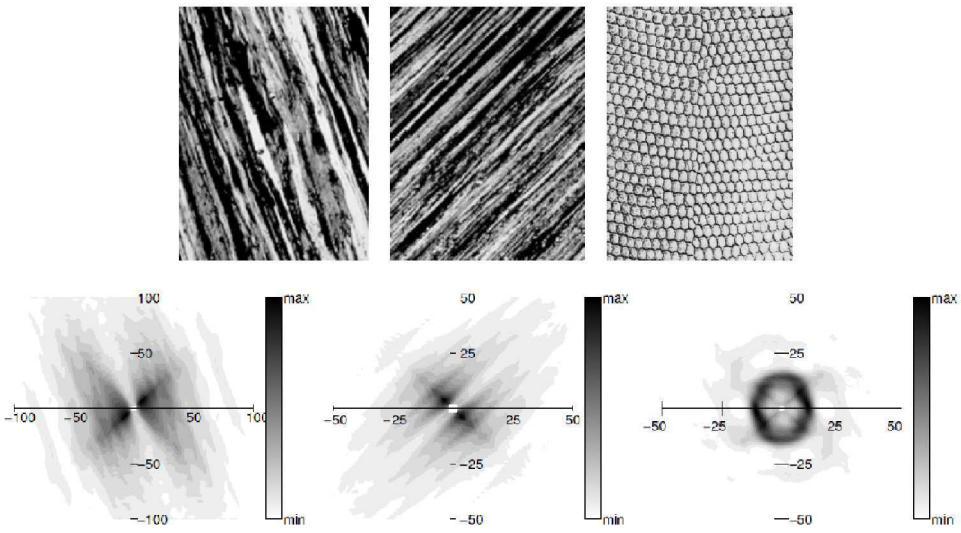


Figure 9.4: Examples of textures and their pattern spectra.

9.4 Massive Marching: A Massively Parallel Watershed Algorithm

The watershed transform, introduced by [9], is widely considered as a sequential, and difficult to parallelize operator. Nonetheless, its central position in the morphological image segmentation explains the number of attempts of its acceleration or parallelization, Moga *et al.* [69], Moga and Gabbouj [67], Meijster and Roerdink [65, 66], Bienik *et al.* [10], Noguet [77], Moga *et al.* [68], Iwanowski and Aswiercz [48] or recently Van Neerbos *et al.* [110].

In Dejnožková [119], we have presented a new parallel algorithm, called Massive Marching for solving the Eikonal equation. Massive Marching propagates the solution of the Eikonal equation from initial conditions, a set of markers in the form of contours, arbitrarily placed with a sub-pixel accuracy. Hence, it can be seen as an algorithm for computing the weighted distance function. With its non equidistant propagation mechanism, Massive Marching does not require any use of heap ordered structure. The computational complexity is close to $\mathcal{O}(1)$ per pixel.

The Massive Marching allows to propagate region labels simultaneously with the calculation of the distance function. Hence, it can also be used for computing the watershed transform.

We have studied the error of Massive Marching and have shown that it ends in a finite time. Furthermore, we have compared the results with the results of algorithms with equidistant propagation mechanism.

Massive Marching can be implemented sequentially, semi-parallely or massively parallely.

Chapter 10

Implementations

10.1 Curve-Evolution PDEs

Methods described by partial differential equations have gained a considerable interest because of indubitable advantages such as an easy mathematical description of the underlying physics phenomena, subpixel precision, isotropy, or direct extension to higher dimensions. Though their implementation within the level set framework, introduced by Osher and Sethian [79] in 1988, offers other interesting advantages, their vast industrial deployment is slowed down by their considerable computational effort.

The importance of finding an optimal solution can be seen on the considerable effort concentrated at this topic. Solutions have been sought either in i) mathematical - Weickert [115] proposes a stable scheme for filters, later extended to contours by Smereka [94], another solution was proposed by Goldenberg *et al.* [40], ii) algorithmic - Precioso and Barlaud use splines [81] or iii) implementation on a specialized hardware. Holmgren and Wallin [46] and Sethian [92] use supercomputers. Rumpf and Strzodka [84, 85], Cates *et al.* [13] or Sigg *et al.* [93] use a graphic hardware. Hwang *et al.* [47] propose an orthogonal architecture designed for numerical solution of PDEs, not inevitably related to the image processing, and Gijbels *et al.* [38] propose a VLSI architecture for nonlinear diffusion.

In Dejnožková [120], we exploit the high parallelization potential of the level set framework and propose a scalable, asynchronous, multiprocessor platform suitable for system-on-chip solutions, see Fig. 10.1. We focus on the HW-based implementation issues of the level set techniques on embedded, one-chip devices that will be easily (i) scalable, to adapt their computational power to the requirements of the chosen application, (ii) programmable with conventional programming tools, (iii) by far less energy consuming than Pentium-based desktop machines with comparable computational power, and (iv) as small sized as possible.

The contribution of the paper Dejnožková [120] is twofold. In its first part, it proposes a unifying insight into the level set framework from the system design point of view, to propose a unique type of iteration with two different types of memory access: random memory access and sequential memory access. Then it analyzes the data flow to define, in the second part, a scalable architecture fitting the real-time needs and taking into account the limited energy autonomy of embedded platforms and the silicium surface on commercially available FPGAs. we have evaluated the performance on two benchmarks. The first one, computation of a weighted distance (used for initialization of the narrow band), to verify the uniformity of the data flow and the distribution of the computation over all the processing units. The second benchmark implements an active-contour-based object-tracking algorithm.

10.2 Spatially-Variant Morphology

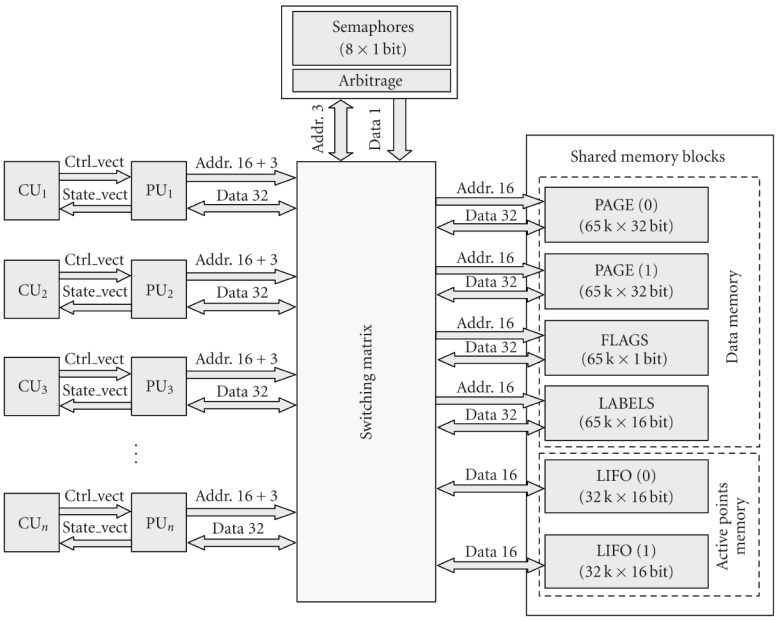


Figure 10.1: Top level view of the architecture, Dejnožková [120].

Mathematical morphology with spatially variant structuring elements has been profoundly studied in the literature over the years, since it has soon become clear that it outperforms translation invariant structuring elements in various applications: filtering of images with anamorphosis [8,83], see illustrations at Figs. 10.2, 10.5, image coding and object restoration from skeleton, [32, 54, 62, 89, 91], see Fig. 10.3, or content-aware filtering [15, 23, 24, 60, 76, 112, 117], see Fig. 10.4.

However, supporting a variable structuring element shape imposes an overwhelming computational complexity. In naive implementation, the cost dramatically increases with the size of the structuring element, e.g. Lerallut *et al.* [60] report the running time of several hours for 3-D MRI data.

Despite this evident efficiency, little optimization effort has been done so far. One can cite Cuisenaire [21] who proposes a fast algorithm for binary spatially-variant morphology based on thresholding the distance transform, widely used for efficient implementation of dilations and erosions. Using the distance function limits the class of shapes to balls of various norms. Various algorithms exist for computing the distance map. They are either i) image scan operations, e.g. [22] or ii) equidistant propagations from the sources, (see surveys [11], [33] for overview and other citations). The former have high memory requirements since they use a large intermediate storage for partial results between the scans. The distance is computed on the entire image, penalizing the performance when small SEs are used. The latter, based on equidistant propagation from the sources do not necessarily compute the distance on the entire image and are more efficient. On the other hand, the equidistant propagation needs using ordered structures and results in extensive random memory accesses (costly for large data).

We have conducted an optimization effort to accelerate SV morphological operators. We have first started with binary SV operators, extended later to functional morphology. In both, we have taken into account the needs of portable or embedded systems. We have proposed new algorithms and evaluated their implementation. We summarize this effort in the following sections.

Spatially-Variant Binary Dilation

In Hedberg *et al.* [41], we have conducted an optimization effort aimed at portable or embedded systems. We have proposed a new algorithm supporting a SV rectangular SE for binary mathe-

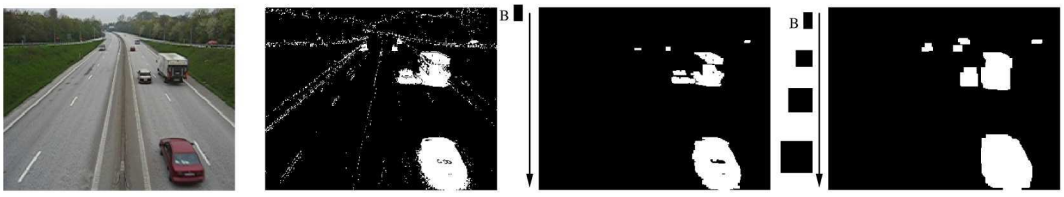


Figure 10.2: Road surveillance application: (a) input image, (b) binarized motion mask, (c) motion mask filtered using a translation-invariant SE. (d) motion mask filtered using a spatially-invariant SE, increasing in size from top to bottom.

Figure 10.3: Object reconstruction from skeleton. (a) A binary object 'tools', (b) its skeleton, weighted by distance to the complement, (c) reconstruction from the skeleton.

Figure 10.4: Contour preserving filter. (a) Original, (b) TI ASF, (c) SV ASF.

matical morphology with very low computational complexity and memory requirements. We have designed a corresponding HW architecture, suitable for embedded or mobile applications. The architecture has several important properties from a HW perspective, i.e.sequential pixel processing, low-computational complexity, and low memory requirement. The complexity is of $\mathcal{O}(n)$, where n is the width of the SE. Implementation results of the proposed architecture are presented in terms of resource utilization when targeted for both FPGA and ASIC.

The algorithm is based on computation of distance function to object edges. Decomposing the computation into columns brings restriction to $\mathcal{L}_1$. Hence, in the HW realization presented below, the allowed SE shapes are rectangles (including $\mathcal{L}_1$-balls – the squares). The paper [41] explains the extension to a richer class of shapes. Other restrictions come out if the algorithm is to be implemented in low complexity and low memory requirement architectures with no intermediate storage.

The input pixels arrive in a stream, and the result pixels are also output in a stream. The complexity (memory consumption, latency and number of operations) – far below other existing algorithms supporting SV SE – rather competes with the fastest optimizations of TI structuring elements: one image line memory, and several image lines latency (latency obviously depending of the SE size). The sequential memory access allows composing cascade filters (alternate sequential filters) with low latency, and without intermediate storage.

For more demanding applications there is an extension to support richer SE shapes (balls,

diamonds) in two raster scans. This algorithm is interesting for various use cases: cascade morphological filters running on systems under heavy time and space constraints such as embedded or communication systems or possibly also low-end user terminals.

An extension of SV dilation to gray-scale images using the same principle goes through computing the distance transform to all changes in the signal value. We develop the principle in the following section.

Spatially-Variant Functional Dilation

In [31] we have proposed a 1-D functional dilation algorithm for spatially varying structuring elements. The algorithm has interesting properties. It processed the signal in stream, that is it reads/writes the input/output signal from the left to the right, produces the result with minimal latency, and can also work in place. The algorithm is extremely efficient, competing with the most efficient TI dilation algorithms. The complexity is $\mathcal{O}(1)$ per pixel, hence the computing time is independent of the structuring element size.

The extension to 2-D is not straightforward. Indeed, a SV 2-D dilation is not separable into two perpendicular SV 1-D dilations. We have discussed these issues in Dokládal [28], where we explain that in this way one only can obtain an approximated, inexact result, where the shape of the SE is deformed. A second limitation is brought by the processing in stream, which also infers restrictions. The structuring function must be a continuous function.

Obviously, the shape inaccuracy of the SE limits the applicability where the exact shape of the SE is necessary. Nonetheless, wherever the shape is not strictly required, the efficiency of algorithm is handy. A good example is processing video streams in embedded systems. Using the exact direct and adjunct dilation or erosion – implemented by definition – is computed in $\mathcal{O}(LN^2)$, which might become prohibitive in embedded systems.

Illustrations: Detection of license plates is an example of application where the SE size is controlled by an external parameter, the distance to the camera. Using TI SE would have caused that i) the license plates come out only in a restricted (camera equidistant) zone and ii) false detections appear frequently in other zones. Using SV SE allows respecting the perspective.

The application is based on a cascade of morphological operators (see [27] or [82], we omit details here), such as: gray valued top-hat, closing and opening with a rectangular SE, with size constant (for TI SE) and progressively increasing (for SV SE). Compare the performance obtained with a TI and SV SE in Fig. 10.5. Optimal memory and latency also ease obtaining real-time performances on low-end systems (e.g. portable cameras).

A second example is a content-aware image preprocessing, e.g. image simplification, noise reduction. Consider an contour preserving filter in the example given by Fig. 10.6. To efficiently filter noise but preserve the contours, the filter needs large structuring elements, that do not cross these contours. The contours, can be detected by any usual contour detecting operator (c). Then, one can design a SE map of squared SE with side size equal half the distance to contours, as given by (d). Any filter using this SE map will preserve these contours. The rest of the image is strongly simplified.

10.3 Parallel implementation of serial morphological filters

In Sec. 9.1, we have seen an original algorithm with interesting properties for efficient computation of 1-D functional dilation. The most interesting properties for implementation on parallel platforms are processing the signal from the left to the right, with minimal latency.

For practical applications on images, however, we need an extension into higher (2-D or 3-D) dimensions. It is widely known that dilations by rectangles and polygons are separable under the Minkowski addition into a concatenation of orthogonal or inclined 1-D dilations. We have seen that the combination of these properties is inherited under concatenation. In this scope, we have proposed in Bartovský *et al.* [5] : i) a stream-preserving separation of rectangles into linear segments, and ii) a spatial parallelism allowing a further increase of performances.

(a) detection using a translation-invariant SE (b) detection using a spatially-variant SE

Figure 10.5: License plates detection under perspective deformation

(a) Original (b) Zoom (c) Contours (d) SE size map

(e) Opening fast (f) Opening exact

Figure 10.6: Contour-aware opening and closing on the Manet's painting "Le joueur de flûte".

In the following paper Bartovský *et al.* [4] we have extended the computation to dilations by polygons. This paper overcomes a number of issues: a correct handling of thick borders needed for decomposition of polygons, a partition of the 2-D image support into independent computation corridors, where every pixel is visited once and only once.

As a result a polygonal structuring element can be assembled from several inclined line segments with the sequential access to data at all levels, see Fig. 10.7. The input image is sequentially read line by line from the left to the right. The result, at the origin of the SE – assume here in the centre of the polygon (d), is available as soon as all data covered by the SE are read, that is, when the reading position reaches the down-right corner (a) of the hexagon, refer to Bartovský *et al.* [4] for details.

Conclusions: The combination of the previous allows obtaining previously unachievable performances of applications consisting of long concatenations of dilations and erosions like ASF or granulometries. We have obtained performances for high-end industrial applications running under severe time constraints. To give a concrete example, we could match real-time performances corresponding to 100Hz 1080p HD TV standard.

10.4 Massively parallel implementation

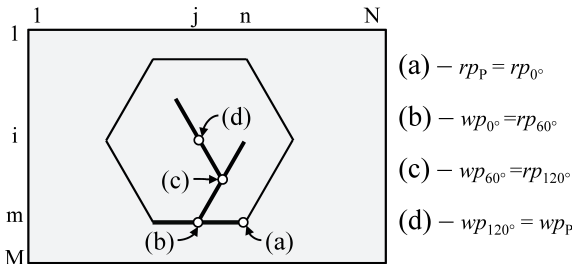


Figure 10.7: Stream concatenation of three L_{α_i} into hexagonal SE P; rp/wp - reading/writing position.

Traditionally – and still quite often – thin, curvilinear objects in images are enhanced/detected by using morphological openings by linear segments. If the orientation angle is a priory unknown, the opening must be repeated a number of times, to test all possible angles. As a by-product of this operation, at the same time, one detects the local preferential orientation. The number of tested angles increases when the objects are thinner or more tortuous, since the structuring element less likely fits in the object. This operation being costly, an efficient implementation is needed especially for high-resolution, industrial applications running under timing constraints.

We address this issue in Karas *et al.* [49]. We start from the 1-D opening algorithms discussed in Secs. 9.2 to 9.3 and implement them on a GPU to benefit from their massive parallelism capabilities. We use the nVIDIA Tesla C2050 GPU device with 14 multi-processors, having 448 cuda cores, allowing running hundreds of threads in parallel.

We can benefit from this potential by mapping individual, and independent image rows (oriented under the angle α) to one thread, see Soille [95]. Small images, that do not offer a sufficient number of rows are subdivided into horizontal, overlapping straps processed independently. After a comparison, we achieve the best performance from the Bartovský opening algorithm, Sec. 9.3 for several reasons: i) the access to both input and output data are regular, making the execution of the threads synchronous, and reducing the thread divergence. ii) the maximum memory footprint (the size of the FIFO) per thread corresponds to the size of the SE. This strongly limits the overall memory footprint, especially important to process large, high-resolution images. Finally, different types of data (input, output and temporary data) have been mapped to different memory spaces, to fully utilise the available memory bandwidth of the device.

With all optimisations we could reach the performance over 1 Gpix/s for openings of arbitrary size, and regardless the orientation for high-resolution images.

10.5 Conclusion

Thanks to the properties of the original 1-D dilation algorithm, we could obtain a tremendous performance increase compared to the current state of the art.

Regarding specialised HW implementations: Compared to existing HW implementations with limited size of SE, [17, 19, 26], we could propose streaming morphological processors on a FPGA with arbitrarily large structuring elements. We could achieve on an FPGA a 200 Mpix/s throughput, for arbitrarily long sequential morphological filter, conforming to real-time specifications of the 100Hz 1080p FullHD TV standard. This is far above what has been reported in the literature up to this date.

Regarding massively parallel platforms, we could obtain the performance over 1 Gpix/s for large, high-resolution images. This allows to conclude that the premises (sequential memory accesses, small memory footprint, stream execution) used to design an algorithm for specialised HW such as FPGA (Bartovský algorithm Sec. 9.3) are *in fine* applicable to other types of parallel hardware such as GPGPU. At the same time, pretending that the same premises are ubiquitous constants is also misleading. On most commonly used platforms such as the PC, the same premises

may not lead to obtaining the highest performances. We have seen an example with the Morard opening, Sec. 9.3, that overrun all other concurrent algorithms on a usual PC CPU.

Part IV

Contractual Research and Applications

10.6 Material Science

10.6.1 The Tocata project

The Tocata (Technologie Optique Couplée à l'Analyse Topologique Automatisée) project is a 3-year (2009-2012), Fond Unique Interministériel (FUI) project, grouping 13 partners. I have provided the administrative and scientific responsibility of this project jointly with E. Decencière. The software development was entirely ensured by Vincent Morard in the scope of his PhD project.

The objective of Tocata is to develop an industrial system for automated surface inspection of metallic parts. The benefits of this goal are two - economical and ecological. The current inspection process is done manually, and it is a rather slow and polluting process (usage of chemicals and consuming water).

The inspected parts come from the aeronautics and nuclear industry. The parts can be very different and come from different assembly groups (and provided by different manufacturers). The size of the parts ranges from centimeters to meters. The manufacturing processes include molding, lamination, forging, welding or a combination of these processes.

The parts can be inspected either new for manufacturing defects or periodically, after a life cycle for fatigue defects. The Tocata project focuses on superficial defects, and encompasses three detection modes. Armines (Center of Mathematical Morphology) was charged by the development of the optical way of defect detection. The defects are of various nature and are not necessarily visible by eye.

The difficulties to levy come from the variability of the observed structures. Indeed, these structures are either fatal defects (cracks, ruptures), after which a part is eliminated, or non-eliminating defects (scratches, corrosion, etc.).

A second difficulty comes from the huge amount of data to process in a limited time. In some applications, tiny defects (unobservable by bare eye) are to be detected in large parts, requiring scanning m^2 surfaces with a μm pixel resolution, which generates an overwhelming amount of data. At the same time the image processing part must conform to the timing constraints of the industrial cycle.

Regarding the variability of the defects, the developed method must be highly polyvalent. Indeed, to the existing defect register can be added other defects in the future. Given the timing constraints we have proposed a sparse statistical learning strategy to discriminate the fatal from the non-eliminating defects while conserving the polyvalence and adaptability. We have used a set of descriptors more or less sensitive to various defects, and a proposed an original, sparse descriptor-selection method (called AdaCOS, confidential) able to take into account both the descriptor's efficiency and its computational cost. The descriptor selection accepts a maximum time budget available for the processing, as specified by the user. Given the timing constraints it selects the most efficient subset of descriptors fitting in the processing-time budget. A priority can be given either on the accuracy of the detection or the timing constraints.

The research results have been implemented in a software application made available to the Tocata consortium for testing purposes. Some results have been published [70–72, 75], and the entire detection procedure is to be patented. The optical detection procedure is going to be used on line by the Tocata industrial partners in complement to the two other detection modes.

A fully functional prototype has been developed at the end of the Tocata project, see Fig. 10.8.

10.6.2 The Colas project

The Colas project is a three year collaboration with the Centre d'expertise et de documentation of Colas a.s. I have provided the administrative and scientific responsibility of this project.

The objective of the project is the assessment of the degradation of the surface pavement of roads by the means of image processing. Next to other measures evaluating the degradation, the objectives here were two-fold: first, the detection and analysis of cracks, and second, the detection of surface torn offs, by the means of texture analysis techniques. The detected cracks are submitted to a variety of measurements and classification.

(a) (b)

Figure 10.8: A fully functional prototype of non-destructive industrial testing. (a) the robotic arm used for the Tocata demonstrator. (b) The camera and lighting equipment mounted on the arm.

I have provided the software development for the crack detection part. The software development for the texture analysis part was provided by Vincent Morard, in the scope of his PhD thesis, under my guidance.

The cracks are classified into groups of severity according to the width. Further, the cracks are analyzed according to the position and local, mutual distance and grouping. An analysis report is produced to identify the major phenomena causing the aging (traffic, meteorological phenomena, and others).

Regarding the second task, the torn offs are not thin but patchy defects stretching over areas of variable size. They are caused essentially by the tires of the vehicles that virtually tear off the pavement material. They are observed as modifications of the texture. This modification is variable and may affect the mean or the variance of local intensity of the texture, size of the grain, frequency properties, etc. Different torn offs can be observed simultaneously in a road.

The difficulties come from the nature of the structures to detect and from the environment. The cracks are thin, erratic and tortuous structures found in the texture of the road surface. Another difficulty comes from the variety of existing surfaces, presence of dirt (soil, tree leaves, vegetation), intrusions (vegetation, sewer grids, paving blocks) or even the white, lane marking.

Again, given the amount of data to analyze (often tens of kilometers of roads scanned with 1 mm/pix resolution) our industrial partner has formulated timing constraints on the execution time.

The detection of cracks is done in several steps. First, the images are downsized using an adaptive subsampling. The cracks are enhanced by the difference of gaussians filter, and detected via an adaptive thresholding combined with an adaptive filtering.

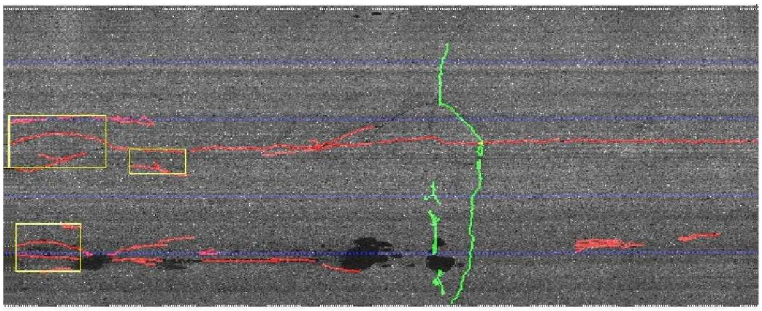
Additionally, a set of auxiliary tools is used such as suppression of the white lane marking, detection of the standard vehicle position, detection and suppression of decorative pavings, etc.

Regarding the texture modification. The a priory unknown properties of the torn offs require adopting a two phase method. In the first phase, a statistical classifier learns the properties of the sane road texture, and of the other textures possibly present in the images (the defects and the intrusions). The choice of the descriptors is user dependent. According to the context, one may (or not) use some descriptors. For example, one may want to use descriptors sensitive to the mean gray level.

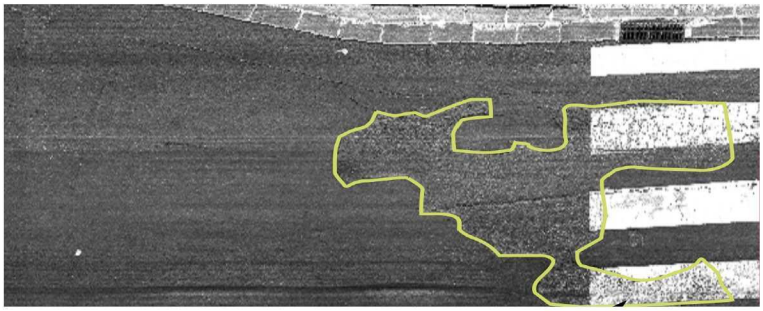
The research developments carried out in the framework of this project were implemented in a software tool, which has been successfully tested on line, and is now routinely used by the Colas

society under the commercial product name REC (Road Eagle Colas), see Fig. 10.9.

(a)



(b)



(c)

Figure 10.9: The Road Eagle Colas system. (a) The road imaging vehicle, equipped with the LRIS camera system, (b) the road cracks detection, (c) the surface torn-offs detection.

10.6.3 The Aedinca project

The Aedinca project is a partnership with the Aedinca society, specialized in research and modeling in the domain of assembly materials. I have provided the administrative task jointly with D. Jeulin, and entirely ensured the software development task.

The perimeter of the collaboration with Aedinca is modeling and optimization of the design of fiber-reinforced composite molded components. Mechanical properties of molded components made from fiber-reinforced composite materials locally depend of the orientation of the fibers. The orientation of the fibers depends of the molding process: i) the flow of the liquid binder during the injection, ii) rheological properties of the liquid binder or obviously the iii) the form of the mold and position of the injection nozzles. The molding process can be up to some extent be modeled by using a commercial software (Moldflow). Nonetheless, this model has limited precision, and small local variation can not be modeled with sufficient accuracy. The verification and optimization of the molding process requires a post verification process on manufactured components.

The evaluation of the properties we have done consists of sampling the component at known

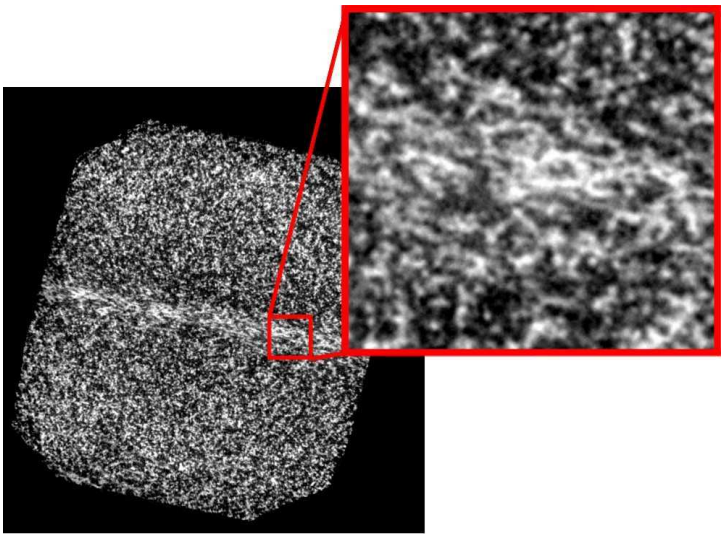


Figure 10.10: Example of a fiber-reinforced composite material illustrating a local modification of the injection flow in the center of the image (a 2-D slice extracted from a 3D X-ray scan).

positions. These sampling positions can typically situate at known zones of weakness of the component. The samples (of size of 1 mm^3) are scanned in a tomograph which yields 3-D images. We are interested in extracting the individual fibers, to analyze their length and local orientation. The segmentation of the fibers is a challenging task. First, the resolution of the reconstruction being at the limits of the capabilities of the device (optics, sensor, wavelength), the images are noisy and fuzzy. Second, the fibers have a non uniform length and are heavily tangled.

Several methods of local orientation analysis are known and have been used for specific purposes. Directional morphological filtering Soille and Talbot [96], [98]. Nonetheless, even though efficient algorithms for computing openings with multiple structuring elements has been proposed by Urbach and Wilkinson [104] the computational cost makes that this approach is unusable for large 3-D data. In addition to that, to evaluate the length of every fiber would require to let vary also the SE length. Other approaches come from the orientation space introduced by Chen and Hsu [14], Van Vliet and Verbeek [111], Chen *et al* [16] or Ginkel [108]. It works well on isolated objects. Using a bank of filters infers a trade off between accuracy and locality. This drawback makes that this approach is not usable for tangled objects. Also, the computational cost, induced by applying a set of filters, is quite high. Adding supplementary dimensions for the orientation increases the memory requirements. The orientation in 2-D is one, and in 3-D two values. Analyzing 3-D images requires working with 5-D data. Perona and Malik [80] or Frangi [36] use extraction of local orientation for enhancement of thin, elongated, tubular objects. The detection of orientation is local, based on second (or higher) derivatives. It is not suitable for noisy images. Stein *et al.* [99] use graphs to analyze the geometry of collagen gel images acquired by a confocal microscope. This work is perhaps the most similar to our approach, described below.

Starting from X-ray micro-tomographic images, the objective is to extract 3-D presentational maps of fibers in components manufactured by molding from fiber-composite materials. The map will serve as a support for the simulation of alignment of fibers in the flowing, liquid matrix during the molding process.

The proposed method proceeds in two steps. First step, it extracts the skeleton of the fibers by a thinning. Second, individual fibers are reconstructed from the skeleton.

The reconstruction process is formulated and implemented using the theory of graphs. It uses basic, local graph operations as the edge or vertex contraction. The graph models a real object. Its geometrical properties must not alter during the simplification. They are encoded as weights associated to the graph. During the contraction of the graph, the weights are iteratively inherited,

until the ultimate state, beyond which no additional simplification is possible.

After that, we perform a statistical analysis of geometric properties of the material, such as distribution of the orientation and length of the fibers.

The segmentation process has been described in [30]. The application we have developed has been successfully tested and is now routinely used in industry.

10.6.4 The Cocascope project

The Cocascope project is a collaborative, ANR funded MATETPRO project starting in the beginning of 2013, grouping laboratories (ECP, CEA/IRFM, CEA/IRFU, INSA Lyon and Armines) and one industrial partner (Nexans). I am providing the administrative and scientific responsibility of this project.

The electrical conductivity of superconducting cables depends of the local mechanical strain the cables undergo. An accurate modeling of their mechanical and electrical behavior is required to optimize their global performances.

Superconducting cables display a multiscale internal structure. At the first level, the cable is made of an assembly of elementary strands arranged together according to more or less complex architectures, depending on their application. At a following scale, strands appear as composite structures formed either by superconducting microfilaments embedded in a metallic matrix, or by a thin superconducting layer deposited onto a metallic substrate. Both reversible and irreversible conductivity losses are encountered for these superconducting components at microscopic scale, depending on the magnitude of local strains.

Given the complexity of local configurations of various types of cables, statistical tools will be developed in order to compare relevant geometrical deformations and to validate simulation results against those identified experimentally.

The project focuses to identify - at the scale of the strands - the mechanical and electrical behavior as a function of cyclic mechanical bending stress induced by axial compression at cryogenic temperatures.

The validation of the statistical tools will be done via geometrical analysis of the cables by the means of image processing. A 3D image of a section will be acquired either by tomography, either via a collection of equidistant axial cuts.

From this image, every filament will be isolated, and its trajectory identified. A statistical analysis will be performed and compared with results obtained by simulation, and those by experimentation. Particularly, a precise analysis of the local curvature will be compared to the results of simulation of stress induced by bending.

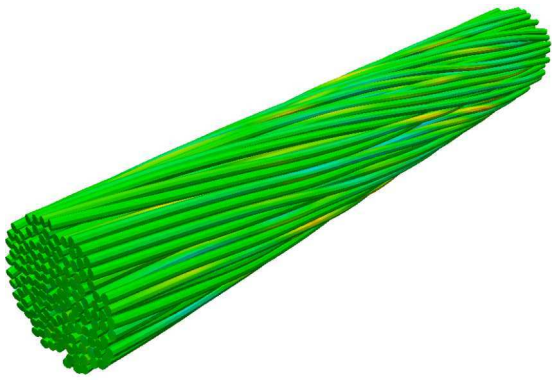


Figure 10.11: An illustration of filaments of a superconducting cable. The local curvature is expressed in false colors.

10.7 Medical&Biomedical Science

10.7.1 The L'Oréal partnership

The collaboration with L'Oréal is a successful partnership between CMM and L'Oréal/Research-&Innovation, started in 2008. I have contributed to this partnership as one of several software developers. I have also provided scientific guidance of the dermis collagen and elastin fiber network analysis task. This task is developed here below.

Recent technological advances in microscopy, and particularly the multi-photon microscopy, has allowed direct *in vivo* imaging - up to several hundreds microns of depth - of structures previously observable in histological cuts and after coloration.

We have used multi-photon microscopy to acquire images of two fibrous structures found in the dermis - collagen and elastin - that provide the skin with mechanical properties such as the rigidity and elasticity. These mechanical properties get progressively lost in aging or damaged[1] skin. The main objectives are understanding of biological processes that take place in the skin, evaluation of medical or cosmetic products but also the development of bio-materials. Within this context we were brought to study various structures in different skin layers.

The quantification of the alteration induced by the aging (or photo-aging) process is only feasible through an accurate characterization of the fibrous structures. The aging fibers can be visually recognized from young fibers by an expert through some visually easily discernable characteristics (such as the length). Other characteristics are not easily observable by eye and need to be characterized numerically.

However, the automated characterization of the fibrous structures is uneasy because of several reasons. The population of fibers is not homogeneous. It is widely accepted that the fibers become shorter in the old skin. However, in young skin, only a limited number of fibers are long. The fibrous structure is geometrically quite a complicated object, difficult to characterize. Only the superficial layer of the dermis is visible. The *in vivo* imaging with the multi-photon microscope yields images with progressively decreasing SNR with depth.

We have used a pool of young and old volunteers (under 25 and over 60) to acquire a database of multi-photon images of the skin from the dorsal and ventral side of the forehand.

We have developed a segmentation technique to identify different skin layers (several layers of the epidermis and of the dermis). We have developed a battery of descriptors and analyzed the structures found in each layer. We have trained a statistical classifier to identify the variance of every descriptor and its potential to characterize the aging-related alteration of the skin.

The proportion of long individuals in the whole population of fibers can be assessed from the distribution of lengths. This is not easy to measure since the fibers are thin, tortuous and heavily entangled. The parsimonious path openings, proposed in Sec. 7, can successfully be used to discriminate the short fibers from the long ones, and measure the distribution of the length.

The developed methods were implemented, and successfully tested in a software which is now routinely used by our partner for various studies. The developments are protected by a patent [37], and have partially been published in [2, 25], and will entirely be published after the acceptation of the patent.

10.8 Other research

10.8.1 The Freia project

FREIA is an ANR funded project making cooperate two research centers (CMM and CRI) of Armines and one industrial partner (Thales). The development of one part of this project was ensured by Jan Bartovský, during his internship in CMM – in the scope of his PhD – under my guidance.

[1] essentially due to sun exposure; so called photo-aging

Following our previous algorithmic achievements in efficient computation of serial morphological filters (developed in Secs. 9.1 and 9.3), implemented in FPGA (Sec. 10.3), we have contributed to FREIA by providing efficient dilation and opening blocks.

The principal scientific achievement stems from the principal properties of the underlying algorithms. We have already seen that both algorithms run with constant complexity $\mathcal{O}(1)$ and have minimal (further irreducible) memory consumption and latency. We have also seen that the combination of these properties is inherited under concatenation, which is particularly interesting for serial filters.

Additionally, both algorithms can be implemented in HW using the finite state machine concept. This brings an advantage of having constant (or almost constant) HW resources w.r.t. the image resolution. Practically, this allows achieving the HW scalability through only using a smaller or larger external memory, keeping the logic resources constant. Given that the image resolution is programmable in the logic, having constant resources allows using only a small FPGA for a whole range of resolutions provided a sufficiently large external memory is available.

Implemented in one HW block, preserving all these properties, we could obtain the first *morphological neighborhood processor with arbitrarily large structuring element*. This is a substantial improvement unexampled in the existing, where only small-sized-neighborhood processors have been published and used [17, 19, 26, 51]. Indeed, prior to our work, large size neighborhood operations where implemented in three ways: i) by using the invariance under scaling (homothecy), ii) by decomposition, or iii) directly in the logic. None of these approaches combines all the properties that we could obtain: irreducible latency and memory, programmable SE size, constant logic resources, and scalability exclusively through a scalable external memory.

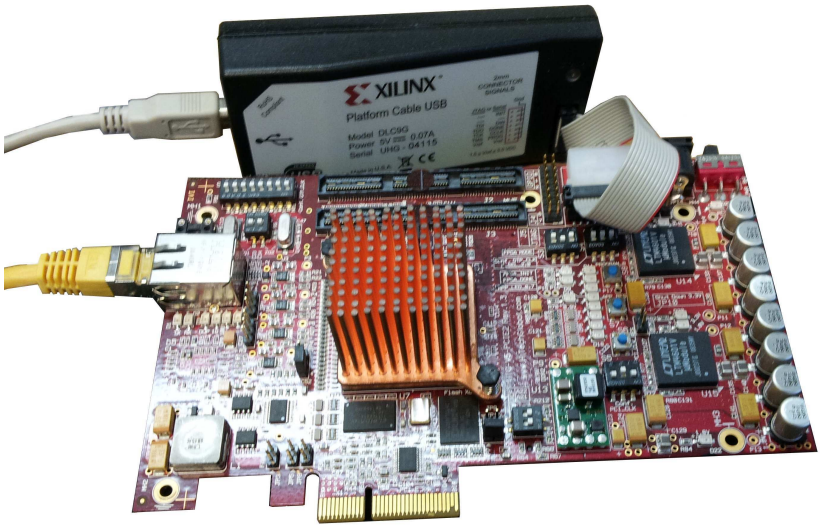


Figure 10.12: The implementation of arbitrary size structuring element processors on a Virtex 6 development board.

10.8.2 The Cosmeca project

We group in this section two subsequent projects, a RNRT project Cosmeca (Continuité de Service de Mobilité Evaluée dans un Contexte Automobile) and a European project GST (Global Systems for Telematics). I have provided the software end electronics development for the Cosmeca and GST projects, and the administrative responsibility for the GST project.

In the context of the Cosmeca project Armines/CMM was in charge of the risk management of drivers of road vehicles, and particularly the prevention of sleepiness, and the risk of hypoglycemia of diabetic drivers.

The diabetes prevents from correct metabolic regulation of the blood glucose levels. The hypoglycemia and hyperglycemia bring about dangerous short-time and long-time complications. One of possible ways how to obtain a correct regulation is the prediction of the blood glucose evolution. This evolution depends on several factors: insulin medication, carbohydrates intake, physical activity and to some extent the physical condition. Consequently, the blood glucose level evolution can only be done on a short time scale, up to the next event (meal, insulin intake, etc.) and ideally several hours. A reliable blood glucose prediction can serve to administrate accurate medication, in function to the current situation but also, reversely to control holters and insulin pumps.

The principal difficulty comes from the fact that the blood glucose level is not observable except in laboratory condition by a medical staff. One can only reliably measure the capillary or interstitial glucose, that only is roughly correlated to the glucose in blood.

Numerous and very different techniques have been considered in the past: i) purely mathematical models such as neural networks or ii) metabolism-based models, such as a compartimental model taking into account other organs (stomach, liver, gut, ... etc., in addition to the pancreas). The neural networks are often used in situations where no mathematical description of the modeled phenomena is available. In turn, they need to be trained on the whole domain of possible input data. However, the acquisition of the training data is a medical and ethical problem. One cannot deliberately provoke a dangerous state in other individuals. Nonetheless, it is precisely in the dangerous levels where the neural networks have to be accurately trained. The compartimental models are reasonably reliable but need more data than those that are routinely observable.

We could identify, and validate in rats, a model able to simulate the level of glucose in blood according to the change of the flow of interstitial glucose under conditions close to the conditions under which the model was identified. We note that the flow of insulin is not a parameter acting in a linear way. Two models obtained under two different flows of insulin will not be a linear combination one of the other. In other words, insulin acts as well on the reaction speed of the organization as on the values of stabilization. Partial results have been published in Chollet *et al.* [18].

Part V

Perspectives

Chapter 11

Perspectives

Après s'être arrêté sur le passé, il arrive le moment de regarder vers le futur. Après ma thèse de doctorat en traitement d'image, et mon post-doc dans ce même domaine, j'ai passé douze ans, c'est à dire, l'essentiel de ma carrière dans le Centre de Morphologie Mathématique, centre de recherche commun de Mines-ParisTech et Armines. L'activité de ce centre de recherche et, par conséquent, l'environnement intime de mes activités est celui de la recherche appliquée. Elle va de pair avec la formation dispensée à Mines-ParisTech, étroitement liée et orientée vers l'industrie.

Au fil des années, je suis arrivé à conviction que ce modèle de recherche constitue une vraie symbiose, de la recherche et le monde économique industriel, mutuellement bénéfique aux deux parties engagées. Les nouveaux besoins et problèmes identifiés par l'industrie constituent une source inépuisable d'inspiration et de nouvelles thématiques pour la recherche. Notons, qu'elle est également génératrice d'emplois - essentiellement pour les jeunes chercheurs de stagiaires masteriens aux post-docs. Inversement, la recherche apporte au monde industriel des nouvelles solutions, menant tant qu'il se peut à la création de nouvelles technologies et ouvertures de nouveaux marchés. J'évolue dans ce contexte, et je ne souhaite guère le changer.

Regardant mes axes de recherche future, ils s'inscrivent bien évidemment dans la continuité avec le passé. Dans les mois à venir je vais approfondir l'axe de la détection et de l'analyse des éléments fins. D'une part, une extension des ouvertures par chemins parcimonieux vers l'ouverture par chemins ultime semble logique et facilement réalisable, ainsi que son application à la mesure des distributions des longueurs. Nous avons également évoqué l'erreur dont souffre la mesure de la longueur. Plusieurs phénomènes sont à l'origine de cette erreur et y contribuent de différentes manières. Différentes idées ont surgi afin de réduire l'erreur de cette mesure.

Toujours dans le même axe concernant les éléments fins et tortueux, il serait intéressant d'explorer l'utilisation d'autres types de connexité afin de réduire la sensibilité au bruit et améliorer les performance de la détection.

De manière plus générale, et à plus long terme, d'autres descripteurs devront être développés. Plus particulièrement, il faudra répondre au besoin de développer des descripteurs des formes capables de décrire des objets individuels dans des populations d'objets inséparables les uns des autres. Des descripteurs applicables dans des situations où l'on n'arrive pas à extraire ou individualiser les objets par segmentation d'image. Au jour d'aujourd'hui dans ces situations on recourt aux descripteurs qui sont principalement basés sur des mesures statistiques. En effet, elles servent très bien à décrire des textures, des milieux aléatoires ou des populations d'objets. Cependant, elles souffrent de deux principaux inconvénients inhérents à leur nature statistique: i) d'une part, elles mesures les propriétés de la population, et sons donc insensibles aux variations isolées des objets individuels et, ii) d'autres part, elles se situent à un niveau sémantique relativement bas. Un niveau sémantique plus élevé, plus proche du raisonnement humain, mais utilisant toutefois des descripteurs géométriques simples, n'ayant pas besoin d'avoir préalablement individualisé les objets sera applicable dans des contextes variés, à l'instar de la classification, contrôle ou diagnos-

tique.

J'avais constaté au début de ce manuscrit une activité d'enseignement relativement limitée. Notons que l'enseignement est pour moi une source de plaisir et de motivation. La motivation elle vient surtout à travers l'échange avec les étudiants, et de leur satisfaction avec le cours. Je me suis vite aperçu que la symbiose de la recherche et du monde industriel et aussi bénéfique pour l'enseignement. Elle permet d'insérer le cours dans un contexte applicatif et concret. Elle apporte une motivation de par son application immédiate, très appréciée par les élèves. Pour ces raisons, je souhaite dans le future élargir mon activité d'enseignement, sans pour autant devenir enseignant à part entière. De manière générale, je souhaite maintenir un équilibre entre mes activités contractuelles, de la recherche et de l'enseignement à raison de 40%, 50% et 10% environ.

Bibliography

[1] B. Appleton and H. Talbot. Efficient path openings and closings. *Mathematical Morphology: 40 Years On*, pages 33–42, 2005.

[2] T. Baldeweck, E. Tancrède, P. Dokládal, S. Koudoro, V. Morard, F. Meyer, E. Decencière, and A. M. Pena. In vivo multiphoton microscopy associated to 3d image processing for human skin characterization. In *Photonics West*, 2012.

[3] J. Bartovský, P. Dokládal, and E. Dokládalová. Fast streaming algorithm for 1-d morphological opening and closing on 2-d support. In *ISMM*, 2011.

[4] J. Bartovský, P. Dokládal, E. Dokládalová, M. Bilodeau, and M. Akil. Real-time implementation of morphological filters with polygonal structuring elements. *Journal of Real-Time Image Processing*, September 2012. DOI: 10.1007/s11554-012-0271-8.

[5] J. Bartovský, P. Dokládal, E. Dokládalová, and V. Georgiev. Parallel implementation of sequential morphological filters. *Journal of Real-Time Image Processing*, pages 1–13, 2011. DOI: 10.1007/s11554-011-0226-5.

[6] J. Bartovsky, D. Schneider, E. Dokládalová, P. Dokládal, V. Georgiev, and M. Akil. Morphological classification of particles recorded by the timepix detector. In *International Symposium on Image and Signal Processing and Analysis, ISPA*, 2011.

[7] S. Beucher. Numerical residues. *Image and Vision Computing*, 25(4):405–415, 2007.

[8] S. Beucher, J.M. Blosseville, and F. Lenoir. Traffic spatial measurements using video image processing. In *SPIE's Advances in intelligent robotics systems*. Cambridge symposium on optical and optoelectronic engineering, Cambridge, Mass., USA, Nov 1987.

[9] S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Int. Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation*, 1979.

[10] A. Bieniek, H. Burkhardt, H. Marschner, G. Schreiber, and M. Nolle. A parallel watershed algorithm. In *In Proc. 10th Scandinavian Conference on Image Analysis (SCIA'97)*, pages 237–244, 1997.

[11] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34(3):344–371, 1986.

[12] E.J. Breen and R. Jones. Attribute openings, thinnings, and granulometries. *Computer Vision and Image Understanding*, 64:377–389, November 1996.

[13] J. E. Cates, A. E. Lefohn, and R. T. Whitaker. GIST: an interactive, GPU-based level set segmentation tool for 3D medical images. *Medical Image Analysis*, 8(3):217–231, 2004.

[14] Chen, Yung-Sheng, and Hsu Wen-Hsing. An interpretive model of line continuation in human visual perception. *Pattern Recogn.*, 22(5):619–639, 1989.

[15] C.-S. Chen, J.-L. Wu, and Y.-P. Hung. Theoretical aspects of vertically invariant gray-level morphological operators and their application on adaptive signal and image filtering. *IEEE Transactions on Signal Processing*, 47(4):1049–1060, 1999.

[16] J. Chen, Y. Sato, and S. Tamura. Orientation space filtering for multiple orientation line segmentation. *IEEE Trans PAMI*, 22, 2000.

[17] S.-Y. Chien, S.-Y. Ma, and L.-G. Chen. Partial-result-reuse architecture and its design technique for morphological operations with flat structuring elements. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(9):1156 – 1169, sept. 2005.

[18] C. Choleau, P. Dokládal, J-C. Klein, W. K. Ward, G. S. Wilson, and G. Reach. Prevention of hypoglycaemia using risk assessment with a continuous glucose monitoring system. *Diabetes*, 51:3263–3273, November 2002.

[19] Ch. Clienti, S. Beucher, and M. Bilodeau. A system on chip dedicated to pipeline neighborhood processing for mathematical morphology. In EURASIP, editor, *EUSIPCO 2008*, Lausanne, August 2008.

[20] F. Cokelaer, H. Talbot, and J. Chanussot. Efficient Robust d-Dimensional Path Operators. *IEEE Journal of Selected Topics in Signal Processing*, 6(7):830–839, November 2012.

[21] O. Cuisenaire. Locally adaptable mathematical morphology using distance transformations. *Pattern recognition, the Journal of the pattern recognition society*, 39(3):405–416, 2006.

[22] P. E. Danielsson. Euclidean Distance Mapping. *Computer Graphics Image Processing*, 14:227–248, 1980.

[23] J. Debayle and J. C. Pinoli. General Adaptive Neighborhood Image Processing - Part I: Introduction and Theoretical Aspects. *Journal of Mathematical Imaging and Vision*, 25(2):245–266, 2006.

[24] J. Debayle and J. C. Pinoli. General Adaptive Neighborhood Image Processing - Part II: Practical Application Examples. *Journal of Mathematical Imaging and Vision*, 25(2):267–284, 2006.

[25] E. Decencière, E. Tancrède-Bohin, P. Dokládal, S. Koudoro, A.-M. Pena, and T. Baldeweck. Automatic 3d segmentation of multiphoton images: a key step for the quantification of human skin. *Skin Research and Technology (to appear)*.

[26] O. Déforges, N. Normand, and M. Babel. Fast recursive grayscale morphology operators: from the algorithm to the pipeline architecture. *Journal of Real-Time Image Processing*, pages 1–10, 2010. DOI: 10.1007/s11554-010-0171-8.

[27] C.-H. Demarty. *Segmentation et Structuration d'un Document Vidéo pour la Caractérisation et l'Indexation de son Contenu Sémantique*. PhD thesis, Ecole des Mines de Paris, France, 2000.

[28] P. Dokládal and E. Dokládalova. Grey-scale Morphology with Spatially-Variant Rectangles in Linear Time. In *Advanced Concepts for Intelligent Vision Systems*, 2008.

[29] P. Dokládal and E. Dokládalová. Computationally efficient, one-pass algorithm for morphological filters. *Journal of Visual Communication and Image Representation (in press)*, 2011.

[30] P. Dokládal and D. Jeulin. 3-d extraction of fibres from microtomographic images of fibre-reinforced composite materials. In Michael Wilkinson and Jos Roerdink, editors, *Mathematical Morphology and Its Application to Signal and Image Processing*, volume 5720 of *LNCS*, pages 126–136. Springer, 2009.

[31] P. Dokládal and E. Dokládalová. Grey-scale 1-D dilations with spatially-variant structuring elements in linear time. In *Eusipco*, August 2008.

[32] E. R. Dougherty and R. A. Lotufo. *Hands-on Morphological Image Processing*. Spie Press, Bellingham, WA, USA, 2003.

[33] R. Fabbri, L. Da F. Costa, J. C. Torelli, and O. M. Bruno. 2D Euclidean Distance Transform Algorithms: A Comparative Survey. *ACM Computing Surveys*, 40(1), Feb. 2008.

[34] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Scale and the differential structure of images. *Image Vision Comput.*, 10(6):376–388, July/Aug. 1992.

[35] A.F. Frangi, W.J. Niessen, R.M. Hoogeveen, T. van Walsum, and M.A. Viergever. Model-based quantitation of 3-d magnetic resonance angiographic images. *Medical Imaging, IEEE Transactions on*, 18(10):946 –956, Oct. 1999.

[36] A.F. Frangi, W.J. Niessen, L.V. Koen, and M.A. Viergever. Multiscale vessel enhancement filtering. *Lecture Notes in Computer Science*, 1496:130–137, 1998.

[37] French patent N^o OA11508. *Procédé pour caractériser l'épiderme et le derme à partir d'images multiphoton tridimensionnelles in vivo de la peau.*

[38] T. Gijbels, P. Six, L. Van Gool, F. Catthoor, H. De Man, and A. Oosterlinck. A VLSI-architecture for parallel non-linear diffusion with applications in vision. In *IEEE Workshop on VLSI Signal Processing VII*, page 398–407, La Jolla, Calif, USA, October 1994.

[39] J. Gil and M. Werman. Computing 2-d min, median, and max filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):504–507, 1993.

[40] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. Fast geodesic active contours. *IEEE Trans. Image Processing*, 10(10):1467–1475, 2001.

[41] H. Hedberg, P. Dokládal, and V. Öwall. Binary morphology with spatially variant structuring elements: Algorithm and architecture. *IEEE Transactions on Image Processing*, 18(3):562–572, March 2009.

[42] H. Heijmans, M. Buckley, and H. Talbot. Path openings and closings. *CWI. Probability, Networks and Algorithms [PNA]*, (E 0403):1–21, 2004.

[43] H. Heijmans, M. Buckley, and H. Talbot. Path openings and closings. *Journal of Mathematical Imaging and Vision*, 22(2):107–119, 2005.

[44] C.L.L. Hendriks. Constrained and dimensionality-independent path openings. *IEEE Transactions on Image Processing*, 19(6):1587–1595, 2010.

[45] J. Hernández and B. Marcotegui. Ultimate attribute opening segmentation with shape information. *Mathematical Morphology and Its Application to Signal and Image Processing*, pages 205–214, 2009.

[46] S. Holmgren and D. Wallin. Performance of high-accuracy pde solvers on a self-optimizing numa architecture. In *Lecture Notes in Computer Science*, volume 2150. Springer, Berlin, Germany, 2001.

[47] K. Hwang, P. S. Tseng, and D. Kim. An orthogonal multiprocessor for parallel scientific computations. *IEEE Trans. Comput.*, 38(1):47–61, 1989.

[48] M. Iwanowski and M. Swiercz. Fast, parallel watershed algorithm based on path tracing. In *ICCVG 2010, Part II, Lecture Notes in Computer Science*, volume 6375, pages 317–324, 2010. DOI: 10.1007/978-3-642-15907-7_39.

[49] P. Karas, V. Morard, J. Bartovský, T. Grandpierre, E. Dokládalová, P. Matula, and P. Dokládal. GPU implementation of linear morphological openings with arbitrary angle. *Journal of Real-Time Image Processing*, 2012. DOI: 10.1007/s11554-012-0248-7.

[50] A. Katartzis, H. Sahli, V. Pizurica, and J. Cornelis. A model-based approach to the automatic extraction of linear features from airborne images. *Geoscience and Remote Sensing, IEEE Transactions on*, 39(9):2073–2079, 2001.

[51] J. C. Klein and R. Peyrard. Pimm[1], an image processing ASIC based on mathematical morphology. In *Second annual IEEE ASIC Seminar and Exhibit*, pages P7 1.1–1.4, Rochester, NY, USA, September 25-28 1989.

[52] J. J. Koenderink. The structure of images. *Biol. Cybern.*, 50:363–370, 1984.

[53] T.M. Koller, G. Gerig, G. Szekely, and D. Dettwiler. Multiscale detection of curvilinear structures in 2-d and 3-d image data. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 864–869. IEEE, 1995.

[54] R. Kresch and D. Malah. Morphological reduction of skeleton redundancy. *Signal Processing*, 38:143–151, 1994.

[55] Jeulin D. Kurdy B. Directional mathematical morphology operations. In *In 5th European Congress For Stereology, Acta Stereologica*, volume 8/2, 1989.

[56] C. Lantuejoul and S. Beucher. On the use of the geodesic metric in image analysis. *Journal of Microscopy*, 121(1):39–49, 1981.

[57] C. Lantuéjoul and F. Maisonneuve. Geodesic methods in quantitative image analysis. *Pattern Recognition*, 17(2):177–187, 1984.

[58] D. Lemire. Streaming maximum-minimum filter using no more than three comparisons per element. *Nordic J. of Computing*, 13(4):328–339, 2006.

[59] F. Lemonnier and J.-C. Klein. Fast dilation by large 1D structuring elements. In *IEEE International Workshop on Nonlinear Signal and Image Processing, Halkidiki, Greece*, 1995.

[60] R. Lerallut, E. Decencière, and F. Meyer. Image filtering using morphological amoebas. *Image Vision Comput*, 25(4):395–404, 2007.

[61] P. Maragos. Pattern spectrum and multiscale shape representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):701–716, 1989.

[62] P. Maragos and R. Schafer. Morphological skeleton representation and coding of binary images. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34:1228–1244, 1986.

[63] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[64] G. Matheron. *Eléments pour une Théorie des Milieux Poreux*. Masson, 1967.

[65] A. Meijster and J. B. T. M. Roerdink. Computation of watersheds based on parallel graph algorithms. In R. W. Shafer P. Maragos and M. A. Butt, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 305–312. Kluwer, 1996.

[66] A. Meijster and J.B.T.M. Roerdink. A proposal for the implementation of a parallel watershed algorithm. In *CAIP Compute Analysis and Patterns*, Prague, September 1995.

[67] A. Moga and M. Gabbouj. A parallel watershed algorithm based on the shortest paths computation. In P. Fritzson and L. Finmo, editors, *Parallel Programming and Applications*. IOS Press, 1995.

[68] A. N. Moga, B. Cramariuc, and M. Gabbouj. Parallel watershed transformation algorithms for image segmentation. *Parallel computing*, 24:1981–2001, 1998.

[69] A.N. Moga, T. Viero, M. Gabbouj, M. Nolle, G. Schreiber, and H. Burkhardt. Parallel watershed algorithm based on sequential scanning. In *IEEE Workshop on Nonlinear Signal and Image Processing*, Halkidiki, Greece, June 1995.

[70] V. Morard. *Détection de structures fines par traitement d'images et apprentissage statistique : application au contrôle non destructif.* PhD thesis, Mines-ParisTech, 2012.

[71] V. Morard, E. Decencière, and P. Dokládal. Geodesic attributes thinnings and thickenings. In *Mathematical Morphology and Its Applications to Image and Signal Processing*, pages 200–211. Springer, 2011.

[72] V. Morard, E. Decencière, and P. Dokládal. Region growing structuring elements and new filters based on their shape. In *Proceedings of Signal and Image Processing (SIP) Dallas USA*, pages 0–0. IASTED, 2011.

[73] V. Morard and P. Dokládal E. Decencière. Parsimonious path openings. *to be submitted*, 2012.

[74] V. Morard, P. Dokládal, and E. Decencière. One-dimensional openings, granulometries and component trees in O(1) per pixel. *submitted to IEEE Journal of Selected Topics in Signal Processing*, 2012.

[75] Vincent Morard, Etienne Decencière, and Petr Dokládal. Efficient geodesic attribute thinnings based on the barycentric diameter. *submitted to JMIV*, 2012.

[76] M. N. T. Natsuyama. Edge preserving smoothing. *Computer Graphics and Image Processing*, 9:394–407, 1979.

[77] D. Noguet. A massively parallel implementation of the watershed based on cellular automata. In *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pages 42–52, July 1997.

[78] N. Ortega-Quijano, B. H. Haj Ibrahim, S. Bancelin, M.-C. Schanne-Klein, A. Nazac, P. Dokládal, E. Decencière, J. L. Arce-Diego, and A. De Martino. Orientational characterization of fibrillar collagen in histopathological samples by shg microscopy and mueller polarimetry. In *Photonics West*, 2012.

[79] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[80] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[81] F. Precioso and M. Barlaud. B-spline active contour with handling of topology changes for fast video segmentation. *Journal on Applied Signal Processing*, 2002(6):555–560, 2002. Special Issue on Image Analysis for Multimedia Interactive.

[82] T. Retornaz. *Détection de textes enfouis dans des bases dimages généralistes. Un descripteur sémantique pour l'indexation.* PhD thesis, Ecole des Mines de Paris, France, Oct. 2007.

[83] J. B. T. M. Roerdink and H. J. A. M. Heijmans. Mathematical morphology for structures without translation symmetry. *Signal Processing*, 15(3):271–277, 1988.

[84] M. Rumpf and R. Strzodka. Level set segmentation in graphics hardware. In *International Conference on Image Pro- cessing (ICIP '01)*, page 1103–1106, Thessaloniki, Greece, October 2001.

[85] M. Rumpf and R. Strzodka. Nonlinear diffusion in graphics hardware. In *EG/IEEE TCVG Symposium on Visualization (VisSym '01)*, page 75–84, Ascona, Switzerlan, May 2001.

[86] P. Salembier, A. Oliveras, and L. Garrido. Antiextensive connected operators for image and sequence processing. *Image Processing, IEEE Transactions on*, 7:555–570, 1998.

[87] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, and R. Kikinis. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical Image Analysis*, 2(2):143–168, June 1998.

[88] M. Schmitt. *Des algorithmes morphologiques à l'intelligence artificielle*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 1989.

[89] D. Schonfeld and J. Goutsias. Morphological representation of discrete and binary images. *IEEE Transactions on Signal Processing*, 39:1369–1379, 1991.

[90] J. Serra. *Image analysis and mathematical morphology*, volume 1. Academic Press, London, 1982.

[91] J. Serra. *Image Analysis and Mathematical Morpohology*. Academic Press, New York, NY, USA, 1982.

[92] J. A. Sethian. Parallel level set methods for propagating inter- faces on the connection machine. Technical report, Department of Mathematics, University of California at Berkeley, Berkeley, Calif, USA, 1989.

[93] C. Sigg, R. Peikert, and M. Gross. Signed distance transform using graphics hardware. In *14th IEEE Visualization Conference (VIS '03)*, page 83–90, Seattle, Wash, USA, October 2003.

[94] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *Journal of Scientific Computing*, 19(1-3):439–456, 2003.

[95] P. Soille, E. Breen, and R. Jones. Recursive implementation of erosions and dilations along discrete lines at arbitrary angles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(5):562–567, 1996.

[96] P. Soille and H. Talbot. Image structure orientation using mathematical morphology. In *Conference on Pattern Recognition*, volume 2, pages 1467–76, June 29-July 2 1998.

[97] P. Soille and H. Talbot. Directional morphological filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1313–1329, 2001.

[98] P. Soille and H. Talbot. Directional morphological filtering. *IEEE PAMI*, 23:1313–29, 2001.

[99] A. M. Stein, D. A. Vader, L. M. Jawerth, D. A. Weitz, and L. M. Sander. An algorithm for extracting the network geometry of three-dimensional collagen gels. *Journal of Microscopy*, 232(3):463–75, 2008.

[100] H. Talbot and B. Appleton. Efficient complete and incomplete path openings and closings. *Image and Vision Computing*, 25(4):416–425, April 2007.

[101] O. Tankyevych. *Filtering of Thin Objects. Applications to Vascular Image Analysis*. PhD thesis, University Paris-Est, France, 2010.

[102] O. Tankyevych, A. Dufour, B. Naegel, H. Talbot, Ch. Ronse, J. Baruthio, P. Dokládal, and N. Passat. Filtering and Segmentation of 3D Angiographic Data; Advances Based on Mathematical Morphology. *MEDIA*, (**to appear**).

[103] O. Tankyevych, H. Talbot, and P. Dokladal. Curvilinear morpho-hessian filter. In *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pages 1011–1014. IEEE, 2008.

[104] E. Urbach and M. Wilkinson. Efficient 2-D Grayscale Morphological Transformations With Arbitrary Flat Structuring Elements. *IEEE Trans. Image Processing*, 17(1):1–8, 2008.

[105] E R. Urbach, J B T M. Roerdink, and M H F. Wilkinson. Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):272–285, 2007.

[106] E.R. Urbach and M.H.F. Wilkinson. Shape-only granulometries and gray-scale shape filters. In *Mathematical Morphology and Its Applications to Image and Signal Processing - proceedings of ISMM'2002*, page 305, 2002.

[107] S. Valero, J. Chanussot, JA Benediktsson, H. Talbot, and B. Waske. Advanced directional mathematical morphology for the detection of the road network in very high resolution remote sensing images. *Pattern Recognition Letters*, 31(10):1120–1127, 2010.

[108] M. van Ginkel, L.J. van Vliet, and P.W. Verbeek. Applications of image analysis in orientation space. In F.M. Vos A.M. Vossepoel, editor, *Fourth Quinquennial Review 1996-2001 Dutch Society for Pattern Recognition and Image Processing, NVPHBV, Delft*, pages 355–370, 2001.

[109] M. van Herk. A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. *Pattern Recog. Letters*, 13(7):517–521, 1992.

[110] J. Van Neerbos, L. Najman, and M. Wilkinson. Towards a Parallel Topological Watershed: First Results. In *10th International Symposium on Mathematical Morphology (ISMM'11)*, France, 2011.

[111] L.J. van Vliet and P.W. Verbeek. Segmentation of overlapping objects. In L.J. van Vliet and I.T. Young, editors, *Abstracts of the ASCI Imaging Workshop*, October 1995.

[112] J. G. Verly and R. L. Delanoy. Adaptive mathematical morphology for range imagery. *IEEE Trans. on Image Processing*, 2(2):272–275, 1993.

[113] L. Vincent. Morphological area openings and closings for grey-scale images. *Shape in Picture: Mathematical Description of Shape in Grey-level Images*, pages 196–208, 1993.

[114] T. Walter and J.C. Klein. Segmentation of color fundus images of the human retina: Detection of the optic disc and the vascular tree using morphological techniques. *Medical Data Analysis*, pages 282–287, 2001.

[115] J. Weickert, B. M. T. H. Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Trans. Image Processing*, 7(3):398–410, 1998.

[116] M.H.F. Wilkinson and M.A. Westenberg. Shape preserving filament enhancement filtering. *Medical Image Computing and Computer-assisted Intervention, Volume 2208 of lecture notes in computer science*, 2208:770–777, 2001.

[117] Y. Wu and H. Maître. Smoothing speckled synthetic aperture radar images by using maximum homogeneous region filters. *Optical Engineering*, 31(8):1785–1792, 1992.

[118] F. Zana and J.C. Klein. Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation. *Image Processing, IEEE Transactions on*, 10(7):1010–1019, 2001.

[119] E. Dejnožková and P. Dokládal. A parallel architecture for curve-evolution PDEs. *International Journal of Image Analysis and Stereology*, 22:121–132, 2003.

[120] E. Dejnožková and P. Dokládal. Embedded real-time architecture for level-set-based active contours. *Journal on Applied Signal Processing*, 17:1–16, 2005.

Part VI

Annexe : Selected Papers

The Annexe gathers a collection of selected papers to provide supplementary details to the summary research activity report above. We group the papers into like categories.

- Detection of Thin Objects
    1. Parsimonious path openings, at page 73
    2. Efficient geodesic attribute thinnings based on the barycentric diameter, at page 85
    3. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology, at page 100
- Efficient Algorithms
    1. Computationally Efficient, One-Pass Algorithm for Morphological Filters, at page 118
    2. One-dimensional openings, granulometries and component trees in O(1) per pixel, at page 129
- Implementations
    1. Binary Morphology with Spatially Variant Structuring Elements: Algorithm and Architecture, at page 139
    2. Parallel Implementation of Sequential Morphological Filters, at page 150
    3. Real-Time Implementation of Morphological Filters with Polygonal Structuring Elements, at page 165
    4. GPU Implementation of Linear Morphological Openings with Arbitrary Angle, at page 176
    5. A parallel architecture for curve-evolution PDEs, at page 190
    6. Embedded real-time architecture for level-set-based active contours, at page 202

Parsimonious path openings

Vincent Morard, Petr Dokládal and Etienne Decencière

***Abstract*—Path openings and closings are morphological tools used to preserve long, thin and tortuous structures in Gray level images. They explore all paths from a defined class, and filter them with a length criterion. However, the great majority of the paths bring redundant information, making the process generally slow.**

***Parsimonious path openings* are introduced in this paper to solve this problem. These operators only consider a subset of the paths considered by classical path openings, thus achieving a substantial speed-up, while obtaining similar results. Moreover, a recently introduced one dimensional (1-D) opening algorithm is applied along each selected path. Its complexity is linear with respect to the number of pixels, independent of the size of the opening. Furthermore, it is fast for any input data accuracy (integer or floating point) and works *in stream*.**

Parsimonious path openings are also extended to *incomplete paths*, *i.e.* paths containing *gaps*. Noise-corrupted paths can thus be processed with the same approach and complexity.

These parsimonious operators achieve a several orders of magnitude speed-up. Examples are shown for incomplete path openings, where computing times are brought from minutes to tens of milliseconds, while obtaining similar results.

***Index Terms*—Parsimonious path openings, path openings, mathematical morphology, complete and incomplete paths.**

I. Introduction

Thin structures extraction is a non-trivial task in image processing. It requires adapted tools, used in a great range of applications, from the biomedical field to the industrial domain. Blood vessels extraction from eye fundus images [1], [2], road detection from remote sensing images [3], [4] or automated cracks detection from metallic pieces for non-destructive testing [5], [6] are some examples.

In the literature, the typical approach to enhance thin structures is to compute the supremum of openings with linear structuring elements (SE) in many orientations [7], [8]. The same strategy can be used with a bank of directional Gabor filters or difference of Gaussians filters [9], [10]. However, tortuous structures are difficult to detect with this kind of approach. Using adaptive mathematical morphology methods improves the detection. Tankyevych et al. [11] introduced hessian based filters to detect curvilinear lines. In [6], the SE are able to adapt their shapes to enhance very thin cracks of any tortuosity. Area openings, introduced by Vincent [12], are considered as the first attribute openings, later generalized by Breen and Jones to obtain attribute thinnings [13]. Indeed, using non-increasing criteria to build attribute thinnings yields interesting filters to detect thin structures. For instance, the inertia of the connected components, weighted by their area, gives an interesting shape descriptor for elongated structures [14], [15]. More recently, thinnings based on geodesic attributes have been shown to efficiently characterize structures according to their length, tortuosity or elongation [5], [16]. Finally, the so-called *path openings* (PO) [17], [18] use underlying oriented acyclic graphs to measure the path length.

All these methods have the same drawback: their lack of robustness with respect to noise. Indeed, thin elongated structures we are looking for can be easily corrupted by noise, resulting in disconnected paths. Talbot and Appleton [19] have proposed *incomplete path openings*, able to deal with gaps in paths. It has logarithmic complexity w.r.t. the length of the paths and linear w.r.t. the width of the gaps, resulting in long computation timings, unsuitable for time-critical applications.

The approach developed in this paper is motivated by the need to detect thin, long, tortuous and possibly noisy structures in a computationally demanding framework. The methods from the state of the art that fulfill the best these requirements are indeed path openings with complete or incomplete paths [19]. Here, we propose *Parsimonious path openings* (PPO) a fast method to detect the same structures, using complete or incomplete paths. PPO only explore a relevant subset of all paths in the image, to reduce the computation time by several orders of magnitude. PPO are fast, accurate and robust to noise.

This paper is organized as follows. We first recall the theory of classical path openings (Sec. II). Then, we describe the extraction of the relevant subset of paths in the image (Sec. III), the filtering strategies available (Sec. IV), the practical considerations (Sec. V) and the operator accuracy (Sec. VI). Finally, we present some results through an application: the detection of cracks from road pavement images. We also study the algorithmic complexity and we propose a timing comparison with classical path openings (Sec. VIII).

II. Basic notions on path openings

Path openings [17], [18] were introduced to offer a higher flexibility compared to the supremum of linear openings. We briefly recall here their definition and characteristics.

A. Connectivity graph and maximal paths

A two-dimensional binary image X can be described as a subset of a rectangular sub-domain D of Z^2. We equip D with a directed acyclic graph $G : D \to \mathscr{P}(D)$, where $\mathscr{P}(D)$ is the power set of D. For any two points x and y of D, we say that x is linked to y on G if, and only if, $y \in G(x)$. G^- is the inverse of G, defined by $G^- : D \to \mathscr{P}(D)$ and for all x in D, $y \in G^-(x)$ (i.e. x is linked to y on G^-) if, and only if, $x \in G(y)$ (i.e. y is linked to x on G). G_X is the subgraph of G obtained when the graph is restricted to X.

Fig. 1 illustrates the classical graphs used in practice for G.

Let us introduce now the definition of a path on G_X:

Definition 1

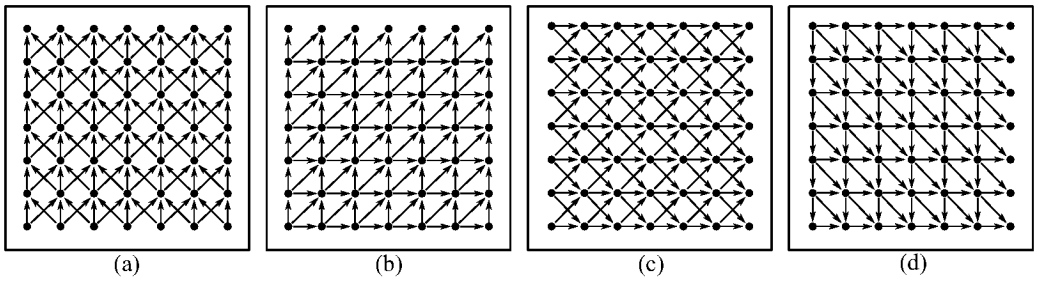


Fig. 1. Example of commonly used graphs: (a) paths having an orientation from south to north **(S-N)**. (b) **SW-NE** paths, (c) **W-E** paths and (d) **NW-SE** paths

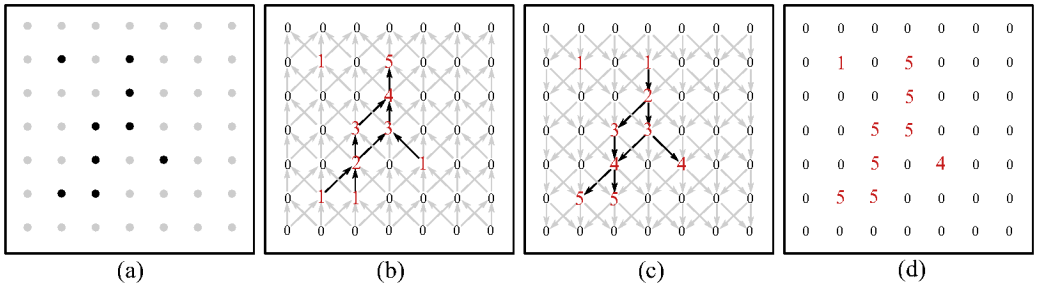


Fig. 2. Path openings computation: (a) input binary image, (b) upstream distance map λ^+, (c) downstream distance map λ^-, and (d) maximal path length map λ.

> *A sequence $\pi = (x_1, x_2, \ldots, x_n)$, $n \in \mathbb{N}$, of points is a path of G_X if, and only if, $\forall i \in \mathbb{N}, 1 \leq i \leq n-1, x_{i+1} \in G_X(x_i)$ (x_i is linked to x_{i+1} on G_X). The path length is equal to n. Points x_1 and x_n are its starting and end points. The set of all paths of G_X is denoted Π_{G_X}.*

On each point x of D we define $\lambda_{G_X}(x)$ (or $\lambda(x)$, when there is no ambiguity) as the maximal length of all the paths of Π_{G_X} going through x. If x does not belong to X, then $\lambda(x)$ is equal to zero. Given that the considered graphs are finite, and without loops, the values of λ are finite.

B. Binary path opening

Map λ can be efficiently computed using a scan of G_X, followed by a scan of G_X^-: let λ^+ (reps. λ^-) be the map which gives, for each point x of D, the maximal length of the paths of Π_{G_X} (resp. $\Pi_{G_X^-}$) ending at x. λ^+ and λ^- are efficiently computed thanks to the following equations:

$$\lambda^+(x) = \max_{y \in G_X(x)} \lambda^+(y) + 1, \quad (1)$$

$$\lambda^-(x) = \max_{y \in G_X^-(x)} \lambda^-(y) + 1. \quad (2)$$

An example of λ^+ and λ^- is shown in Fig. 2 (b and c). Finally, λ is simply computed as follows, for all $x \in D$:

$$\lambda(x) = \lambda^+(x) + \lambda^-(x) - 1. \quad (3)$$

For a given pixel x, $\lambda(x)$ gives the length of the longest paths going through it. If we remove from X all the points where λ is smaller than a given constant L, we obtain an operator which is idempotent, increasing and anti-extensive (see Heijmans *et al.* [18] for the details), therefore, it is an opening, called a path opening (PO) of size L, and written $\Gamma_L^{PO}(X)$:

$$\Gamma_L^{PO}(X) = \{x \in X | \lambda_{G_X}(x) \geq L\}. \quad (4)$$

Such an opening, when based on any of the graphs illustrated in Fig. 1, is far from being rotation invariant, an often welcome property. In order to improve on this aspect, one can use the fact that the supremum of openings is still an opening [20]: four openings are in practice computed, based on the four graphs depicted in Fig. 1, and their supremum is computed.

In the following Γ_L^{PO} will denote the path opening of size L based on this supremum.

C. Grey level path openings

Let $f : D \to V$ be a gray level image, where V is a finite subset of $\mathbb{Z}$, typically $\{0, 1, \ldots, 255\}$.

Let $X^h = \{x \mid f(x) \geq h\}$ be the upper level set obtained by thresholding f at level h. Given that the binary opening Γ_L^{PO} is increasing, it commutes with thresholding. Thence, the extension to gray level images is direct:

$$\forall x \in D,\ \gamma_L^{PO}(f)(x) = \vee\{h \in V \mid x \in \Gamma_L^{PO}(X^h(f))\}. \quad (5)$$

Using equation 5 to compute gray level path openings is easy, but it is never used in practice due to the high computation time. In [21], an efficient update of λ^+ and λ^- is proposed, which achieves a large speed up factor. This update is improved in [22] to be able to easily work with n-D images. Another solution is proposed in [23]. These three papers bring notable improvements on path opening timings, but the algorithms are still too slow for many applications. A solution is proposed to this speed issue in the next sections.

In what follows, we present and illustrate this work with the extraction of bright structures in an image, with no loss of generality. Path closings (resp. parsimonious path closings) are computed using path openings (resp. parsimonious path openings) on the inverted image.

III. Parsimonious set of paths

The principal idea behind the notion of *parsimonious paths* is to work with a restricted set of paths instead of exploring all of them. In fact, the set of paths will be so sparse, that most points in the image will not be crossed by any of them.

In the original definition of path openings, the number of paths grows exponentially with the size of the image. However, only few out of these paths bring relevant information. In the following, we deal with the problem of building a relevant subset of paths.

For the extraction of bright structures, the relevant paths have to follow the brightest structures of the image. Since this will usually leave other pixels devoided of a path, we will speak about *parsimonious path openings*.

Two strategies of selection of relevant paths and a generalization are proposed in this section. Hereafter, D, the definition domain or support of f, will be a rectangular subset of $\mathbb{Z}^2$, and G will be a directed acyclic graph.

A. Locally maximal paths

The strategy called *locally maximal paths* (LMP) performs a local search for bright structures. Definition 1 relative to a path is extended as follows:

Definition 2

$\pi^{LMP} = (x_1, \ldots\ x_n)$ is a locally maximal path if, and only if, the starting point of the path belongs to a boundary of D and if, $\forall x_i \in \pi^{LMP},\ 0 \leq i \leq n$, we have:

$$x_{i+1} \in \underset{x_j \in G(x_i)}{\arg\max}\{f(x_j)\}. \tag{6}$$

Equation 6 is used to iteratively construct a path from a starting point. The path ends when there is no successor to x_n. We note that several successors of a pixel x_i may have the same gray-scale value. In that case, the principal orientation is preserved by selecting the central pixel defined by the graph.

Pixels from the boundary of D are used as starting points for each selected graph. Thus, we define Π_f^{LMP}:

Definition 3

The set $\Pi_f^{LMP} = \{\pi_1^{LMP}, \ldots, \pi_p^{LMP}\}$, is the set of locally maximal paths of f.

Fig. 4(b) proposes an illustration of this set. Pixels which belong to at least one path of the set appear as white ; other pixels are black. The original image shows a DNA molecule observed with an electron microscope [17] (Fig. 4(a)). We note that the number of paths in the image is very low in comparison with the number of paths considered by path openings. We also observe that most pixels are black (no path crossing them). This method is not only sparse with respect to paths, but also with respect to pixels.

With this strategy, the search for the next pixel of a path is only local and the required time is very low. However, such paths are not very robust to noise. For instance, impulsive noise can disturb and deviate a path from a thin structure. To improve noise robustness, we make a global search for the paths with a second strategy, namely *globally maximal paths* (GMP).

B. Globally maximal paths

To build a path, we use graph theory to search for the *highest path* between two pixels of the image. To explain the notion of *highest path*, let us see the image as a topographical surface where high (resp. low) gray-scale values correspond to high (resp. low) altitudes. A globally maximal path (GMP) is a path between two points such that the average gray-scale value is the highest one among all available paths. The Dijkstra algorithm allows such search. However, given that the graph is directed and acyclic, specific algorithms can be used to provide fast algorithms. They are part of dynamic programming approaches and known as *longest path algorithms* [24]–[27]. The definition of a globally maximal path is given as follows:

Definition 4

$\pi^{GMP} = (x_1, \ldots\ x_n)$ is a globally maximal path if, and only if, the starting and end points of the path belong to a boundary of D and if we have:

$$\pi^{GMP} \in \underset{\pi \in \Pi_G}{\operatorname{argmax}} \left(\frac{1}{\operatorname{card}(\pi)} \sum_{x_i \in \pi} f(x_i) \right). \tag{7}$$

For a given point from the boundary of D, several globally maximal paths can be found. In practice, we select the path that preserves the principal orientation of the graph. Computing a GMP from all boundary pixels of the support D and for every considered graph, we obtain Π_f^{GMP}:

Definition 5

$\Pi_f^{GMP} = \{\pi_1^{GMP}, \ldots, \pi_p^{GMP}\}$, is the set of globally maximal paths of f.

This set is depicted on Fig. 4(g). We observe that the paths tend to go straight towards a bright structure, and then they follow it as far as they can. It is a global approach, robust to noise. With this strategy, the size of zones with no information (no path going through them) is larger than with locally maximal paths. We call these zones *blind regions*, since structures localized in these regions are not analyzed. With globally maximal paths, large and bright structures attract all paths; short and a bright structures found in their vicinity might not be seen.

GMP need more computation time than LMP and the size of blind regions is larger. Nevertheless, the robustness w.r.t noise is much higher than with LMP. Below we introduce a new general formalism that allows for intermediate strategies. We call this generalization the *β-maximal paths* (βMP).

C. β-maximal paths

The idea of βMP is to compute globally maximal paths on stripes of width β pixels. With a given graph, say south to north (Fig. 1(a)), the image is divided into several, horizontally oriented stripes of height β, as illustrated in Fig. 3.

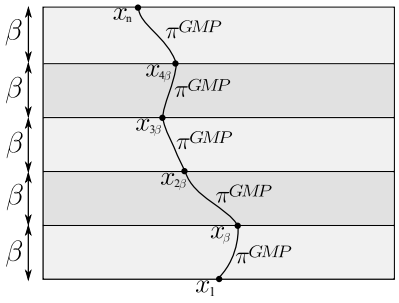


Fig. 3. Construction of β maximal paths with the concatenation of globally maximal paths.

From a starting point x_1 localized on the bottom boundary of the support D, we compute $\pi^{GMP} = (x_1, \ldots, x_\beta)$ on the first stripe. Then, x_β is the new starting point and we iterate this process until there is no successor to x_n with the graph G.

Definition 6

$\pi^{\beta MP} = (x_1, \ldots, x_n)$ is a β maximal path if, and only if, the starting point of the path belongs to a boundary of D and if $\pi^{\beta MP}$ is the concatenation of globally maximal paths π^{GMP} on stripes of size β.

Computing βMP from all boundary pixels of the support D and for every graph G, we get $\Pi_f^{\beta MP}$:

Definition 7

$\Pi_f^{\beta MP} = \{\pi_1^{\beta MP}, \ldots, \pi_p^{\beta MP}\}$, is the set of all the β maximal paths of f.

By definition, β maximal paths generalize previous methods: LMP are obtained with β=1 and GMP with β=∞. This method unifies the path extraction strategy and is used to compute the set of paths of Fig. 4. Thus, we control the trade-off between noise robustness and blind regions size. The choice of β is application dependent. On a noisy image, a high value for β is preferable. On the contrary, if the signal to noise ratio is high, a small value will reduce the size of blind regions.

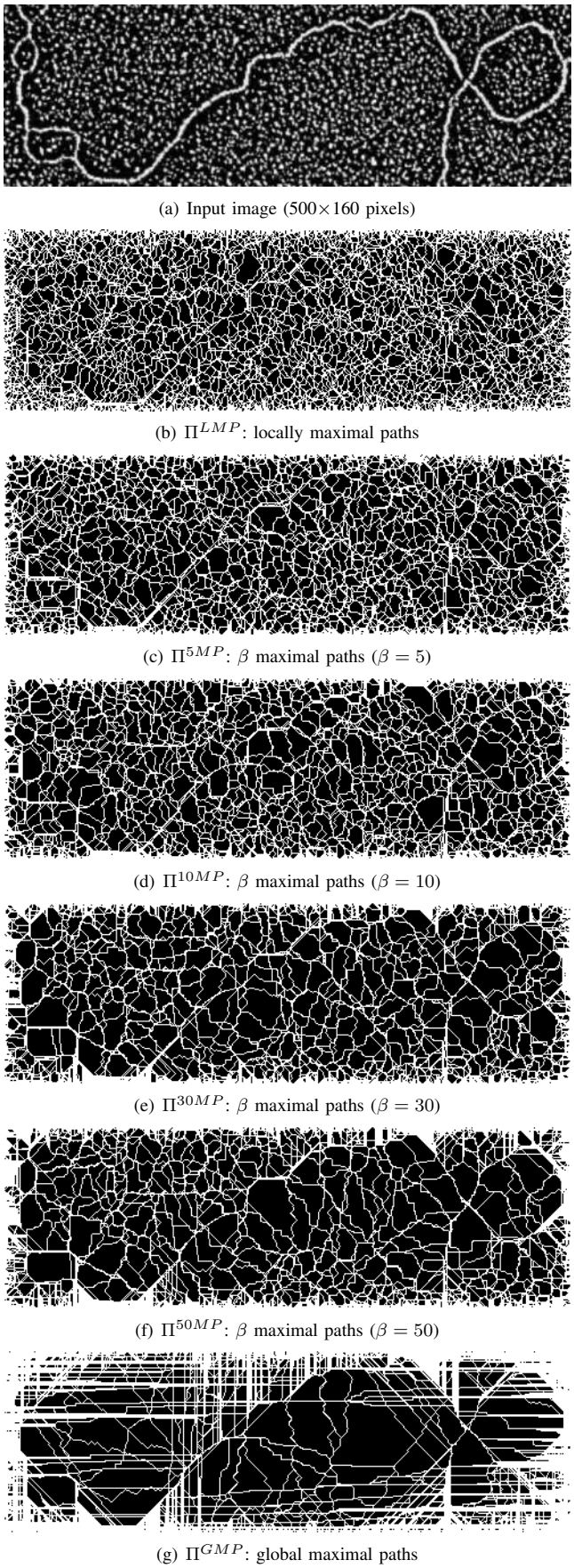
Now that we have proposed a general strategy to compute parsimonious paths, we have to:

- process them;
- build a final 2D image from the results on paths.

These problems are treated in the following section.

IV. Paths operators

We will see in this section how, from an operator working on single paths, we can build an operator working on the whole image.



(a) Input image (500×160 pixels)

(b) Π^{LMP}: locally maximal paths

(c) Π^{5MP}: β maximal paths ($\beta = 5$)

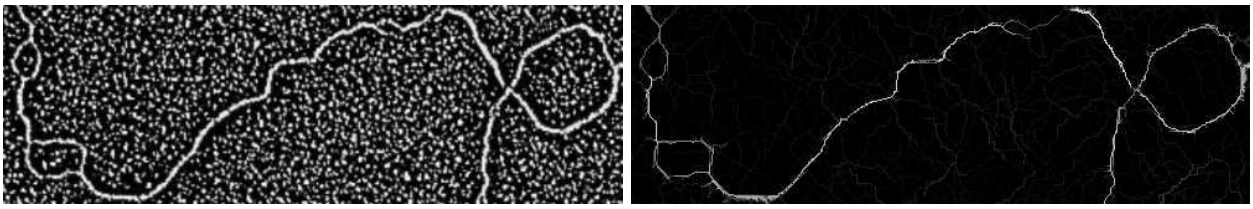
(d) Π^{10MP}: β maximal paths ($\beta = 10$)

(e) Π^{30MP}: β maximal paths ($\beta = 30$)

(f) Π^{50MP}: β maximal paths ($\beta = 50$)

(g) Π^{GMP}: global maximal paths

Fig. 4. Illustration of the sets of parsimonious paths on a given image (a). Pixels in white belong to at least one path of the set; other pixels appear in black.



(a) Input image (500×160 pixels) (b) PPO γ_L^{Π} L=50

Fig. 5. A molecule (a) on textured background. The PPO nicely filters background noise, but the contrast of some molecule sections is partially lost.

A. General strategy

Recall that D is the support of function f, also written $\mathrm{spt}(f)$, and $\Pi = \{\pi_i\}$ denotes a collection of paths of D. The restriction of f to the path π, denoted by $f_{/\pi}$, can be considered as a one-dimensional signal.

Let $\xi f_{/\pi}$ be the application of a 1-D operator ξ to f restricted to π. Using ξ we obtain a result for each path π_i of Π. Notice that for any x belonging to the intersection of two different paths π_i and π_j, one generally obtains $\xi f_{/\pi_i}(x) \neq \xi f_{/\pi_j}(x)$. Hence, we need a method to produce a single value.

Let $\mathrm{spt}(\Pi)$ denote the support of Π, i.e. the set of all points of D which belong to at least one path of Π. The illustrations of different sets of parsimonious paths given in Fig. 4 correspond in fact to this domain. We can extend $\xi f_{/\pi}$ to $\mathrm{spt}(\Pi)$ by taking:

$$\xi f(x) = \bigotimes_{\substack{\pi \ni x \\ \pi \in \Pi}} \xi(f_{/\pi})(x), \tag{8}$$

where $\otimes$ is a binary operator such as $\bigvee$ or $\bigwedge$ (i.e. supremum or infimum). In practice, the choice of the binary operator will depend upon the desired properties of the resulting operator. Several examples are described in the following sections.

We now have a result on all points belonging to at least one path. Indeed, Eq. 8 does not define $\xi f(x)$, for x outside the support of Π.

For some applications this might be enough, but we also may need to compute a value outside $\mathrm{spt}(\Pi)$. This choice is very application-dependent.

The first and simpler strategy consists of using a constant value outside $\mathrm{spt}(\Pi)$, e.g. the minimum or maximum of V, or even the original f. Another strategy is using a morphological reconstruction under/above f in order to propagate the results to the entire D. We will come back to this strategy in section V-D.

B. Parsimonious path openings

The first step to obtaining *parsimonious path openings* (PPO) is the choice of a convenient ξ. We naturally take the 1-D opening of size L, γ_L and use $\bigvee$ in Eq. 8 to compute a value for each $x \in \mathrm{spt}(\Pi)$:

$$\gamma_L^{\Pi}(f)(x) = \begin{cases} \bigvee\limits_{\substack{\pi \ni x \\ \pi \in \Pi}} \gamma_L(f_{/\pi})(x), & x \in \mathrm{spt}(\Pi) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

we pad D outside $\mathrm{spt}(\Pi)$ by zeros to ensure the anti-extensiveness of the operator.

Based on the fact that γ_L^{Π} is built from openings using supremums, and that outside $spt(\Pi)$ the result is set to the minimal value of V, i.e. zero, it can be demonstrated that this operator is an opening.

Note that if the set of paths Π is the complete set of paths associated to a graph, such as in Fig. 1, then $\mathrm{spt}(\Pi)$=D, no padding is necessary, and one obtains a classical path opening.

Concerning path closings, as in the classical path opening definition by Heijmans *et al.* [18], they can be obtained by duality: $\varphi(f)$=$-\gamma(-f)$. However it should be noted that Π_f and Π_{-f} are not the same. Whereas Π_f selects the bright structures Π_{-f} selects the dark ones.

C. Interlude: are parsimonious path openings really openings?

The set of paths Π is a function of the image f, written Π_f. So, what can we say about $\gamma_L^{\Pi_f}$? It can be shown that although the anti-extensivity remains a valid property, the increasingness and idempotence are lost. Therefore this more general operator is not an opening any more.

From a practical point of view, to avoid an unexpected behavior of serially composed operators, once Π_f is defined, it should be set constant. For example, for building granulometries [28] (see Section VI later) it is necessary to use the same Π in all stages. Similarly, when computing alternate sequential filters, it is logical to compute once the Π_f and Π_{-f} and use them, respectively, in computing all subsequent openings and closings.

Hence, provided the above mentioned precautions are met, PPO are openings.

D. Parsimonious Incomplete path openings (PIPO)

In presence of noise, long, thin structures are likely to appear disconnected, implying an underestimation of their length. See for example the result of a PPO in Fig. 5. Notice that while the background texture has been filtered out, some portions of the molecule are lost as well. We propose solutions below.

Consider a signal disconnected by one "noisy" pixel (Fig. 6, top). A path opening γ_L fails to detect a part of it (Fig. 6, middle, blue). To solve this problem, Talbot and Appleton [19] introduced Incomplete Path Openings (IPO), tolerating gaps (Fig. 6, middle, dashed red). However, the computation complexity of IPO is high, giving in practice one order of magnitude longer timings.

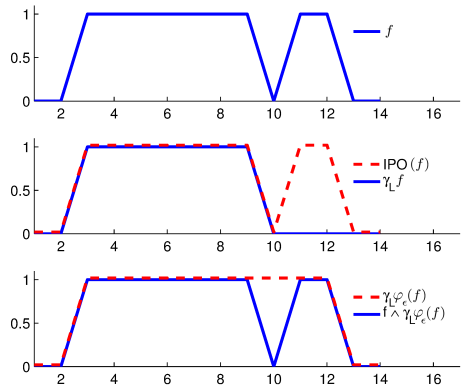


Fig. 6. Complete and incomplete parsimonious path openings. From top to bottom: input signal f. The opening $\gamma_L(f)$, L=5 fails to detect a portion of the signal due to noise. A closing $\varphi_\epsilon(f)$ of size ϵ=2, followed by the opening, $\gamma_L\varphi_\epsilon(f)$, detects correctly the entire structure. The infimum with the original $f \wedge \gamma_L\varphi_\epsilon(f)$ is used to ensure the anti-extensiveness of the result.

To achieve the same objective within our framework, we propose to use a small closing φ_ϵ of size ϵ to close the gap, followed by an opening of size L, see Fig. 6 bottom, dashed red. The 2-D operator version is built by using the supremum:

$$\zeta^{\Pi}_{L,\epsilon}(f)(x) = \begin{cases} \bigvee\limits_{\substack{\pi \ni x \\ \pi \in \Pi}} \gamma_L\varphi_\epsilon(f_{/\pi})(x), & x \in \mathrm{spt}(\Pi) \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

See Sec. VIII below for results on real images.

The original IPO verify the anti-extensivity of openings. To obtain an anti-extensive result from Eq. 10 one can take the infimum with the original image to obtain a parsimonious incomplete path opening $\gamma^{\Pi}_{L,\epsilon}f = f \wedge \zeta^{\Pi}_{L,\epsilon}f$ (see Fig. 6 bottom, blue).

V. Practical considerations

The construction of PPO comprises two steps: path extraction and path processing. We give here some details on the algorithms used to speed up both steps. Next, we introduce a parsimony parameter and we allow the detection of thick structures by assigning a value to all pixels of the resulting image, by means of the morphological reconstruction.

A. Path extraction: efficient algorithm

1) Local maximal paths: The LMP strategy (Sec. III-A) extracts paths from the input image with a local search, by using the definition corresponding to Eq. 6. Thanks to the locality, the LMP extraction is extremely efficient.

On the other hand, the extraction of β maximal paths by using the definition corresponding to Eq. 7 is slow. Below, we propose an efficient implementation using a fast, two-step algorithm.

2) Globally maximal paths: The extraction of GMP is in principle similar to the classical path opening on binary sets (Sec. II-B). This method was first introduced by Schmitt [29] to extract the longest path in a binary set, and extended later to gray level images by Vincent and Jeulin [30] to detect fracture lines in porous media or to extract correlogram tracks from noisy sonar data [27].

Starting from the input image f we compute the upstream λ^+ and downstream λ^- weighted distance maps with G and G^-, by extending equations 1 and 2:

$$\lambda^+(x) = \max_{y \in G(x)} \lambda^+(y) + f(x), \quad (11)$$

$$\lambda^-(x) = \max_{y \in G^-(x)} \lambda^-(y) + f(x). \quad (12)$$

Then, we sum these two distance maps to get, for each pixel of the image, the weighted length of the longest and brightest path going through it:

$$\lambda(x) = \lambda^-(x) + \lambda^+(x). \quad (13)$$

Finally, GMP are obtained by using the LMP search (Eq. 6), on the map λ.

To obtain βMP, we divide the image into several, β-pixel wide stripes oriented perpendicularly to the main direction of the selected graph, and we compute GMP on each of these stripes.

To conclude, the LMP extraction and the map λ are both computed in $\mathcal{O}(1)$ per pixel. Consequently, GMP (and βMP) are extracted with the same $\mathcal{O}(1)$ complexity per pixel as LMP. Nonetheless, the execution time of GMP (and βMP) is longer due to the time needed to compute λ.

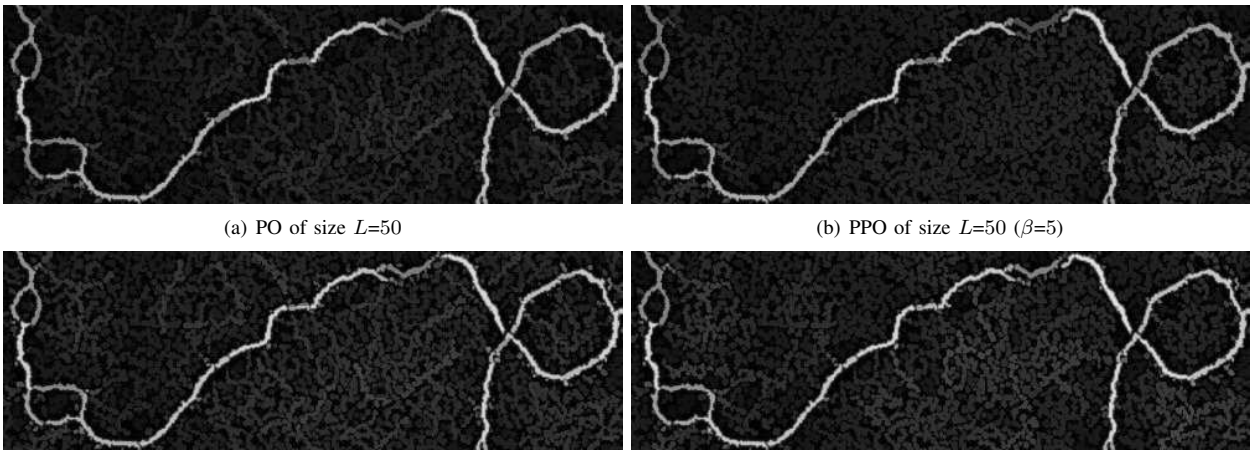
B. Path filtering: efficient algorithm

The βMP strategy individualizes each path π of the set $\Pi^{\beta MP}$. The image alongside a path, $f_{/\pi}$, is a 1D signal. Efficient algorithms are available in the literature for 1D openings or closings that run in one scan of the signal with a $\mathcal{O}(1)$ cost per pixel. The fastest algorithm for 1D openings is the one invented by Van Droogenbroeck and Buckley [31]. However, it uses a histogram, consequently the whole signal to be processed must be known in advance, and the pixel values are limited to 8-bits. Instead, we propose using another fast algorithm [32], [33], which offers the following additional advantages: (i) it computes the output signal progressively, each time a pixel is added to a path; (ii) it can handle the signal borders in two different ways, by padding with zeros or with ∞; (iii) it can handle any input data accuracy (integer or floating point) with no extra cost; (iv) the complexity is independent of the size of the opening; (v) it is fast and GPU compliant, as shown by Karas et al. [34].

In the next section, a parsimony parameter k is introduced to further reduce the computation time.

C. Parsimony parameter k

By observing the images of Fig. 4, we note that many paths, with close starting points, converge to follow the same structures. This phenomenon increases with increasing β. Thus, many paths from the set $\Pi^{\beta MP}_f$ do not bring new information and can be removed. Since paths with close starting points have a high probability of being attracted by the same structures, k will be the distance between starting points of two adjacent paths.



(a) PO of size L=50 (b) PPO of size L=50 (β=5)

(c) IPO of size L=50 (ϵ=2) (d) PIPO of size L=50 (ϵ=2, β=5)

Fig. 7. Comparison between path openings (first column) with parsimonious path openings with a reconstruction step (second column) for complete or incomplete paths.

Thus, we divide by k the total number of paths, as well as the cost of the algorithms. On the other hand, the PPO accuracy is slightly reduced (see section VI).

D. Morphological reconstruction

PPO is a sparse operator and yields a thin representation of objects. Should one need a thick detection, PPO results can be reconstructed under the original image. Efficient implementations of the morphological reconstruction are available in the literature [35], [36]. Their complexity is linear with reference to the image size.

Fig. 7 illustrates the DNA molecule extraction from the noisy background using parsimonious path openings followed by a reconstruction to ease the comparison with classical path openings. Figs. 7(a) and 7(b) compare PO and PPO with complete paths whereas Figs. 7(c) and 7(d) illustrate the results for incomplete paths with a tolerance ϵ of 1 pixel. Very similar results are obtained with PPO in comparison with PO with a significant reduction of the timings (see Sec. VII-B).

Using incomplete paths improves the detection since thin structures are reconnected. However, it also reconnects noisy pixels, preserving some structures in the image background. Thus, tuning the parameter ϵ is a trade-off between the size of the gaps to fill and the level of noise.

In the following section, the accuracy of PPO with respect to length measurements is studied

VI. Accuracy of PPO

In order to evaluate the accuracy of PPO, we will apply them to binary images containing segments of known length, and compare the measured length distribution using PPO with the theoretical one.

Size distributions are usually computed by a residual approach with a collection of increasing-size filters commonly known as granulometry, introduced by Matheron [28], [37]. Let us us consider a binary image f, and a set of paths Π_f computed on f. The family of PPO $\{\gamma_L^\Pi\}_{L\geq 1}$ is a granulometry. Note that we have droped the f index on Π, as the set of paths is computed once and for all, and then kept constant, for reasons explained in section IV-C. The corresponding size distribution is, for non-negative integers L:

$$SD_L = Meas(\gamma_L^\Pi - \gamma_{L-1}^\Pi). \tag{14}$$

In the present case, the measure *Meas* is the number of connected components of the binary image. Eq. (14) measures the length distribution.

PPO suffer from two types of error. The first one is the anisotropy of the length measurement, also present in the original PO. It comes from the discretisation of the path on the $\mathbb{Z}^2$ grid. The second error is due to the parcimonious scan of the support. The following text analyses the phenomena at the origin of these errors and their impact on the accuracy.

Let x be a measurement, and m the correct value. The relative measure error is:

$$err = \frac{x-m}{m}. \tag{15}$$

A positive *err* means overestimation, whereas a negative *err* means underestimation.

In our case x and m are distributions (expressed as probability density functions or frequencies in case of histograms). A number of measures exist to compare probability density functions or histograms (see [38] for a review of most common distances or divergences). However, none of these suits our case for the following reasons: i) The metric behind the majority of the distances considers the probability density function as a vector in an orthogonal space. Nonetheless, the histogram bins in our case are not orthogonal. ii) A distance is always positive, which does not reflect the difference between under- and over-estimation of a measure. iii) No distance or metric is correlated to the usually used relative measure error as in eq. 15. This means that for singleton distributions (containing only one point) we would not obtain the same value as with eq. 15.

Consequently, we define an equivalent of Eq. 15 for two histograms X and M, with respective mean values equal

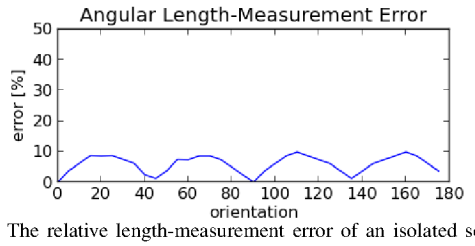


Fig. 8. The relative length-measurement error of an isolated segment with respect to the orientation.

to $\overline{X}$ and $\overline{M}$. We define the relative error of the measured distribution X to the ground truth distribution M as:

$$err = \frac{\overline{X} - \overline{M}}{\overline{M}}. \quad (16)$$

This error evaluation has the following properties: The difference $\overline{X} - \overline{M}$ is insensitive to the standard deviation of either distribution, that can be evaluated separately. It does not require the histograms to be normalized nor aligned. X and M can have different count sum and either X and M or both can be scalars. If both are scalars then eq. 16 is equivalent to eq. 15.

In the first experiment we evaluate the relative measure error on the length of an isolated, rotating straight segment of constant length, placed in the center of the image. The results are shown in Fig. 8. We can observe that the measurement is correct for multiples of $45°$. For all other orientations, the length is overestimated because of the limited set of elementary orientations in the corresponding graph (see Fig. 1). The error is however smaller than 10%.

The second experiment evaluates the error to measure the length distribution on a population of co-linear straight segments. We use a model of constant-length L=80, equally-oriented segments, randomnly placed in the support without touching each other. The length distribution M of the model is a Dirac impulse located at L (red in Fig. 9), the measured length distribution X is given in green. The relative measure error w.r.t. the orientation (eq. 16) is given in Fig. 10 (green curve). Again, the length distribution is exact if the segments are oriented in a multiple of 45 degrees. For other orientations the two mentioned errors occur: i) the length is overestimated due to the discretization of the support (see bin 90 in Fig. 9 bottom); ii) a few segments have been counted in short bins. This is a direct consequence of the parsimony of the support scan. Some segments are not entirely followed by any path. For example, in Fig. 11 the segment in the middle is only partially followed by the two dotted and dashed paths. These fragments are filtered, and counted in short histogram bins.

This fractioning can be attenuated if a morphological reconstruction is used, as proposed in section V-D. In Fig. 11, the middle segment is entirely reconstructed from its central portion used as the marker. A PPO followed by a morphological reconstruction is a connected operator [39], and yields more accurate results. In Figs. 9 and 10 the results obtained

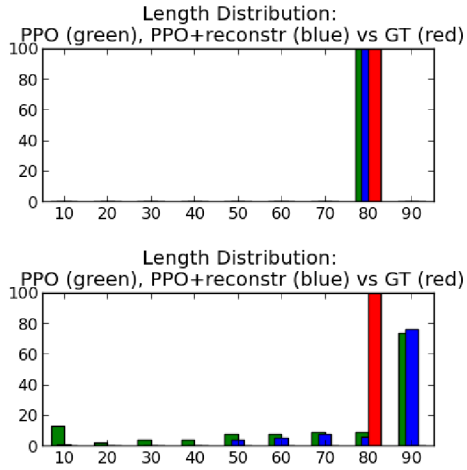


Fig. 9. The length distribution of a population of straight, co-linear segments of constant length (L=80), oriented at $45°$ (top), and $55°$ (bottom).

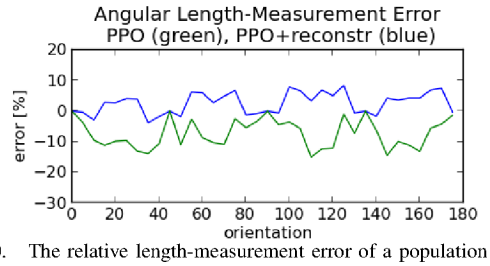


Fig. 10. The relative length-measurement error of a population of co-linear segments with respect to the orientation.

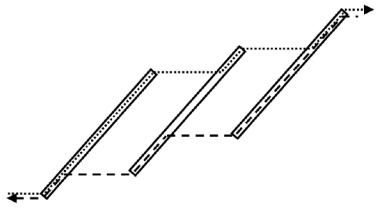


Fig. 11. Occlusions bias the length measure. Neither the dotted nor the dashed path can entirely follow the middle segment.

with the PPO followed by the morphological reconstruction are shown in blue. Consequently, in the third experiment we use the opening-reconstruction version. Note, however, that the fragmentation cannot be completely avoided. Indeed, it can happen that no marker remains to reconstruct the segment.

The third experiment evaluates the relative error w.r.t. the spatial density of the segments. We use a collection of realisations of a random model of straight segments uniformly distributed in a support of 512×512 pixels (see Fig. 12). The model contains a population of $N \in \{50, 100, 150, \ldots, 450\}$ segments of random length l drawn from the normal distribution $\mathcal{N}(\mu, \sigma)$ with $L = \mathcal{N}(40, 20)$, bounded in the (5, 90) interval. The segments have a random uniformly distributed orientation and are placed in the support without touching each other.

We observe that the fragmentation in sparsely populated media does not occur, as illustrated in Fig. 13 (top), and increases progressively with the density of the population (see the overestimated bin 10 in Fig. 13 (bottom)).

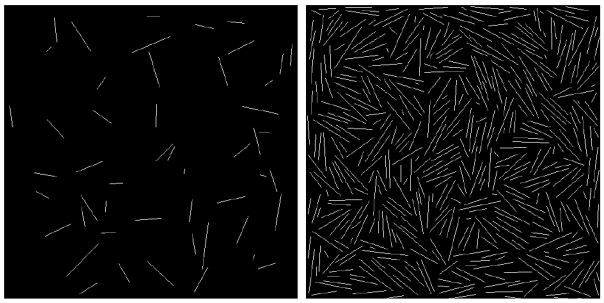


Fig. 12. Two realisations of the random gaussian length distribution model with N=50 and N=450 segments.

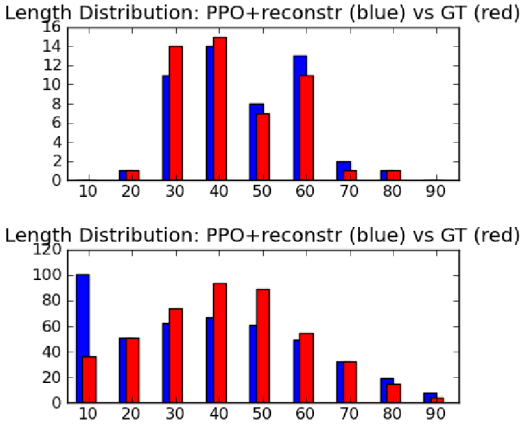


Fig. 13. The length distributions measured with PPO+reconstruction (blue) vs ground-truth (red) for N=50 (top) and 450 (bottom) segments.

To complete the error evaluation w.r.t. the density of the media see the two following Figs. 14 and 15 that confirm that with increasing number of segments N the error increases towards stronger underestimation. Both figures are provided for various values of the parameters k and β. One can observe that the error also increases with increasing parsimony (increasing k) and decreasing locality (increasing β).

It is interesting to observe that even with a high density of segments, as shown in Fig. 12 (right), i.e. a situation deemed unfavorable to a parsimonious approach, PPO parameters can be chosen in such a way that the absolute error is smaller than 10%.

VII. Complexity and Timings

A. Complexity

In the original version of PO, the number of paths passing through one pixel is exponential w.r.t. the length L. Consequently, if implemented in a naive way, PO have exponential complexity. An implementation with logarithmic complexity w.r.t. L has been proposed in [21]. More recently, the complexity of incomplete paths has been decreased to logarithmic w.r.t. L and linear w.r.t. the tolerance ϵ [19].

Here we analyze the complexity of PPO, split into two parts: path extraction and path filtering.

We have shown above that the collection of paths βMP can be extracted in $\mathcal{O}(1)$ per pixel (Sec. V-A), and that an image

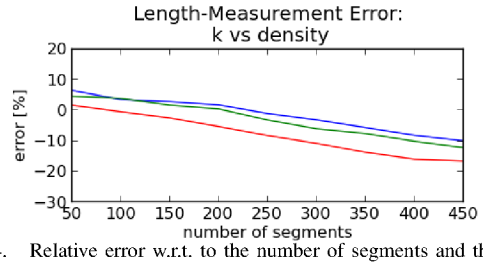


Fig. 14. Relative error w.r.t. to the number of segments and the parsimony parameter: k=1, 5, 10 (blue, green, red).

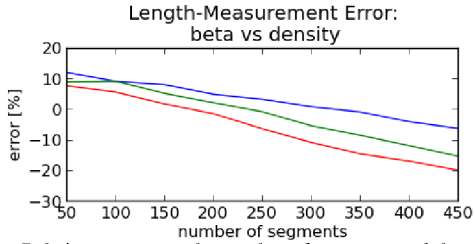


Fig. 15. Relative error w.r.t. the number of segments and the parameter β= 1, 20, 50 (blue, green, red).

f can be filtered alongside a path π also in $\mathcal{O}(1)$ per sample (Sec. V-B). To complete the analysis of complexity we need to count the number of paths, and evaluate their length.

For a given image f, with D=spt(f) a $W\times H$ rectangle (width×height), we have $2W$ vertical and $2W$ horizontal paths. Vertical paths are H pixels long, and horizontal paths are W pixels long. After dropping multiplicative constants, this yields a linear complexity of $\mathcal{O}(HW)$ when only horizontal and vertical paths are used.

When diagonal paths are used, the complexity slightly increases. We count a total of $4(H+W)$ diagonal paths (for four principal diagonal orientations). The diagonal paths have unequal length, bounded though by H+W. This yields a slightly higher complexity of $\mathcal{O}((H+W)^2)$, yet still a linear factor of the support size card(D)=HW.

Therefore, the complexity of PPO an PIPO is proportional to the number of pixels in the images, and independent of the length L and the tolerance ϵ of incomplete paths, which is an improvement with respect to the state of the art.

B. Timings

PPO have a low complexity and have been designed to address the timing issues of classical path openings. We compare here the timings of both approaches. Fig. 16 shows a benchmark for complete and incomplete paths. The gain is huge and we verify that PPO run in constant time with respect to L and ϵ. For complete paths, the computation time is divided by 77 between PO and PPO, with k=10. The gap between incomplete path openings and parsimonious incomplete path openings is even larger. We divide the average time by almost 3000.

In another experiment, we benchmark PPO against parameters β, ϵ and k, with the same image of size 768×576. We check in Fig. 17 that the timings are independent of β (except for β=1, which does not use the weighted distance map λ). Regarding the gap tolerance ϵ, the size of the closing does not

increase the computation time. On the contrary, an increase of ϵ results in a reduction of the computation time (see [33] for explanation). With the parsimony parameter k, timings decrease with $\frac{1}{k}$ as expected from the theoretical complexity.

The last benchmark, Fig. 18, exhibits the execution time against the image size. The timings confirm the linear complexity of the algorithm. The timings were measured on a real image, obtained by taking a crop from Fig. 19(b).

VIII. Results

We use these filters to detect road-pavement cracks, as illustrated in Fig. 19. These cracks form thin, tortuous and possibly disconnected objects, difficult to detect due to the presence of the surrounding texture. Fig. 19(a) shows a raw image as acquired by the camera, and (b) the preprocessed image – uniformization of intensity, and thin structures enhancement by means of a morphological top hat – provided as input to the following steps.

A single opening γ_L, with a large size L=80, removes the texture noise. Some cracks are missing though (Fig. 19(c)). A fair detection of all cracks present in the image can be obtained with a PIPO $\gamma_{L,\epsilon}$, of size L=140, and tolerance ϵ=3 (Fig. 19(d)).

IX. Conclusions

This paper presents a new family of parsimonious operators for image processing, based on paths. In comparison with classical path openings, only a relevant subset of paths is used to retrieve similar information.

The extraction of paths is decoupled from the path filtering, which brings two major advantages: i) a new, general scan strategy allows tuning the search continuously from local to global, which gives the possibility to find a trade-off between accuracy and robustness to noise; ii) it allows defining different operators alongside the same collection of paths, to operate on the same objects. For instance, the combination of a path opening with a closing allows reconnecting discontinued bright structures, thus achieving results similar to those obtained with incomplete path openings, tolerant to missing pixels in paths.

An efficient algorithm is used to compute openings or closings alongside a path in $\mathcal{O}(1)$ per pixel to decrease the complexity of both openings, and incomplete path openings to a constant, independent of the length and the tolerance parameters. Consequently, the timings are several orders of magnitude lower in comparison with classical (incomplete) path openings, with comparable results. Hence, PPO are usable in high-throughput industrial applications. Other interesting properties include: i) the ability to accept arbitrary data accuracy (integer or floating point), and ii) the ability to handle the border effect in two different ways (extending the support with 0 or with ∞).

We provide a thorough study of the accuracy to show that processing a conveniently chosen subset of paths can provide a result sufficient for certain applications. Such a parsimonious approach has allowed us the bridge the gap

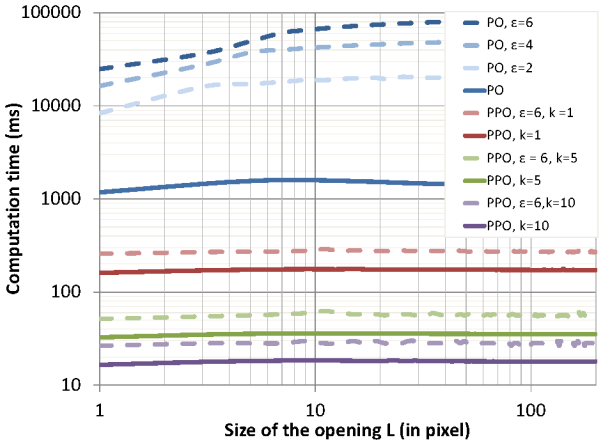


Fig. 16. Benchmark between path openings and parsimonious path openings (β=1) for complete and incomplete paths. The timings are plotted in ms w. r. t. L (log-log scale). The input image size is 768×576 pixels. The benchmarks were made on a laptop computer (Intel Core 2 Duo T7700 CPU @2.4GHz) by taking the average value from 100 realizations.

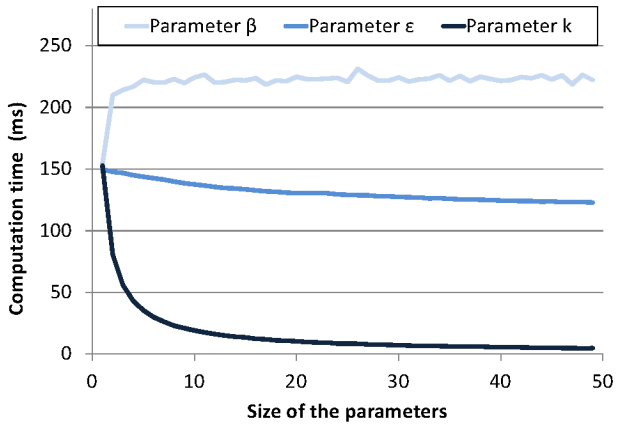


Fig. 17. Influence of the parameters (β, ϵ and k) on the computation time of parsimonious path openings.

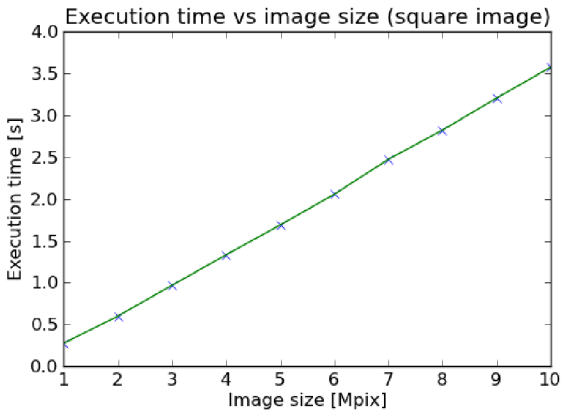
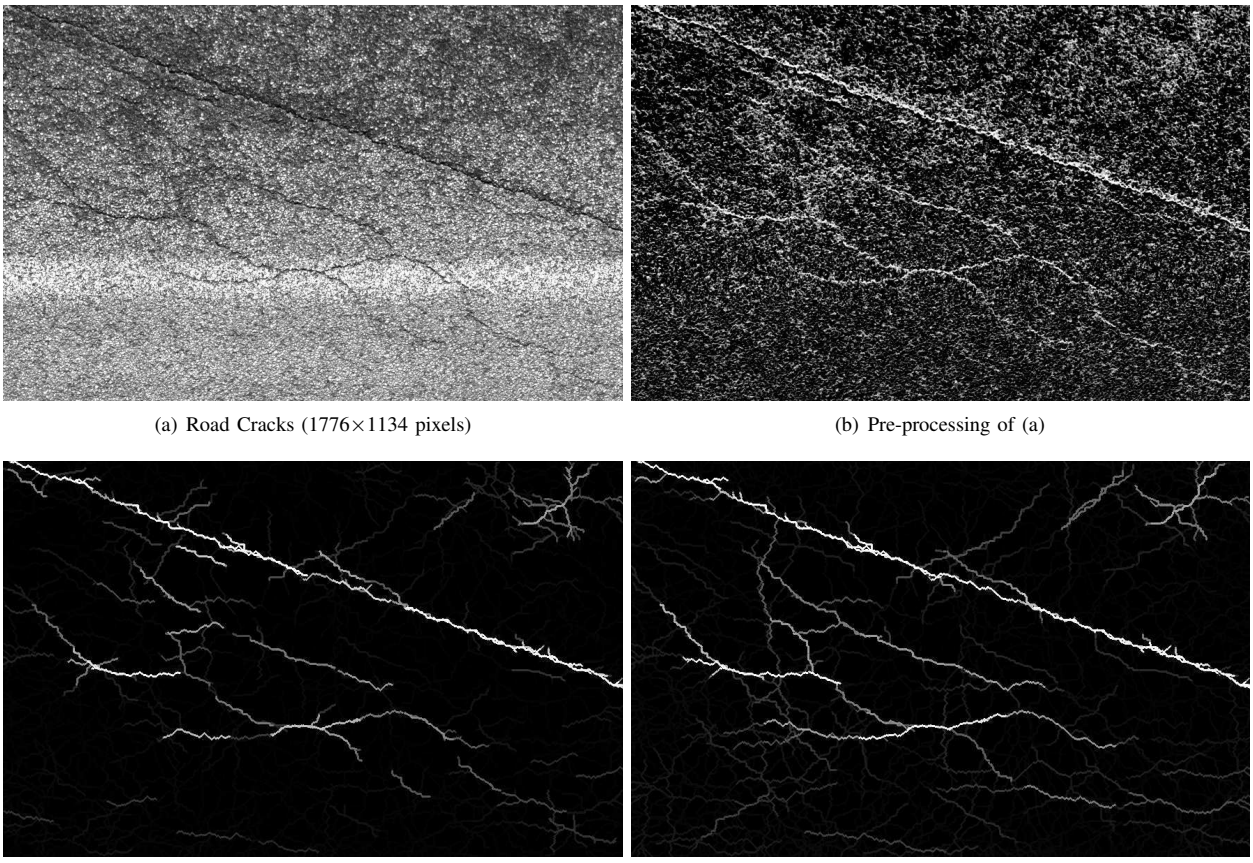


Fig. 18. Execution time of parsimonious path openings (β=1, ϵ=0, k=1) vs image size. Test image obtained by taking a square crop from Fig. 19(b).



(a) Road Cracks (1776×1134 pixels) (b) Pre-processing of (a)

(c) PPO γ_L, L=80 (d) PIPO $\gamma_{L,\epsilon}$, L=140, ϵ=3

Fig. 19. The detection of road cracks with various path operators. All operators use the collection of parsimonious paths $\Pi^{\beta MP}$ with β=36 and k=1. (The contrast of all images has been enhanced for printing purposes).

between an interesting methodological tool (path openings) and a practical, computation intensive, real-world application.

This work opens different research perspectives. For instance, the proposed strategy only uses paths seeded at the image border. Other starting points could be considered, for example with a random placement, as is in the geodesic voting approach [40]. This could help obtaining information on blind regions. Other morphological parsimonious image representations can be also considered, based for example on image extrema, or image ridges and valleys.

ACKNOWLEDGMENT

This work was made possible thanks to the support of the "Pôle ASTech" and the "Pôle Nucléaire de Bourgogne", and has been financed by the French "Département de Seine et Marne"

The authors are grateful to Hugues Talbot (A2SI ESIEE, IGM) for providing his incomplete path opening code for benchmarking.

The road crack images are property of Colas SA.

REFERENCES

[1] C. Sinthanayothin, J. Boyce, H. Cook, and T. Williamson, "Automated localisation of the optic disc, fovea, and retinal blood vessels from digital colour fundus images," *British Journal of Ophthalmology*, vol. 83, no. 8, pp. 902–910, 1999.

[2] T. Walter and J.-C. Klein, "Segmentation of color fundus images of the human retina: Detection of the optic disc and the vascular tree using morphological techniques," in *Medical Data Analysis*, ser. Lecture Notes in Computer Science, J. Crespo, V. Maojo, and F. Martin, Eds. Springer Berlin Heidelberg, 2001, vol. 2199, pp. 282–287. [Online]. Available: http://dx.doi.org/10.1007/3-540-45497-7_43

[3] A. Katartzis, H. Sahli, V. Pizurica, and J. Cornelis, "A model-based approach to the automatic extraction of linear features from airborne images," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 39, no. 9, pp. 2073–2079, 2001.

[4] S. Valero, J. Chanussot, J. Benediktsson, H. Talbot, and B. Waske, "Advanced directional mathematical morphology for the detection of the road network in very high resolution remote sensing images," *Pattern Recognition Letters*, vol. 31, no. 10, pp. 1120–1127, 2010.

[5] V. Morard, E. Decencière, and P. Dokládal, "Geodesic attributes thinnings and thickenings," in *Mathematical Morphology and Its Applications to Image and Signal Processing*. Springer, 2011, pp. 200–211.

[6] V. Morard, E. Decencière, and P. Dokládal, "Region growing structuring elements and new operators based on their shape," in *Proceedings of Signal and Image Processing*. ACTA Press, 2011, pp. 1–8.

[7] J. D. Kurdy B., "Directional mathematical morphology operations," in *In 5th European Congress For Stereology, Acta Stereologica*, vol. 8/2, 1989.

[8] P. Soille and H. Talbot, "Directional morphological filtering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1313–1329, 2001.

[9] T. Koller, G. Gerig, G. Szekely, and D. Dettwiler, "Multiscale detection of curvilinear structures in 2-D and 3-D image data," in *Computer Vision. Proceedings of the Fifth International Conference on*. IEEE, 1995, pp. 864–869.

[10] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[11] O. Tankyevych, H. Talbot, and P. Dokládal, "Curvilinear morpho-hessian filter," in *Biomedical Imaging: From Nano to Macro. 5th IEEE International Symposium on*. IEEE, 2008, pp. 1011–1014.
[12] L. Vincent, "Morphological area openings and closings for grey-scale images," *Shape in Picture: Mathematical Description of Shape in Grey-level Images*, pp. 196–208, 1993.
[13] E. Breen and R. Jones, "Attribute openings, thinnings, and granulometries," *Computer Vision and Image Understanding*, vol. 64, pp. 377–389, Nov. 1996.
[14] M. Wilkinson and M. Westenberg, "Shape preserving filament enhancement filtering," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001*, vol. 2208, 2001, pp. 770–777.
[15] E. Urbach and M. Wilkinson, "Shape-only granulometries and grey-scale shape filters," in *Mathematical Morphology and Its Applications to Image and Signal Processing*, 2002, pp. 305–314.
[16] V. Morard, E. Decencière, and P. Dokládal, "Efficient geodesic attribute thinnings based on the barycentric diameter," *Journal of Mathematical Imaging and Vision*, pp. 1–15, 2012.
[17] H. Heijmans, M. Buckley, and H. Talbot, "Path openings and closings," *Probability, Networks and Algorithms*, no. E 0403, pp. 1–21, 2004.
[18] ——, "Path openings and closings," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2, pp. 107–119, 2005.
[19] H. Talbot and B. Appleton, "Efficient complete and incomplete path openings and closings," *Image and Vision Computing*, vol. 25, no. 4, pp. 416–425, 2007.
[20] J. Serra, *Image analysis and mathematical morphology*. Academic Press, London, 1982, vol. 1.
[21] B. Appleton and H. Talbot, "Efficient path openings and closings," *Mathematical Morphology: 40 Years On*, pp. 33–42, 2005.
[22] C. Luengo Hendriks, "Constrained and dimensionality-independent path openings," *Image Processing, IEEE Transactions on*, vol. 19, no. 6, pp. 1587–1595, 2010.
[23] F. Cokelaer, H. Talbot, and J. Chanussot, "Efficient Robust d-Dimensional Path Operators," *Selected Topics in Signal Processing, IEEE journal of*, vol. 6, no. 7, November 2012.
[24] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
[25] G. Gallo and S. Pallottino, "Shortest path algorithms," *Annals of Operations Research*, vol. 13, no. 1, pp. 1–79, 1988.
[26] M. Buckley and J. Yang, "Regularised shortest-path extraction," *Pattern Recognition Letters*, vol. 18, no. 7, pp. 621–629, 1997.
[27] L. Vincent, "Minimal path algorithms for the robust detection of linear features in gray images," *Computational Imaging and Vision*, vol. 12, pp. 331–338, 1998.
[28] G. Matheron, *Random sets and integral geometry*. New York: Wiley, 1974.
[29] M. Schmitt, "Des algorithmes morphologiques à l'intelligence artificielle," Ph.D. dissertation, Ecole Nationale Supérieure des Mines de Paris, 1989.
[30] L. Vincent and D. Jeulin, "Minimal paths and crack propagation simulations," *Acta Stereologica*, vol. 8, no. 2, pp. 487–494, 1989.
[31] M. Van Droogenbroeck and M. Buckley, "Morphological erosions and openings: fast algorithms based on anchors," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2, pp. 121–142, 2005.
[32] V. Morard, P. Dokládal, and E. DecenciÃ¨re, "Linear openings in arbitrary orientation in O(1) per pixel," in *Acoustics, Speech and Signal Processing, IEEE International Conference on*. IEEE, 2011, pp. 1457–1460.
[33] V. Morard, P. Dokládal, and E. Decencière, "One-dimensional openings, granulometries and component trees in o (1) per pixel," *Selected Topics in Signal Processing, IEEE journal of*, pp. 1–10, 2011.
[34] P. Karas, V. Morard, J. Bartovský, T. Grandpierre, E. Dokládalová, P. Matula, and P. Dokládal, "GPU Implementation of Linear Morphological Openings with Arbitrary Angle," *Journal of Real-Time Image Processing*, 2012.
[35] L. Vincent, "Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms," *Image Processing, IEEE Transactions on*, vol. 2, no. 2, pp. 176–201, 1993.
[36] K. Robinson and P. Whelan, "Efficient morphological reconstruction: a downhill filter," *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1759–1767, 2004.
[37] G. Matheron, *ElÃ©ments pour une ThÃ©orie des Milieux Poreux*. Masson, 1967.
[38] S.-H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions," *Mathematical Models and Methods in Applied Sciences, Intl. journal of*, vol. 1, no. 4, pp. 300–307, 2007.
[39] P. Salembier and J. Serra, "Flat zones filtering, connected operators and filters by reconstruction," *IEEE Transactions on Image Processing*, vol. 3, no. 8, pp. 1153–1160, 1995.
[40] Y. Rouchdy and L. Cohen, "Image segmentation by geodesic voting. application to the extraction of tree structures from confocal microscope images," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE, 2008, pp. 1–5.

PLACE PHOTO HERE

Vincent Morard Biography text here.

PLACE PHOTO HERE

Petr Dokládal Biography text here.

PLACE PHOTO HERE

Etienne Decencière Biography text here.

Noname manuscript No.
(will be inserted by the editor)

Efficient geodesic attribute thinnings based on the barycentric diameter

Vincent Morard · Etienne Decencière · Petr Dokládal

Received: date / Accepted: date

Abstract An *attribute opening* is an idempotent, anti-extensive and increasing operator, which removes from an image connected components which do not fulfil a given criterion. When the increasingness property is dropped, we obtain a – more general – *attribute thinning*. In this paper, we propose efficient grey scale thinnings based on geodesic attributes.

Given that the geodesic diameter is time consuming, we propose a new geodesic attribute, the *barycentric diameter* to speed up the computation time. Then, we give the theoretical error bound between these two attributes, and we note that in practice, the barycentric diameter gives very similar results in comparison with the geodesic diameter. Finally, we present the algorithm with further optimisations, to obtain a 60× speed up.

We illustrate the use of these thinnings in automated non-destructive material inspection: the detection of cracks. We discuss the advantages of these operators over other methods such as path openings or the supremum of openings with segments.

Keywords Geodesic attributes · geodesic diameter · barycentric diameter · elongation · tortuosity · thinnings · thickenings · mathematical morphology

1 Introduction

Automated optical inspection for non-destructive testing is an economically important, fast developing domain. One of its main tasks, the detection of cracks, requires efficient image processing methods. The detection of cracks (usually long, thin and randomly tortuous structures) in the presence of noise is a challenging task.

Generally speaking, filtering of unwanted objects (e.g. noise) while preserving the desired information is a frequent pre-processing step. Mathematical morphology [12,25,26] is based on a set approach and classically uses structuring elements to obtain information on the morphology of the objects. In their overview of morphological filtering, Serra and Vincent [27] noticed that simple openings and closings with square, disk or hexagonal structuring elements, are often good enough for the filtering task. However, noise reduction and feature enhancement properties can be improved by adapting the shape and size of the structuring elements to the image content (see e.g. [2], [3] and [14]). Openings and closings by reconstruction are also considered as a part of shape-adaptive morphology. This led Vincent [33] to propose area openings and later Breen and Jones more general attribute openings and thinnings [4]. Later, Urbach and Wilkinson [32,34] proposed thinnings based on the inertia to detect elongated objects in the image.

In order to enhance thin structures of random orientation, one can use the supremum of linear openings over all orientations. Although this suffices when the structures have a bounded local curvature, it fails for randomly tortuous thin cracks. In order to relax the curvature limit, Heijmans et al. [5] introduced *path openings*. Instead of rigid linear structuring elements, they use flexible paths inferred from an underlying connectivity graph. The paths are kept if, and only if, their length is longer than a given constant. Nonetheless, very tortuous structures still cannot be entirely detected.

The geodesic diameter, initially proposed by Lantuéjoul and Beucher [8], is particularly useful to measure the length of thin structures. We associated, in Morard

Centre de Morphologie Mathématique
Mathématiques et Systèmes, MINES ParisTech;
35, rue Saint-Honoré, 77305 Fontainebleau CEDEX - France
E-mail: Vincent.Morard@mines-paristech.fr
E-mail: Etienne.Decenciere@mines-paristech.fr
E-mail: Petr.Dokladal@mines-paristech.fr

et al. [13], attribute thinnings and geodesic attributes to build a new family of operators. These geodesic attribute thinnings increase the shape flexibility by removing any constraint on the tortuosity. Albeit developed to detect cracks, they can be used to detect other kinds of similar fibrous structures. Nevertheless, the geodesic diameter being computationally expensive, it is unsuitable for time-critical industrial applications.

This paper extends [13], by introducing and deeply analysing a new, fast and accurate geodesic attribute: *the barycentric diameter*. We also propose further experiments, results and an efficient algorithm, which is faster than path openings and readily usable for time-critical industrial applications.

We start by reviewing attribute thinnings and geodesic binary attributes in sections 2 and 3. Section 4 introduces the barycentric diameter. Section 5 explains the method to construct geodesic attribute thinnings. Section 6 illustrates their interests through two applications and provides a detailed comparison with path openings, initially developed to solve similar problems. Lastly, section 7 focuses on an efficient implementation, the complexity and the timings.

2 Background: attribute thinnings

Even if most definitions used in this paper can be given in a continuous domain, for practical reasons we consider a discrete domain.

Let $I : D \rightarrow \{0; 1\}$ be a binary image, with D a finite (typically rectangular) subset of $\mathcal{Z}^2$. A *binary operator* (or operator, when there is no ambiguity) is a function that transforms a binary image into another binary image. The set X contained in I is $X = \{x \in D | I(x) = 1\}$ and we denote X^c its complement in D. We associate to D a local neighbourhood describing the connection between adjacent pixels. In this study, each pixel is connected to its eight nearest neighbours. With this 8-connectivity, we introduce the collection of the connected components (CC) of X as $\{X_i\}$. From now on, by *object* we understand one connected component from the collection $\{X_i\}$.

2.1 Connected components and attributes

We wish to keep or delete the objects in an image according to intuitive *attributes* like length, tortuosity, elongation or circularity. Given some connected component X_i, these attributes will be respectively denoted $L(X_i)$, $T(X_i)$, $E(X_i)$ and $C(X_i)$. Their definitions will be given later. These attributes allow to define *criteria* like "longer than or equal to" ($L(.) \geq \lambda$), or "less tortuous than" ($T(.) < \lambda$), with some $\lambda \in \mathcal{R}^+$. Formally, a criterion χ is a function mapping the set of connected components of D into $\{0, 1\}$, where 0 can be interpreted as *false* and 1 as *true*.

The binary operator based on criterion χ is then defined as:

$$\psi_\chi(X_i) = \begin{cases} X_i & \text{if } \chi(X_i) \text{ is true} \\ \emptyset & \text{otherwise,} \end{cases} \quad (1)$$

for all CCs X_i included in D.

2.2 Binary attributes thinnings

Based on the binary operator ψ_χ, the corresponding attribute thinning of X is

$$\rho_\chi(X) = \bigcup_i \psi_\chi(X_i). \quad (2)$$

This is equivalent to scanning, one by one, the different CCs of X, and either preserving them intact or removing them, depending on the criterion χ.

Attribute thinnings are anti-extensive and idempotent (see [4] for the proof). Moreover, if the criterion is increasing, then the corresponding attribute thinning is also increasing and the operator becomes an attribute opening.

The dual transform of a thinning is called a thickening, and is obtained using the complementation:

$$\delta_\chi(X) = [\rho_\chi(X^c)]^c. \quad (3)$$

In what follows, we restrict our study to thinnings for simplicity, since the computation of thickenings is straightforward with equation 3.

In the next section, we will discuss the extension of thinnings to grey level images.

2.3 Grey level attribute thinnings

A grey level image f is a mapping from D into $V = \{0, \ldots, M-1\}$. Image f can be decomposed into a collection of sets obtained by thresholding. Hereafter, $X^h(f)$ denotes the threshold of f above $h \in V$:

$$X^h(f) = \{x \mid x \in D, f(x) \geq h\}, \quad (4)$$

and $X_i^h(f)$ the i-th connected component of $X^h(f)$.

Any increasing binary operator, such as an opening, can be generalised to grey level images by applying it to all the threshold sets $X^h(f)$, and stacking the results to recompose the grey-level image again [11]:

$$(\rho_\chi^{\text{direct}}(f))(x) = \vee\{h \in V \mid x \in \rho_\chi(X^h(f))\}. \quad (5)$$

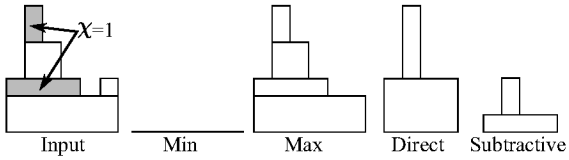


Fig. 1 Results of an attribute thinning with different filtering rules. From left to right: input signal with two CCs that fulfil the criterion χ (in grey), min, max, direct and subtractive rules.

However, binary attributes are not necessarily increasing, therefore their extension to grey level images is not straightforward. Some filtering rules are reported in the literature to construct this extension [4,23,32], see Fig. 1:

1. ***Min* rule:** A connected component X_i is removed if $\chi(X_i) = 0$ or if it exists a connected component X_j such that $X_i \subset X_j$, which is removed.
2. ***Max* rule:** A connected component X_i is removed if $\chi(X_i) = 0$ and if all the CCs X_j such that $X_j \subset X_i$ are also removed.
3. ***Direct* rule:** This is the simplest rule available to extend a binary operator to grey scale images. It uses the same equation as for the opening, eq. 5.
4. ***Subtractive* rule:** A CC X_i is removed if $\chi(X_i) = 0$. All the other connected components such that $X_j \subset X_i$ are lowered by the value of the contrast of X_i.

Other rules have been introduced for more specific treatments: the median rule [22] is a pre-filter to improve the robustness of the decision, the Viterbi rule [23] solves an optimisation problem for the same purpose, finally, the k-subtractive and the k-absorption rules [19] work with k-flat zones. All these extensions lead to the same result when applied to increasing operators, such as openings.

The choice of the rule depends on the application. However, it is shown in [32,31], that the subtractive rule is preferable. This is the only rule that fulfils two intuitive requirements: (i) after filtering, all the structures that do not meet the criterion are removed; and (ii) the difference image $f - \rho_\chi(f)$ contains only the structures that do not meet the criterion.

Furthermore, our final application requires the extraction of the cracks from the background and this rule is perfectly suitable for this task. Therefore, we are using the *subtractive rule* in this paper, to extend thinnings to grey scale images.

Geodesic attributes and the barycentric diameter are introduced in the two following sections. They will be used afterwards to build new attribute thinnings.

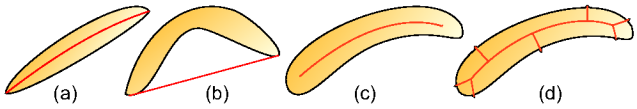


Fig. 2 Illustration of possible definitions of the length of an object. (a) and (b): the length of the segment between its ends points. (c) and (d), the measurement of its skeleton. These definitions are not always suitable.

3 Geodesic attributes

Geodesic attributes can be defined in both continuous and discrete domains. In this paper, the computations are done in $\mathcal{Z}^2$, with the 8-connectivity.

3.1 Geodesic diameter

Lantuéjoul and Maisonneuve, in [9] asked the question: "What is the length of an object?" The first idea is to measure the length of the segment connecting its end points, Fig. 2(a). However, this definition is not satisfactory, since this segment is not always included in the object, Fig. 2(b). Moreover, the definition of the end points is not trivial. Another possible measurement is the cardinal of the set of points of a homothetic skeleton of the object, Fig. 2(c). However, even though the skeleton is included in the object, it is not continuous in the sense that a slight modification of the shape can introduce important variations to the skeleton (and to its length), Fig. 2(d). Moreover, the skeleton does not necessarily span over the entire object (the skeleton of a disk is its centre).

We sum up below the exposition by Lantuéjoul and Maisonneuve and we present the generalisation of the geodesic diameter, proposed by Soille in [28].

Let X be an object (i.e. a connected component) and let $x, y \in X$. Path connectivity implies that there exists some paths between these two points, Fig. 3 (a). Among them, those with the minimal length are the *geodesic arcs* from x to y within X, Fig. 3 (b). By definition, they have all the same *geodesic length* written $d_X(x, y)$. For a given x belonging to X, we can compute the length of the longest geodesic arcs starting at x:

$$l_x(X) = \sup_{y \in X} d_X(x, y), \tag{6}$$

which corresponds to the distance to the farthest points from x, within X (for the geodesic distance).

We can compute $l_x(X)$ for all $x \in X$. Any point x of X where $l_x(X)$ reaches its global minimum is a *geodesic centre* of X. On the contrary, any couple of points where $d_X(x, y)$ reaches its global maximum are the *geodesic extremities* of X.

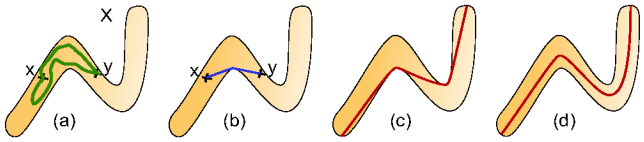


Fig. 3 (a) Two paths between x and y; (b) geodesic arc between these two points; (c) longest geodesic arc of X, whose length is the geodesic diameter of X: (d) generalised geodesic distance, longest geodesic arc.

The *geodesic diameter* of X is defined as the length of its longest geodesic arc (Fig. 3 (c)):

$$L(X) = \sup_{x \in X} l_x(X), \tag{7}$$

which corresponds to the largest geodesic distance between any two points of X:

$$L(X) = \sup_{x,y \in X} d_X(x,y). \tag{8}$$

The geodesic diameter gives, in our opinion, a nice estimation of the intuitive length of an object. It has, in addition, three advantages [9]: it is a general definition valid for every object; it is continuous, since a small change in the shape of the object (excluding any change of topology) will cause, at most, a small change of the measure of the geodesic diameter; finally, the computation of $L(X)$ leads to other attributes, such as the geodesic elongation and the geodesic tortuosity.

Soille in [28], has extended these notions with the generalised geodesic distance. First, he uses the Euclidean distance inside the object, inverts this distance map and finally, computes propagations with the geodesic time [29] from all the boundary points of the object. Therefore, the geodesic diameter tends to follow the medial axis of the object, like in Fig. 3 (d). This increases the accuracy of the measurement, since it measures the same length before and after bending a fibre. In practice though, very similar results are obtained for the detection of thin objects like cracks (see Fig. 8 for a comparison of both methods on a fingerprint image), but the computation times are higher ($\times 2$ at least, table 2). Therefore, we will not use a generalised geodesic distance to compute the length of the objects, unless high precision is required.

It should also be noted that the geodesic distance, and therefore the geodesic diameter, are defined in the same way in a n-dimensional space; their generalisation to higher dimensions is straightforward.

3.2 Geodesic elongation

The geodesic diameter gives a satisfactory definition of the length of an object but it does not tell much about its shape. By combining it with the area $S(X)$ of X, we obtain some information on its elongation. The geodesic elongation, introduced by Lantuéjoul and Maisonneuve [9], is computed as follows:

$$E(X) = \frac{\pi L^2(X)}{4S(X)} \tag{9}$$

The longer and narrower the object is, the higher is the elongation. The lower bound is reached with the disk[1]. Note that this definition can naturally be generalised to higher dimensions [20].

3.3 A new geodesic attribute: the geodesic tortuosity

We propose a new feature, derived from the geodesic diameter: the *geodesic tortuosity*. The pair of points $\{x, y\}$ is a pair of geodesic extremities of X if, and only if $d_X(x,y) = L(X)$. Note that some objects may have more than one pair of geodesic extremities (e.g. a disk). Let $\Xi(X) = \{\{x_0, y_0\}, \{x_1, y_1\}, \ldots\}$ be the set of geodesic extremities of X. Let $L_{Eucl}(X)$ denote the minimal Euclidean distance between geodesic extremities:

$$L_{Eucl}(X) = \min_{\{x,y\} \in \Xi(X)} \|x, y\|. \tag{10}$$

We define the geodesic tortuosity as the ratio between the geodesic diameter and $L_{Eucl}(X)$:

$$T(X) = \frac{L(X)}{L_{Eucl}(X)}. \tag{11}$$

In the case of the two-dimensional Euclidian space, the geodesic tortuosity of convex objects is equal to one.

3.4 Geodesic attributes properties and comments

Provided a rotation-invariant metric is used for the distances, e.g. the $\mathcal{L}_2$, all these attributes are rotation invariant in a Euclidean space; the geodesic elongation and tortuosity attributes are moreover scale invariant.

One can derive other attributes from the geodesic diameter, such as the geodesic circularity, which is the inverse of the geodesic elongation: $C(X) = \frac{1}{E(X)}$. It is also scale and rotation invariant. It reaches its upper bound 1 with the disk. Moreover, we can combine two attributes together to emphasise different structures. For instance, the product between the geodesic diameter and the geodesic tortuosity emphasises long and tortuous structures: $LT(X) = L(X)T(X)$.

[1] The normalisation factors are such that in a two-dimensional Euclidian space, the geodesic elongation of a disk is equal to 1

However, the computation of the geodesic diameter is computationally intensive and we are interested in efficient implementation of the thinnings based on geodesic attributes. Therefore, we introduce in the next section the *barycentric diameter* – a new attribute, approximating the geodesic diameter, but much faster to compute.

4 Barycentric diameter

The aim of the barycentric diameter is to replace the geodesic diameter by a similar measure with a lower complexity. We first describe the available algorithms for computing the geodesic diameter. Then, we introduce the new attribute.

4.1 Review on the computation of the geodesic diameter

The direct approach to compute the geodesic diameter of an object X consists of computing, for each point of the border, the geodesic distance within X to all other points of X (starting from one pixel, this operation is called a *propagation*). The highest geodesic distance computed, is the geodesic diameter. The timings of this algorithm are high and depend on both the area of the object and the number of boundary pixels. We notice that there are many redundant propagations, leading to an inefficient implementation. Schmitt [24] showed that it is enough to consider a subset of the border points as starting points for the propagations. However, despite the important speed-up thus achieved, still too many propagations remain to compute.

Maisonneuve and Lantuéjoul designed an efficient parallel algorithm to compute the geodesic diameter in a hexagonal grid [9]. Let X be a non-porous connected component (without "holes"), and Y the set of border points of X. Using a particular propagation in the hexagonal grid, starting from Y, the algorithm gives the geodesic diameter in a single propagation. However, this algorithm requires that X is not porous. Otherwise, the propagation never stops, turning infinitely around the holes in X. This characteristic makes the algorithm inadequate for our application. Indeed, a group of cracks can represent a porous connected object, e.g. Fig. 10(a).

Given that we have not found a method to efficiently compute the geodesic diameter, we propose what we consider a convenient - an efficient - approximation of the geodesic diameter: the barycentric diameter.

4.2 Definition of the barycentric diameter

For any point x of X, we can compute the length $l_x(X)$ of the longest geodesic arc starting at x in a single propagation. Could this value replace $L(X)$? Is it a good approximation? How to choose a convenient x?

Let us consider a maximal geodesic arc of X, and let y_1 and y_2 be its extremities. This means that $d_X(y_1, y_2)$ is equal to $L(X)$. Given that the geodesic distance is a distance [9], thanks to the triangular inequality we have, for any x belonging to X:

$$d_X(y_1, y_2) \leq d_X(y_1, x) + d_X(x, y_2). \tag{12}$$

Moreover, by definition $l_x(X)$ is larger than both $d_X(y_1, x)$ and $d_X(x, y_2)$, and smaller than $L(X)$, therefore we finally obtain:

$$L(X) \geq l_x(X) \geq \frac{L(X)}{2}. \tag{13}$$

Thus, the relative error $(L(X) - l_x(X))/L(X)$ obtained when approximating $L(X)$ with $l_x(X)$ is smaller than 50%.

In order to improve this approximation, it is tempting to iterate the propagation, starting this time from the farthest points from x. If Y is the subset of X containing the farthest points from x:

$$Y = \{y \mid \ y \in X,\ d_X(x, y) = l_x(X)\}, \tag{14}$$

we can introduce the iterated maximal geodesic distance starting from x, defined as:

$$l_x^2(X) = \sup_{y \in Y} l_y(X). \tag{15}$$

One can easily show that:

$$L(X) \geq l_x^2(X) \geq l_x(X) \geq \frac{L(X)}{2}, \tag{16}$$

and more generally that $l_x^n(X)$ converges, $\forall n \in \mathcal{Z}^+$:

$$L(X) \geq l_x^n(X) \geq l_x^{n-1}(X) \geq \frac{L(X)}{2}. \tag{17}$$

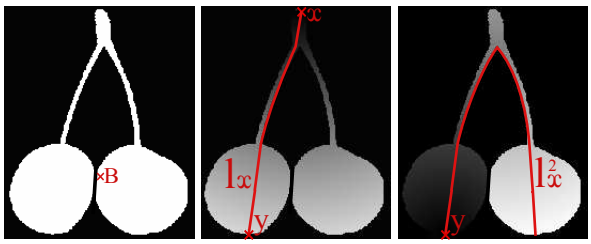
Nonetheless, as it will be seen below, the limit is not necessarily equal to $L(X)$. In practice though, we observe that the convergence is fast, and that going beyond $l_x^2(X)$ is not interesting. We will come back to this point at the end of the section. Fig. 4 provides an example where $l_x^2(X)$ gives a much better approximation than $l_x(X)$, and where further iterations do not improve the result.

In the following, the approximations of $L(X)$ will be based on $l_x^2(X)$. But we still have to choose x.

Does the choice of x within X have an influence on the quality of the approximation? Experiments have shown that indeed it is the case. Several strategies have been tested:

Table 1 Summary of the relative error between the geodesic diameter and 5 approximations of this geodesic diameter. These statistics are computed on 51400 binary shapes on total. The automatic generation of these shapes is presented in the Appendix.

Methods		L_{Bar}	$L_{BarNearest}$	$L_{GeoCentre}$	$L_{GeoCentreNearest}$	L_{Random}
Convex objects	Mean (%)	**0.24**	**0.24**	0.47	0.56	1.92
	Std (%)	0.85	**0.84**	1.41	1.53	3.36
	Max (%)	**10.04**	11.20	15.9	17.24	22.59
Pixel aggregation	Mean (%)	0.43	**0.40**	0.74	0.85	2.68
	Std (%)	1.09	**1.03**	1.58	1.71	3.59
	Max (%)	10.76	**9.27**	12.87	12.87	24.59
Ball aggregation	Mean (%)	**0.22**	**0.22**	0.23	0.59	1.21
	Std (%)	**0.72**	1.26	1.52	2.31	4.53
	Max (%)	**7.17**	12.75	13.84	16.85	28.83
Random walk	Mean (%)	**0.13**	0.28	0.24	0.53	1.31
	Std (%)	**0.66**	1.03	1.08	1.61	3.27
	Max (%)	**9.97**	21.7	21.48	21.48	29.96
Perlin noise	Mean (%)	**0.12**	0.16	0.21	0.40	0.76
	Std (%)	**0.74**	0.80	1.21	1.50	2.61
	Max (%)	10.87	**10.14**	16.55	13.61	31.90
Database MPEG7	Mean (%)	**0.25**	0.43	0.34	0.58	0.66
	Std (%)	**1.12**	1.93	1.44	1.91	2.38
	Max (%)	20.36	22.31	**19.16**	19.21	30.11
All Objects	Mean (%)	**0.23**	0.29	0.37	0.58	1.42
	Std (%)	**0.86**	1.49	1.37	1.76	3.29
	Max (%)	**20.36**	22.31	21.48	21.48	31.90



(a) Input (b) First propagation step (c) Second propagation step

Fig. 4 A black and white cherry: illustration of the barycentric diameter. Point B is barycentre of the CC and the starting point of the first propagation (x) is the farthest point from B. Then, the first propagation gives l_x and the second propagation, starting from y, leads to a much better approximation of the geodesic diameter: l_x^2

- When x is one of the farthest points from the barycentre of X, we obtain what we call the barycentric diameter $L_{Bar}(X)$. Note that the barycentre of X does not necessarily belong to X, so we use the Euclidean distance when looking for these farthest points. By taking one of the farthest points from the barycentre, we suppose that it will be close to an extremity of X (even though it is not always the case for non-convex shapes). Alternatively, instead of taking one of the farthest points from the barycentre, we can take one of the closest, thus obtaining another approximation: $L_{BarNearest}(X)$.
- A geodesic centre of X can also be used as reference point, instead of the barycentre. We obtain then two other approximations, $L_{GeoCentre}(X)$ and $L_{GeoCentreNearest}(X)$. Note that this strategy is only proposed for comparison; in practice, computing the geodesic centre is computationally intensive.
- Finally, we have also considered as starting point the first point of the object found with a raster scan, called $L_{Random}(X)$.

Therefore, five methods are available to approximate the geodesic diameter with only two propagation steps. To compare them, we apply them to six different sets of binary objects and we compute the geodesic diameter as well as the approximations defined above. For each object, we store $L(X)$ and $l_x^2(X)$ and we finally compute some statistics for each database : the mean relative error $(L(X) - l_x^2(X))/L(X)$, its standard deviation, and its maximum.

Five sets correspond to realisations from a random connected component model. Each one contains 10000 realisations. We give in the Appendix further explanations on the automatic generation of these objects. The sixth set is a standard database of 1400 binary objects (MPEG7 CE Shape-1 Part B database[2]).

Table 1 summarises the results. A first general observation concerns the fact that relative errors are, in practice, much lower than the 50% theoretical limit. All mean errors are in fact smaller than 1%. Even the maximal error is far from reaching the theoretical limit. Note also that the barycentric diameter achieves the smallest

[2] Database available in `www.imageprocessingplace.com/root files V3/image databases.htm`

error for most sets of objects: in average, the error is at least reduced by 20% compared to the other methods.

The maximal relative error on all 51400 objects obtained with the barycentric diameter is only 20%. Can the 50% relative error theoretical limit be reached? In Fig. 5 we exhibit an example, where $L_{Bar}(X)$ can be arbitrarily close to $\frac{L(X)}{2}$. Note also that this counter-example proves that $l_x^n(X)$ does not necessarily converge towards $L(X)$. Indeed, in this case the limit is $L_{Bar}(X)$, reached after only one iteration.

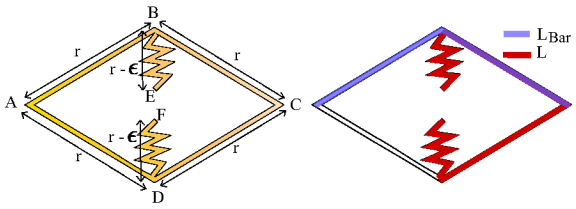


Fig. 5 Example of an object X, where the barycentric diameter does not converge towards the geodesic diameter. The geodesic diameter is equal to the distance $d_X(E,B) + d_X(B,C) + d_X(C,D) + d_X(D,F) = 4r - 2\epsilon$ whereas the barycentric diameter is equal to $d_X(A,B) + d_X(B,C) = 2r$. The relative error is : $\lim_{\epsilon \to 0} \frac{L - L_{Bar}}{L} = \lim_{\epsilon \to 0} \frac{r-\epsilon}{2r-\epsilon} = 0.5$

Concerning the convergence speed of the different approximations, we note that in practice, the barycentric diameter shows also the best performance, as illustrated in Fig. 6. This figure presents the convergence rate of the different approximations on the objects of the MPEG7 database. Note also that these curves show that, as stated previously, the second iteration brings an important improvement, but that further steps do not seem necessary.

We observe that a high relative error is reached when the propagation starts from a geodesic centre and converges towards another geodesic centre. Now, using the farthest point from the barycentre usually avoids a geodesic centre as starting point of the propagation. This observation provides us with some clues to understand why the barycentric diameter is, in practice, close to the geodesic diameter.

To conclude this section, the barycentric diameter appears to be, in practice, a very good and efficient replacement of the geodesic diameter. Hereafter, geodesic elongations and geodesic tortuosities are computed using L_{Bar} instead of L. In the next section, new thinnings, based on geodesic attributes, are introduced.

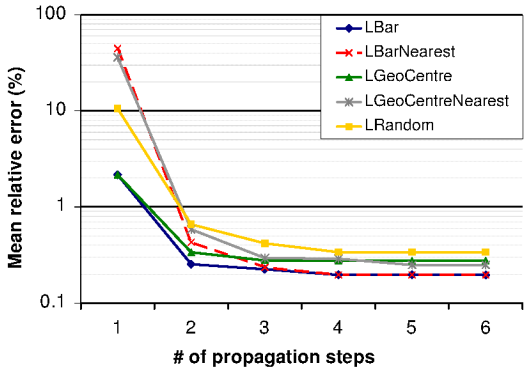


Fig. 6 Convergence of the geodesic diameter approximations. The relative error (in % and in log scale) is considerably reduced with 2 propagations (from l_x to l_x^2). However increasing the number of iterations beyond 2 does not bring a huge improvement on the accuracy of the measurement. Therefore, we choose l_x^2 with x being the farthest point from the barycentre. These curves are computed using all the objects of the MPEG7 database.

5 Geodesic attribute thinnings

To the extent of the authors knowledge, this is the first time that the geodesic diameter, its approximation L_{Bar}, as well as the derived attributes (geodesic elongation and tortuosity) are used to build attribute thinnings.

We recall that ρ_χ stands for the attribute thinning using criterion χ. We will consider criteria such as $L(X) \geq \lambda$, where λ is some real value, chosen according to the application.

5.1 Results on binary images

Fig. 7 is a toy example containing thin, tortuous objects, and other non-thin objects. We want to separate these structures with the following criteria:

- Suppress objects whose barycentric diameter value is smaller than 80 pixels and smaller than 120 pixels (second column);
- Suppress objects which are not elongated, e.g. whose geodesic elongation is smaller than 4 and smaller than 7 (third column);
- Suppress all non tortuous objects, i.e. whose geodesic tortuosity is smaller than 1.3 and smaller than 2 (fourth column);
- Suppress all objects having a circularity measurement smaller than 0.4 and smaller than 0.6 (fifth column).

We can observe through this simple experiment that geodesic attribute thinnings are intuitively parametrised, and provide an interesting tool to classify objects.

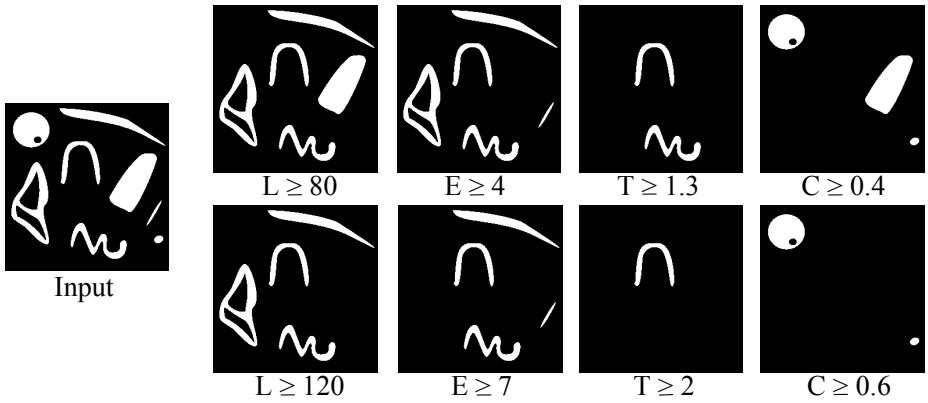


Fig. 7 Filtering with geodesic attributes criteria: (first column) initial image, (second column) barycentric diameter thinnings, (third column) geodesic elongation thinnings, (fourth column) geodesic tortuosity thinnings and (fifth column) geodesic circularity thinnings.

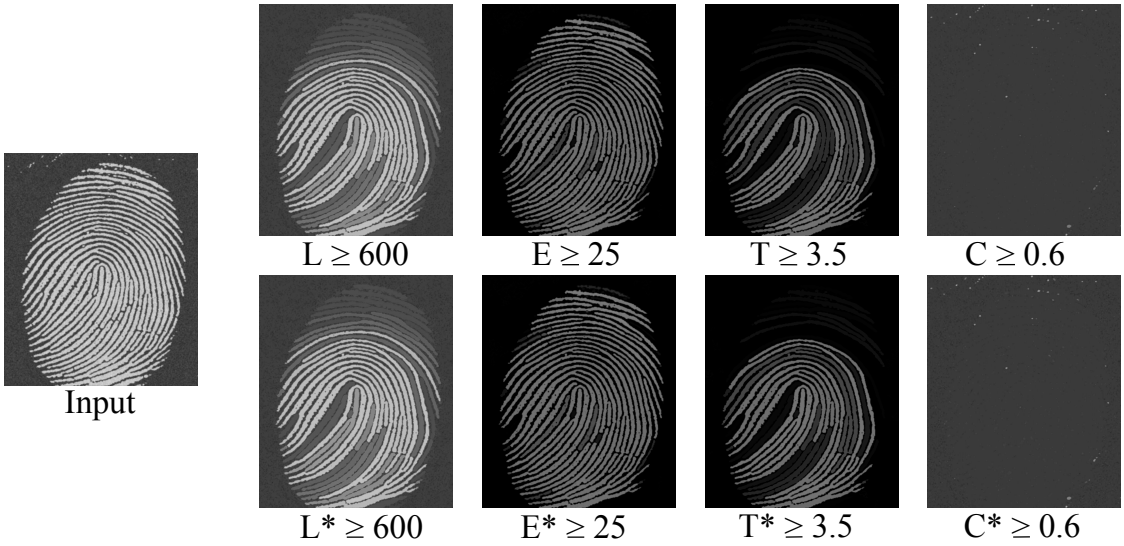


Fig. 8 Illustration of the thinnings based on geodesic attributes on a grey scale image. The first line uses classical geodesic distance and the second line presents the results of thinnings based on the generalised geodesic distance [28], written with a $*$ (e.g. L^*, E^*, T^* and C^*)

5.2 Grey level images

We present some results on the application of geodesic attribute thinnings to a grey scale image. Fig. 8 shows a fingerprint with long and tortuous structures, and the resulting images after application of thinnings with length, elongation, tortuosity and circularity criteria, using the subtractive rule. Different structures are enhanced and we can make the distinction between long, elongated or tortuous structures with these filters. Finally, the circularity attribute naturally erases most structures from the image.

In section 3, we have presented both the classical geodesic distance [9] and the generalised geodesic distance [28]. We compare, in Fig. 8, thinnings with both methods and we note the similarity of the results; the main visible difference corresponds to a fingerprint ridge around the center of the tortuosity image. The generalised geodesic distance is more accurate, but the computation times are longer (see Tab. 2). According to whether one prefers speed or accuracy, one may choose the classical or the generalised geodesic distance.

6 Results

First, we propose an application[3] where geodesic attribute thinnings are particularly adapted. Fig. 9(a) shows the image of a DNA molecule acquired with an electron microscope[4]. These structures are long, thin,

[3] A demonstration version is available `http://cmm.ensmp.fr/~morard/DemoGeoThinnings.html`

[4] Image from the Institute for Molecular Virology. University of Wisconsin - Madison `http://www.biochem.wisc.edu/faculty/inman/empics/dna-prot.htm`

and extremely tortuous. In order to evaluate the results we have visually compared them with two other methods, classically used for this sort of application:

1. The supremum of openings (Fig. 9(b), 60-pixel long segments oriented every 2 degrees) only preserves straight parts, and fails with tortuous structures. Note that the length λ is very low compared to the other methods. Longer segments (100-pixel long) would completely remove the molecule. A smaller length would preserve the background noise.
2. The path opening, see Heijmans et al. [5], (Fig. 9(c), length 160 pixels), tolerates some tortuosity. It yields better result than the previous method. The noise is considerably reduced but some parts of the molecule are also discarded. This operator is not able to follow the entire structure, and therefore underestimates its length.
3. The barycentric diameter thinning (Fig. 9(d), size 600 pixels) yields a better result, since the molecule is correctly extracted. It relaxes any constraints on the tortuosity, and offers a better flexibility than path openings.
4. The geodesic elongation thinning (Fig. 9(e), elongation superior to 50), filters out all the noise and offers a very efficient detection. Any other non elongated structures are similarly deleted.
5. The geodesic tortuosity thinning (Fig. 9(f), tortuosity superior to 2.5), filters out every structure that is not tortuous. Hence, this molecule is easily extracted with this tool.

These attribute thinnings have been used in an industrial non-destructive material-inspection application to detect long, narrow and randomly tortuous cracks. Fig. 10 offers a crack enhancement example with the same operators. Similar conclusions on the results can be pointed out: geodesic thinnings yield the best detection. The tortuosity thinning (Fig. 10(f)) enhances the crack as well as some noise in the background image. However, the noise has a lower grey level value than the crack and a simple threshold suffices to extract the crack.

7 Algorithm, practical considerations and optimisation

We describe the algorithm used to build geodesic attribute thinnings. We compare it with the algorithm previously used by the authors [13], where the geodesic diameter is computed exhaustively from the contour points. Furthermore, we propose a comparison with path openings [5].

7.1 Geodesic attribute thinnings algorithm

Attribute filters are often implemented using a tree representation of the image called *max-tree* (see e.g. Salembier et al. [23]). The max-tree creation relies on a recursive flooding procedure starting from the lowest pixel in the image. Its worst-case time complexity is quadratic $\mathcal{O}(N \times M)$, with M the cardinal of the set of values and N the number of pixels of the image, see [15]. However, the worst case is rare and in practice, Salembier *et al.* algorithm is faster than the algorithm of Najman and Couprie [15] for 8 bits images, even if the construction of max-tree is done in quasi linear time with the size of the image, using an union find approach. Other efficient implementations exist, see e.g. Ngan et al. [16].

A max-tree structure is adapted for attributes that can be *updated* each time a new pixel is aggregated to a CC. This is not the case of the barycentric diameter which needs to be *recomputed* and needs to access all the pixels of a connected component. This requires some modifications of the max-tree algorithm and prohibits the recursion. The algorithm Najman and Couprie [15] maintains several tree-like structures at a time and accessing the pixels is uneasy either.

Instead, we start with the relief completely submerged by water, and let the water progressively sink. As soon as appears the first (global) maximum, its connected component is progressively reconstructed and tested on χ. When other local maxima appear, they are not process yet and we reconstruct all connected component at lower threshold sets that are supersets of the global maxima (see Fig. 11 component 1). Finally, other local maxima are processed in the same way but the aggregation stops when a CC is already processed (Fig. 11 components 2 and 3).

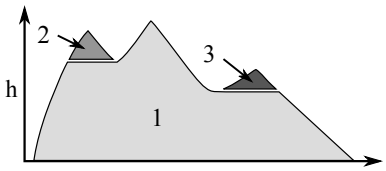
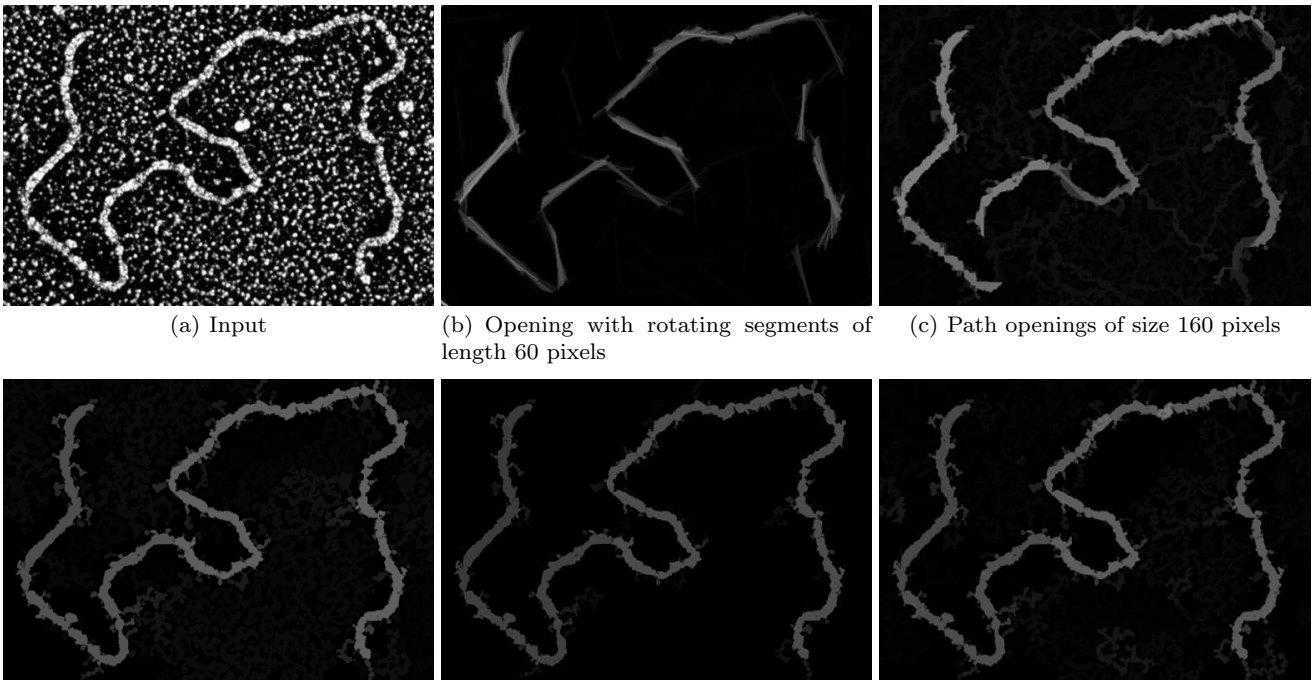


Fig. 11 Illustration of the algorithm principle for a one-dimensional signal with three local maxima. The components in light grey (marked by one) are analysed first, followed by the components marked by 2 and 3.

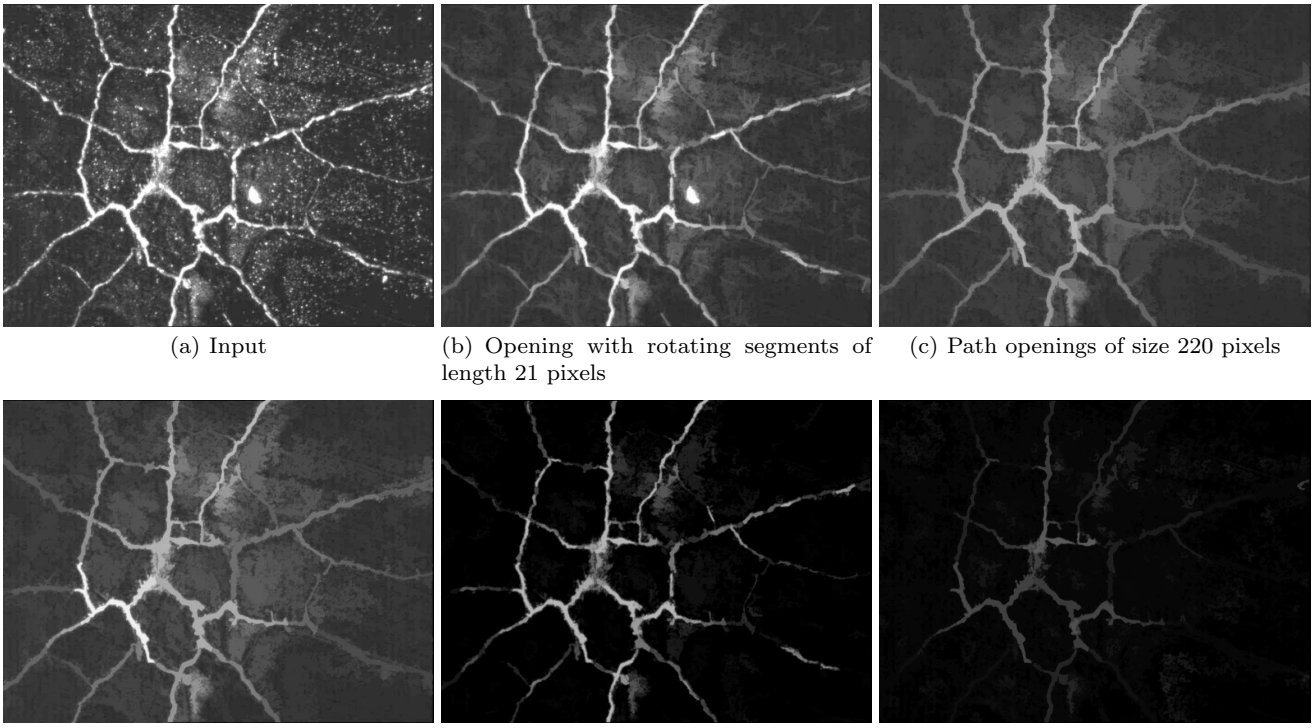
The Alg. 1 simulates this principle. It uses a priority queue $HQueue$ that supports operations (modifying the content), and queries (not modifying the content): the operation $Push(x, p)$ inserts an element x with a given priority p, and $(x, p) = Pop()$ retrieves the currently highest-priority element x and its priority p. The



(a) Input (b) Opening with rotating segments of length 60 pixels (c) Path openings of size 160 pixels

(d) Barycentric diameter thinning (subtractive rule) ($L \geq 600$) (e) Geodesic elongation thinning (subtractive rule) ($E \geq 50$) (f) Geodesic tortuosity thinning (subtractive rule) ($T \geq 2.5$)

Fig. 9 DNA molecule extraction: we compare five different methods to detect this thin structure. We note that path openings and segment opening completely underestimate the length of this tortuous structure. By opposition, geodesic attribute thinnings easily enhance this molecule and correctly evaluate its length.



(a) Input (b) Opening with rotating segments of length 21 pixels (c) Path openings of size 220 pixels

(d) Barycentric diameter thinning (subtractive rule) ($L \geq 220$) (e) Geodesic elongation thinning (subtractive rule) ($E \geq 4$) (f) Geodesic tortuosity thinning (subtractive rule) ($T \geq 1.1$)

Fig. 10 Crack detection: to detect these thin structures, we use five different methods. Geodesic attribute thinnings yield the best enhancement.

queries: $p = Prio()$ returns the currently highest priority, and $Empty()$ tests the emptiness of $HQueue$.

In the beginning, the output image g is initialised by the input f, and the local maxima of f are extracted by a function $FindLocalMax$ (code lines 4 and 5). The list $LocalMax$ contains one pixel per every maximum, and is sorted by decreasing order of the level.

The *for* cycle (lines 8 to 28) runs over all the local maxima to process. Line 9 reads from $LocalMax$ the position p of next maximum to process. The *repeat-until* cycle (lines 14 to 28) simulates the decreasing water level to progressively reconstruct the associated component associated, and tests it against χ (line 26). The extraction of the component uses two auxiliary flag arrays $State : D \to \mathcal{Z}$ and $Level : D \to V$ (lines 2 to 3), initialised to 0 for all image pixels (lines 6 to 7). State receives -1 whenever a pixel is pushed in the queue (line 11), and later the index of the local maximum to which component it belongs to (line 18), cf. Fig. 11.

We initialise the region growing algorithm by pushing p in the hierarchical queue with the priority $f(p)$ (line 10), by marking this pixel (lines 11 to 12) and by initialising CC to the empty set. Then, a region-growing algorithm (code lines 15 to 25) extracts a connected component CC to which the pixel p belongs to. Criteria χ based on the barycentric diameter L_{Bar} need to access the connected component as a set. When the region growing at this level is finished (line 25), CC contains all pixels of this connected component such that $f(x) \geq h$, $\forall x \in CC$. Then, if the test $\chi(CC)$ is false, we apply the subtractive rule by removing the contrast of this connected component for all pixels belonging to CC. The function *contrast(CC)* returns the difference between the level h and the highest grey level of the neighbours of CC.

The same process is repeated until all regional maxima are processed.

The Fnct. L_{Bar} illustrates how to compute the barycentric diameter when χ is based on this attribute. First, it computes the barycentre $\overline{p}$ of CC, i.e. the mean of the coordinates (code line 1). Then, it searches one of the farthest point from the barycenter with the Euclidean distance (line 2). Finally, it computes twice the geodesic distance inside CC (see Eq. 6, and lines 3 and 4), and returns the result l^2_χ of the second iteration.

Note : Algorithm 1 is set up to compute the subtractive rule. However, if one wants to apply the direct rule, only two modifications of the algorithm are required: the output image is initialised with 0 and if $\chi(CC)$ is true, we write the CC in g by taking the maximum value between h and g.

Algorithm 1: $g \leftarrow$ GeoAttrThinning (f, χ)

Input: $f : D \to V$ - input image
χ - criterion
Result: $g : D \to V$ - output image

```
   /* Declare auxiliary variables                    */
 1 HQueue = {};          /* Init HQueue to empty */
 2 State : D → Z;
 3 Level : D → V;
   /* Initialize                                     */
 4 g = f;
 5 LocalMax = FindLocalMax(f); /* get sorted list
   of local maxima */
 6 State(:) = 0;
 7 Level(:) = 0;
   /* Run over all local maxima                      */
 8 for i = 1 ... Card(LocalMax) do
 9     p = LocalMax(i);
10     HQueue.Push(p, f(p));
11     State(p) = −1 ;              /* pixel enqueued */
12     Level(p) = f(p);
13     CC = {};
       /* Simulate decreasing water level            */
14     repeat
           /* Extract a CC to test on χ              */
15         repeat
16             (q, h) = HQueue.Pop();
17             CC = CC ∪ {q};
18             State(q) = i;
19             foreach n ∈ Neighbour(q) do
20                 if f(n)>Level(n) and State(n)≠i then
21                     hmin = min(h, f(n));
22                     HQueue.Push(n, hmin);
23                     State(n) = −1 ; /* pixel enqueued */
24                     Level(n) = hmin;
25         until h < HQueue.Prio() ;
26         if not χ(CC) then
27             g(CC) = g(CC) − contrast(CC)
28     until HQueue.Empty() ;
```

Function $l^2_x \leftarrow L_{Bar}(CC)$

Input: CC - A connected component
Result: l^2_x: Barycentric diameter of CC

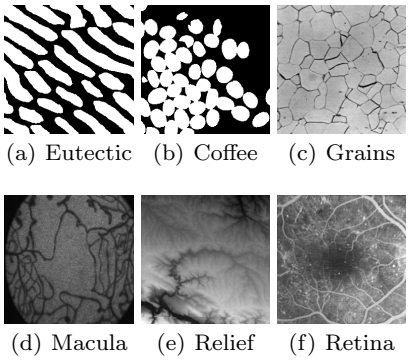
```
1 b = p̄, p ∈ CC;
2 x = arg max_{y∈CC} ||y, b||;
3 x^1 = arg sup_{y∈CC} d_CC(x, y);
4 x^2 = arg sup_{y∈CC} d_CC(x^1, y);
5 l^2_x = d_CC(x^1, x^2);
6 return l^2_x
```

7.2 Complexity

The complexity analysis is split into two parts; the complexity of the CC's extraction (Alg 1) and the complexity of the computation of the attribute.

Table 2 Running times for different images of size 256x256 for a geodesic diameter thinning or thickening for $\lambda = 20$. See section 7.4 for details. Timings are in seconds. Laptop computer: Intel Core2 Duo T7700 @ 2.40GHz

	From [13]		From this paper	
Images	$L_{no\ opt}$	L_{opt}	L_{Bar}	L^*_{Bar}
Coffee	36	0.079	0.0028	0.0062
Eutectic	37	0.015	0.0029	0.0069
Grains	3650	5.4	0.041	0.328
Macula	3850	2.7	0.031	0.220
Relief	1024	0.99	0.022	0.139
Retina	1820	1.56	0.024	0.136



(a) Eutectic (b) Coffee (c) Grains

(d) Macula (e) Relief (f) Retina

Fig. 12 Images used to build table 2. They have all the same size: 256×256 pixels.

First, we detail the complexity of Alg. 1. The function *FindLocalMax* fills the array *LocalMax* using a region growing process with a queue, and sorts the array *LocalMax* with a linear complexity radix sort [17], which gives a complexity of $\mathcal{O}(N)$ for this function ($N = Card(D)$). Finally, from the local maxima, we use a relief emerging approach using a hierarchical queue, which adds $log(M)$ in the complexity. Every local maximum is processed sequentially and the worth case is a highly nested signal. Then, for the worst signal, a pixel is analysed $N/4$ times in average, which leads to a complexity of $\mathcal{O}(N^2\ log(M))$, just like the algorithm presented by Breen and Jones [4]. We note however, that the computation times are very far from this upper bound[5] and in practice, we observe a linear evolution of the computation time with the number of pixels.

All geodesic attributes are based on the length measurement. Therefore, all attributes are computed with the complexity of Fcnt 2.

For a connected component having n pixels, the computation of the attribute value is done in 4 steps. The barycenter and the farthest point both require one scan over all pixels, giving $\mathcal{O}(n)$. Then, two propagations are performed to get the barycentric diameter. For the propagation step, according whether $\mathcal{L}_1$ or $\mathcal{L}_2$ is used for the distance, one needs a queue (or a priority queue, resp.) giving a complexity of $\mathcal{O}(1)$ (or $\mathcal{O}(log(m)$ resp.) per pixel, with m the mean number of elements in the priority queue.

This is an improvement in comparison with the complexity of the geodesic diameter. With a propagation from all boundary points, the complexity is $\mathcal{O}(n^2)$ (the worst case is a thin line, which needs n propagations). With the Schmitt's method, the number of boundary points is reduced however, the worth case still requires $\sqrt{n}$ propagations (for a disc).

[5] We made an experiment on more than 100 natural images showing that a pixel is processed less than two times in average.

7.3 Optimisations

The first and probably, the most important optimisations is that we can stop the propagation step whenever the criterion value is reached. This acceleration is only available for the barycentric diameter, or the geodesic elongation and it does not change the complexity of the algorithm.

The acceleration also comes from other optimisations, which are computer's tricks:

- We compute simple attributes such as the barycentre or the area during the region growing process, in order to avoid useless scan of all the pixels of CC in Fnct. 2.
- We also compute the contrast of the CC during the region growing process.
- In Fnct. 2, we remove the distance map during the propagation step. Instead, we enqueue the distance value with its position whenever a pixel is stored in a queue or in the hierarchical queue.
- We manage our own queue, which is specialised for this algorithm. Therefore, we can remove all the elements from a queue in only one assignation.
- For 8 bits images, we can merge the buffers *State* and *Level*. Then, we use the first byte to store the buffer *Level* and the other bytes for the variable *State*. Then, we get the value of two variables in only one access of the memory.

These optimisations does not change the complexity of this algorithm, however, the timings have been reduced by a very large factor, as shown in the next section.

7.4 Timings

We compare in this section all the benefits brought by the optimisations, accelerations presented in section 7

and the barycentric diameter (section 4). Table 2 collects the timings for the images of Fig. 12, where we compare four methods: a thinning based on the geodesic diameter with no optimisation ($L_{no\ opt}$), the geodesic diameter with the stop of the propagation when the front wave is larger than λ (L_{opt}), the barycentric diameter with all the discussed optimisations (L_{Bar}) and the barycentric diameter using the generalised geodesic distance (L^*_{Bar}).

Timings have been reduced by a large factor between thinnings using L_{opt} and L_{Bar} (around 60 in average for these 6 images). Moreover, using a generalised geodesic distance increases the computation time by at least a factor of 2, in comparison with L_{Bar}. The overhead is introduced by the computation of the distance map for every connected component. This distance map also cuts down the benefits of the optimisation, where the propagation step is stopped if the criterion is fulfilled during the propagation.

7.5 Efficiency comparison with path openings

Path openings [5] have been developed to solve applications similar to those treated in this paper. In section 6, it has been shown that they provide indeed interesting results, even if they cannot detect objects as tortuous as those detected by geodesic attributes thinnings. For the comparison to be complete, we now compare their computational performance.

In Fig. 13, we plot a comparison between the thinnings based on geodesic attributes (barycentric diameter) and an efficient implementation of path openings [30,6]. The timings were computed on the image presented in Fig. 10(a). On this crack image (768x576 pixels), thinnings are faster than path openings and this tendency is the same for other images.

Finally, we can state that the resulting implementation of geodesic attribute thinnings is at least as efficient as other operators that have been designed with similar objectives, which makes them interesting for a large number of industrial applications.

8 Conclusion and future work

A new geodesic attribute is introduced, the barycentric diameter to approximate the costly geodesic diameter. We give the theoretical upper error bound of this approximation, and show that the typical error is negligible.

This, as well as other shape attributes, such as the geodesic length, elongation and circularity, is combined

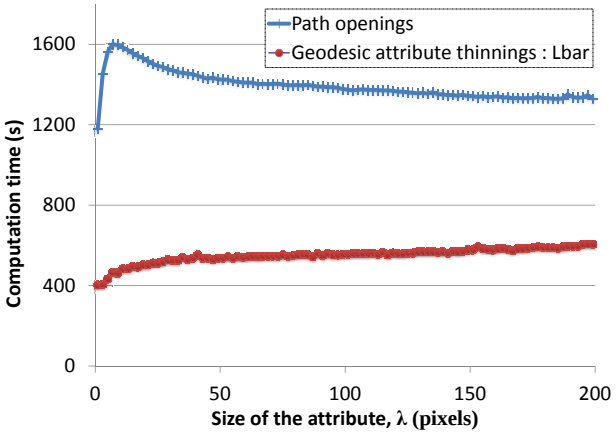


Fig. 13 Benchmark on a crack image (768x576 pixels, 8 bits) with regard to the size of the attribute, for path openings and thinnings based on the barycentric diameter with the subtractive rule.

with attribute thinnings to obtain a new family of operators. They are ideal for characterising long, elongated or tortuous objects. We show that their shape flexibility is superior to that of classical linear openings or path openings.

We use the subtractive filtering rule, proposed by Urbach and Wilkinson [32], and we provide an efficient algorithm for computing the resulting operators. We report competitive timings, which allow using these operators in time-critical, industrial applications. The geodesic attribute thinnings are not only more flexible than path openings, but also faster. Furthermore, geodesic attribute thinnings readily generalise to 3-dimensional images, by only changing the connectivity.

At present, geodesic thinnings fail to extract thin, curvilinear structures disconnected by noise. Indeed, their length will be underestimated. A possible solution shall consist in using a generalised connectivity, like the second generation connectivity introduced by Ouzounis and Wilkinson [18].

Future work will concern the generalisation of the proposed algorithm to efficiently compute other operators based on geodesic attribute thinnings. On the one hand, morphological pattern spectra [10] estimate the size and shape distribution of the searched structures. On the other hand, ultimate openings [1,7] extract structures with the highest contrast. Typically, these operators require the computation of a family of thinnings of increasing size. Using this algorithm, we can compute pattern spectra and ultimate thinnings within only one "relief emerging" process.

Acknowledgements This work was made possible thanks to the support of the "Pôle ASTech" and the "Pôle Nucléaire

de Bourgogne", and has been financed by the French "Département de Seine et Marne"

The authors are grateful to Ms Raviart, working in the "Centre des matériaux, MINES ParisTech", for help with the optical microscope.

The authors also wish to thank the anonymous reviewers for their comments.

References

1. Beucher, S.: Numerical residues. Image and Vision Computing **25**(4), 405–415 (2007)
2. Beucher, S., Blosseville, J.M., Lenoir, F.: Traffic spatial measurements using video image processing. Intelligent Robots and Computer Vision, Proc. SPIE **848**, 648–655 (1987)
3. Braga-Neto, U.M.: Alternating sequential filters by adaptive-neighborhood structuring functions. In: Mathematical Morphology and Its Applications to Image and Signal Processing, vol. 5, pp. 139–146. Maragos, Schafer and Butt, Atlanta (1996)
4. Breen, E.J., Jones, R.: Attribute openings, thinnings, and granulometries. Computer Vision and Image Understanding **64**(3), 377–389 (1996)
5. Heijmans, H., Buckley, M., Talbot, H.: Path openings and closings. Journal of Mathematical Imaging and Vision **22**(2), 107–119 (2005)
6. Hendriks, C.L.L.: Constrained and dimensionality-independent path openings. Image Processing, IEEE Transactions on **19**(6), 1587–1595 (2010)
7. Hernández, J., Marcotegui, B.: Ultimate attribute opening segmentation with shape information. Mathematical Morphology and Its Application to Signal and Image Processing pp. 205–214 (2009)
8. Lantuejoul, C., Beucher, S.: On the use of the geodesic metric in image analysis. Journal of Microscopy **121**(1), 39–49 (1981)
9. Lantuéjoul, C., Maisonneuve, F.: Geodesic methods in quantitative image analysis. Pattern Recognition **17**(2), 177–187 (1984)
10. Maragos, P.: Pattern spectrum and multiscale shape representation. Pattern Analysis and Machine Intelligence, IEEE Transactions on **11**(7), 701–716 (1989)
11. Maragos, P., Ziff, R.D.: Threshold superposition in morphological image analysis systems. Pattern Analysis and Machine Intelligence, IEEE Transactions on **12**(5), 498–504 (1990)
12. Matheron, G.: Random sets and integral geometry, vol. 1. Wiley New York (1975)
13. Morard, V., Decencière, E., Dokládal, P.: Geodesic attributes thinnings and thickenings. In: Mathematical Morphology and Its Applications to Image and Signal Processing, vol. 6671, pp. 200–211. Springer (2011)
14. Morard, V., Decencière, E., Dokládal, P.: Region growing structuring elements and new operators based on their shape. In: Signal and Image Processing, pp. 78–85. ACTA Press (2011)
15. Najman, L., Couprie, M.: Building the component tree in quasi-linear time. Image Processing, IEEE Transactions on **15**(11), 3531–3539 (2006)
16. Ngan, N., Dokládalová, E., Akil, M., Contou-Carrère, F.: Fast and efficient FPGA implementation of connected operators. Journal of Systems Architecture **57**(8), 778 – 789 (2011)
17. Nilsson, S.: Radix sorting and searching. Ph.D. thesis, Department of Computer Science, Lund University, Lund, Sweden (1996)
18. Ouzounis, G.K., Wilkinson, M.H.F.: Mask-based second-generation connectivity and attribute filters. Pattern Analysis and Machine Intelligence, IEEE Transactions on **29**(6), 990–1004 (2007)
19. Ouzounis, G.K., Wilkinson, M.H.F.: Hyperconnected attribute filters based on k-flat zones. Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**(2), 224–239 (2011)
20. Parra-Denis, E.: Analyse morphologique 3D de particules de forme complexes: application aux intermétalliques dans les alliages d'aluminium. Ph.D. thesis, Université Jean Monnet de Saint-Etienne (2007)
21. Perlin, K.: Course in advanced image synthesis. In: ACM SIGGRAPH Conference, vol. 18 (1984)
22. Salembier, P., Oliveras, A., Garrido, L.: Motion connected operators for image sequences. In: European Signal Processing Conference, vol. 96, pp. 1083–1086 (1996)
23. Salembier, P., Oliveras, A., Garrido, L.: Antiextensive connected operators for image and sequence processing. Image Processing, IEEE Transactions on **7**, 555–570 (1998)
24. Schmitt, M.: Des algorithmes morphologiques à l'intelligence artificielle. Ph.D. thesis, Ecole Nationale Supérieure des Mines de Paris (1989)
25. Serra, J.: Image analysis and mathematical morphology, vol. 1. Academic Press, London (1982)
26. Serra, J.: Image analysis and mathematical morphology, vol. 2 Theoretical Advances. Academic Press, London (1988)
27. Serra, J., Vincent, L.: An overview of morphological filtering. Circuits Syst. Signal Process. **11**(1), 47–108 (1992)
28. Soille, P.: Generalized geodesic distances applied to interpolation and shape description. Mathematical morphology and its applications to image processing pp. 193–200 (1994)
29. Soille, P.: Generalized geodesy via geodesic time. Pattern Recognition Letters **15**(12), 1235–1240 (1994)
30. Talbot, H., Appleton, B.: Efficient complete and incomplete path openings and closings. Image and Vision Computing **25**(4), 416–425 (2007)
31. Urbach, E.R., Roerdink, J.B.T.M., Wilkinson, M.H.F.: Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images. Pattern Analysis and Machine Intelligence, IEEE Transactions on **29**(2), 272–285 (2007)
32. Urbach, E.R., Wilkinson, M.H.F.: Shape-only granulometries and grey-scale shape filters. In: Proc. Int. Symp. Math. Morphology (ISMM), vol. 2002, pp. 305–314 (2002)
33. Vincent, L.: Grayscale area openings and closings, their efficient implementation and applications. In: First Workshop on Mathematical Morphology and its Applications to Signal Processing, pp. 22–27 (1993)
34. Wilkinson, M.H.F., Westenberg, M.A.: Shape preserving filament enhancement filtering. Medical Image Computing and Computer-assisted Intervention, lecture notes in computer science **2208**, 770–777 (2001)

Appendix: random connected component models

Many strategies exist to generate random binary shapes. The results of the comparison between the geodesic diameter and its approximations are linked to the method used to generate these shapes. Therefore, we use five different methods to have a high variety of objects. For each method, 5 realisations are presented in Fig. 15. The size of the support of these random shapes is a 500 by 500 pixels square.

Convex shape

A random number of points (between 10 and 100) are randomly and uniformly picked on D. The final connected component is the convex hull of these points.

Pixel aggregation

This method is used to generate relatively dense objects, which are almost convex. The set is initialised with a single point. At each iteration, a randomly chosen neighbour point is added to the set. The procedure is iterated a random number of times.

Ball aggregation

This method uses the same process as the pixel aggregation method, except that we iteratively aggregate a ball instead of a point to the set. The ball radius is chosen randomly between 5 and 40 pixels, for each ball. The generated shapes are much more complex than the shapes generated using the pixel aggregation method.

Random walk

We start from a ball in the centre of the domain. Then, we use a Brownian motion to choose the next location of the ball. At each iteration, the radius of the ball is chosen randomly.

Perlin noise

Perlin noise [21] is a procedural texture primitive. It has a pseudo random appearance that is highly controllable and multiscale. Fig. 14 provides an illustration of a realisation of this noise. By thresholding this image, we get a set of objects and we select the biggest CC of this set (Fig. 14). Some resulting objects can be very smooth, whereas others can have a high tortuosity.

Fig. 14 Object generation with Perlin noise. Left to right: Perlin noise, thresholding, selection of the biggest CC.

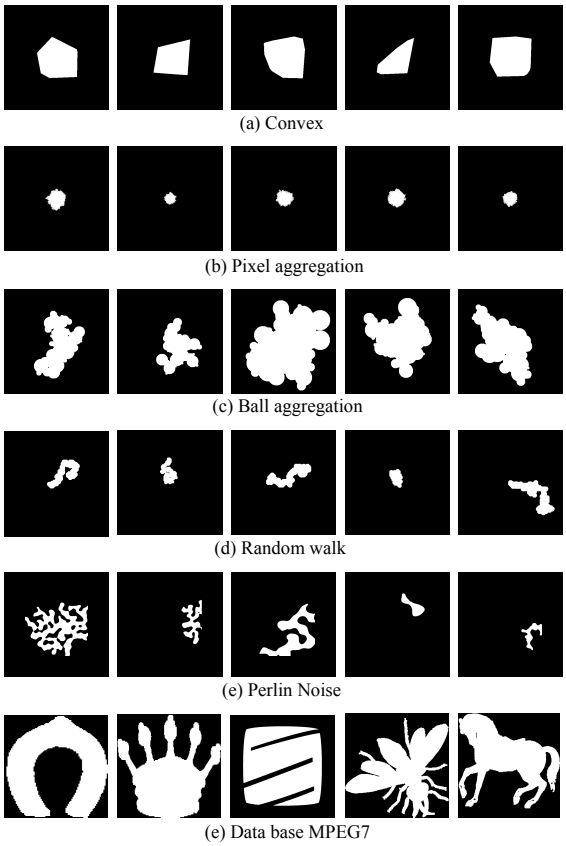


Fig. 15 Examples of random connected components, obtained with five different models.

Medical Image Analysis xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Medical Image Analysis

journal homepage: www.elsevier.com/locate/media

Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology

A. Dufour [a,b,1], O. Tankyevych [c,d,1], B. Naegel [a], H. Talbot [c], C. Ronse [a], J. Baruthio [b], P. Dokládal [e], N. Passat [a,*]

[a] Université de Strasbourg, LSIIT, UMR 7005, CNRS, France
[b] Université de Strasbourg, LINC, UMR 7237, CNRS, France
[c] Université Paris-Est, LIGM, UMR 8049, CNRS, France
[d] Université Paris-Est, LISSI, EA 3956, France
[e] Mines ParisTech, CMM, Fontainebleau, France

ARTICLE INFO

Article history:
Received 5 December 2011
Received in revised form 25 July 2012
Accepted 20 August 2012
Available online xxxx

Keywords:
Vessel filtering
Vessel segmentation
Mathematical morphology
3D angiography
Spatially variant morphology

ABSTRACT

In the last 20 years, 3D angiographic imaging has proven its usefulness in the context of various clinical applications. However, angiographic images are generally difficult to analyse due to their size and the complexity of the data that they represent, as well as the fact that useful information is easily corrupted by noise and artifacts. Therefore, there is an ongoing necessity to provide tools facilitating their visualisation and analysis, while vessel segmentation from such images remains a challenging task. This article presents new vessel segmentation and filtering techniques, relying on recent advances in mathematical morphology. In particular, methodological results related to spatially variant mathematical morphology and connected filtering are stated, and included in an angiographic data processing framework. These filtering and segmentation methods are evaluated on real and synthetic 3D angiographic data.

1. Introduction

The important rise of medical imaging during the 20th century, mainly induced by physics breakthroughs related to nuclear magnetic resonance and X-rays, has led to the development of imaging modalities devoted to visualise vascular structures. The analysis of such angiographic images is of great interest for several clinical applications. Initially designed to generate 2D data, these imaging modalities progressively led to the acquisition of 3D images, enabling the visualisation of vascular volumes.

However, such 3D data are generally quite large, being composed of several millions of voxels, while the useful vascular information generally represents less than 5% of the whole volume. In addition to this sparseness, the frequent low signal-to-noise ratio and the potential presence of artifacts (due to acquisition, patient movements, etc.) make the analysis of such images a challenging task. In order to assist practitioners in the use of such data (radiologists, clinicians, etc.), it is therefore necessary to design software tools enabling them to exploit as well as possible the relevant information embedded in these images.

One of the main ways to perform such a task is to develop filtering and/or segmentation methods, i.e., routines that enhance or extract the vessels from angiographic images. In particular, such methods are required to be as ergonomic as possible, for instance by providing user-friendly and time-saving interactive *modus operandi*.

Recently, several methodological works have been conducted in the field of mathematical morphology. Some of them, and especially those related to spatially-variant mathematical morphology and connected filtering, can be efficiently involved in the design of relevant tools for vessel filtering and segmentation from 3D angiographic data, especially Computed Tomography Angiography (CTA) and Magnetic Resonance Angiography (MRA). This article aims at presenting some of these new mathematical morphology concepts, and their applicative use in this complex field of medical image processing.

The remainder of this article is organised as follows. Section 2 proposes a synthetic state of the art related to mathematical morphology in medical image processing, and vessel segmentation from 3D angiographic data. Section 3 provides the background notions required to describe the methods developed in the sequel of the article. The next two sections represent the main contributions of this article. Section 4, which is an improved version of the conference articles (Tankyevych el al., 2009b,a), describes a vessel filtering method based on a hybrid strategy, merging both new spatially-variant mathematical morphology algorithms and

* Corresponding author. Tel.: +33 (0)3 68 85 44 96; fax: +33 (0)3 68 85 44 55.
E-mail addresses: alice.dufour@unistra.fr (A. Dufour), olena.tankyevych@u-pec.fr (O. Tankyevych), b.naegel@unistra.fr (B. Naegel), h.talbot@esiee.fr (H. Talbot), cronse@unistra.fr (C. Ronse), j.baruthio@unistra.fr (J. Baruthio), petr.dokladal@mines-paristech.fr (P. Dokládal), passat@unistra.fr (N. Passat).
[1] Equal first authors.

1361-8415/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.media.2012.08.004

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

derivative-based approaches. Section 5, which is an improved version of the conference articles (Passat and Naegel, 2011b; Dufour et al., 2011a), describes an example-based interactive vessel segmentation method relying on a component-tree-based technique. In particular, it describes a way to consider fuzzy examples and how the filtered images provided in Section 4 can be used as extensive masks for connectivity improvement. Section 6 describes and discusses experimental results related to vessel segmentation and filtering performed on angiographic phantom images and *in vivo* cerebral MRA data. Concluding remarks emphasising contributions and remaining challenges are proposed in Section 7.

2. State of the art

2.1. Mathematical morphology in medical imaging

Mathematical morphology is a well-established theory of non-linear, order-based image analysis (Serra, 1982; Najman and Talbot, 2010). It relies on basic operations (namely erosions, dilations, openings, closings), involving geometric patterns (structuring elements, or SEs for brief). These low-level SE-based operations made it possible to design some of the first image processing segmentation methods (e.g., for 2D vessel segmentation (Thackray and Nelson, 1993)), and remained frequently used in this domain (e.g., for 2D (Zana and Klein, 2001) and 3D vessel segmentation (Cline et al., 2000), or for skull stripping (Dogdas et al., 2005)).

Based on these basic mathematical morphology operations, higher-level image processing techniques have been developed and used in the context of medical image processing. Watersheds (Vincent and Soille, 1991) have been used for many applications, including 3D vessel segmentation (Passat et al., 2007), 3D vertebrae labelling (Naegel, 2007), 4D heart segmentation (Cousty et al., 2010), or 3D brain structure segmentation from newborn brain MRI (Gui et al., 2011). The grey-level hit-or-miss transform (Naegel et al., 2007b) has been also utilised, essentially in the field of 3D vessel segmentation (Naegel et al., 2007c; Bouraoui et al., 2010). Finally, connected filters (for a recent survey, see (Salembier and Wilkinson, 2009)) and especially those based on component-trees (described in Section 3.4) have been involved in several (bio)medical applications, including 3D vessel filtering and segmentation (Wilkinson and Westenberg, 2001; Urbach and Wilkinson, 2002; Caldairou el al., 2010), 3D brain structures segmentation (Dokládal et al., 2003), 2D melanocytic nevi segmentation (Naegel et al., 2007a), or interactive visualisation of 3D data (Westenberg et al., 2007).

The use of mathematical morphology in these techniques has been, in particular, motivated by the ability of the involved operators to efficiently integrate and model *a priori* knowledge enabling an efficient detection of the structures of interest with a wide range of usage policies (automated, semi-automated, knowledge-based and/or interactive ones).

2.2. Filtering and segmentation of 3D angiographic data

Filtering and segmentation of vascular structures (generally from MR and CT angiography) has been an active research field since the end of the 1980s (see, e.g., (Eichel et al., 1988; Kitamura et al., 1988) for pioneering works). These intensive efforts were motivated by the potential use of such segmentation results, e.g., for pathology detection and quantification, or for surgical planning. A complete state of the art is beyond the scope of this article. The reader may find up-to-date surveys on 3D angiographic segmentation in (Lesage et al., 2009b; Tankyevych et al., 2011).

Most image processing and analysis concepts have been considered in the development of 3D vessel segmentation methods. Non-exhaustively, one can cite: region-growing (Tizon and Smedby, 2002), deformable models (Lorigo et al., 2001; Descoteaux et al., 2008), statistical analysis (Chung et al., 2004; Sabry Hassouna et al., 2006), minimal path-finding (Li and Yezzi, 2007), vessel tracking (Flasque et al., 2001; Manniesing et al., 2007), differential analysis (Sato et al., 1998), or mathematical morphology (discussed in Section 2.1). Despite this wide range of methodological contributions, results provided by segmentation methods generally remain perfectible. The handling of under-segmentation (e.g., in the case of small vessels, of signal variation, or of partial volume effect) and over-segmentation (e.g., in the case of neighbouring with other anatomical structures, or of high intensity artifacts); robustness to image degradations (low signal-to-noise ratio); low computational cost; guarantee of termination and convergence, are all desirable properties that are not often satisfied together.

Consequently, a reasonable trend over the last few years has been to use synergies across methodologies. Indeed, hybrid vessel segmentation methods present a range of solutions for overcoming certain weaknesses of each method and combining their advantages. One of the most popular hybrid strategies is based on the combination of multi-scale differential analysis with deformable models, such as level-sets (Chen and Amini, 2004), B-spline snakes (Frangi et al., 1999) and maximum geometric flow (van Bemmel et al., 2003). Deformable methods combined with multi-scale statistical region-based analysis was proposed in (Hernandez and Frangi, 2007). Tracking strategies reinforced by gradient flux of circular cross-sections was considered in (Lesage, 2009a), while in (Friman et al., 2009) multiple hypothesis tracking was used with Gaussian vessel profile and statistical model fitting. In (Wong and Chung, 2007), a probabilistic method for axis finding was proposed within a minimal path finding strategy. We note finally that mathematical morphology has also been used in combination with other techniques, for instance in (Kobashi et al., 2001), watersheds and neural networks were combined, and in (Sun and Sang, 2008), multi-scale morphology was used together with Gabor wavelets.

An alternative way to improve vessel segmentation efficiency consists of injecting high-level guiding knowledge in the segmentation process. This can be achieved by designing vascular atlases devoted to explicitly guide segmentation tools (Passat et al., 2005; Passat et al., 2006). Also, instead of using atlases, it is possible to use segmentation examples, thus leading to the design of example-based segmentation processes. A last strategy is to directly take advantage of users skills in order to guide the segmentation process, thus leading to interactive methods. This last strategy however requires the interaction to be simple and quick, since medical experts generally cannot afford to spend much time with segmentation tasks.

These considerations motivate, in particular, the new filtering and segmentation methods described in the next sections. Indeed, in Section 4, a hybrid strategy, mixing differential analysis and mathematical morphology is proposed for 3D vessel filtering. In Section 5, an interactive and example-based segmentation method, relying on connected filtering, is described. These two methods take advantage of recent methodological advances in mathematical morphology. We show that they can also be conveniently fused, leading to improved results.

3. Background notions

This section includes the formal notions required to correctly describe the filtering and segmentation approaches developed in Sections 4 and 5. In particular, Section 3.1 provides general notations, while Sections 3.2, 3.3, 3.4 focus on differential analysis and mathematical morphology concepts, respectively.

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

3.1. Notations

Let $E = \prod_{i=1}^{3}[[0, d-1]]$ (with $d \in \mathbb{N}^*$) be a subset of $\mathbb{Z}^3$. The set E provides a (discrete) model for the part of $\mathbb{R}^3$ where the considered 3D images are defined. An element of E (called point, or voxel), is noted $x = (x_1, x_2, x_3)$.

Let $V \subset \mathbb{Z}$ be a finite interval of integers. Without loss of generality, we can assume that $V = [[0, M-1]]$. The set V provides a (discrete) model for the value space of the considered 3D images. An element of V (called value, or grey level), is noted v.

A (grey-level) image I is defined as a function

$$\begin{vmatrix} I : E \to V \\ \quad x \mapsto v \end{vmatrix} \tag{1}$$

and we note $I : E \to V$ or $I \in V^E$. The set E is called the support of I. By abuse of notation, a (binary) image $B : E \to \{0, 1\}$ will also be considered as the set $B^{-1}(\{1\}) = \{x \in E \mid B(x) = 1\}$.

The thresholding function at value $v \in V$ is defined by

$$\begin{vmatrix} \lambda_v : V^E \to 2^E \\ I \mapsto \{x \in E \mid v \leqslant I(x)\} \end{vmatrix} \tag{2}$$

The cylinder function of support $X \subseteq E$ and of value $v \in V$ is defined by

$$\begin{vmatrix} C_{X,v} : E \to V \\ x \mapsto \begin{cases} v & \text{if } x \in X \\ 0 & \text{otherwise} \end{cases} \end{vmatrix} \tag{3}$$

The impulse function at point $p \in E$ and of value $v \in V$ is defined by

$$\begin{vmatrix} i_{p,v} : E \to V \\ x \mapsto \begin{cases} v & \text{if } x = p \\ 0 & \text{otherwise} \end{cases} \end{vmatrix} \tag{4}$$

In particular, we have $i_{p,v} = C_{\{p\},v}$.

3.2. Hessian-based analysis

One of the main challenges in image analysis is to design operators that are translation, rotation and scale-invariant. Translation invariance is satisfied by all convolution kernels, by definition. Rotation invariance can be guaranteed either by using rotation-invariant kernels or when the preferred direction is fixed relatively to the image. Scale invariance can be satisfied by derivatives of Gaussian filters. Linear combinations of derivatives of Gaussian filter kernels constitute, in particular, the basic feature detectors within linear scale-space theory (Lindeberg, 1994).

The Hessian matrix $\mathscr{H}$ is obtained from the Gaussian second derivative analysis of a 3D image F at each voxel in the principal directions

$$\mathscr{H} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \frac{\partial^2 F}{\partial x_1 \partial x_3} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & \frac{\partial^2 F}{\partial x_2 \partial x_3} \\ \frac{\partial^2 F}{\partial x_3 \partial x_1} & \frac{\partial^2 F}{\partial x_3 \partial x_2} & \frac{\partial^2 F}{\partial x_3^2} \end{bmatrix} \tag{5}$$

This Hessian matrix $\mathscr{H}$ can be decomposed into three eigenvalues, λ_1, λ_2 and λ_3 (with $|\lambda_1| \leqslant |\lambda_2| \leqslant |\lambda_3|$) associated to three eigenvectors $\mathbf{e_1}, \mathbf{e_2}$ and $\mathbf{e_3}$. When λ_1 is close to zero and much smaller than λ_2 and λ_3, the locally characterised shape is a line-like (bright) structure, e.g., a vessel in angiographic data. Its orientation is then given by $\mathbf{e_1}$ ($\mathbf{e_2}$ and $\mathbf{e_3}$ then form a basis for the plane orthogonal to the line-like bright structure).

When appropriately designed and applied at multiple scales, combinations of the three eigenvalues, often called *vesselness* function, should give the strongest response at one particular scale corresponding to the plate-, blob-like and/or tubular objects (Sato et al., 1998; Frangi et al., 1999). Hereafter, and in the remainder of this article, we consider the vesselness function proposed in (Frangi et al., 1999) (which has been experimentally assessed as the most robust in the current applicative context). For a 3D grey-level image, observed at a point x, and a scale σ (directly linked to the standard deviation of the considered Gaussian kernel), this vesselness function v is formulated as follows

$$v(x, \sigma) = \begin{cases} 0 & \text{if } \lambda_2 > 0 \text{ or } \lambda_3 > 0 \\ \left(1 - \exp\left(\frac{-R_A^2}{2\alpha^2}\right)\right) \cdot \exp\left(-\frac{R_B^2}{2\beta^2}\right) \cdot \left(1 - \exp\left(\frac{-S^2}{2c^2}\right)\right) & \text{otherwise} \end{cases} \tag{6}$$

with

$$\begin{cases} R_A = \frac{|\lambda_2|}{|\lambda_3|} \\ R_B = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}} \\ S = \sqrt{\sum_{j=1}^{3} \lambda_j^2} \end{cases} \tag{7}$$

where R_A differentiates between planar and line-like objects, R_B differentiates blob-like ones, and S accounts for the intensity difference between objects and background. The parameters α, β and c influence the sensitivity of the filter to the corresponding measures.

As stated in Formula (6), the filter can be applied at different scales, which then provide results in a large range of object sizes. After normalisation, the maximal vesselness value is selected for each point x. The corresponding scale then provides an estimate of the object width.

3.3. (Spatially-variant) mathematical morphology

We introduce hereafter some notions of mathematical morphology and spatially-variant mathematical morphology (SVMM).

Definition 1 (*Adjunction*). Let $\mathscr{L}$ and $\mathscr{M}$ be two complete lattices (i.e., partially ordered sets $(X, \leqslant)$, such that every subset S of X has an infimum in X denoted $\bigwedge S$, and a supremum in X denoted $\bigvee S$). Two operators $\delta : \mathscr{L} \to \mathscr{M}$ and $\varepsilon : \mathscr{M} \to \mathscr{L}$ form an *adjunction* (ε, δ) if and only if for all $x \in \mathscr{L}$ and all $y \in \mathscr{M}$, we have

$$(\delta(x) \leqslant y) \iff (x \leqslant \varepsilon(y)) \tag{8}$$

From these notions, it is then possible to introduce the basic operators of morphology, namely, dilation, erosion, opening and closing.

Definition 2 (*Erosions, dilations, openings and closings*). With the same hypotheses as in Definition 1, the operator δ commutes with the supremum operator $\bigvee$ and is called a *dilation*, while the operator ε commutes with the infimum operator $\bigwedge$ and is called an *erosion*. Moreover, the operator $\gamma = \delta\varepsilon$ is called an *opening*, and the operator $\varphi = \varepsilon\delta$ is called a *closing*.

We have the following general properties of openings and closings.

Property 3. *Let $\mathscr{L}$ and $\mathscr{M}$ be two complete lattices. Let γ and φ be the opening and closing induced by an adjunction for $\mathscr{L}, \mathscr{M}$. Let $x, x' \in \mathscr{L}$ and $y, y' \in \mathscr{M}$. Then we have*

$$(\textit{Idempotence}) \begin{cases} \gamma\gamma = \gamma \\ \varphi\varphi = \varphi \end{cases} \tag{9}$$

$$(\textit{Increasingness}) \begin{cases} (x \leqslant x') \Rightarrow (\varphi(x) \leqslant \varphi(x')) \\ (y \leqslant y') \Rightarrow (\gamma(y) \leqslant \gamma(y')) \end{cases} \tag{10}$$

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

$$((Anti-)extensivity)\begin{cases} x \leqslant \varphi(x) \\ \gamma(y) \leqslant y \end{cases} \tag{11}$$

These properties are useful from both algebraic and practical points of view. Indeed, the behaviour of morphological operators is well defined. It is therefore possible to design new operators (e.g., gradients, top-hats) exploiting differences between these operators.

We now assume that $\mathscr{L} = \mathscr{M}$. In the "grey-level" case (e.g., in flat morphology (Heijmans, 1991)), $\mathscr{L}$ is the family of grey-level images V^E equipped with the point-wise partial order on functions $\leqslant$. Let $B \subseteq E$ be a binary set, also called *structuring element* (SE). Let $\delta_B, \varepsilon_B : V^E \to V^E$ be the dilation and the erosion induced by B. The dilation of the impulse function $i_{p,v}$ (with $p \in E$ and $v \in V$) is defined as $\delta_B(i_{p,v}) = C_{B_p,v}$, with $B_p = \{x + p \mid x \in B\}$. From this expression, we derive the classical definitions of the dilation and erosion for a function $f : E \to V$

$$\delta_B(f)(x) = \bigvee_{p \in B} f(x - p) = \bigvee_{p \in \check{B}_x} f(p) \tag{12}$$

$$\varepsilon_B(f)(x) = \bigwedge_{p \in B} f(x + p) = \bigwedge_{p \in \check{B}_x} f(p) \tag{13}$$

where $\check{B} = \{-p \mid p \in B\}$ is the *transpose* of B.

In the (more general) case of SVMM (Bouaynaya et al., 2008; Bouaynaya and Schonfeld, 2008), the involved SE B is often denoted as *structuring function* and is actually defined as $B : E \to 2^E$. Consequently, $B(x)$ is the structuring element considered at point $x \in E$. The *transpose* of a structuring function B, still noted $\check{B} : E \to 2^E$, is now defined, for all $x \in E$ by

$$\check{B}(x) = \{y \in E \mid x \in B(y)\} \tag{14}$$

Here, the dilation of the impulse function $i_{p,v}$ is defined as $\delta_B(i_{p,v}) = C_{B(p),v}$. From this expression, we derive the spatially-variant definition of the dilation and erosion for a function $f : E \to V$

$$\delta_B(f)(x) = \bigvee_{p \in \check{B}(x)} f(p) \tag{15}$$

$$\varepsilon_B(f)(x) = \bigwedge_{p \in B(x)} f(p) \tag{16}$$

With the definitions given above, the standard and SV morphological erosion and dilation form an adjunction. Then, we can define, in both cases, the morphological opening and closing, as

$$\gamma_B = \delta_B \varepsilon_B \tag{17}$$

$$\varphi_B = \varepsilon_B \delta_B \tag{18}$$

The transpose $\check{B}$ of B is used in Formula (15) for the computation of δ_B. In addition to being computationally expensive, it can, under some conditions, be of larger extent than any of the B functions, although if the family of B is bounded, so is $\check{B}$. This becomes computationally problematic for implementing filters based on adjunctions of dilations and erosions in order to compute a closing or an opening. However, by considering the following – equivalent – alternative writing of Formula (15)

$$\delta_B(f) = \bigvee_{p \in E} C_{B(p),f(p)} \tag{19}$$

it turns out that the computation of $\delta_B(f)$ is performed independently of $\check{B}$, then leading to the following result.

Proposition 4. *The SV dilation and the SV adjunct erosion can be computed with the same algorithmic cost $\mathscr{O}(MN)$ where $N = |E|$, and $M = \mathscr{O}(\max_{x \in E}\{|B(x)|\})$.*

We note that for this property to be true, it is required that the *structuring function* defining B must remain constant when computing Formula (17).

3.4. Component-trees

Let us consider a given connectivity on $\mathbb{Z}^3$, for instance the standard 6- or 26-connectivity (Kong and Rosenfeld, 1989) (in the sequel of this article, some alternative morphological connectivities will also be considered). For a given binary image B defined on E, we denote by $\mathscr{C}[B]$ the set of the connected components (i.e., the maximal connected sets) of B with respect to this connectivity. Note that a grey-level image $I \in V^E$ can be expressed as

$$I = \bigvee_{v \in V} \bigvee_{X \in \mathscr{C}[\lambda_v(I)]} C_{X,v} \tag{20}$$

Let $\mathscr{K} = \bigcup_{v \in V} \mathscr{C}[\lambda_v(I)]$ be the set of the connected components generated by the thresholdings of I at all values $v \in V$. The Hasse diagram of the partially ordered set $(\mathscr{K}, \subseteq)$ is a tree (i.e., a connected acyclic graph), and more especially a rooted tree, the root of which is the supremum $\lambda_0(I) = E$. This tree is called the *component-tree* of I.

Definition 5 (*Component-tree*). Let $I \in V^E$ be a grey-level image. The *component-tree* of I is the rooted tree $T = (\mathscr{K}, L, R)$ such that:

$$\mathscr{K} = \bigcup_{v \in V} \mathscr{C}[\lambda_v(I)] \tag{21}$$

$$L = \{(X, Y) \in \mathscr{K}^2 \mid (Y \subset X) \wedge (\forall Z \in \mathscr{K},\ Y \subseteq Z \subset X \Rightarrow Y = Z)\} \tag{22}$$

$$R = \sup(\mathscr{K}, \subseteq) = \lambda_0(I) = E \tag{23}$$

The elements of $\mathscr{K}$ (resp. of L) are the *nodes* (resp. the *(oriented) edges*) of T. The node R is the *root* of T. For any $N \in \mathscr{K}$, we set $ch(N) = \{N' \in \mathscr{K} \mid (N, N') \in L\}$; $ch(N)$ is the set of the *children* of N.

An example of component-tree defined for a 2D image is illustrated on Figs. 1 and 2. Component-trees can be used to develop image processing/analysis procedures based on filtering or segmentation strategies (Jones, 1999). Such procedures generally consist of determining a subset $\widehat{\mathscr{K}} \subseteq \mathscr{K}$ among the nodes of the component-tree $T = (\mathscr{K}, L, R)$ of a considered image $I : E \to V$. (In the case of filtering, some alternative strategies devoted to visualisation have also been proposed to preserve all the nodes, by only modifying their associated grey-level value (Westenberg et al., 2007).)

When performing segmentation, the (binary) resulting image $B \subseteq E$ is defined as the union of the nodes of $\widehat{\mathscr{K}}$, i.e., as

$$B = \bigcup_{X \in \widehat{\mathscr{K}}} X \tag{24}$$

In this context, determining the nodes to be preserved is a complex issue, which can be handled by considering attributes (Urbach el al., 2005) (i.e., qualitative or quantitative information related to each node) to characterise the nodes of interest. An alternative solution, based on an example-based paradigm, can also be considered. We describe such a strategy in Section 5.

4. Vessel filtering: a morpho-Hessian approach

4.1. Motivation

3D angiographic imaging modalities (e.g., MRA, CTA) provide a detailed visualisation of vascular networks up to the resolution of the generated data. However, the small size and complexity of vascular structures, coupled to noise, acquisition artifacts, and blood signal heterogeneity (especially signal discontinuity) make the analysis of such data a hard task, thus justifying intensive efforts devoted, in particular, to filtering (i.e., vessel enhancement).

Vessel filtering has often been considered *via* the use of Gaussian second derivative analysis, and more specifically *via* the analysis of the Hessian matrix (see Section 3.2). This approach enables

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

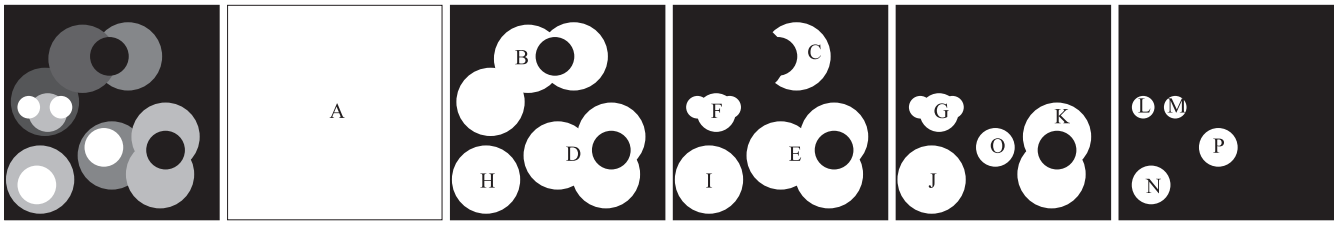


Fig. 1. (a) A grey-level image $I : E \to V = [[0, 4]]$ (from 0, in black, to 4, in white). (b–f) Threshold images $\lambda_v(I)$ (in white) for v varying from 0 (b) to 4 (f).

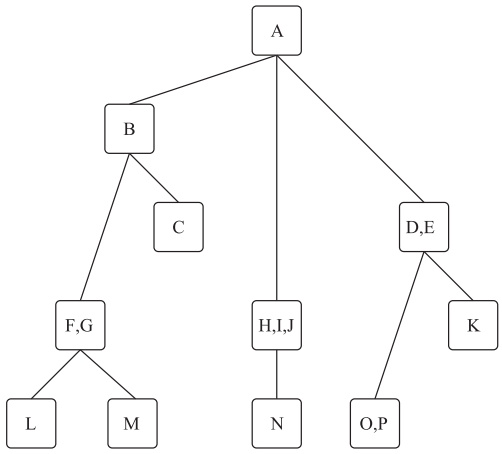


Fig. 2. The component-tree T of I (see Fig. 1a). The letters (A–P) in nodes correspond to the associated connected components in Fig. 1b–f.

the detection of thin objects and their principal directions, at different scales. Compared with first derivative (i.e., gradient) approaches, the Hessian matrix can also capture some shape characteristics. In particular, the eigenvalues of the Hessian matrix can be combined into vesselness functions in order to discriminate such shapes (Lorenz et al., 1997; Sato et al., 1998; Frangi et al., 1999; Krissian et al., 2000).

An alternative to these linear approaches is proposed by spatially-variant mathematical morphology (see Section 3.3). The algorithms defined in this framework are formulated with the purpose of filtering images in a way that depends on the location in the dataset (Maragos and Vachier, 2009; Verdú-Monedero and Angulo, 2008; Dokládal and Dokládalová, 2008). Such filtering techniques provide solutions for reducing noise and reconnecting vessels despite signal decrease/loss, by taking advantage of local shape knowledge.

The combination of linear and non-linear techniques is motivated by several facts. Hessian analysis is robust and fast for object direction detection as well as multiple scales, whereas orientation analysis using purely mathematical morphology methods would require directional sampling, which is prohibitive in 3D. Conversely, for reconnection and noise reduction, anisotropic diffusion, that has been previously used together with Hessian analysis, e.g., in (Manniesing et al., 2006), requires several iterations and is subject to convergence issues, while a spatially-variant closing or opening converges in a single iteration.

This combination of simple modeling and quick convergence comes in contrast to more geometric models, such as found in (Gooya et al., 2008). Indeed in this approach a level-set formulation of a directional diffusion process is proposed, which is shown to be able to reconnect vessels at long ranges. However this comes at the price of a long convergence time and a parametric model.

4.2. Methodology

In this section, we propose a hybrid (morpho-Hessian) filtering method devoted to 3D angiographic image analysis. It especially aims to retrieve the smallest (low-intensity) vessels and correctly reconnect them. Based on the analysis of the Hessian, we distinguish vessel-like objects from the background and compute their local orientation. Then, we perform a spatially-variant morphological closing (assuming that vessels are bright structures on a dark background) according to these local directions.

A first version of this work was proposed in Tankyevych et al. (2009a,b). In this article, we have improved it by regularizing the orientation field (see Section 4.2.3).

4.2.1. Outline of the method

The method takes as input:

- a 3D grey-level angiographic image $I_{in} : E \to V$, e.g., a MRA or CTA image.

Depending on the quality of the input (1T vs. 3T for instance), it can be useful to denoise this input image. We have used the bilateral filter in order to better preserve contrast and edges (Tomasi and Manduchi, 1998). The proposed filter is fully automatic. It is however parametric, in order to allow the user to choose the size of the vessels to detect, and the gap length between vessels to reconnect. The process, visually summarised in Fig. 3, is divided into three main steps:

1. The Hessian matrix of I_{in} is computed for each point of E at different scales, resulting into a vesselness image I_{ves} and leading to define three images corresponding to the principal vessel directions: I_{x_1}, I_{x_2} and I_{x_3} (see Section 4.2.2).
2. From a thresholded version of I_{ves} and the direction images $I_{x_1}, I_{x_2}, I_{x_3}$, and with the help of morphological thinning and dilation, dense and regular vessel direction fields $I^d_{x_1}, I^d_{x_2}$ and $I^d_{x_3}$ are obtained (see Section 4.2.3).
3. A family of structuring elements, composed of segments of fixed length, oriented with respect to $I^d_{x_1}, I^d_{x_2}$ and $I^d_{x_3}$, is involved in an SV morphological closing operation carried out on I_{in} (see Section 4.2.4).

The method finally provides as output:

- a 3D grey-level filtered image $\mathscr{F}(I_{in}) : E \to V$ of the input image I_{in}, such that $I_{in} \leqslant \mathscr{F}(I_{in})$ (i.e., $I_{in}(x) \leqslant \mathscr{F}(I_{in})(x)$ for any $x \in E$), and enabling in particular to reconnect relevant high intensity line-like structures of I_{in}.

4.2.2. Step 1: vessel detection

Given a set of different scales $\mathscr{S}$ enabling to characterise vessels among different radii, the image I_{in} is first convolved with a Gaussian kernel $G(x, s) = (2\pi s^2)^{-N/2} \cdot \exp\left(-\frac{|x|^2}{2s^2}\right)$ at each scale $s \in \mathscr{S}$. For

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

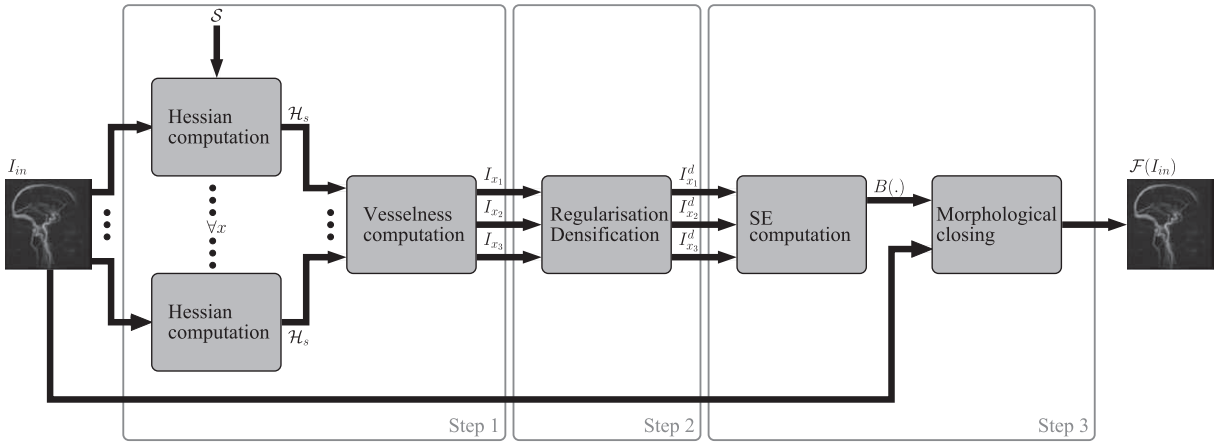


Fig. 3. Visual outline of the filtering method described in Section 4.2. Step 1: vessel detection (see Section 4.2.2). Step 2: directional field regularization (see Section 4.2.3). Step 3: vessel reconnection (see Section 4.2.4).

each point $x \in E$, its Hessian matrix $\mathscr{H}_s$ is then computed. The eigen form of this matrix, i.e.,

$$\mathscr{H}_s = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \tag{25}$$

enables the computation of a score in x from the vesselness function defined in Formula (6). The maximal score among the scales of $\mathscr{S}$ is chosen for each point x as its best response $v_{max}(x) = \max_{s\in\mathscr{S}}\{v(x,s)\}$.

The associated basis vectors $(\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3})$ (forming the basis of the eigen form of $\mathscr{H}_s$) are assumed to define the orientation of the characterised shape in x. In particular, the vectors $\mathbf{e_1}$ are stored in three images as the principal directions (along the principal axes): $I_{x_1}, I_{x_2}, I_{x_3} : E \to [-1, 1]$, defined, for all $x \in E$, by

$$\mathbf{e_1}(x) = I_{x_1}(x).\mathbf{e_{x_1}} + I_{x_2}(x).\mathbf{e_{x_2}} + I_{x_3}(x).\mathbf{e_{x_3}} \tag{26}$$

where $(\mathbf{e_{x_1}}, \mathbf{e_{x_2}}, \mathbf{e_{x_3}})$ is the canonical basis of $\mathbb{R}^3$.

4.2.3. Step 2: directional field correction

In order to propagate objects outside their own boundary with the spatially-variant morphological closing, it is necessary to benefit from a direction vector field that extends beyond these boundaries. In our case, the directional information is necessary only as far as the dilation can reach. In the literature, Verdú-Monedero and Angulo (2008) have proposed to use a gradient vector flow (Xu and Prince, 1998) to this end. Similarly, Deguchi et al. (2002) have proposed to use the structure tensor within a diffusion scheme to obtain a dense direction field.

However, simply diffusing the directions obtained by second-order derivatives is not sufficient. Indeed, directions are typically unreliable at the end of tubular object segments. This is likely to cause problems in methods that use them in further procedures. To solve this, we *regularize* the direction vector field. Here, in order to obtain both an extended and regularized direction field, we have come up with a procedure based on a combination of morphological operations.

First, similar to (Ouzounis and Wilkinson, 2007) in 2D, we ignore directions when they are not strong. For this, as the expression of vesselness (see Formula (6)) expresses the probability of being a vessel for each point (thus varying between 0 and 1), the vesselness image I_{ves} is thresholded so that most of the vessel-like objects are preserved. By using the thresholded vesselness result I^t_{ves}, we ensure that we use the tubular objects as markers for direction field propagation. Then, we perform a binary morphological thinning of the direction field (I_{x_1}, I_{x_2} and I_{x_3}) guided by the thresholded vesselness result I^t_{ves} with a structuring element of fixed size. We then perform the adjunct dilation – solely of the direction field – guided by the thresholded vesselness result. This results into an extended and regularized direction fields $I^d_{x_1}, I^d_{x_2}$ and $I^d_{x_3}$. A schematic representation of the operation is illustrated in Fig. 4.

4.2.4. Step 3: vessel reconnection

In this last step, an SV morphological closing operation is performed over the image I_{in} with the aim of reconnecting vessels. First, a morphological dilation is applied with a structuring function $B : E \to 2^E$ (see Section 3.3), providing, for each $x \in E$, a structuring element $B(x)$ centred on x, of fixed length, and oriented according to $\mathbf{e_1}(x)$. The (discrete) direction $\mathbf{e_1}$ of $B(x)$ is approximated from the images of regularized direction fields, $I^d_{x_1}, I^d_{x_2}$ and $I^d_{x_3}$, by defining a discrete segment.

The SE-based adjunct dilation, resulting in the image $\delta_B(I_{in})$, is followed by the adjunct erosion ε_B. Both computations (δ_B and ε_B), whose results are formally defined by Formula (19) and (16) respectively, then provide the final filtering result $\mathscr{F}(I_{in}) = \varphi_B(I_{in})$ with a low algorithmic cost (see Proposition 4). Also note that this processing ensures idempotence, guaranteeing that the filter obeys morphological rules. This methodology is validated in Section 6.

5. Interactive vessel segmentation: a component-tree based approach

5.1. Motivation

Most 3D vessel segmentation techniques are designed to be globally automated (except, sometimes, for initialization and/or termination, or for the determination of parameters). Automation is generally justified by the difficulty for medical experts to spend too much time in guiding such segmentation methods. In contrast, such automatic methods do not take enough advantage of the user's expertise (e.g., clinician, radiologist, etc.), thus leading to possible segmentation errors (in addition to a frequently high computational cost).

The recent rise of interactive segmentation in several application fields modifies this conception of vessel segmentation (McGuinness and O'Connor, 2010). Indeed, in contrast to standard automatic segmentation, interactive segmentation strongly relies on the user's skills. In particular, the user must generally initialize the process, by providing (background and/or object) markers which strongly influence the results (for instance in the case of

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

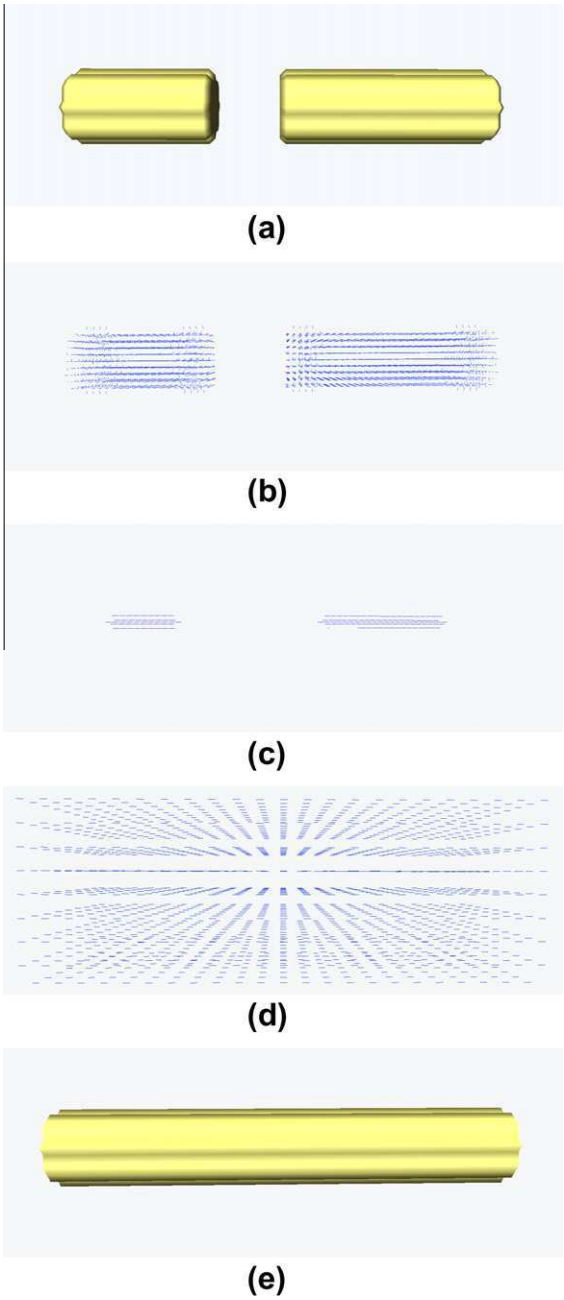


Fig. 4. Vector field regularization. (a) Original broken cylinder. (b) Orientation field in the broken cylinder. Observe they are incorrect at the extremities. (c) Restriction of the orientation to the thinning of the cylinder. (d) Propagation of the orientation field by guided dilation. (e) Filtered reconstructed cylinder by SV oriented closing.

watersheds (Vincent and Soille, 1991), graph-cuts (Boykov and Jolly, 2001) or binary partition tree algorithms (Salembier and Garrido, 2000)). Interaction also requires such methods to be very efficient, especially in terms of computational cost (see, e.g., (Westenberg et al., 2007)).

Such guidance is potentially time consuming for the user, especially in the case of 3D images. However, a compromise between automatic and interactive segmentation exists through the concept of example-based segmentation, which has been considered in several application fields before being applied to medical imaging (see, e.g., (Faisan et al., 2011)). Indeed, the use of segmentation examples leads to an automatic pre-segmentation of the image, which can be used as object markers.

The *component-tree* (see Section 3.4) is a graph-based structure which models some characteristics of a grey-level image by considering its binary level-sets obtained from successive thresholding operations (see, e.g., (Najman and Talbot, 2010), Chapter 7). It has been involved, in particular, in the development of morphological operators (Breen and Jones, 1996; Salembier et al., 1998), and used for designing segmentation procedures in several medical applications (see Section 2.1).

By definition, component-trees are particularly well-suited for the design of methods devoted to process and/or analyse grey-level images based on *a priori* hypotheses related to the topology (connectedness) and the specific intensity (locally/globally minimal or maximal) of structures of interest. Several works related to component-trees have been devoted to enable their efficient computation (Salembier et al., 1998; Najman and Couprie, 2006; Wilkinson et al., 2008). In practice, the choice of the algorithm should be based on the input image depth. Salembier's algorithm, which is based on a recursive flooding of the image, is one of the most efficient when the image pixels are coded on 8 bits. In our experiments, which involve such images, it is twice as fast as Najman's algorithm, which is based on Tarjan's union-find operations. Therefore, in the sequel, all the results are computed using Salembier's algorithm. We propose in the sequel an interactive segmentation method, based on component-trees, that combines the advantages of example-based segmentation in terms of automation (since it avoids manual marker positioning or presegmentation by the user) and the ability to take into account the skills of the user in a quite simple and intuitive fashion. Indeed the only interaction consists of a thresholding process, which only requires a few seconds. In particular, Section 5.2 proposes an improved version of the theoretical framework developed in (Passat and Naegel, 2011b), which now deals with fuzzy examples instead of only binary ones. This use of fuzzy examples, together with the potential use of mask-based connectivity derived from the extensive filtering procedure proposed in Section 4 also enables us to obtain an improved version of the segmentation approach initially developed in (Dufour et al., 2011a). This segmentation strategy is described in Section 5.3.

5.2. Theory

A way to consider the previously described segmentation problem is to search the set of nodes $\widehat{\mathscr{K}} \subseteq \mathscr{K}$, which generates a binary object being as similar as possible to a given approximate precomputed segmentation. In (Passat and Naegel, 2011b), this issue is formalised as the resolution of the following optimisation problem

$$\widehat{\mathscr{K}} = \arg \min_{\mathscr{K}' \subseteq \mathscr{K}} \left\{ d\left(\bigcup_{N \in \mathscr{K}'} N, M \right) \right\} \tag{27}$$

where $M \subseteq E$ is a (binary) approximate segmentation, and d is a measure on 2^E. This measure considers the amount of false positives/negatives induced by $X = \bigcup_{N \in \mathscr{K}'} N$ with respect to M

$$d^\alpha(X, M) = \alpha \cdot |X \setminus M| + (1 - \alpha) \cdot |M \setminus X| \tag{28}$$

where $\alpha \in [0, 1]$ controls the trade-off between the tolerance to false positives and false negatives.

This binary formulation of the segmentation problem can be generalised in order to consider fuzzy examples, and not only binary ones. Indeed, by defining (without loss of correctness) the binary segmentation $X = \bigcup_{N \in \mathscr{K}'} N \subseteq E$ induced by a set of nodes $\mathscr{K}' \subseteq \mathscr{K}$ as a function $X : E \to \{0, 1\}$, and the considered (fuzzy) example as a grey-level image $M : E \to [0, 1]$, Eq. (27) remains valid while the measure d^α (on 2^E) provided in Formula (28) can be extended as the measure D^α (on $[0, 1]^E$) defined by

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

$$D^{\alpha}(X,M) = \int_0^1 d^{\alpha}(\lambda_v(X),\lambda_v(M)) \cdot dv \tag{29}$$

$$= \alpha \sum_{x \in X^{-1}(\{1\})} (1-M)(x) + (1-\alpha) \sum_{x \in X^{-1}(\{0\})} M(x) \tag{30}$$

The function $\mathscr{F}^{\alpha}$ proposed hereafter (that generalises the one proposed in (Passat and Naegel, 2011b)) makes it possible to build a binary image whose connected components form a set $\widehat{\mathscr{K}}$, which is a solution of Eq. (27). (The proofs of the following propositions, which rely on the framework initially considered in (Guigues et al., 2006), follow the same scheme as the ones developed in (Passat and Naegel, 2011b)) for Propositions 6 and 7 and in (Passat et al., 2011) for Proposition 8.

Proposition 6. *Let $I \in V^E$ be a grey-level image. Let $T = (\mathscr{K}, L, R)$ be the component-tree of I. Let $M : E \to [0,1]$. Let $\alpha \in [0,1]$. Let $\mathscr{F}^{\alpha} : \mathscr{K} \to 2^{\mathscr{K}}$ and $c^{\alpha} : \mathscr{K} \to \mathbb{R}^+$ be the functions recursively cross-defined, for all $N \in \mathscr{K}$, by*

$$\mathscr{F}^{\alpha}(N) = \begin{cases} \{N\} & \text{if } \alpha \cdot n(N,M) < (1-\alpha) \cdot p^*(N,M) + \sum_{N' \in ch(N)} c^{\alpha}(N') \\ \bigcup_{N_l \in ch(N)} \mathscr{F}^{\alpha}(N_l) & \text{otherwise} \end{cases} \tag{31}$$

and

$$c^{\alpha}(N) = \begin{cases} \alpha.n(N,M) & \text{if } \alpha \cdot n(N,M) < (1-\alpha) \cdot p^*(N,M) + \sum_{N' \in ch(N)} c^{\alpha}(N') \\ (1-\alpha) \cdot p^*(N,M) + \sum_{N' \in ch(N)} c^{\alpha}(N') & \text{otherwise} \end{cases} \tag{32}$$

where

$p^*(N,M) = \sum_{x \in N \setminus \bigcup_{N' \in ch(N)} N'} M(x)$ *and* $n(N,M) = \sum_{x \in N} (1-M)(x)$.

Let $M^{\alpha} = \bigcup_{N \in \mathscr{F}^{\alpha}(E)} N$. Then, we have

$$D^{\alpha}(M^{\alpha},M) = c^{\alpha}(E) = \min_{\mathscr{K}' \subseteq \mathscr{K}} \left\{ D^{\alpha}\left(\bigcup_{N \in \mathscr{K}'} N, M \right) \right\} \tag{33}$$

We notice that such a solution can be computed in linear time.

Proposition 7. *The set $\mathscr{F}^{\alpha}(E) = \mathscr{C}[M^{\alpha}]$ (and thus M^{α}) are computed with the linear algorithmic complexity $\mathscr{O}(\max\{|\mathscr{K}|, |E|\})$.*

Moreover, the increasing property of thresholding is actually inherited by the developed method.

Proposition 8. *Let $I \in V^E$ be a grey-level image. Let $M : E \to [0,1]$. Let $\alpha_1, \alpha_2 \in [0,1]$. Then we have*

$$(\alpha_1 < \alpha_2) \Rightarrow (M^{\alpha_2} \subseteq M^{\alpha_1}) \tag{34}$$

Remark 9. A consequence of this property is the ability to store $k > 2$ different results obtained for k increasing values $0 \leqslant \alpha_1 < \alpha_2 < \ldots < \alpha_{k-1} < \alpha_k \leqslant 1$, as a grey-level image $S_k : E \to [[1,k]]$ defined, similarly to Formula (20), by

$$S_k = \bigvee_{i=1}^{k} C_{M^{\alpha_i},i} \tag{35}$$

where $M^{\alpha_i} \subseteq E$ is the binary result of the segmentation method for the parameter α_i. In such a situation, we can avoid storing k distinct binary images, and the interactive choice of the result by the user is made (in real-time) by actually performing a standard thresholding of S_k among the values $[[1,k]]$.

5.3. Methodology

In this section, we focus on the description of a methodology for vessel segmentation by means of a fuzzy example. This method is automated in the first part of its process, and only requires user interaction once a set of binary results has been precomputed and stored in a grey-level image. The interactive part of the method is, in particular, a single thresholding step where the user can tune a parameter controlling the trade-off between false positives and false negatives between the segmentation example and the expected result.

5.3.1. Outline of the method

The method takes as input:

- a 3D grey-level angiographic image $I_{in} : E \to V$, e.g., a MRA or CTA image;
- a 3D vessel segmentation example consisting of a fuzzy (i.e., grey-level) image $B_{ex} : E \to [0,1]$ of vascular structures similar to those present in I_{in}, and the grey-level image $I_{ex} : E \to V$ from which this segmentation has been obtained.

(In Section 5.3.2, it will be observed that it may be also required to provide images $J_{in}, J_{ex} : E \to V$ for visualising the morphological structures neighbouring the vessels visualised in I_{in}, I_{ex}.) The only parameter is a threshold value $\alpha \in [0,1]$ (see Section 5.2), which needs to be tuned by the user at the end of the segmentation. The process, visually summarised in Fig. 5, is divided into two main steps:

- The first one consists of fitting the fuzzy image B_{ex} onto the image I_{in} using a registration procedure (see Section 5.3.2).
- Once B_{ex} is correctly positioned, the second step mainly consists of the interactive α-tuned segmentation process described in Section 5.2 (see Section 5.3.3).

The method finally provides as output:

- the 3D binary vessel segmentation $B_{out} \subseteq E$, associated to I_{in}, and induced by the example I_{ex} and the chosen parameter α.

5.3.2. Step 1: example fitting

Fitting the binary example B_{ex} onto I_{in} requires a registration/warping process to be performed. Registration of vascular images is a complex task. Indeed, while registration algorithms have arguably reached a satisfactory degree of efficiency for the processing of *compact, dense* images, such as morphological cerebral data (Holden, 2008), the development of efficient registration procedures in the case of *sparse* – and specifically angiographic – data seems to remain a globally open question, despite few recent works (Aylward et al., 2003; Jomier and Aylward, 2004; Suh et al., 2010). This is *a fortiori* the case for interpatient registration (it is indeed infrequently the case that I_{in} and I_{ex} are images of a same patient).

In some cases, the angiographic images contain a sufficient amount of morphological information, (e.g., Time-of-Flight MRA, depending on the acquisition parameters (Ozsarlak et al., 2004)). In most other cases, it may be necessary to associate to each angiographic data, namely I_{in} and I_{ex}, a corresponding morphological profile of the same patient (e.g., a T1 MRI acquisition). Since it is usual in clinical practice to acquire such data (which are even directly available via certain acquisition protocols (Dumoulin et al., 1991)) during an angiographic image acquisition, thus providing couples of morphological/angiographic data $(I_{in}, J_{in}), (I_{ex}, J_{ex})$, such a requirement remains acceptable.

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

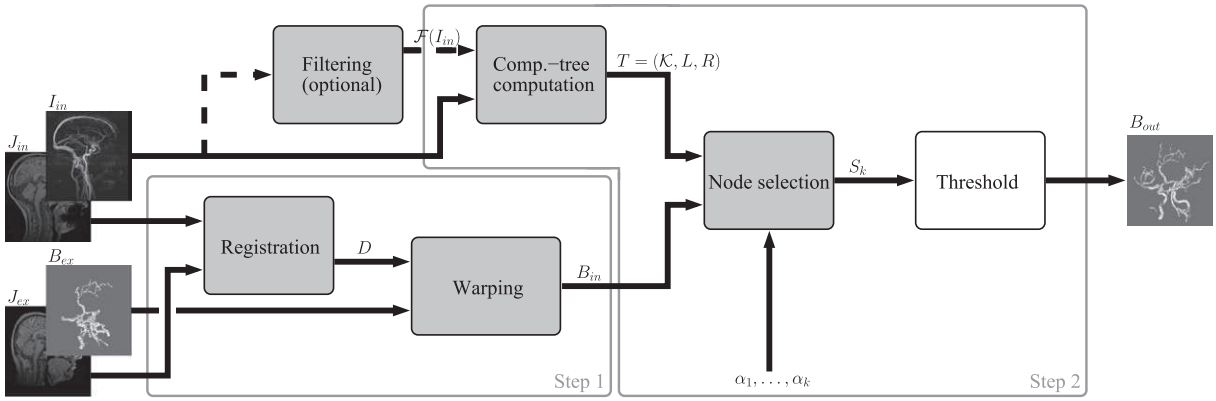


Fig. 5. Visual outline of the segmentation method described in Section 5.3. Step 1: example fitting (see Section 5.3.2). Step 2: interactive segmentation (see Section 5.3.3). Grey boxes: automatic steps; white box: interactive step. Continuous arrows: standard workflow; dash arrow: optional supplementary step (filtering, see Sections 4.2 and 5.4, and Fig. 3).

Under these conditions (and as proposed in this work), registration algorithms devoted to morphological images can be involved in the current step. These algorithms consist of determining a deformation field $\mathscr{D} : E \to E$ such that the composition of $\mathscr{D}$ and I_{ex} is "semantically" equal to I_{in}, or, more formally, that for all $x \in E$, we have

$$I_{in}(x) \simeq (I_{ex} \circ \mathscr{D}^{-1})(x) \tag{36}$$

Such registration procedures are based on various strategies leading to different degrees of accuracies, (i.e., rigid, affine, or nonrigid deformation fields $\mathscr{D}$). In the proposed work we have considered the nonrigid approach developed in (Noblet et al., 2005), as well as standard approaches for rigid and affine registration.

Once $\mathscr{D}$ is (automatically) computed from the angiographic data I_{in} and I_{ex} (or, in our case, from the morphological ones J_{in} and J_{ex}), the segmentation example B_{ex} remains to be fitted onto the vascular image I_{in} to be segmented. This is actually equivalent to computing $B_{in} : E \to [0, 1]$ in the same way as in Formula (36), which is done here by following a standard (polynomial) interpolation approach (Noblet et al., 2006).

5.3.3. Step 2: interactive segmentation

Once B_{in} has been computed, the segmentation of I_{in} can be carried out, guided both by the example B_{in} (which provides a model of the structures of interest in I_{in}), and by the user (who controls the adequacy in terms of false positives/negatives between this model and the expected result).

In the proposed methodological framework, this task is based on the approach developed in Section 5.2 (the example M defined in this section actually corresponds to the example B_{in} considered here). In particular, once the component-tree $T = (\mathscr{K}, L, R)$ of I_{in} is computed, the purpose is to determine the set of nodes $\widehat{\mathscr{K}} \subseteq \mathscr{K}$ defined in Eq. (27), i.e., the set of binary connected components which leads to the best possible segmentation (M^{α}, in Section 5.2, which corresponds to B_{out}, here) with respect to the chosen measure D^{α}, that controls the trade-off of false positives/negatives between this solution M^{α}/B_{out} and the example M/B_{ex}.

Practically, as stated in Remark 9, it is necessary to compute several segmentation results for distinct values of the parameter α, leading to a grey-level image (see Formula (35)). The level-sets of this image allows the user to choose the best segmentation B_{out} by a simple thresholding.

5.4. Mask-based connectivity

As stated at the beginning of Section 3.4, the connectivities considered when computing the component-tree of I_{in} are the "standard" ones (in our case, those induced by the well-known notions of 6- or 26-adjacency).

A (morphological) alternative definition for connectivity has been proposed with the notion of *second-generation* connectivity (Ronse, 1998; Serra, 1998; Braga-Neto and Goutsias, 2002). In this context, *mask-based* connectivity (Ouzounis and Wilkinson, 2007) proposes to use some (grey-level) mask functions in order to characterise the connected sets. In the binary case, and by only considering masks which are supersets of an image, we derive from (Ouzounis and Wilkinson, 2007) the following definition.

Definition 10. (Mask-based connectivity) Let $X \subseteq E$ be a binary image. Let $\omega(X) \supseteq X$ be a mask of X. The ω-connected components of X, noted $\mathscr{C}_{\omega}[X]$, are the sets $X \cap Y$, for any connected component Y of $\omega(X)$.

In the sequel, for a given (grey-level) image $I : E \to V$, we consider the extensive masks $\Omega(I) : E \to V$, i.e., such that $I \leqslant \Omega(I)$. We call Ω-connected components of I the set of all the ω-connected components of $\lambda_v(I)$ induced by the masks $\omega(\lambda_v(I)) = \lambda_v(\Omega(I))$, at all values $v \in V$. Typical examples of masks verifying these properties are those induced by:

- (flat) dilations, e.g., $I, \delta(I), \ldots, \delta^k(I), \ldots$;
- (flat) closings, e.g., $I, (\epsilon \circ \delta)(I), \ldots, (\epsilon^k \circ \delta^k)(I), \ldots$;

with a (well-chosen) structuring element, avoiding in particular translation effects.

As for any other connectivity, it is possible to build the component-tree of I_{in} induced by the Ω-connected components of the successive level-sets of I (Passat and Naegel, 2011a). Note in particular that each element of the set $\mathscr{K}_{\Omega}$ of the Ω-connected components of I_{in} will be composed of one or several connected components of I. More precisely, the component-tree and the (Ω-connected) component-tree of I_{in} induce a (surjective) morphism between $(\mathscr{K}, \subseteq)$ and $(\mathscr{K}_{\Omega}, \subseteq)$.

Broadly speaking, mask-based connectivity involving such extensive masks makes it possible to semantically reconnect structures which are physically disconnected in I_{in}. In particular, we consider as mask the filtered image $\mathscr{F}(I_{in})$, computed with the method proposed in Section 4. These masks present all the required properties, and allow us to correct the disconnection effects resulting from noise, artifacts and/or signal loss. This leads us to an

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

improved variant of the segmentation method proposed above. This variant consists of computing B_{out} from B_{ex} and I_{in} equipped with the connectivity provided by the filtered mask $\mathscr{F}(I_{in})$. This strategy is experimentally assessed in the next section.

6. Experiments and results

This section presents experiments carried out to assess the behaviour of the proposed two methods. The filtering method described in Section 4 is evaluated in Section 6.2 from a quantitative point of view, on synthetic images, and from a qualitative point of view on samples of 3D (MR) angiographic data. The segmentation method described in Section 5 is evaluated in Section 6.3, on 3D (MR) angiographic data from the qualitative and quantitative points of view, depending on the assessed criteria, and on the available ground-truth.

6.1. Data

We describe hereafter the data considered for the experimental validations carried out in this section. To the best of our knowledge, there does not exist any unified framework for the validation of 3D cerebral angiographic image analysis in the context of real data. The closest framework – devoted to coronary angiography image analysis (Schaap et al., 2009) – is indeed not suitable for the current issues. As far as possible, validations have been carried out in a quantitative fashion on synthetic data, in particular for vessel filtering. To deal with the validations of vessel segmentation, which indeed require real data, we chose to consider MRA images. For some of these data, hand-made ground-truths have been designed by medical experts. Since this is an intensive, time consuming task, only three ground-truths are indeed available: two for TOF MRAs and one for a couple of (TOF and PC) MRAs acquired from the same subject. Note also that we chose (i) to consider MRAs of various resolutions, in order to assess the ability of the method to deal with a large spectrum of images, and (ii) to focus on non-injected images, which present a growing interest due to concerns about the safety of gadolinium-based contrast agents (Miyazaki and Lee, 2008).

6.1.1. Synthetic data

We used the synthetic dataset considered in (Aylward and Bullitt, 2002) for vessel filtering validations (Section 6.2). This dataset is based on a $100 \times 100 \times 100$ isotropic image visualising a tortuous, branching vessel-like object of varying radii (0.5–4 voxels), which does not simulate a specific anatomical structure. The object contained in this image is depicted in Fig. 6a. A slice of the corresponding 3D grey-level image (at different levels of noise) is provided in Fig. 6b–e. The object cross-section intensities present a parabolic profile, ranging from 150 at the object borders, to 200 at its medial axes, while the background intensity is 100, which corresponds to a standard (intensity) model for MRAs, in small vessels neighbourhoods.

6.1.2. Phase-Contrast MRA

A set of 10 PC MRA data was considered for vessel segmentation validations (Section 6.3). The MRA exams were performed on a 1 T whole-body scanner (Gyroscan NT/INTERA 1.0 T from Philips, gradient slope 75 T/m/s). The flow encoding sequence (T1FFE/PCA) uses a TR of 10 ms and a TE of 6.4 ms. A sagittal MIP of one of these images are illustrated in Fig. 7a. The acquired images of dimensions varying from $256^2 \times 150$ to $256^2 \times 180$ voxels, were made of non-isotropic voxels of edges varying from 0.9 to 1.3 mm.

6.1.3. Time-of-Flight MRA

A set of 3 TOF MRA data was considered for vessel segmentation validations (Section 6.3). The MRA exams were performed on a 3 T whole-body scanner (Siemens verio 3 T, gradient slope 200 T/m/s). The flow encoding sequence (Flash/TOF) uses a TR of 21 ms and a TE of 3.6 ms. A sagittal MIP of one of these images is depicted in Fig. 7b. The acquired images of dimensions varying from $256^2 \times 208$ to $348 \times 284 \times 296$ voxels, were made of isotropic voxels of edges varying from 0.5 to 1.0 mm.

6.2. Vessel filtering

We first assess the filtering method described in Section 4. This is done in the context of vessel reconnection in 3D angiographic data. Contrary to the case of vessel segmentation (see Section 6.3), quantitative validations are hardly tractable on real data. Some qualitative (i.e., visual) validations are provided on few 3D samples of real data (same TOF MRAs as those used below, for segmentation experiments), at the end of this section. However, most of the validations presented hereafter, and in particular the quantitative ones, are performed on the synthetic dataset considered in (Aylward and Bullitt, 2002).

In addition to the discrete sampling of the continuous object, which generates errors (due, e.g., to partial volume effects), a Gaussian noise is added to the data, with different standard deviations, namely $\sigma = 10,\ 20, 40$ and 80, in the considered images (see Fig. 6b–e). Note that $\sigma = 20$ corresponds to the expected noise in MR or CT data, while $\sigma = 40$ is closer to the noise level expected in ultrasound data. The standard deviation $\sigma = 80$ was also tested, in order to explore the limits of the method in the worst cases, which do sometimes happen in clinical applications.

These experiments aim at estimating the efficiency of the filtering methodology, and in particular the cost of the reconnections in terms of supplementary noise. This is done in a quantitative fashion in Section 6.2.1, and in a more visual (and then subjective) fashion in Section 6.2.2.

When performed on a standard personal computer (equipped with a processor 3.0 GHz and 4 GB of memory), Step 1 (vessel detection) requires 10 s per scale, Step 2 (directional field correction) requires 10 s, while Step 3 (vessel reconnection) requires 30 s, for an image 256^3. It may be recalled that all these steps are fully automated.

6.2.1. Experiments on synthetic data

In order to carry out these first validations, we consider the four 3D images $I_\sigma : E \to V$, namely I_{10}, I_{20}, I_{40}, and I_{80}, illustrated in Fig. 6b–e, where σ is the standard deviation of the Gaussian noise in the image I_σ. Let $G \subseteq E$ be the object visualised in this image, viewed as a binary (ground-truth) object for I_σ.

When performing a thresholding of I_σ at a given value $v \in V$, we obtain a binary (segmented) result $\lambda_v(I) \subseteq E$ which approximates G. From a quantitative point of view, this approximation is expressed in terms of true positives ($G \cap \lambda_v(I)$) and false positives ($\lambda_v(I) \setminus G$).

For experiments, parameters of vesselness were set to $\alpha = \beta = 0.25$ and $\gamma = 5$. We performed a multi-scale Hessian analysis with scales $\sigma \in [1, 4]$ in geometric progression with 5 steps.

We first consider the global quality of the filtering procedures, in terms of proportions of false positives/negatives induced by a subsequent thresholding operation. These are visualised on ROC curves, allowing for a comparison of the proposed filtering with (i) the thresholding of the non-filtered image; (ii) the thresholding of a Hessian-based vesselness function (Frangi et al., 1999); and (iii) the thresholding of the image filtered by anisotropic diffusion (Manniesing et al., 2006). The four induced ROC curves are computed and compared for the four levels of noise $\sigma = 10, 20, 40$ and 80. The results of these experiments are shown on Fig. 8.

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

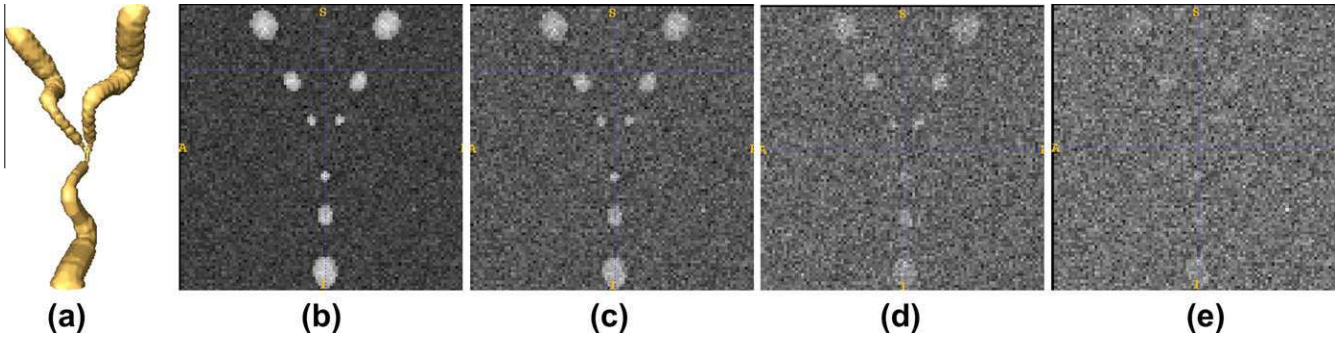


Fig. 6. (a) Synthetic 3D object (Aylward and Bullitt, 2002) used for validations in Section 6.2. (b–e) Slices of the grey-level image for various levels of additive white Gaussian noise: (b) $\sigma = 10$, (c) $\sigma = 20$, (d) $\sigma = 40$, and (e) $\sigma = 80$.

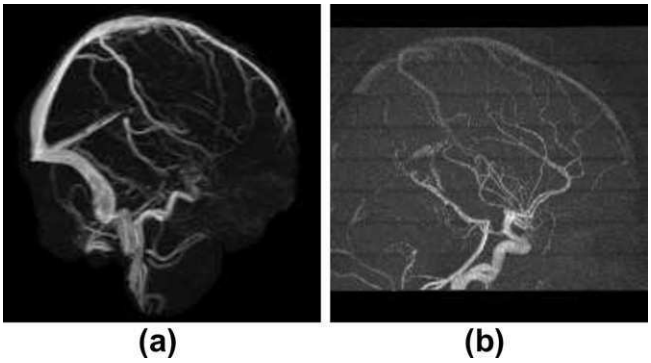


Fig. 7. Examples of data considered for vessel segmentation (maximal intensity projection, sagittal plane). (a) Phase-Contrast MRA (resolution: 1.0 mm, no contrast agent injection). (b) Time-of-Flight MRA (resolution: 0.5 mm, no contrast agent injection).

The main purpose of the proposed morpho-Hessian filtering method is to reconnect vessel-like structures without introducing too much noise. The presence of noise in I_σ may lead to a disconnection of the segmented object $\lambda_v(I)$ into several connected components, especially as the object becomes thinner. On the other hand, while reconnecting objects is desirable, it is also an inverse operation that may lead to connecting noise to the main object. In order to assess the "cost" of these reconnections in terms of supplementary noise, for each one of the images previously considered (namely the non-filtered image, vesselness function, anisotropic diffusion image, and morpho-Hessian filtered image), we show the first high accuracy reconnection on each ROC curve corresponding to threshold value of the segmented image $\lambda_v(I_\sigma)$. In other words, these points correspond to the best accuracy result for which the main object is still connected. These reconnection points are depicted by triangular dots in Fig. 8.

For all noise levels, i.e., σ = 10–80, we observe that the proposed morpho-Hessian filter exhibits at least as good results compared to the other methods. This is true whether we consider the overall shape of the ROC or only the point at which reconnection is ensured. We further note that all other methods degrade more rapidly than the morpho-Hessian as noise increases.

6.2.2. Experiments on real data

In order to conclude this first part of the validations devoted to the proposed morpho-Hessian filtering method, we now consider a few examples obtained from real images. The samples depicted in Fig. 9 show isosurface renderings of 3D angiographic data (namely TOF images) altered by signal heterogeneity, resulting in visible disconnections, depicted in yellow. The corresponding morpho-Hessian filtered images associated to these data are superimposed in blue. Qualitatively, these results confirm expectations derived from the synthetic data, namely that the morpho-Hessian filter reconnects vessel without amplifying noise significantly.

6.3. Vessel segmentation

We now assess the segmentation method described in Section 5. The experiments aim at estimating not only the global efficiency of the methodology, but also the influence of some key-elements of the technique, such as registration accuracy, example accuracy, interpatient anatomical variability, and effects of connectivity policies. Contrary to Section 6.2, only real data are considered. This is justified by the fact that realistic example-based segmentation cannot be performed on phantoms.

The experiments are carried out in the context of artery segmentation from 3D angiographic data of the brain, namely PC MRAs (10 images) and TOF MRAs (3 images). Some examples of images of the considered datasets are illustrated in Fig. 7. These datasets vary in resolution (millimetric for PC MRAs, and half-millimetric for TOF MRAs) and quality (low SNR for PC MRAs, and higher SNR for TOF MRAs, all data having been acquired without contrast agent injection). Also note that some data visualise both veins and arteries (PC MRAs, with a better contrast on venous structures), or essentially arteries (TOF MRAs, where veins are visible but at a much lower intensity).

For each angiographic image, a morphological image of the patient (acquired during the same session) is also considered for registration purpose (a PC MRA magnitude image in the case of PC MRAs, and a T1 MRI in the case of TOF MRAs). We note that ground-truths are available for only a part of these MRA data.[2]

When performed on a standard personal computer (equipped with a processor 3.0 GHz and 4 GB of memory), the registration step requires between 2 and 3 min, however it can be carried out prior to the segmentation step that involves the user. The segmentation part of the method, namely the component-tree construction and the computation of a series of potential results for various α values, requires respectively 10 s for the component-tree construction and 1 s by α values. The interactive part, i.e., the α-thresholding within these results is actually carried out in real time.

6.3.1. Evaluation of the component-tree approach

These first experiments aim at assessing the relevance of the example-based interactive approach, i.e., to validate the

[2] Note, moreover, that the use of vascular ground-truth, and in particular manual segmentations performed by experts, is neither a necessary nor a sufficient condition to guarantee the correctness of validations. See, e.g., (Caldairou el al., 2010), where some of the authors point out the significant variability in inter-experts segmentation results, which strongly bias the quantitative measures provided by standard criteria.

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

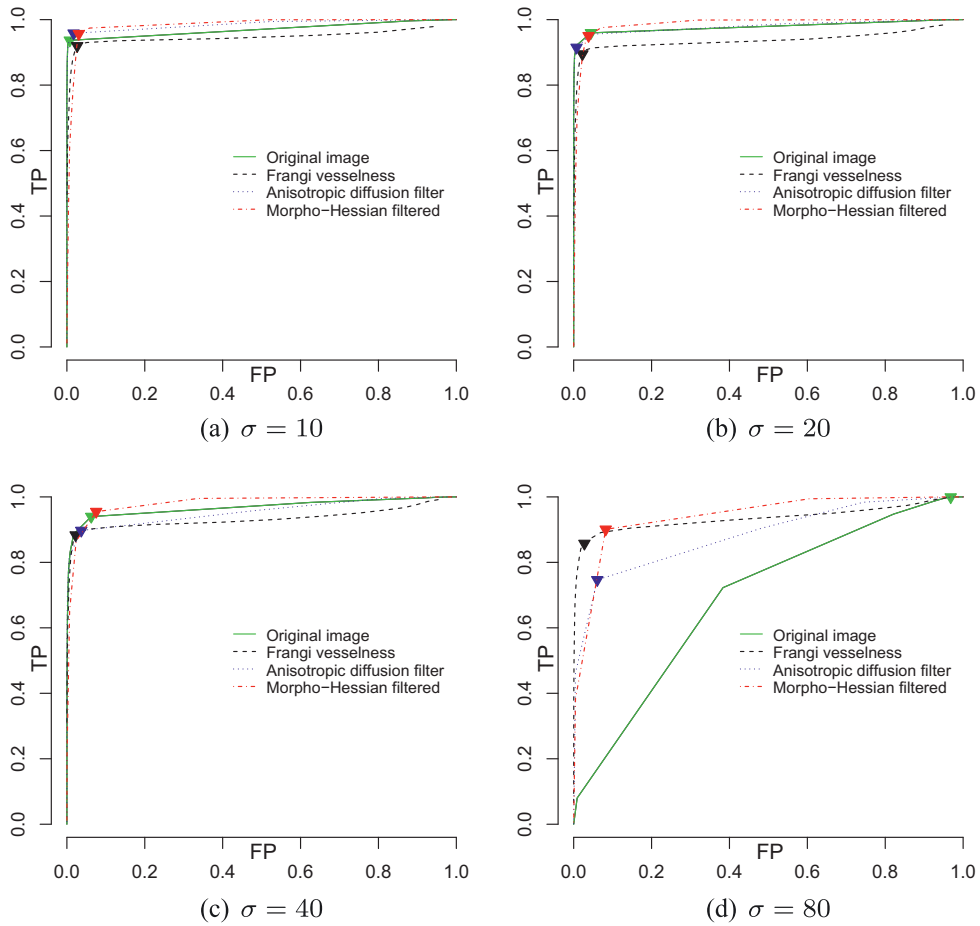


Fig. 8. ROC curves of thresholding operations performed on the original image I_σ, on its Hessian-based vesselness function, on the image obtained by anisotropic diffusion of I_σ, and on the image $\mathscr{F}(I_\sigma)$ obtained by multiscale morpho-Hessian filtering. The triangular dot on each curve indicates the point at which a correct reconnection of the segmented structure has been obtained. Results for different levels of noise: (a) $\sigma = 10$, (b) $\sigma = 20$, (c) $\sigma = 40$, (d) $\sigma = 80$.

segmentation theory exposed in Section 5.2. In order to do so, we consider an experimental context where neither the registration nor the example quality may affect the results. This is done by focusing on intra-patient and intra-image experiments.

Practically, for a given MRA image, for which a (binary) ground-truth segmentation is available, we perform the example-based interactive segmentation by using, as example, this same ground-truth. This is actually done for one (low resolution, low SNR) PC MRA, and three (high resolution, high SNR) TOF MRA.

Ideally, one may retrieve as result the ground-truth involved as examples. This (expected) correlation is expressed here by using the standard measures of sensitivity (*Sen*) and positive predictive value (*PPV*)

$$Sen = \frac{tp}{tp+fn} \text{ and } PPV = \frac{tp}{tp+fp} \tag{37}$$

where tp,fp, and fn are the true positives, false positives and false negatives, respectively. Due to the "one-dimensional" nature of the small vessels, which may bias the relevance of these volumic-based measures, both sensitivity and positive predictive values are computed on the 3D results (3D *Sen* and 3D *PPV*) and on the skeletonised ones (1D *Sen* and 1D *PPV*). These results are summarised in Table 1 and illustrated in Fig. 10.

They tend to show the correct behaviour of the method in the case of a correct and well positioned example. Note that the results are slightly less satisfactory for low resolution images. This is shown here for PC #1, but also note that the 1D measures are globally higher than the 3D ones, emphasising the ability of the method to correctly detect the structure of the vessels, despite possible volumetric inaccuracies. It may also be noticed that the use of an example to guide the segmentation process allows users to select specific structures of interest among a set of homogeneous ones, for instance here, arteries among the whole arteriovenous network.

6.3.2. Evaluation of the example quality

Secondly, we intend to evaluate the impact of the example quality on the segmentation accuracy, still without considering the effects of registration and of interpatient anatomical variability. We also assess the relevance of considering fuzzy examples vs. binary ones.

In order to do so, we consider two images of a single patient for which the segmentation ground-truth is available. One of these images is a (high resolution, high SNR) TOF MRA (TOF #1, considered above) thus associated to an accurate ground-truth. The other one is a (low resolution, low SNR) PC MRA (PC #1, considered above), associated with some less accurate ground-truth (which may be seen as a "blurred subset" of the TOF MRA ground-truth).

From these two images and associated ground-truth, we then perform four series of segmentations, which emphasise the behaviour of the segmentation method when applied on an accurate/non-accurate image with an accurate/non-accurate example. Moreover, in each series, these experiments are performed for a binary example (namely, a ground-truth), but also for fuzzy

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

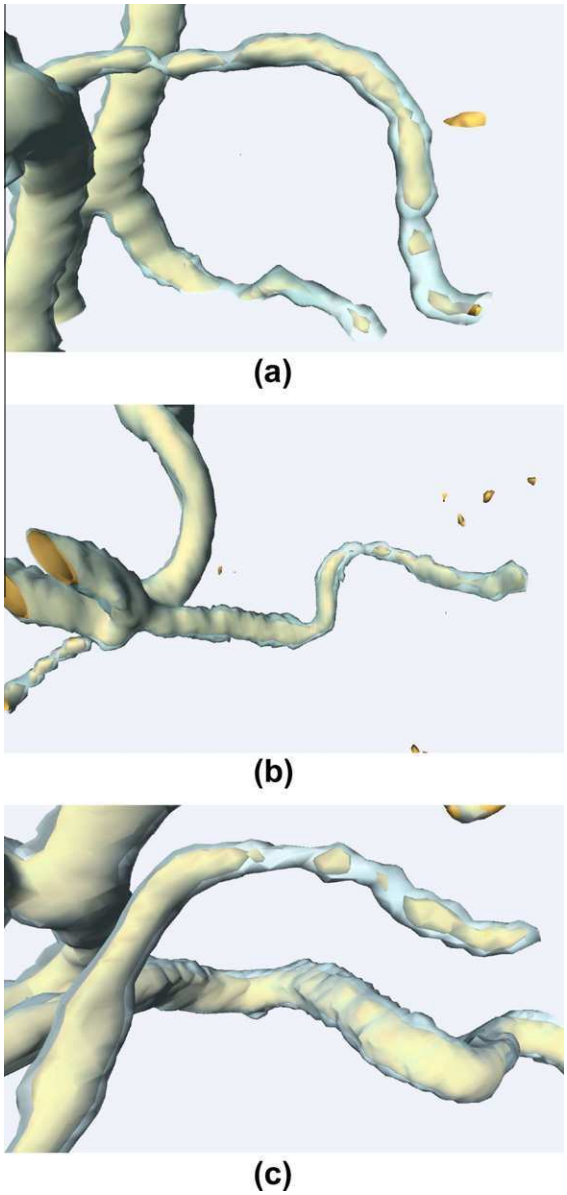


Fig. 9. Morpho-Hessian filtering results on real images. 3D isosurface rendering of the morpho-Hessian filter (in blue) superimposed over the initial image (in yellow). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Sensitivity and positive predictive value measures for experiments of Section 6.3.1 (see text and Fig. 10). The α value is the one for which the best segmentation has been obtained. In the case of TOF #2 and #3 any $\alpha \in]0,1[$ provides exact results.

Image	α	3D		1D	
		Sen	*PPV*	*Sen*	*PPV*
TOF #1	0.665	99.972	99.317	99.736	99.832
TOF #2	–	100.00	100.00	100.00	100.00
TOF #3	–	100.00	100.00	100.00	100.00
PC #1	0.510	95.391	97.542	98.877	97.305

examples obtained from this ground-truth by a grey-level (fuzzy) dilation with a Gaussian-shape SE. In particular, several SEs K_n ($n \in 2\mathbb{Z}+1$) were considered, modeling kernels of support $n \times n \times n$. For the sake of concision, we present only the most relevant results with respect to the 3D and 1D validations, namely K_5.

The obtained results, still expressed in terms of sensitivity and positive predictive value, are shown in Table 2 and partially illustrated in Fig. 11.

As mentioned above, the value of the standard measure are better in 1D than in 3D, emphasising the ability of the method to detect the structure of the vessels. Moreover, the obtained results reveal several facts. First, the example has to present the same degree of details as the visualised vessels, in order to maximise the ability to correctly segment them (see, in particular the first and fourth sets of rows in Table 2, vs. the second and third ones). Second, the use of fuzzy examples increases the sensitivity, but leads to a reduction of the positive predictive value. This is expected behaviour, with respect to the proposed connected-filtering strategy. Indeed, fuzzy examples capture a larger part of the space than binary ones, allowing the method to catch possibly disconnected vascular details (thus increasing *tp*, while decreasing *fn*), but also possibly catching (disconnected) noisy structures, then increasing *fp*. Despite this antagonist behaviour, it appears that fuzzy examples result in a better compromise between the appearance of false positives and false negatives (i.e., similar mean value and lower standard deviation between *Sen* and *PPV*). From this point on we then only consider fuzzy examples for the remaining validations.

6.3.3. Impact of the interpatient anatomical variability

We now intend to estimate the robustness of the segmentation method to vascular anatomical variations. These variations, which are low for the largest vessels, tend to become higher for the smaller ones, in particular by comparison to other (cerebral or non-cerebral) anatomical structures. Our experiments were carried out from a quantitative point of view on the four same images as above, and from a qualitative point of view on a dataset of 10 PC MRA images.

Experiments consist of performing segmentation with different examples, namely one example obtained from a single PC MRA image (not considered for segmentation here), and another example which is a "mean image" obtained from the preliminary segmentation of 20 PC MRA images (not considered for segmentation here). These two examples are fitted on the images to be segmented by performing rigid registration.

The obtained results for the first four images are given in Table 3. For the other 10 images, since no ground-truth is available, the validations were performed from a visual analysis (partially illustrated in Fig. 12).

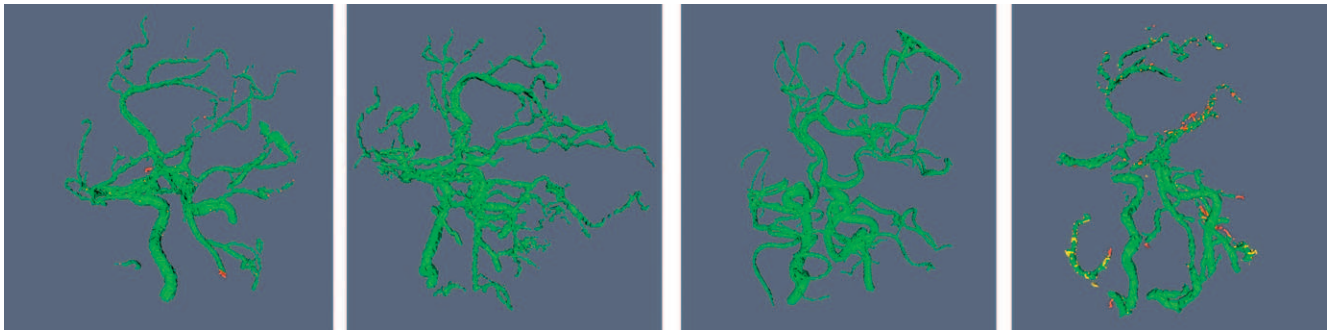
It appears from Table 3 that the use of a mean image as example provides similar results as a single image example in terms of *Sen*, and significantly improves them in terms of *PPV*. This argues in favour of using such averaged examples, which are, in some ways, comparable to vascular atlases (Chillet el al., 2003; Passat et al., 2006). These model more accurately the anatomical variability among a whole population. (These conclusions, stated here in the context of examples fitted by rigid transformation, remain the same for other kinds of registration policies.)

6.3.4. Evaluation of the registration

These experiments show that (i) the use of a mean image as example provides better results than the use of a single segmentation, and (ii), the use of fuzzy examples also increases the robustness of the method to both inter-individual variability and potential example inaccuracies. In our next experiments, we then only focus on mean and fuzzy examples.

Our purpose is here to evaluate the impact of the registration accuracy on the segmentation quality. By considering the same datasets of 4 and 10 images as above, three segmentations are computed. We fit the example using a rigid, an affine, and a non-rigid registration procedure. The obtained results for the first 4

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004



(a) TOF #1 (b) TOF #2 (c) TOF #3 (d) PC #1

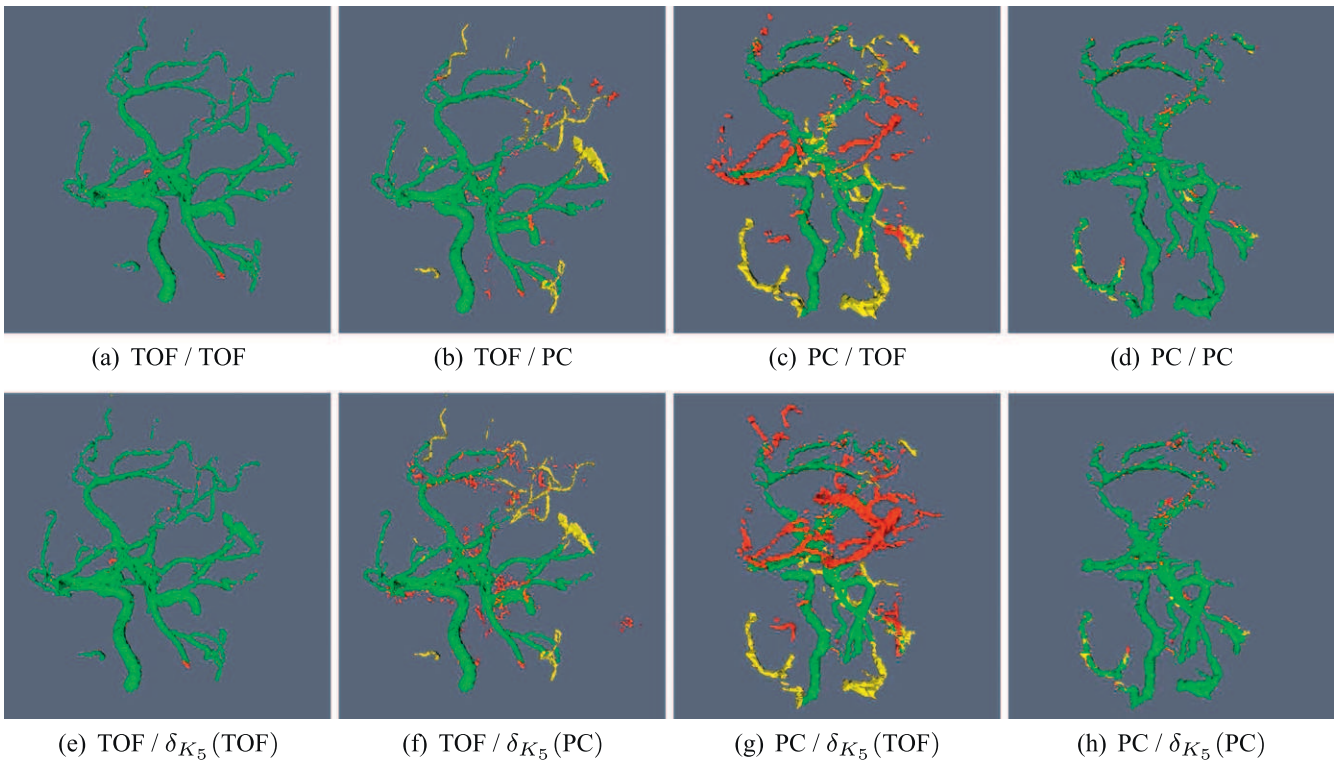
Fig. 10. Segmentation results in intra-patient cases, by using the associated ground-truths as examples (see Section 6.3.1 and Table 1). In green: true positives. In yellow: false negatives. In red: false positives. (a–c) TOF MRAs (TOF #1 to #3). (d) PC MRA (PC #1). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Sensitivity and positive predictive value measures for experiments of Section 6.3.2 (see text and Fig. 11). The α value is the one for which the best segmentation has been obtained. The best results for each data are depicted in bold fonts.

Image	Example	α	3D		1D	
			Sen	*PPV*	*Sen*	*PPV*
TOF #1	TOF #1	0.665	99.972	**99.317**	99.736	**99.832**
TOF #1	δ_{K_5}(TOF#1)	0.970	**99.991**	99.271	**99.772**	99.796
TOF #1	PC #1	0.810	90.848	**96.966**	94.923	**97.825**
TOF #1	δ_{K_5}(PC#1)	0.775	**91.192**	95.588	**95.339**	97.237
PC #1	TOF #1	0.845	58.936	**81.499**	85.956	**89.260**
PC #1	δ_{K_5}(TOF#1)	0.965	**71.983**	66.149	**91.983**	84.747
PC #1	PC #1	0.510	95.391	**97.542**	98.877	**97.305**
PC #1	δ_{K_5}(PC#1)	0.250	**96.237**	96.014	**99.021**	96.992

images are shown in Table 4. The results for all 10 images, analysed in a visual fashion, are partially illustrated in Fig. 13.

First, it appears that both for 3D and 1D scores, nonrigid registration is never the best registration policy. This emphasises the fact that despite good results in the case of morphological data, dense image nonrigid registration techniques are as yet not well-suited to handle vascular structures efficiently. Indeed, they may lead to registration noise at the highest resolution, which is same as that of the vessels. This should motivate in particular further development in registration techniques, specifically devoted to vascular structures. Such enhancements would enable us to better take into account the specificities (sparseness, noise, etc.) of angiographic data, as already stated in Section 5.3.2.



(a) TOF / TOF (b) TOF / PC (c) PC / TOF (d) PC / PC

(e) TOF / δ_{K_5}(TOF) (f) TOF / δ_{K_5}(PC) (g) PC / δ_{K_5}(TOF) (h) PC / δ_{K_5}(PC)

Fig. 11. Segmentation results depending on example quality (see Section 6.3.2 and Table 2). In green: true positives. In yellow: false negatives. In red: false positives. (a and e) TOF #1 segmentation using the TOF #1 ground-truth as example. (b and f) TOF #1 segmentation using the PC #1 ground-truth as example. (c and g) PC #1 segmentation using the TOF #1 ground-truth as example. (d and h) PC #1 segmentation using the PC #1 ground-truth as example. (a–d) Use of binary examples. (e–h) Use of fuzzy examples (dilated by K_5). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

Table 3
Sensitivity and positive predictive value measures, plus mean values and standard deviations, for experiments of Section 6.3.3 (see text). The α value is the one for which the best segmentation was obtained. The best (mean) results are depicted in bold fonts, for the mean image example.

Example	Image	α	3D		1D	
			Sen	*PPV*	*Sen*	*PPV*
Mean image	TOF #1	0.500	90.402	75.135	94.834	79.140
	TOF #2	0.870	100.00	83.063	100.00	93.260
	TOF #3	0.825	97.140	72.986	99.230	78.180
	PC #1	0.895	69.828	69.770	90.722	82.839
	Mean (std. dev.)	–	89.343 (13.618)	**75.238** (5.663)	**96.177** (4.278)	83.355 (6.902)
One image	TOF #1	0.710	90.579	71.218	94.744	77.672
	TOF #2	0.915	97.290	77.590	99.118	91.802
	TOF #3	0.900	100.00	68.081	100.00	75.136
	PC #1	0.915	70.475	60.608	89.982	79.041
	Mean (std. dev.)	–	**89.586** (13.342)	69.374 (7.058)	95.961 (4.601)	**85.07** (7.783)

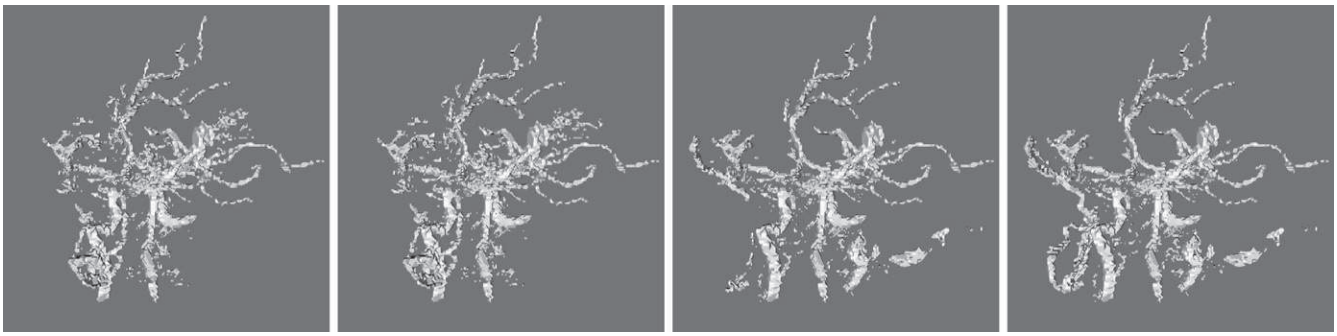


Fig. 12. Segmentation results on one of the 10 tested PC MRAs depending on the kind of example (see Section 6.3.3). (a and b) Example consisting of one segmented image: (a) binary, (b) fuzzy (K_5). (c and d) Example consisting of a mean image obtained from several segmented images: (c) binary, (d) fuzzy (K_5). The α value indicated for each subfigure is the one for which the best segmentation was obtained.

Table 4
Sensitivity and positive predictive value measures for experiments of Section 6.3.4 (see text and Fig. 13). The α value is the one for which the best segmentation was obtained. The best (mean) results are depicted in bold fonts.

Registration	Image	α	3D		1D	
			Sen	*PPV*	*Sen*	*PPV*
Rigid	TOF #1	0.500	90.402	75.135	94.834	79.140
	TOF #2	0.870	100.00	83.063	99.921	93.260
	TOF #3	0.825	97.140	72.986	99.230	78.180
	PC #1	0.895	69.828	69.770	90.722	82.839
	Mean (std. dev.)	–	89.343 (13.618)	**75.238** (5.663)	96.177 (4.278)	**83.355** (6.902)
Affine	TOF #1	0.500	90.402	73.908	94.836	78.534
	TOF #2	0.880	97.739	81.081	99.399	93.011
	TOF #3	0.950	100.00	65.387	100.00	71.833
	PC #1	0.920	70.192	69.387	90.712	82.979
	Mean (std. dev.)	–	**89.583** (13.561)	72.441 (6.730)	**96.237** (4.345)	81.589 (8.886)
Nonrigid	TOF #1	0.430	91.610	73.352	95.694	79.496
	TOF #2	0.970	96.574	75.914	98.930	91.503
	TOF #3	0.680	97.736	67.420	99.358	72.882
	PC #1	0.675	68.103	76.585	88.367	87.544
	Mean (std. dev.)	–	88.506 (13.858)	73.318 (4.171)	95.587 (5.084)	82.856 (8.317)

Second, it appears that rigid registration provides similar results as affine registration in terms of *Sen*, while it improves them in terms of *PPV*. This can in particular be explained by the preservation properties of this (simple) registration policy, in areas of the brain where registration remains an ill-posed problem due to the potential signal homogeneity of morphological tissues. In this context, the fuzzy policy considered for examples definitions indeed makes it possible to handle registration inaccuracies (in the same way as for inter-individual variability). Note that this finally leads to low computational costs, since the registration step contributes heavily to the total time cost of the method.

6.3.5. Evaluation of the connectivity

In these last experiments, we finally assess the influence of the neighbourhood connectivity on the quality of the segmentation results. In particular, we compare on the one hand the segmentation results obtained using the component-tree of an image I using the connectivity induced by the 6- and the 26-adjacencies, respectively; and on the other hand the results obtained using the component-tree defined by the connectivity induced by the (extensive) mask $\mathscr{F}(I)$ of I, computed in the way described in Section 4.

These comparisons are performed on the three TOF MRAs (TOF #1, TOF #2 and TOF #3), by using the same mean image example

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

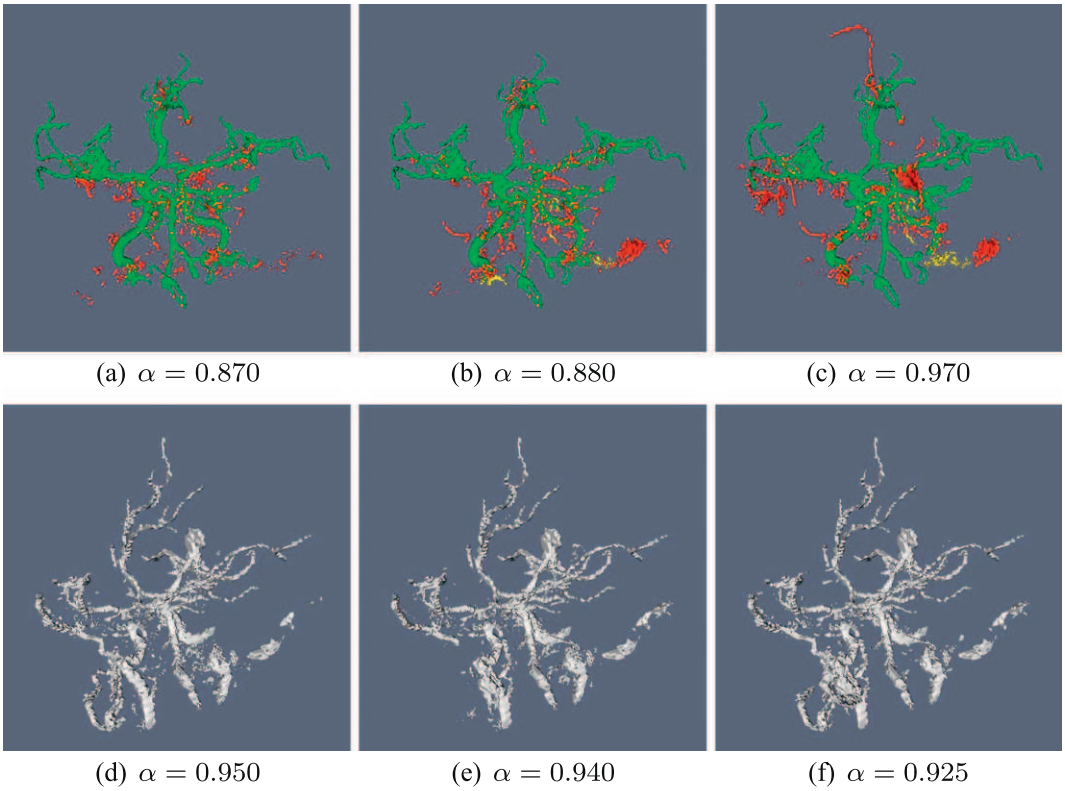


Fig. 13. Segmentation results depending on the registration policy (see Section 6.3.4). (a–c) Results on a TOF MRA (TOF #2, see also Table 4). In green: true positives. In yellow: false negatives. In red: false positives. (d–f) Results on one of the 10 tested PC MRAs. (a and d) Rigid registration. (b and e) Affine registration. (c and f) Nonrigid registration. The α value indicated for each subfigure is the one for which the best segmentation has been obtained. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 5
Sensitivity and positive predictive value measures for experiments of Section 6.3.5 (see text). The α value is the one for which the best segmentation was obtained. The best results are depicted in bold fonts.

Connectivity	Image	α	3D		1D	
			Sen	*PPV*	*Sen*	*PPV*
6-	TOF #1	0.745	**93.441**	36.682	94.014	44.304
	TOF #2	0.950	93.695	38.881	94.791	68.221
	TOF #3	0.865	**98.753**	57.129	97.538	59.865
26-	TOF #1	0.500	90.402	75.135	94.834	79.140
	TOF #2	0.870	**100.00**	**83.063**	**100.00**	93.260
	TOF #3	0.825	97.140	**72.986**	**99.230**	**78.180**
Filter	TOF #1	0.145	92.707	**76.180**	**95.226**	**81.417**
	TOF #2	0.730	**100.00**	83.053	**100.00**	**93.919**
	TOF #3	0.825	97.140	56.089	**99.230**	71.734

as above, dilated with K_5, and fitted by rigid registration. Numerical results are given in Table 5.

From a quantitative point of view, the sensitivity scores are all comparable, however, PPV scores are much lower for the 6-connectivity. This is due to the fact that 6-connected components are likely to be much smaller, and therefore close to noise. The use of 6-connectivity is therefore not recommended. With respect to using 26-connectivity vs. filtered connectivity, we observe that for the same sensitivity, PPV results can vary either way. Based on our experiments, results vary according to the noise level. We observe that the morpho-Hessian does not help much when images are already quite clean. It helps more in the case of high noise, e.g. with TOF #1. Since reconnections with the morpho-Hessian filter happen for thin vessels, they occur at the extremities of the network. These reconnections are visible but have relatively little effects on the PPV numbers.

7. Conclusion

Two methods have been described for 3D angiographic image filtering and segmentation. Both rely on recent advances in mathematical morphology. In particular, they take advantage of the mixture of discrete and continuous approaches (filtering method, Section 4.2.4), and of the low algorithmic cost of the involved strategies (filtering method – Proposition 4, and segmentation method – Proposition 7) leading to time-saving (fast, and automatic or interactive) image processing and analysis tools. We have shown how these two methods could be easily interfaced (Section 5.4) to directly integrate the filtering results in the segmentation process. Moreover, new results related to vector field regularization (Section 4.2.3), and fuzzy example handling (Section 5.2) have been obtained, thus extending the results initially proposed for both filtering (Tankyevych el al., 2009b; Tankyevych et al., 2009a) and segmentation (Passat and Naegel, 2011b; Dufour et al., 2011a) approaches.

These methods were evaluated on synthetic and real angiographic data, emphasising their relevance. The ability to discriminate specific parts of the vascular structures (example-based approach) and to integrate the user's skills with a low time cost has, in particular, led to use them in processes involving possibly large image datasets, for instance, the generation of statistical vascular atlases (Dufour et al., 2011b).

The following further work may also lead to improvements of these methods. Regarding vessel orientation computation (Section 4.2.2), the consideration of not only second-order derivatives,

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

but also first-order ones may provide better robustness to noise, and then improve vessel orientation estimates. Moreover, as for linear scale-space approach, a more elaborate analysis could be used, involving automatic scale selection (Lindeberg, 1998). Regarding the morphological part of the filtering method, (Section 4.2.4), the size of the spatially-variant structuring elements could be made to vary (Dokládal and Dokládalová, 2008) according to the eigenvalues of the Hessian matrix. In addition, it is envisaged to use variable and more flexible structuring element shapes, such as paths instead of segments.

Regarding vessel segmentation (Section 5.3.3), instead of computing several segmentation results for different (chosen/sampled) α values, an alternative solution may be to provide the exhaustive (finite) set S of possible binary segmentations, modeled as a grey-level image $I_{out} : E \rightarrow [1, |S|]$. The mask-based connectivity approaches may also be more intensively involved in the proposed segmentation paradigm, by considering not only extensive masks (see Definition 10), but also antiextensive or mixed ones (Ouzounis and Wilkinson, 2007). Extensions to hyperconnectivity (Ouzounis and Wilkinson, 2011) or multi-scale connectivities (Passat and Naegel, 2011a) may also be investigated.

Finally, vascular image registration (Section 5.3.2) also remains a challenging issue, in the case of the proposed image segmentation technique. The way to use not only morphological information from standard images, but also (sparse and varying) vascular information from angiographic data, will also be considered in (longer term) further work, with the purpose of improving the accuracy of the example fitting.

Acknowledgements

The angiographic data used for these experiments were provided by the radiology service of the Civil Hospital, Strasbourg, France (PC MRA data), and the *In Vivo* Imaging Platform of Strasbourg University, Strasbourg, France (TOF MRA data).

The research leading to these results was partially funded by a PhD grant of the *Région Alsace* and the *Centre National de la Recherche Scientifique* (CNRS), and has also received funding from the *Agence Nationale de la Recherche* (Grant Agreement ANR-2010-BLAN-0205).

References

Aylward, S.R., Bullitt, E., 2002. Initialization, noise, singularities, and scale in height ridge traversal for tubular object centerline extraction. IEEE Transactions on Medical Imaging 21, 61–75.

Aylward, S.R., Jomier, J., Weeks, S., Bullitt, E., 2003. Registration and analysis of vascular images. International Journal of Computer Vision 55, 123–138.

van Bemmel, C.M., Spreeuwers, L.J., Viergever, M.A., Niessen, W.J., 2003. Level-set-based artery-vein separation in blood pool agent CE-MR angiograms. IEEE Transactions on Medical Imaging 22, 1224–1234.

Bouaynaya, N., Charif-Chefchaouni, M., Schonfeld, D., 2008. Theoretical foundations of spatially-variant mathematical morphology. Part I: Binary images. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 823–836.

Bouaynaya, N., Schonfeld, D., 2008. Theoretical foundations of spatially-variant mathematical morphology. Part II: Gray-level images. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 837–850.

Bouraoui, B., Ronse, C., Baruthio, J., Passat, N., Germain, P., 2010. 3D segmentation of coronary arteries based on advanced mathematical morphology techniques. Computerized Medical Imaging and Graphics 34, 377–387.

Boykov, Y., Jolly, M., 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In: Proceedings of the ICCV, pp. 105–112.

Braga-Neto, U., Goutsias, J., 2002. Connectivity on complete lattices: new results. Computer Vision and Image Understanding 85, 22–53.

Breen, E.J., Jones, R., 1996. Attribute openings, thinnings, and granulometries. Computer Vision and Image Understanding 64, 377–389.

Caldairou, C., Passat, N., Naegel, B., 2010. Attribute-filtering and knowledge extraction for vessel segmentation. In: Proceedings of the ISVC. Springer, pp. 13–22.

Chen, J., Amini, A.A., 2004. Quantifying 3-D vascular structures in MRA images using hybrid PDE and geometric deformable models. IEEE Transactions on Medical Imaging 23, 1251–1262.

Chillet, D., Jomier, J., Cool, D., Aylward, S., 2003. Vascular atlas formation using a vessel-to-image affine registration method. In: Proceedings of the MICCAI. Springer, pp. 335–342.

Chung, A.C.S., Noble, J.A., Summers, P.E., 2004. Vascular segmentation of phase contrast magnetic resonance angiograms based on statistical mixture modeling and local phase coherence. IEEE Transactions on Medical Imaging 23, 1490–1507.

Cline, H.E., Thedens, D.R., Meyer, C.H., Nishimura, D.G., Foo, T.K., Ludke, S., 2000. Combined connectivity and a gray-level morphological filter in magnetic resonance coronary angiography. Magnetic Resonance in Medicine 43, 892–895.

Cousty, J., Najman, L., Couprie, M., Clément-Guinaudeau, S., Goissen, T., Garot, J., 2010. Segmentation of 4D cardiac MRI: automated method based on spatio-temporal watershed cuts. Image and Vision Computing 28, 1229–1243.

Deguchi, K., Izumitani, T., Hontani, H., 2002. Detection and enhancement of line structures in an image by anisotropic diffusion. Pattern Recognition Letters 23, 1399–1405.

Descoteaux, M., Collins, D.L., Siddiqi, K., 2008. A geometric flow for segmenting vasculature in proton-density weighted MRI. Medical Image Analysis 12, 497–513.

Dogdas, B., Shattuck, D.W., Leahy, R.M., 2005. Segmentation of skull and scalp in 3-D human MRI using mathematical morphology. Human Brain Mapping 26, 273–285.

Dokládal, P., Bloch, I., Couprie, M., Ruijters, D., Urtasun, R., Garnero, L., 2003. Topologically controlled segmentation of 3D magnetic resonance images of the head by using morphological operators. Pattern Recognition 36, 2463–2478.

Dokládal, P., Dokládalová, E., 2008. Grey-scale morphology with spatially-variant rectangles in linear time. In: Proceedings of the ACIVS. Springer, pp. 674–685.

Dufour, A., Passat, N., Naegel, B., Baruthio, J., 2011a. Interactive 3D brain vessel segmentation from an example. In: Proceedings of the ISBI, pp. 1121–1124.

Dufour, A., Tankyevych, O., Talbot, H., Ronse, C., Baruthio, J., Passat, N., 2011b. A statistical arteriovenous cerebral atlas. In: Proceedings MICCAI-CVII, pp. 73–80.

Dumoulin, C.L., Souza, S.P., Darrow, R.D., Pelc, N.J., Adams, W.J., Ash, S.A., 1991. Simultaneous acquisition of phase-contrast angiograms and stationary-tissue images with Hadamard encoding of flow-induced phase shifts. Journal of Magnetic Resonance Imaging 1, 399–404.

Eichel, P.H., Delp, E.J., Koral, K., Buda, A.J., 1988. A method for a fully automatic definition of coronary arterial edges from cineangiograms. IEEE Transactions on Medical Imaging 7, 313–320.

Faisan, S., Passat, N., Noblet, V., Chabrier, R., Meyer, C., 2011. Topology-preserving warping of binary images according to one-to-one mappings. IEEE Transactions on Image Processing 20, 2135–2145.

Flasque, N., Desvignes, M., Constans, J.M., Revenu, M., 2001. Acquisition, segmentation and tracking of the cerebral vascular tree on 3D magnetic resonance angiography images. Medical Image Analysis 5, 173–183.

Frangi, A.F., Niessen, W.J., Hoogeveen, R.M., van Walsum, T., Viergever, M.A., 1999. Model-based quantitation of 3-D magnetic resonance angiographic images. IEEE Transactions on Medical Imaging 18, 946–956.

Friman, O., Hindennach, M., Kühnel, C., Peitgen, H.O., 2009. Multiple hypothesis template tracking of small 3D vessel structures. Medical Image Analysis 14, 160–171.

Gooya, A., Liao, H., Matsumiya, K., Masamune, K., Masutani, Y., Dohi, T., 2008. A variational method for geometric regularization of vascular segmentation in medical images. IEEE Transactions on Image Processing 17, 1295–1312.

Gui, L., Lisowski, R., Faundez, T., Huppi, P.S., Lazeyras, F., Kocher, M., 2011. Automatic segmentation of newborn brain MRI using mathematical morphology. In: Proceedings ISBI, pp. 2026–2030.

Guigues, L., Cocquerez, J.P., Le Men, H., 2006. Scale-sets image analysis. International Journal of Computer Vision 68, 289–317.

Heijmans, H., 1991. Theoretical aspects of gray-level morphology. IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 568–582.

Hernandez, M., Frangi, A.F., 2007. Non-parametric geodesic active regions: method and evaluation for cerebral aneurysms segmentation in 3DRA and CTA. Medical Image Analysis 11, 224–241.

Holden, M., 2008. A review of geometric transformations for nonrigid body registration. IEEE Transactions on Medical Imaging 27, 111–128.

Jomier, J., Aylward, S.R., 2004. Rigid and deformable vasculature-to-image registration: a hierarchical approach. In: Proceedings of the MICCAI. Springer, pp. 829–836.

Jones, R., 1999. Connected filtering and segmentation using component trees. Computer Vision and Image Understanding 75, 215–228.

Kitamura, K., Tobis, J.M., Sklansky, J., 1988. Estimating the 3D skeletons and transverse areas of coronary arteries from biplane angiograms. IEEE Transactions on Medical Imaging 7, 173–187.

Kobashi, S., Kamiura, N., Hata, Y., Miyawaki, F., 2001. Volume-quantization-based neural network approach to 3D MR angiography image segmentation. Image and Vision Computing 19, 185–193.

Kong, T.Y., Rosenfeld, A., 1989. Digital topology: introduction and survey. Computer Vision, Graphics, and Image Processing 48, 357–393.

Krissian, K., Malandain, G., Ayache, N., Vaillant, R., Trousset, Y., 2000. Model-based detection of tubular structures in 3D images. Computer Vision and Image Understanding 80, 130–171.

Lesage, D., Angelini, E.D., Bloch, I., Funka-Lea, G., 2009a. Design and study of flux-based features for 3D vascular tracking, in: Proceedings of the ISBI, pp. 286–289.

Lesage, D., Angelini, E.D., Bloch, I., Funka-Lea, G., 2009b. A review of 3D vessel lumen segmentation techniques: models, features and extraction schemes. Medical Image Analysis 13, 819–845.

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

Li, H., Yezzi, A.J., 2007. Vessels as 4-D curves: global minimal 4-D paths to extract 3-D tubular surfaces and centerlines. IEEE Transactions on Medical Imaging 26, 1213–1223.
Lindeberg, T., 1994. Scale-Space Theory in Computer Vision. Kluwer Academic Publishers.
Lindeberg, T., 1998. Feature detection with automatic scale selection. International Journal of Computer Vision 30, 79–116.
Lorenz, C., Carlsen, I.C., Buzug, T.M., Fassnacht, C., Weese, J., 1997. Multi-scale line segmentation with automatic estimation of width, contrast and tangential direction in 2D and 3D medical images. In: Proceedings of the CVRMed-MRCAS. Springer, pp. 233–242.
Lorigo, L.M., Faugeras, O.D., Grimson, W.E.L., Keriven, R., Kikinis, R., Nabavi, A., Westin, C.F., 2001. CURVES: curve evolution for vessel segmentation. Medical Image Analysis 5, 195–206.
Manniesing, R., Viergever, M.A., Niessen, W.J., 2006. Vessel enhancing diffusion: a scale space representation of vessel structures. Medical Image Analysis 10, 815–825.
Manniesing, R., Viergever, M.A., Niessen, W.J., 2007. Vessel axis tracking using topology constrained surface evolution. IEEE Transactions on Medical Imaging 26, 309–316.
Maragos, P., Vachier, C., 2009. Overview of adaptive morphology: trends and perspectives. In: Proceedings of the ICIP, pp. 2241–2244.
McGuinness, K., O'Connor, N.E., 2010. A comparative evaluation of interactive segmentation algorithms. Pattern Recognition 43, 434–444.
Miyazaki, M., Lee, V.S., 2008. Nonenhanced MR angiography. Radiology 248, 20–43.
Naegel, B., 2007. Using mathematical morphology for the anatomical labeling of vertebrae from 3D CT-scan images. Computerized Medical Imaging and Graphics 31, 141–156.
Naegel, B., Passat, N., Boch, N., Kocher, M., 2007a. Segmentation using vector-attribute filters: methodology and application to dermatological imaging. In: Proceedings of the ISMM. INPE, pp. 239–250.
Naegel, B., Passat, N., Ronse, C., 2007b. Grey-level hit-or-miss transforms – Part I: Unified theory. Pattern Recognition 40, 635–647.
Naegel, B., Passat, N., Ronse, C., 2007c. Grey-level hit-or-miss transforms – Part II: Application to angiographic image processing. Pattern Recognition 40, 648–658.
Najman, L., Couprie, M., 2006. Building the component tree in quasi-linear time. IEEE Transactions on Image Processing 15, 3531–3539.
Najman, L., Talbot, H. (Eds.), 2010. Mathematical Morphology: From Theory to Applications. ISTE/J. Wiley & Sons.
Noblet, V., Heinrich, C., Heitz, F., Armspach, J.P., 2005. 3-D deformable image registration: a topology preservation scheme based on hierarchical deformation models and interval analysis optimization. IEEE Transactions on Image Processing 14, 553–566.
Noblet, V., Heinrich, C., Heitz, F., Armspach, J.P., 2006. Retrospective evaluation of a topology preserving non-rigid registration method. Medical Image Analysis 10, 366–384.
Ouzounis, G.K., Wilkinson, M.H.F., 2007. Mask-based second-generation connectivity and attribute filters. IEEE Transactions on Pattern Analysis and Machine Intelligence 29, 990–1004.
Ouzounis, G.K., Wilkinson, M.H.F., 2011. Hyperconnected attribute filters based on k-flat zones. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 224–239.
Ozsarlak, O., Van Goethem, J.W., Maes, M., Parizel, P.M., 2004. MR angiography of the intracranial vessels: technical aspects and clinical applications. Neuroradiology 46, 955–972.
Passat, N., Naegel, B., 2011a. Component-hypertrees for image segmentation. In: Proceedings of the ISMM. Springer, pp. 284–295.
Passat, N., Naegel, B., 2011b. Selection of relevant nodes from component-trees in linear time. In: Proceedings of the DGCI. Springer, pp. 453–464.
Passat, N., Naegel, B., Rousseau, F., Koob, M., Dietemann, J.L., 2011. Interactive segmentation based on component-trees. Pattern Recognition 44, 2539–2554.
Passat, N., Ronse, C., Baruthio, J., Armspach, J.P., Foucher, J., 2007. Watershed and multimodal data for vessel segmentation: Application to the superior sagittal sinus. Image and Vision Computing 25, 512–521.
Passat, N., Ronse, C., Baruthio, J., Armspach, J.P., Maillot, C., 2006. Magnetic resonance angiography: from anatomical knowledge modeling to vessel segmentation. Medical Image Analysis 10, 259–274.
Passat, N., Ronse, C., Baruthio, J., Armspach, J.P., Maillot, C., Jahn, C., 2005. Region-growing segmentation of brain vessels: An atlas-based automatic approach. Journal of Magnetic Resonance Imaging 21, 715–725.
Ronse, C., 1998. Set-theoretical algebraic approaches to connectivity in continuous or digital spaces. Journal of Mathematical Imaging and Vision 8, 41–58.
Sabry Hassouna, M., Farag, A.A., Hushek, S., Moriarty, T., 2006. Cerebrovascular segmentation from TOF using stochastic models. Medical Image Analysis 10, 2–18.
Salembier, P., Garrido, L., 2000. Binary partition tree as an efficient representation for image processing, segmentation and information retrieval. IEEE Transactions on Image Processing 9, 561–576.
Salembier, P., Oliveras, A., Garrido, L., 1998. Anti-extensive connected operators for image and sequence processing. IEEE Transactions on Image Processing 7, 555–570.
Salembier, P., Wilkinson, M.H.F., 2009. Connected operators: a review of region-based morphological image processing techniques. IEEE Signal Processing Magazine 26, 136–157.
Sato, Y., Nakajima, S., Shiraga, N., Atsumi, H., Yoshida, S., Koller, T., Gerig, G., Kikinis, R., 1998. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. Medical Image Analysis 2, 143–168.
Schaap, M., Metz, C.T., van Walsum, T., van der Giessen, A.G., Weustink, A.C., Mollet, N.R.A., Bauer, C., Bogunović, H., Castro, C., Deng, X., Dikici, E., O'Donnell, T., Frenay, M., Friman, O., Hernández Hoyos, M., Kitslaar, P.H., Krissian, K., Kühnel, C., Luengo-Oroz, M.A., Orkisz, M., Smedby, Ö., Styner, M., Szymczak, A., Tek, H., Wang, C., Warfield, S.K., Zambal, S., Zhang, Y., Krestin, G.P., Niessen, W.J., 2009. Standardized evaluation methodology and reference database for evaluating coronary artery centerline extraction algorithms. Medical Image Analysis 13, 701–714.
Serra, J., 1982. Image Analysis and Mathematical Morphology. Academic Press.
Serra, J., 1998. Connectivity on complete lattices. Journal of Mathematical Imaging and Vision 9, 231–251.
Suh, J.W., Scheinost, D., Qian, X., Sinusas, A.J., Breuer, C.K., Papademetris, X., 2010. Serial non rigid vascular registration using weighted normalized mutual information. In: Proceedings of the ISBI, pp. 25–28.
Sun, K.Q., Sang, N., 2008. Morphological enhancement of vascular angiogram with multiscale detected by Gabor filters. Electronics Letters 44, 86–87.
Tankyevych, O., Talbot, H., Dokládal, P., Passat, N., 2009a. Direction-adaptive grey-level morphology. Application to 3D vascular brain imaging. In: Proceedings of the ICIP, pp. 2261–2264.
Tankyevych, O., Talbot, H., Dokládal, P., Passat, N., 2009b. Spatially variant morpho-Hessian filter: efficient implementation and application. In: Proceedings of the ISMM. Springer, pp. 137–148.
Tankyevych, O., Talbot, H., Passat, N., Musacchio, M., Lagneau, M., 2011. Angiographic image analysis. In: Dougherty, G. (Ed.), Medical Image Processing: Techniques and Applications. Springer, pp. 115–144 (Chapter 6).
Thackray, B.D., Nelson, A.C., 1993. Semi-automatic segmentation of vascular network images using a rotating structuring element (ROSE) with mathematical morphology and dual feature thresholding. IEEE Transactions on Medical Imaging 12, 385–392.
Tizon, X., Smedby, Ö., 2002. Segmentation with gray-scale connectedness can separate arteries and veins in MRA. Journal of Magnetic Resonance Imaging 15, 438–445.
Tomasi, C., Manduchi, R., 1998. Bilateral filtering for gray and color images. In: Proceedings ICCV, pp. 839–846.
Urbach, E.R., Boersma, N.J., Wilkinson, M.H.F., 2005. Vector attribute filters. In: Proceedings of the ISMM. Springer SBM, pp. 95–104.
Urbach, E.R., Wilkinson, M.H.F., 2002. Shape-only granulometries and gray-scale shape filters. In: Proceedings of the ISMM. CSIRO Publishing, pp. 305–314.
Verdú-Monedero, R., Angulo, J., 2008. Spatially-variant directional mathematical morphology operators based on a diffused average squared gradient field. In: Proceedings of the ACIVS. Springer, pp. 542–553.
Vincent, L., Soille, P., 1991. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 583–598.
Westenberg, M.A., Roerdink, J.B.T.M., Wilkinson, M.H.F., 2007. Volumetric attribute filtering and interactive visualization using the max-tree representation. IEEE Transactions on Image Processing 16, 2943–2952.
Wilkinson, M.H.F., Gao, H., Hesselink, W.H., Jonker, J.E., Meijster, A., 2008. Concurrent computation of attribute filters on shared memory parallel machines. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 1800–1813.
Wilkinson, M.H.F., Westenberg, M.A., 2001. Shape preserving filament enhancement filtering. In: Proceedings of the MICCAI. Springer, pp. 770–777.
Wong, W.C.K., Chung, A.C.S., 2007. Probabilistic vessel axis tracing and its application to vessel segmentation with stream surfaces and minimum cost paths. Medical Image Analysis 11, 567–587.
Xu, C., Prince, J.L., 1998. Snakes, shapes, and gradient vector flow. IEEE Transactions on Image Processing 7, 359–369.
Zana, F., Klein, J.C., 2001. Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation. IEEE Transactions on Image Processing 10, 1010–1019.

Please cite this article in press as: Dufour, A., et al. Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology. Med. Image Anal. (2013), http://dx.doi.org/10.1016/j.media.2012.08.004

J. Vis. Commun. Image R. 22 (2011) 411–420

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Computationally efficient, one-pass algorithm for morphological filters

Petr Dokládal [a,*], Eva Dokládalová [b]

[a] Center of Mathematical Morphology, Department Mathematics and Systems, Mines PARISTECH, 35, rue St. Honoré, 77 300 Fontainebleau Cedex, France
[b] Unité mixte de recherche CNRS-UMLV-ESIEE, UMR 8049, Université Paris-Est, Cité Descartes B.P.99, 93 162, Noisy le Grand Cedex, France

ARTICLE INFO

Article history:
Received 15 April 2010
Accepted 16 March 2011
Available online 22 March 2011

Keywords:
Mathematical morphology
Serial filters
Nonlinear filters
Real-time implementation
Streaming
Algorithm

ABSTRACT

Many useful morphological filters are built as long concatenations of erosions and dilations: openings, closings, size distributions, sequential filters, etc. This paper proposes a new algorithm implementing morphological dilation and erosion of functions. It supports rectangular structuring element, runs in linear time w.r.t. the image size and constant time w.r.t. the structuring element size, and has minimal memory usage.

It has zero algorithm latency and processes data in stream. These properties are inherited by operators composed by concatenation, and allow their efficient implementation. We show how to compute in one pass an Alternate Sequential Filter (ASF^n) regardless the number of stages n.

This algorithm opens the way to such time-critical applications where the complexity and memory requirements of serial morphological operators represented a bottleneck limiting their usability.

1. Introduction

Since its introduction in late sixties, the mathematical morphology provides a complete set of image processing tools from filtering [1,2], multi-scale image analysis [3] to pattern recognition [4–6]. They have been used in unrivalled number of applications [7,8]. The most significant examples include biomedical and medical imaging, video surveillance, industrial control, video compression [9], stereology or remote sensing [10].

Nonetheless, not all useful operators can be easily implemented in real time with reasonable memory requirements. In demanding image-interpretation applications requiring a high correct-decision liability, one often uses robust but costly multi-criteria and/or multi-scale analysis.

These applications often consist of a serial concatenation of alternating atomic operators dilation and erosion with progressively increasing computing window called structuring element (SE).

Such operators cannot be parallelized due to the sequential data dependency of the individual atomic operators. The only possibility is to minimize the latency of each atomic operator and consider computing in stream. The latency minimization reduces the time to wait for individual pixel results. The stream computing allows transferring them immediately to the next atomic operator, as soon as they are available and before the entire image is processed. Thus, these atomic operators can work simultaneously, on data delayed in time. In such implementation, one has to sum the individual working memories of every atomic operator. Then also the memory may become penalizing for large, high-resolution images.

The work presented in this paper, aims to propose a new dilation/erosion algorithm with a constant processing time, low latency and low memory requirements for implementation of the individual atomic operators. Consequently, it allows to implement advantageously the following:

1. Alternate Sequential Filters (ASFs) – that are a concatenation of openings and closings with a progressively increasing structuring element, useful for multi-scale analysis [1].
2. Size distributions (granulometries) – that are a concatenation of openings allowing to assess the size distribution of a population of objects [3,11,12].
3. Statistical learning – a selected set of morphological operators ξ_i can be separately applied to an image f. Then for every pixel $f(x,y)$, the vector of values $(\xi_i(f)(x,y))$ can serve as vector of descriptors for pixel-wise learning and classification [6]. Obtaining them may be computationally intensive.

1.1. Paper organization

The remainder of the Introduction lists the most known fast algorithms of morphological dilation and erosion and discusses their properties, followed by the explanation of Novelties in this paper.

The Preliminaries, Section 2, introduce the basic principles of dilations, erosions and their combinations.

* Corresponding author. Fax: +33 1 64 69 47 07.
E-mail addresses: petr.dokladal@mines-paristech.fr (P. Dokládal), eva.dokladalova@esiee.fr (E. Dokládalová).

1047-3203/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2011.03.005

The Section 3 outlines the principle of the new algorithm: (i) the 2-D decomposition preserving sequential access to data and zero latency, (ii) elimination of useless values, (iii) the conversion of an anti-causal structuring element into a causal one, necessary to preserve the sequential access to data, and (iv) the encoding used to reduce the memory requirements and acceleration of computations.

Sections 4 to 5 discuss the properties and Section 6 presents the case of the four stage ASF as an example.

The paper concludes by Benchmarks, general Conclusions and Future extensions, Sections 7 to 9. The commented pseudo code is given in the Appendix.

1.2. Existing work

The mathematical morphology relies on two fundamental, complementary operations: erosion and dilation. They are local, defined within a computing window, specified by the so-called structuring element (SE), characterized by its size, its shape and origin. It is well known that, with the increasing size of the SE, the direct implementation leads to an extremely high computing cost. Although the fastest existing algorithms [13–16] concentrate mainly on the reduction of the number of comparisons, few of them deal with the latency and memory requirements [17]. Moreover, the minimization of comparison number is not always proportional to the overall performance improvement [15].

In the following paragraphs, we concentrate on the presentation of the existing state of the art. We discuss it from the classical point of view of complexity, based on the number of comparisons. We bring it face to face with the latency and the memory requirements. For each algorithm, we analyze the possibility of its stream implementation since it is a key feature allowing efficient chaining of the atomic operators.

Let us start by the definition of the used basic terms. Consider a system $Y = f(X)$ with X and Y the input and output data streams. By *latency* understand the distance between the same positions in the two streams. It is a dimensionless value, expressed in number of data samples. It is the sum of several factors:

(1) *operator latency* – is induced by non causal operators due to the fact that the value to output depends on future signal samples. Consider a basic max filter $y_j = \max(x_{j-w/2}, \ldots, x_{j+w/2})$. One cannot output y_j before having read all x_i until $x_{j+w/2}$,
(2) *algorithm latency* – some algorithms continue reading the input even after all needed input data are available. Several morphological dilation/erosion algorithms run in two (forward and backward) data scans, e.g. [13,18]. Typically, in [18], before processing one image line, one needs to read the entire line. For 2-D dilation by a rectangle, implemented separately in the horizontal and vertical direction, one would need to wait the bottom of the image before writing the result. In [13] these forward and backward scans can be done on w pixels long intervals.

For example, the naive implementation of the morphological dilation (Eq. 3) has a considerable computation complexity $\mathcal{O}(w\text{-}1)$ per pixel, with w the SE width (or area in 2-D), but no algorithm latency.

The operator latency – inherent to the operator – is incompressible. Consequently, the optimization effort should focus on the algorithm latency and the computational complexity.

The first concern related to the latency is therefore the time response of the system. Another concern related to the latency is the memory requirements. This can intuitively be explained by the fact that the latent (meaning "hidden") data need to be temporarily stored somewhere to not to get lost. Obviously, a large latency requires large storage. An interesting conclusion is that using larger SE will have larger memory requirements.

1.2.1. State of the art

The scientific community has adopted several approaches to speed up the erosion/dilation computation. The first one, we call *direct computation*, consists of a straightforward optimization of the computation given the SE shape.

The second approach relies on the *SE decomposition* into a sequence of reduced SE. Consequently, the optimization effort concentrates on the computation of this smaller SE. The special attention is paid to the SE decomposition into a series of 1-D SE, very popular in numerous applications [19,20]. It allows better data access, reuse of intermediate results and is easy to parallelize.

In the following, refer to Table 1 and 2 summarizing the properties of some algorithms cited below. By data memory understand the temporary storage for input or output data if the algorithm uses random data access. For instance the direct implementation needs random access to input data, whereas the output is written sequentially. It includes also the image transposition used by some algorithms. The working memory is any supplementary memory space required by the algorithm. It includes the data structures like FIFOs, LUTs, histograms, etc. Temporary constants, scalar variables, counters, etc., are omitted.

Direct 2-D computation. Optimized algorithms reduce the computing redundancy by using some well-suited data structures to keep the intermediate results. The most natural way is the approach used by Huang et al. [21] for median filtering, by Chaudhuri et al. [22] for rank-order filtering, and later by Van Droogenbroeck and Talbot [23]. They use a histogram to store the values within the span of the SE at some position in the image. During the translation of the SE over later image positions the histogram is updated by inclusion/deletion of the values of the entering/leaving points. The family of available shapes for the SE is arbitrary. On the other hand, using histogram makes that the input data have to be integers.

SE decomposition. It has soon become evident that the SE decomposition offers another possibility to obtain a fast implementation of more complex SEs both on specialized hardware as well as on sequential computers, and the literature soon became abundant see e.g. [24–31]. The speedup is obtained by dividing the effort in two independent key aspects, an efficient decomposition and the algorithm used for computing the atomic operations.

Various types of decompositions have been proposed. Perhaps the most known decomposition of linear sets is the *linear decomposition* which comes from the associativity of the dilation, see Matheron [28]. Pecht [29] has proposed a more efficient logarithmic decomposition based on the *extreme set* of some SE. For example, for a polygon, the extreme set contains the vertices.

Van den Boomgaard and Wester [30] show that the Pecht decomposition can be improved for convex shapes. They propose a decomposition of an arbitrary shape into the union of convex shapes taken from a fixed collection of basis, efficiently decomposable shapes. Coltuc and Pitas [31] propose a factorization based algorithm running efficiently for 2^n signals. Soille et al. [27] propose an extension of the 1-D van Herk algorithm to 2-D. The SE is a line oriented in an arbitrary angle. The decomposition is obtained by saving the 2-D image as an 1-D array, and recomputing the pixel indices correspondingly to the given orientation of the line. Another efficient algorithm has been recently proposed by Urbach and Wilkinson [16]. It decomposes a flat, arbitrary-shape SE by using a set of 1-D chords. The min/max statistics of the chords are stored in LUT.

1-D algorithms. The 1-D algorithms compute the partial 1-D dilations after the SE decomposition into lines.

One of the earliest, and most often used 1-D algorithms, is the van Herk algorithm [13] proposed in 1992. The same algorithm completed by theoretical background was also published by Gil

Table 1
2-D algorithms comparison.

Algorithm	SE type	Complexity per pixel	Algorithm latency	Data memory	Working memory
Naive 2-D	User	$\mathcal{O}(WH)$	0	MN	0
Urbach- Wilkinson	User	$\mathcal{O}(N_c + \log_2(L_{max}(C)))$	MN	MN	$NH\log_2 W$
Van Droogenbroeck–Talbot	User	$\mathcal{O}(H\log_2(G))^*$	0	NH	WHG
This paper 2-D	Rect.	$\mathcal{O}(1)$	0	0	$2(NH + W)$

$W \times H$ = SE size (Width × Height); $N \times M$ = image size; G = number of gray levels; $L_{max}(C)$ = maximum chord length; N_c = number of chords; * square SE.

Table 2
Fast 1-D algorithms comparison.

Algorithm	SE type	Comparisons per pixel	Algorithm latency	Overall memory
Naive 1-D	User	$W - 1$	0	N
van Herk Gil–Wermann	Sym	$3 - \frac{4}{W}$	W	$N + 2W$
Gil–Kimmel	Sym Even/Odd	$1,5 + \frac{\log_2 W}{W} + \mathcal{O}(\frac{1}{W})$	W	$N + 3W$
Lemire	Left	3	0	$N + W$
Lemonnier	Sym	nc	N	$2N$
Van Droogen–Broeck–Buckley	Sym Even/Odd	nc	0	$2N + G$
This paper 1-D	User	$\mathcal{O}(1)$	0	$2W$

Sym = symmetric SE; Left = Left sided SE; User = user defined; W = SE size; N = line size; G = number of gray levels; nc = not communicated.

and Werman [32], and later improved by Gevorkian et al. [33] and Gil and Kimmel [14]. The computational complexity is independent of the SE size. It requires two passes on the input data: causal and anti-causal. Consequently, computing in stream is impossible. Another, similar algorithm was proposed in [34] using ring-type buffers. Recently, Clienti et al. [35] propose an interesting modification of the Van Herk algorithm reducing the memory to $2W$, implemented on an FPGA.

A different approach has been used by Lemonnier [18]. It identifies and propagates local extrema as long as it is required by the SE size. Again, two passes are needed: causal and anti-causal. Hence, the algorithm latency is N. The stream execution is impossible.

Van Droogenbroeck and Buckley [15] publish an anchor based algorithm for erosions and openings. The anchors are these portions of signal that remain unchanged by the operator. The algorithm gives good performance in terms of the computing time. The erosion can not run *in place* and stream processing is probably impossible. The principal disadvantage is in using histograms (suited only for integer values, and making the algorithm irregular).

Lemire proposes a fast, stream-processing algorithm for a left-sided SE [17], and later for symetric SE [36]. Both versions simultaneously compute 1-D dilation and erosion, run on floating point data and have low memory requirements and zero latency. However, even though the algorithms are supposed to run in stream, the intermediate storage of coordinates of local extrema actually represents a random access to the input data.

1.3. Latency issues

In order to asses the latency of 2-D algorithms we need to consider separately these different algorithm categories:

- For decompositions of rectangles using $R = H \oplus V$ (R = rectangle, H/V = horizontal/vertical segment, respectively) the latency in 2-D is a multiplicative factor of the latency in 1-D and the image width. For two pass algorithms, e.g. Lemonnier [18], the 2-D latency equals one image frame. For locally two-pass algorithms, [13,32,33,27] a specific decomposition would have to be found to optimize the latency in 2-D. The same holds also for other 1-D algorithms that in 1-D allow streaming processing [17,36,15].
- Regarding the direct computation in 2-D, though UW [16] could theoretically read/write the input/output images sequentially, the possibility of streaming processing is not mentioned; they also use random accesses to intermediate data. Van Droogenbroeck–Talbot [23] and the naive implementation write output sequentially, but use random accesses to input data.
- The computational complexity, used in the Table 1, may introduce to the result a supplementary delay – the *time to compute the result*. Whereas the two latencies are relative to the stream rate, the additional delay depends on the implementation and the computation platform. It can be (1) negligeable as in most $R = H \oplus V$ decompositions, where the latency prevails, or (2) dominant like in the direct implementation with large SE or in 3-D.

1.4. Novelty of this paper

Although one can find several 1-D algorithms running with zero algorithm latency, none of the above cited algorithms combines all the features necessary for efficient implementation of composed operators in the form $\xi = \delta_{B_n}\varepsilon_{B_{n-1}} \ldots \delta_{B_2}\varepsilon_{B_1}$ for 2-D images.

Suppose the atomic operators δ, ε implemented using an algorithm with sequential access to data. This allows to run in parallel the entire ξ despite its internal sequential data dependence. If the atomic algorithm, in addition, has zero algorithm latency, then the entire chain ξ inherits the same properties: sequential data access and zero algorithm latency. This is an interesting property, since computing ξ suddenly becomes very efficient: in stream, with only the (further irreducible) operator latency of ξ. See the application example Fig. 4.

In this scope, the novelty of this paper is multiple. It is the only algorithm that combines all necessary features for efficient, parallel implementation of serial morphological operators. It uses a strictly *sequential* access to data, and can also run *in place*. The output is produced with *zero algorithm latency*. The algorithm runs in *linear time* w.r.t. the image size and *constant time* w.r.t. the SE size.

Its additional features include: very low *memory requirements*. A natural support of *floating point* data (not all previous algorithms can support floating point data). *The origin* can be arbitrarily placed within the structuring element, which is useful for even sized SE or specific SE decompositions.

2. Preliminaries

2.1. Morphological dilation and erosion

Let $\delta, \varepsilon : \mathcal{L} \to \mathcal{L}$ be a dilation and an erosion, performed on functions $f \in \mathcal{L}$, defined as $f : D \to V$. Below assume $D = \text{supp}(f) = Z^n$, $n = 1,2,\ldots$ and $V = Z$ or R. δ_B, ε_B are parameterized by a structuring element B, assumed rectangular and flat i.e. $B \subset D$ and translation-invariant.

Functional (operating on functions) erosion and dilation by a flat SE defined by extension to functions of the Minkowski set addition/subtraction definitions are given by

$$[\delta_B(f)](x) = \left[\bigvee_{b\in B} f_b\right](x) \tag{1}$$

$$[\varepsilon_B(f)](x) = \left[\bigwedge_{b\in \widehat{B}} f_b\right](x) \tag{2}$$

where $\widehat{\ }$ denotes the transposition of the structuring element, equal to a set reflection $\widehat{B} = \{x | -x \in B\}$, and f_b denotes the translation of the function f by some vector $b \in D$. Hence, the definitions Eqs. (1, 2) can be implemented by

$$[\delta_B(f)](x) = \max_{b\in B} f(x-b) \tag{3}$$

$$[\varepsilon_B(f)](x) = \min_{b\in B} f(x+b) \tag{4}$$

Dilations and erosions combine to form other operators. We shall focus on combinations obtained by concatenations that this algorithm implements optimally.

The basic products obtained by concatenation[1] are opening $\gamma_B = \delta_B\varepsilon_B$ and closing $\varphi_B = \varepsilon_B\delta_B$. Hence from, one forms the so called Alternating Filters obtained as $\gamma\varphi$, $\varphi\gamma$, $\gamma\varphi\gamma$ and $\varphi\gamma\varphi$. The number of combinations obtained from two filters is rather limited. Other filters can be obtained by combining two *families* of filters. This leads to morphological Alternate Sequential Filters (ASF), originally proposed by [37], and studied in [1] Chap. 10. In general, it is a family of operators parameterized by some $\lambda \in Z^+$, obtained by alternating concatenation of two families of increasing, resp. decreasing filters $\{\xi_i\}$ and $\{\psi_i\}$, such that $\psi_n \leqslant \cdots \leqslant \psi_1 \leqslant \xi_1 \leqslant \cdots \leqslant \xi_n$.

The most known ASF are those based on openings and closings, obtained by taking $\psi = \gamma$ and $\xi = \varphi$:

$$ASF^{\lambda} = \gamma^{\lambda}\varphi^{\lambda}\ldots\gamma^{1}\varphi^{1} \tag{5}$$

starting with a closing, and

$$ASF^{\lambda} = \varphi^{\lambda}\gamma^{\lambda}\ldots\varphi^{1}\gamma^{1} \tag{6}$$

starting with an opening.

This brief survey of theory allows to intuitively appreciate the complexity and the challenge involved by the usage of such long compound operators. If the algorithm does not deal at the same time with the memory management as well as with the latency, the overall performances could be (and generally they are) significantly lowered. We address this problem in Section 6 where we show how to efficiently implement the concatenation of dilations and erosions.

3. Principle of the algorithm

According to the algorithm classification presented in Section 1, the proposed algorithm belongs to the *SE decomposition* approach using an improved *1-D dilation algorithm*.

3.1. Separation of 2-D into 1-D

Recall that separable operators are run in all directions separately. This requires intermediate data storage between individual runs. If the 1-D parts use sequential data access, it allows to compose n-D dilations also using sequential data access. This eliminates the necessity of intermediate data storage.

The input image is read in the raster scan order, line by line. Every line is dilated horizontally. The result of the horizontal dilation is immediately read, pixel by pixel, by the vertical dilation in the corresponding column. The result of the vertical dilation part is written to the output. The output image is also written in the raster scan order.

Fig. 1a illustrates the computation of the result at position (k,l). Assume that the input data have already been read until line i, column j. The ongoing computations (depicted by $\times$) are: the horizontal dilation part (Fig. 1b) is running on line i, with the reading position (i,j) in the input image, writing position (i,l), immediately read by the vertical dilation (Fig. 1c) with reading position (i,l) and writing position (k,l), directly written to the output. The lines 1 to $i-1$ have already been horizontally dilated, and all columns have already been vertically dilated up to the line k.

3.2. 1-D algorithm

3.2.1. Elimination of useless values

An efficient coding of the function profile can avoid a number of comparisons during the computation of a dilation or an erosion. One can drop all values that will never take over in the result of the max or min, Eqs. (3, 4).

Consider a 1-D, connected structuring element B containing its origin. Then, computing $\delta_B f(x)$ needs only those values of $f(x_i)$ that can be seen from x when looking over the topographic profile of f. The valleys shadowed by mountains contain unneeded values, see Fig. 2. Notice that the masked values depend on f, and not on B.

Now, lets place ourselves in the context of streaming algorithms. For simplicity assume a *causal* SE, i.e. containing its origin at the right hand side. For causal SE, one only needs to look leftwards, over the past samples. The search of the useless values can be formalized by the Property 1 showing that values useless at some time instant x remain useless also for the "future".

Proposition 1 (*Useless values*). *In computing the dilation $\delta_B f$, with $f: Z^+ \to R$, by some causal, connected structuring element B (a linear segment) containing its origin, no $f(i)$ such that $f(i) \leqslant f(j)$, and $i < j$, will influence the dilation*

$$\delta_B f(x), \quad \text{for } \forall x \geqslant j \tag{7}$$

Proof. From Eq. 3, any x such that $i < j \leqslant x$, if $i \in B(x)$ then $j \in B(x)$. If $f(i) < f(j)$, then $f(i) < \max_{b\in B} f(x-b)$, and $f(i)$ has no impact on the dilation result.

This means that all $f(i)$ such that

$$f(i) \leqslant f(j), \quad \text{with } i < j, \tag{8}$$

may be dropped from the computations. □

This is a strong proposition that allows a considerable reduction of the computational redundancy. One comparison $f(i) \leqslant f(j)$, done upon reading $f(j)$, avoids computing $j - i$ useless comparisons for any later $B(x)$ that covers i and j.

Two important points are to be noticed.

1. $\forall x \geqslant j$ in Eq. 7 means that all values that become useless at the position j remain useless in the future, $\forall x \geqslant j$.
2. Using a bounded and causal $B \subset Z$, i.e. an interval $B(x) = [x-b, x]$, with $b < \infty$, means that for computing $\delta_B(x)$, one can also discard all values outside the SE span, i.e. $f(x_i)$, with $x_i < x - b$.

Remark 1. This proposition does not hold for non causal SE. The values useless at time x may become useful for some $k > x$. This algorithm utilizes the commutation of dilation with translation to convert an anti-causal SE to a causal one.

[1] To be read from right to left.

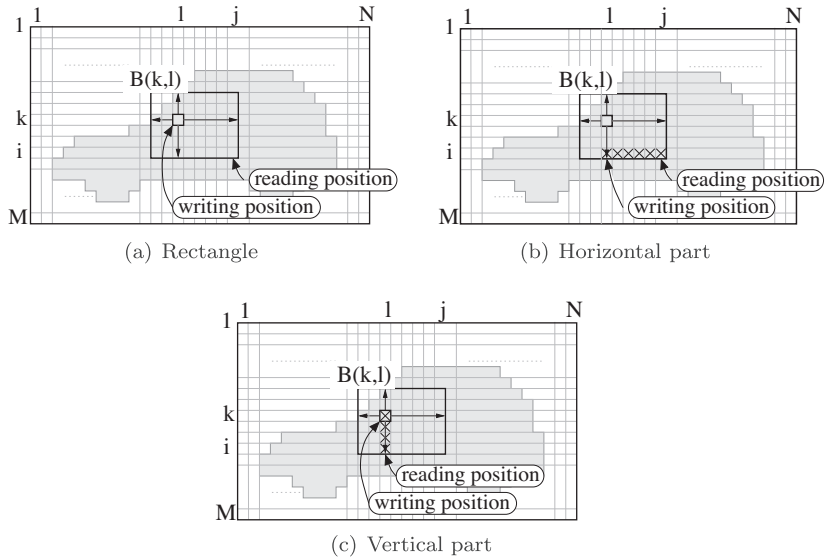


Fig. 1. Separation of computing: (a) rectangular element into (b) the horizontal and (c) vertical part. ×× denote the data stored in the queue of the corresponding line or column.

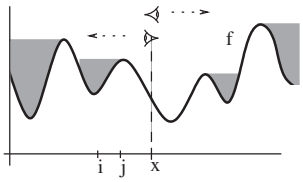


Fig. 2. Computing the dilation $\delta_B f(x)$: values in valleys shadowed by mountains when looking from x over the topographic relief of f are useless.

3.2.2. Anti-causal to causal SE conversion

Every SE B, $B \subset D$ is equipped by an origin $x \in D$. Assuming a sequential access to the input data, the dilation $\delta_B f(x)$ depends of points read before but also after x. We say that B is non causal.

One can transform a non causal SE to a causal SE by utilizing the property that dilation commutes with translation ($t \in D$)

$$\delta_{B+t} f(x) = \delta_B f(x - t) \tag{9}$$

The translation consists of writing the result at the correct place in the output. The horizontal and the vertical shifts are handled by the 1-D horizontal or vertical dilation part, implemented by the Function 1: page 8.

3.2.3. Function coding

Similarly as binary objects can be coded by using the distance to their boundaries, functions need to be coded by computing the distance to every change of the value. Using the Property 1 and relative indexing, the samples $f(x)$ used in computation $\delta_B f(x_i)$ are coded by pairs (*distance, value*) as given in Fig. 3.

The arrows indicate those values that enter in the computation of $\delta_B f(x_i)$. The values non indicated by an arrow are smaller or equal to $f(j) = 6$ and have no impact on the result $\delta_B f(x)$, for $x \geqslant j$.

The Eq. 8 is used by the 1-D dilation algorithm to exclude from the computation all useless values.

4. Algorithm complexity and latency

In this section we shall analyze the latency and the complexity of the algorithm.

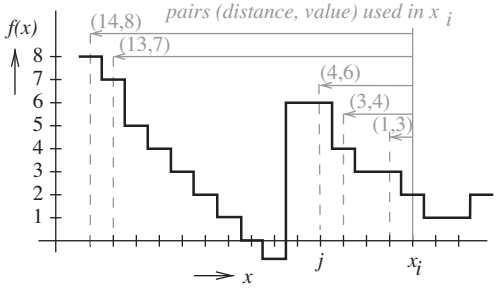


Fig. 3. Function coding. The useless values are discarded.

4.1. Latency

The overall algorithm latency is function of two factors: (i) the latency of the 1-D dilation (Fnct. 1), and (ii) the latency of the 2-D decomposition (Algorithm 2).

- 1-D dilation: the Function 1 writes the output as soon as the reading position *rp* reaches the last position covered by the structuring element (code lines 6 to 7). This corresponds to the last output-to-input data dependency position, i.e. the operator latency.

Remark 2. The *while* loop, clearing from the FIFO the useless values, operates on past signal samples. Consequently, it does not enter in the latency count. The latency of the Function 1 is therefore strictly equal to the operator latency.

- 2-D dilation: the 2-D dilation is decomposed in the way that result of the horizontal 1-D dilation is directly fed to the corresponding 1-D vertical dilations. Their results are recombined into the output stream. Therefore, the algorithm latency of the 2-D decomposition is zero.

4.2. Computation complexity

The 2D_Dilation algorithm iterates over all coordinates of the output image. The inner complexity of the 2D algorithm is $\mathcal{O}(N)$,

where N is the number of pixels in the image. At every coordinate, the 2D_Dilation calls twice the 1D_Dilation function: once for the vertical dilation and once for the horizontal dilation part.

The 1D_Dilation part (Function 1: 1D_Dilation) contains a sequence of $\mathcal{O}(1)$ operations, and one *while* loop (lines 1–2), clearing useless values from the FIFO. We shall see that this *while* loop is executed at most once per pixel, making the complexity of the 1D_Dilation algorithm constant per pixel.

Every incoming pixel is stored in the FIFO once and only once (line 5). Every pixel is cleared from the FIFO once and only once, either (i) when it becomes "too old", i.e. uncovered by the current SE span (lines 3–4) or (ii) when it gets masked by another, higher value (lines 1–2).

The deletion (lines 1–2) of every pixel can be delayed. Delete pixels from FIFO later or sooner has no impact on the algorithm complexity. However, it occurs that several pixels are deleted at the same time. This implies using the loop *while* (lines 1–2). The loop iterates at most once per pixel. Other pixels that become "too old" are deleted at lines 3–4. Hence, both ways of deletion have the same complexity $\mathcal{O}(1)$ per pixel.

This allows to make the following conclusions:

(1) The FIFO size is upper-bounded by the SE width. This determines the memory requirements (detailed in the next section).
(2) For every pixel, the number of the iterations of the *while* loop is lower-bounded by zero and upper-bounded by the SE width.
(3) The average number of iterations of the *while* remains in $[0,1]$ per pixel.
(4) The worst-case complexity of the 1D_Dilation per pixel is bounded by $\mathcal{O}(W)$.

Hence, we shall conclude that the overall complexity of the 2-D dilation algorithm is $\mathcal{O}(N)$, i.e. linear w.r.t the size of the image (N pixels) and constant w.r.t the SE size.

Note: although both ways of deletion have the same complexity $\mathcal{O}(1)$, they do not have the same cost (in terms of instruction count). The deletion by shadowing is slower in C because of the overhead of the loop *while*. The different timings obtained on various data (constant, random or natural images) are due to this overhead. For nonincreasing intervals (e.g. a constant image – see Benchmarks) the loop (lines 3–4) never executes. The probability of either deletion being data dependent explains the slight variation of the execution time on the image content.

5. Memory requirements

In 1-D, the FIFO size is upper-bounded by the width of the SE, which is the memory-worst case encountered whereever there are no useless data to eliminate. This occurs at all monotonically decreasing intervals of the signal that are longer than the SE width. This is equivalent to results obtained in [15], where for computing $\varepsilon_B f(x)$, only the values $f(x_i)$, with $x_i = B(x)$, are needed. Similar results hold for the dilation.

In 2-D, the memory requirements are given by the decomposition of the rectangle as $R = H \oplus V$ (R = rectangle, H/V = horizontal/vertical segment respectively). The raster-scan data access makes that the computations window (the SE) slides over the image from left to right and from top downwards.

The vertical part of the 2-D dilation runs at all columns simultaneously, one pixel per each column at a time. All columns have the memory-worst case equivalent to the height of the SE.

Consider an erosion (or a dilation) of an $N \times M$ image (width by height) by a $W \times H$ rectangular (width by height) SE. The memory requirements M are

$$M = 2(NH + W)$$

This means, N memory blocks of size $2H$ (vertical part) and one memory block of size $2W$ (horizontal part). The multiplicative factor 2 comes from the fact that the stored data are indexed by their coordinates (see Fig. 3, and Fnct. 1, line 5).

For example, an erosion of an 800×600 image by a 20×20 square will require $2 \times (800 \times 20 + 20) = 32{,}040$ bytes. Compared to this, storing an 800×600 image is costly, requiring 480,000 bytes (with 1 byte/pixel coding). Neither the input nor the output image need to be stored in memory.

The memory requirements graphically correspond to storing the image data from the lines currently intersected by the SE.

6. Application

In the following we give as example the implementation of an ASF^{λ} given by Eq. (5). Rewrite the filter as a concatenation of erosions and dilations

$$ASF^{\lambda} = \delta_{B_\lambda}\varepsilon_{B_\lambda}\varepsilon_{B_\lambda}\delta_{B_\lambda}\ldots\delta_{B_1}\varepsilon_{B_1}\varepsilon_{B_1}\delta_{B_1} \tag{10}$$

and reduce it into its canonical form

$$ASF^{\lambda} = \delta_{B_\lambda}\varepsilon_{B_\lambda\oplus B_\lambda}\delta_{B_\lambda}\ldots\delta_{B_1}\varepsilon_{B_1\oplus B_1}\delta_{B_1}$$

This ASF^{λ} can be implemented in a stream in one raster scan of the input image. The writing position of the preceding operator in the cascade becomes the reading position of the following operator. The operator latency of the entire *ASF* will be given by the one introduce by the result of Minkowski sum of all SE in Eq. 10, that is $\bigoplus_{i=1}^{n} B_i$.

Fig. 4 illustrates the propagation of real image data through an ASF^4 after having read approximately one third of the input image. The SE is a square of size $s+1$ for the s-th stage. The individual operators, with sequential data dependence, are running simultaneously. There is no intermediate data storage between the stages; the intermediate results are pipelined.

7. Benchmarks

This section illustrates the execution time of this algorithm, w.r.t. various criteria, measured on an Intel Core 2 2 GHz CPU, with 2 GB 800 MHz Dual Port RAM, running Linux. The time reported below is the processor time spent in the dilation/erosion algorithm as reported by a profiler (obtained as a mean after several runs to reduce inaccuracy).

The first experiment, see Fig. 5, illustrates the running time w.r.t. the content of the image. We have used a constant and a white-noise image to illustrate the fastest and the worst-case running time, and a natural image to illustrate the "expected" running time on a natural scene. The measured time follows a linear func-

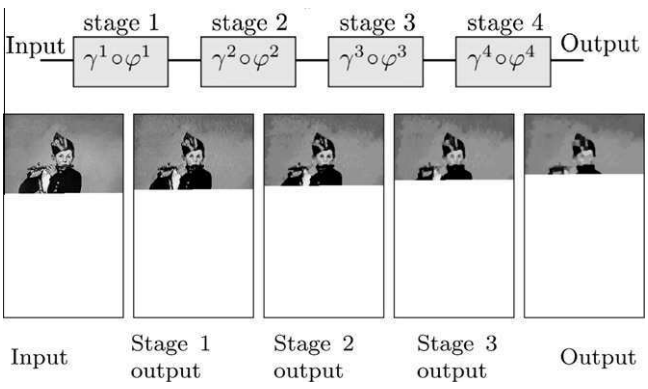


Fig. 4. Propagation of data throughout ASF^4 after having read approximately one third of input image (Manet's painting "Le fifre").

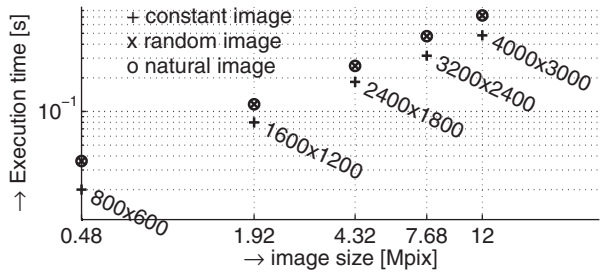


Fig. 5. Execution time of erosion, with respect to the content of the image. Data read from memory.

Table 3
Execution time in ms for 2-D dilation of the 'mountain.pgm' image (800 × 600) by a 21 × 21 square for various data coding types.

Data type/ CPU type	Int	Float	Unsigned char	Double
Intel Core2 Duo 2.4 GHz (32 bits) Dual Core	17.49	18.05	25.77	20.32
AMD Opteron 2.4 GHz (64 bits)	22.34	23.64	25.02	22.06

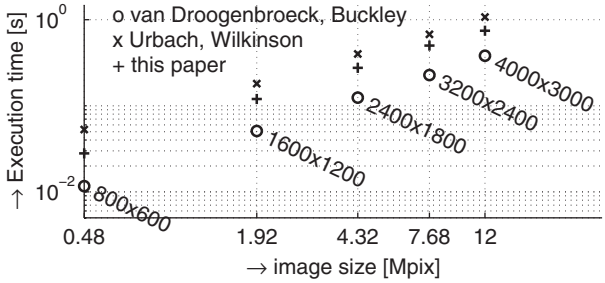


Fig. 6. Execution time of erosion, with respect to the size of image. Structuring element 21 × 21. Data read from memory.

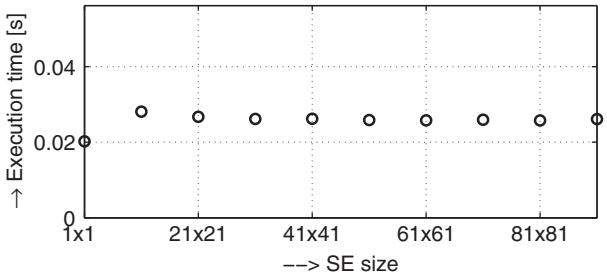


Fig. 7. Execution time of erosion with respect to the size of structuring element. Image size 800 × 600.

tion of the image size. Note also that the performance on the natural image actually coincides with the worst case performance obtained on the white noise. (The various sizes of the natural image were obtained by tiling side to side the original photography mountain.pgm from [38].)

We have evaluated the performance of the algorithm against different data types, see Table 3. The best performance has been obtained for the word width corresponding to the used CPU architecture, i.e. the integer and float for a 32 bit CPU, and the double for a 64 bit CPU. The penalty obtained for the unsigned char (8 bits) is due to inefficient memory access on both architectures.

We have compared our algorithm with Urbach-Wilkinson[2] [16] and Van Droogenbroeck–Buckley[3] [15], see Fig. 6.

[2] Code courtesy of Erik Urbach.
[3] Code available at [38].

The best timing was obtained with [15], the worst with [16], which allows – on the other hand – SE of arbitrary shapes. The algorithm from this paper remains competitive with the speed of the other algorithms, even though the speed has been traded off for memory consumption and latency.

Finally, the last experiment, see Fig. 7, illustrates that the complexity of the algorithm is independent of the SE size. The SE is a centered, $10k + 1 \times 10k + 1$ square, for $k = 0 \ldots 9$.

Note: the FIFO queues are implemented in C by using pointer-addressed arrays.

8. Conclusions

This paper proposes a new algorithm for functional dilation or erosion by a flat, rectangular structuring element for 2-D data (easily extensible to n-D images, and SE in form of n-D boxes).

The algorithm has zero algorithm latency and strictly sequential access to data. The combination of these two properties allows their inheritance to operators composed by concatenation. The entire concatenation chain, despite its internal sequential data dependence, can run simultaneously. The algorithm runs in linear time w.r.t. the image size and constant time w.r.t. the SE size.

Regarding serial filters, if all the operators in the concatenation run simultaneously – then result can also be obtained in constant time w.r.t. the length of the concatenation.

The algorithm has low memory requirements, which eases its implementation on systems with space constraints, such as embedded or mobile devices, intelligent cameras, etc. The linear time and zero latency allow efficient implementations on demanding industrial systems with severe time constraints.

Even though the speed is not the principal concern here, the algorithm remains competitive in term of execution time compared to the recent proposed fast implementations.

9. Future extensions

9.1. Other SE shapes

The algorithm, described above with rectangles, can also be extended to other shapes decomposable into linear segments (e.g. polygons as in [27]).

9.2. Spatially variant SE

This paper is the third step of a wider work towards an efficient implementation of morphological operations with spatially variant structuring elements that are useful for adaptive filters. The first step has been the stream implementation of dilation/erosion of sets [39]. It has the same algorithmic properties: zero latency and optimal memory, sequential access to data. The second step was the extension to the functional morphology; preliminary results have already been published: 1-D spatially variant morphology [40] and approximations of 2-D spatially variant rectangles [41].

The present algorithm is a simplified version of [41] limited from spatially variant to translation invariant SE. This simplification has brought a 10 × speed increase.

The goal is to obtain an algorithm for spatially-variant functional dilations and erosions with structuring elements of unconstrained shapes.

9.3. Real-time HW accelerator

This algorithm can be easily implemented as a finite state machine, interesting for HW implementation. The sequential access allows to read the image from a camera, process it, and write it out for

further processing or visualization. This allows processing large, or *infinite* industrial images without storing them in the memory.

Acknowledgement

The authors thank the anonymous reviewers for their comments and remarks that allowed to improve the paper.

Appendix A

A.1. Notation convention and preliminaries

$\leftarrow$ shall denote the assignment and =, <, $\leqslant$, ... tests. Curly braces denote a collection of values, e.g. $A \leftarrow \{5,8\}$. To obtain an element in a collection by its index we use brackets, e.g. $A[1] = 5$. The empty set is denoted by {}. In a function call, parentheses denote the collection of arguments; funct() is a function call without arguments.

FIFO: The algorithm uses FIFO (First In First Out) queues. The FIFO supports the following operations: *push* – insert a new element, *pop* – retrieve the oldest element, and *dequeue* – retrieve the most recent element, and queries: *front* – read the oldest element, *back* – read the newest element. The operations modify the content of the FIFO whereas the queries do not. `fifo` $\leftarrow$ {} initializes the fifo to empty.

In this algorithms, the elements inserted/read into/from the fifo are always couples {*value*, *position*}. Hence, e.g. the query `fifo.front()[l]` yields the *value* of the oldest element in the queue.

Input/**output** images are assumed 2D, read and written in the raster scan order one pixel at a time by x $\leftarrow$ in_stream.read() and out_stream.write(*x*). The reading/writing position is always implicitly incremented by 1.

A.2. Algorithm description

This section details the algorithm principles in link to the algorithm pseudo-code, see page 8.

A.2.1. 2-D Dilation algorithm

The 2-D Dilation, Algorithm 1, is to decompose the 2-D SE into columns and to assemble the partial 1-D computations into a 2-D stream, cf. Fig. 1.

The horizontal and vertical dilation parts are computed by the same function 1 D_DILATION. It encodes the input data from the current line or column and stores them in the FIFO. There is one FIFO for the horizontal part dilation – `h_fifo`. For the vertical part, there is an 1 ... *N* array – `v_fifo` – one FIFO per image column.

The input image is read at lines 10 to 12. Missing data (to the right of the image) are completed by the padding constant, line 14. The horizontal dilation is computed at line 16.

If the horizontal dilation part outputs a valid (non empty) value, line 19, it is sent to the vertical dilation part, computed by the same function, line 20. The vertical dilations also may require padding – typically below the image – where the horizontal dilation is not called. Instead, dFx is directly set to the padding value, line 18.

Provided the vertical dilation outputs a valid result, line 21, it is directly written to the output image, line 22.

A.2.2. 1-D Dilation function

Assume computing $dF = \delta_B F$, where F, $dF : [1,\ldots,N] \rightarrow R$. The structuring element B is a linear segment, `SEl + SE2 + l` pixels long, with `SEl`, `SE2` the offsets of the origin from the left- or the right-most end.

Calling conventions: The function `lD_Dilation`, see Function 1: page 8, is a function computing one sample of `dF`, to be written at writing position `wp`.

Upon every call, `rp` must be incremented by one by the calling function. Similarly, every time that `lD_Dilation` outputs a valid sample, `wp` is to be incremented by one.

The current fifo queue needs to be passed by reference.

Principle: The function proceeds in three steps:

1. *Dequeue all smaller or equal values*, lines 1 to 2. Removes from the FIFO all values that become useless.
2. *Delete too old value*, lines 3 to 4, removes from the FIFO the value that gets uncovered by the current SE *B*(*wp*).
3. *Enqueue the current sample* in the FIFO in the form of a couple {*value, position*}.
4. Provided enough data have been read, line 6, return the dilation result `dF`, line 7. At any moment, this value is found at the oldest position of the FIFO.

Note: To obtain erosion instead of dilation:

1) Function 1: dF $\leftarrow$ 1D_DILATION, line 1, replace $\leqslant$ by $\geqslant$.
2) Algorithm 2, line 1, set the padding constant PAD to ∞.

A.3. Algorithm pseudo-code

Function 1: dF $\leftarrow$ 1D_DILATION(*rp, wp, F, SE1, SE2, N, fifo*)

Input: rp, wp - reading/writing position; F - input signal value (read at rp); SE1, SE2 - SE size towards left and right; N - length of the signal; fifo - the FIFO
Result: dF - output signal value (to be written at wp)

```
   // Dequeue all queued smaller or equal values
1  while fifo.back()[1] ≤ F do
2  |  fifo.dequeue()

   // Delete too old value
3  if (wp - fifo.front()[2] > SE1) then
4  |  fifo.pop()

   // Enqueue the current sample
5  fifo.push({F, rp})

6  if rp = min (N, wp + SE2) then
7  |  return ( fifo.front()[1] ) ;                    // return a valid value
8  else
9  |  return ({}) ;                                   // return empty
```

Algorithm 2: 2D_Dilation

```
Input: in_stream - input image pixels stream
       M,N - height, width of the image
       SE1, SE2, SE3, SE4 - struct. element size
Result: out_stream
1  const. PAD ← 0 ;                                   // Set the Padding Constant
2  vfifo ← array [1..N] of FIFO ;                     // array of N empty FIFOs for the vertical dilation part
3  line_rd ← 1 ;                                      // read line counter
4  line_wr ← 1 ;                                      // written line counter
   // iterate over all image lines
5  while line_wr ≤ M do
6      hfifo ← FIFO ;                                 // FIFO for the horizontal part
7      col_rd ← 0 ;                                   // read column counter
8      col_wr ← 1 ;                                   // written column counter
       // iterate over all columns
9      while col_wr ≤ N do
           // horizontal dilation on the line_wr line
10         if line_rd ≤ M then
11             if col_rd < N then
12                 F ← in_stream.read()
13             else
14                 F ← PAD ;                          // Padding constant
15             col_rd ← min(col_rd +1, N)
16             dFx ← 1D_Dilation (col_rd, col_wr, F, SE1, SE3, N, hfifo)
17         else
18             dFx ← PAD ;                            // Padding constant
           // vertical dilation of the col_wr column
19         if dFx ≠ {} then
20             dFy ← 1D_Dilation (min(line_rd,M), line_wr, dFx, SE2, SE4, M, vfifo[col_wr ])
21             if dFy ≠ {} then
22                 out_stream.write(dFy)
23             col_wr ← col_wr +1
24     line_rd ← line_rd+1
25     if dFy ≠ {} then
26         line_wr ← line_wr +1
```

References

[1] J. Serra, Image Analysis and Mathematical Morphology, 2, Academic Press, NY, 1988.
[2] E. Dougherty, Mathematical Morphology in Image Processing, Taylor and Francis, Inc, 1992.
[3] P. Maragos, Pattern spectrum and multiscale shape representation, IEEE Trans. Pattern Anal. Mach. Intell 11 (7) (1989) 701–716.
[4] J. Serra, Morphological filtering: an overview, Signal Process. 38 (1) (1994) 3–11.
[5] S. Mukhopadhyay, B. Chanda, An edge preserving noise smoothing technique using multiscale morphology, Signal Process. 82 (4) (2002) 527–544.
[6] A. Cord, D. Jeulin, F. Bach. Segmentation of random textures by morphological and linear operators, in: Proceedings of International Symposium on Mathematical Morphology ISMM, 2007, pp. 387–398.
[7] R. Haralick, S. Sternberg, X. Zhuang, Image analysis using mathematical morphology, IEEE Trans. Pattern Anal. Mach. Intell 9 (4) (1987) 532–550.
[8] M. Wilkinson, J. Roerdink, editors. Mathematical Morphology and Its Application to Signal and Image Processing, in: Proceedings of 9th International Symposium on Mathematical Morphology, Springer, 2009.
[9] P. Salembier, P. Brigger, J. Casas Montse Pardas, J. Casas, Morphological operators for image and video compression, IEEE Trans. Image Process. 5 (1996) 881–897.
[10] P. Soille, M. Pesaresi, Advances in mathematical morphology applied to geoscience and remote sensing, IEEE Trans. Geosci. Remote Sensing 40 (9) (2002) 2042–2055.
[11] R. Sabourin, G. Genest, F. Prêteux, Off-line signature verification by local granulometric size distributions, IEEE Trans. Pattern Anal. Mach. Intell 19 (9) (1997) 976–988.
[12] L. Vincent, Granulometries and opening trees, Fundamenta Informaticae 41 (1-2) (2000) 57–90.
[13] M. van Herk, A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels, Patt. Recog. Lett. 13 (7) (1992) 517–521.
[14] J. Gil, R. Kimmel, Efficient dilation, erosion, opening, and closing algorithms, IEEE Trans. Pattern Anal. Mach. Intell 24 (12) (2002) 1606–1617.
[15] M. Van Droogenbroeck, M. Buckley, Morphological erosions and openings: fast algorithms based on anchors, J. Math. Imaging Vis. 22 (2-3) (2005) 121–142.
[16] E. Urbach, M. Wilkinson, Efficient 2-D grayscale morphological transformations with arbitrary flat structuring elements, IEEE Trans. Image Process. 17 (1) (2008) 1–8.
[17] D. Lemire, Streaming maximum-minimum filter using no more than three comparisons per element, Nordic J. Comput. 13 (4) (2006) 328–339.
[18] F. Lemonnier, J.-C. Klein, Fast dilation by large 1D structuring elements, in: IEEE International Workshop on Nonlinear Signal and Image Processing, Halkidiki, Greece, 1995.
[19] L. Najman. Skew detection. European Patent, Filled at 27, 2002 as a European filing the French Patent Office, 2002.
[20] B. Obara, Identification of transcrystalline microcracks observed in microscope images of a dolomite structure using image analysis methods based on linear structuring element processing, Comput. Geosci. 33 (2) (2007) 151–158.
[21] T. Huang, G. Yang, G. Tang, A fast two-dimensional median filtering algorithm, IEEE Trans. Acoustics Speech Signal Process. 27 (1) (1979) 13–18.
[22] B. Chaudhuri, An efficient algorithm for running window pel gray level ranking 2-D images, Patt. Recogn. Lett. 11 (2) (1990) 77–80.
[23] M. Van Droogenbroeck, H. Talbot, Fast computation of morphological operations with arbitrary structuring elements, Patt. Recog. Lett. 17 (14) (1996) 1451–1460.
[24] X. Zhuang, Decomposition of morphological structuring elements, J. Math. Imaging Vis. 4 (1994) 5–18.

[25] X. Zhuang, R. Haralick, Morphological structuring element decomposition, Comput. Vis. Graphics Image Process. 35 (1986) 370–382.
[26] G. Anelli, A. Broggi, G. Destri, Decomposition of arbitrarily shaped binary morphological structuring elements using genetic algorithms, IEEE Trans. Pattern Anal. Mach. Intell 20 (2) (1998) 217–224.
[27] P. Soille, E. Breen, R. Jones, Recursive implementation of erosions and dilations along discrete lines at arbitrary angles, IEEE Trans. Pattern Anal. Mach. Intell 18 (5) (1996) 562–567.
[28] G. Matheron, Random Sets and Integral Geometry, John Wiley and Sons, New York, 1975.
[29] J. Pecht, Speeding up successive Minkowski operations, Patt. Recog. Lett. 3 (2) (1985) 113–117.
[30] R. van den Boomgaard, D. Wester, Logarithmic Shape Decomposition, in: C. Arcelli, L.P. Cordella, G. Sanniti di Baja (Eds.), Aspects of Visual Form Processing, World Scientific Publishing Co., Singapore, 1994, pp. 552–561. Capri, Italy.
[31] D. Coltuc, I. Pitas, On fast running max-min filtering, IEEE Trans. Circuits Syst. II Analog Digital Signal Process. 44 (8) (1997) 660–663.
[32] J. Gil, M. Werman, Computing 2-D min, median, and max filters, IEEE Trans. Pattern Anal. Mach. Intell 15 (5) (1993) 504–507.
[33] D. Gevorkian, J. Astola, S. Atourian, Improving Gil–Werman algorithm for running min and max filters, IEEE Trans. Pattern Anal. Mach. Intell 19 (5) (1997) 526–529.
[34] T. Miyatake, M. Ejiri, H. Matsushima, A fast algorithm for maximum–minimum image filtering, Syst. Comput. Japan 27 (13) (1996) 74–85.
[35] C. Clienti, M. Bilodeau, S. Beucher, An efficient hardware arhcitecture without line memories for morphological image processing, In Advanced Concepts for Intelligent Vision Systems, 2008.
[36] D. Lemire, Faster retrieval with a two-pass dynamic-time-warping lowerbound, Patt. Recog. 42 (2009) 2169–2180.
[37] S. Sternberg, Grayscale morphology, Comput. Vis. Graph. Image Process 35 (3) (1986) 333–355.
[38] R. Dardenne and M. Van Droogenbroeck. The libmorpho library. Available for download from <http://www2.ulg.ac.be/telecom/research/libmorpho.html>.
[39] H. Hedberg, P. Dokládal, V. Öwall, Binary morphology with locally adaptive structuring elements algorithm and architecture, IEEE Trans. Image Process. 18 (3) (2009).
[40] P. Dokládal, E. Dokládalova, Grey-Scale 1-D dilations with spatially-variant structuring elements in linear time, in: European Signal Processing Conference 2008, 2008.
[41] P. Dokládal, E. Dokládalova, Grey-scale morphology with spatially-variant rectangles in linear time, in: Advanced Concepts for Intelligent Vision Systems, 2008.

One-dimensional openings, granulometries and component trees in $\mathcal{O}(1)$ per pixel

Vincent Morard, Petr Dokládal and Etienne Decencière

Abstract—We introduce a new, efficient and adaptable algorithm to compute openings, granulometries and the component tree for one-dimensional (1-D) signals. The algorithm requires only one scan of the signal, runs in place in $\mathcal{O}(1)$ per pixel, and supports any scalar data precision (integer or floating-point data).

The algorithm is applied to two-dimensional images along straight lines, in arbitrary orientations. Oriented size distributions can thus be efficiently computed, and textures characterised.

Extensive benchmarks are reported. They show that the proposed algorithm allows computing 1-D openings faster than existing algorithms for data precisions higher than 8 bits, and remains competitive with respect to the algorithm proposed by Van Droogenbroeck when dealing with 8-bit images. When computing granulometries, the new algorithm runs faster than any other method of the state of the art. Moreover, it allows efficient computation of 1-D component trees.

Index Terms—Algorithms, Mathematical Morphology, Opening, Granulometry, Component Tree, Oriented size distribution, Filtering.

1 Introduction

In the framework of mathematical morphology [1], [2], any anti-extensive, increasing and idempotent operator is an algebraic *opening*. This fundamental family of operators is often based on a structuring element (SE) probing the image at different places; in this case, it is called a morphological opening. Using a segment as structuring element is useful to detect straight structures, or to find the local orientation of thin objects. Indeed, many practical applications involve a directional analysis (Material characterisation, crack detection, biological applications [3], [4]).

In this paper, the presentation is limited to openings, but all results can be directly applied to their dual operators, the closings.

Openings can be used to build *granulometries* [5]–[7]. This tool was initially introduced to study porous media [5] and it can be seen as a sieving process. Given some powder composed of particles of different radii, sieves of decreasing size are used to perform a size analysis of these particles, by measuring the quantity of powder left in each sieve. Many image processing applications involve granulometries, size distribution, image segmentations or texture characterisations [7]–[9].

The author are with MINES ParisTech, CMM - Centre of Mathematical Morphology, 35, rue St. Honoré, 77305-Fontainebleau-Cedex, France.

Multi-scale image analysis can also be based on the *component tree* (or *max-tree*). Introduced by Salembier [10], it captures some essential features of an image. This tree structure is used in many applications including image filtering, image segmentation, video segmentation, and image compression [11]–[13]. It is also at the basis of the topological watershed [14].

All these operators are time consuming with naive implementations and many authors have developed fast and efficient algorithms to deal with this issue.

For morphological openings (i.e. openings using a structuring element), Pecht [15] defined in 1985 a logarithmic decomposition of the structuring element. This decomposition removes most of the redundancy. Later, Van Herk [16] on the one side, and Gil and Werman [17] on the other side, reduced the complexity to a constant per pixel. This algorithm, called hereafter HGW, is independent of the size of the structuring element for the computation of one-dimensional (1-D) erosions and dilations. Later, Clienti et al. improved HGW algorithm by removing the backward scanning to ensure a low latency [18]. Then, algorithms have also been proposed to compute openings in only one pass of the entire image, without computing successively the erosion and the dilation. Van Droogenbroeck and Buckley developed an algorithm based on so-called anchors [19]. The algorithm uses image histogram. It is extremely efficient on 8-bit data, but its performance decreases with higher precision data. Later, Bartovský et al. [20] worked on a new streaming algorithm with a minimal latency and a low memory requirement.

For granulometries, a straightforward approach consists in computing a set of openings of different sizes, and measuring the residues between two successive openings. This is a very computationally intensive task. Vincent proposed an efficient algorithm based on the recursive analysis of the regional maxima of the signal [21]. This algorithm is faster by several orders of magnitude over the naive implementation and highly contributed to the diffusion of this tool in the image processing community.

In the literature, many authors have worked on the 1-D component tree. Among them, Najman and Couprie [22] built an algorithm in quasi linear time and more recently, Menotti et al. [23] downed the complexity to a

constant per pixel.

In this never-ending struggle for faster algorithms, we propose a new algorithm, which competes favourably with existing ones when computing openings, and also allows computing granulometries and component trees - all this in one dimension. It can also deal with various data accuracy (it is not limited to integers). In fact, in can be applied to any kind of data, as long as the values belong to an ordered group. A first, shorter, description of this work was previously presented by the authors [24].

This paper is organised as follows: section 2 recalls the basic notions on attribute openings and granulometries, whereas section 3 describes the algorithm to build granulometries, openings and the component tree for 1-D signals. Then, section 4 applies this algorithm to two-dimensional (2-D) images, and finally, sections 5 and 6 study the complexity and the timings through a comparison with the state-of-the-art.

2 Basic notions

In this section, the definitions of one-dimensional attribute openings and size distributions are recalled. Moreover, it is explained how to apply them to two-dimensional images.

2.1 Attribute openings

Let $X : D \to \{0;1\}$ be a binary signal, where D is an interval of $\mathcal{Z}$ such as $[1, N]$. We define $\{X_i\}$ as the collection of connected components (CC) of X and X_i the i^{th} element of this set. Note that in 1-D these CCs are intervals of $\mathcal{Z}$. We wish to keep or delete these CCs according to an attribute associated to a criterion χ (*e.g.* *"the length is larger than λ"*). Formally, χ is a function mapping the set of CCs of D into $\{false, true\}$, which allows to define a function ψ:

$$\psi_\chi(X_i) = \begin{cases} X_i & \text{if } \chi(X_i) \text{ is true} \\ \emptyset & \text{otherwise,} \end{cases} \quad (1)$$

for all CCs X_i included in D. Based on this function, the corresponding attribute opening can be introduced for all binary signals X:

$$\gamma_\chi(X) = \bigcup_{X_i \in \{X_i\}} \psi_\chi(X_i). \quad (2)$$

Hereafter, the chosen attribute will be the length of the CC. The resulting criterion will be "is longer than or equal to λ", and the corresponding opening will be denoted γ_λ. However, other attributes can be used, for example, *whether the CC contains a point from another set* (which would allow building openings by reconstruction).

From now on, consider $f : D \to V$, with V equal to $\mathcal{Z}$ or $\mathcal{R}$. Let $X^h = \{x | f(x) \geq h\}$ be the set obtained by thresholding f at level h. Recall that an opening is an increasing, anti-extensive and idempotent operator. Thanks to the increasingness, the opening commutes with the thresholding. Thence, the extension of openings to grey level images is direct:

$$\gamma_\chi(f) = \vee\{h \in V \mid x \in \gamma_\chi(X^h(f))\}. \quad (3)$$

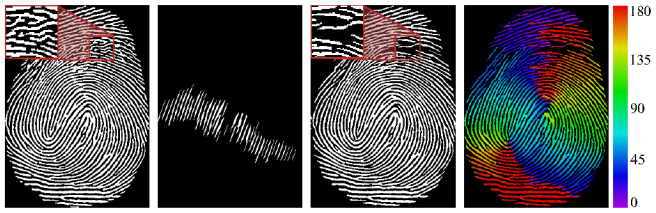
We will introduce in section 3 an efficient algorithm to compute these linear openings. In order to apply it to two-dimensional images, we can consider a given orientation α, and decompose the image into one-dimensional signals, following this orientation (see section 4 for details). Therefore, we get a directional opening written γ^α_χ. Further directional operators can be built with these linear openings. The first one is $\vee\gamma_\chi(f)$, which is computed by taking the supremum of the attribute openings in all orientations:

$$\vee\gamma_\chi(f) = \bigvee_{\alpha \in [0,180[} \gamma^\alpha_\chi(f) \quad (4)$$

The second one, $\zeta_\chi(f)$, can extract the local orientation of the structures by storing, on each pixel, the angle of the opening producing the highest grey value:

$$\zeta_\chi(f) = \operatorname*{argsup}_{\alpha \in [0,180[} \gamma^\alpha_\chi(f). \quad (5)$$

Fig. 1 illustrates the results of these operators on a fingerprint image.



(a) Input (b) γ^α_χ (c) $\vee\gamma_\chi$ (d) ζ_χ

Fig. 1. Application of linear openings with the criterion χ : *"length* $\geq$ 21 pixels". (a) initial image with a zoom on a part of the image. (b) oriented filtering: only straight structures longer than 21 pixels oriented at $\alpha = 70$ degrees are kept. (c) preservation of all linear structures longer than 21 pixels. (d) local orientation in false colours.

2.2 Size distribution

A size distribution, also called granulometry, is built from openings. As proposed by Matheron [5], a family $(\gamma_\nu)_{\nu \geq 0}$ of openings is a granulometry if, and only if:

$$\forall \nu \geq 0, \forall \mu \geq 0, \ \nu \geq \mu \Rightarrow \gamma_\nu \leq \gamma_\mu. \quad (6)$$

Later, Maragos introduced the pattern spectrum [7]:

$$(PS(f))(\nu) = \frac{-d(Meas(\gamma_\nu(f)))}{d\nu}, \quad \nu > 0, \quad (7)$$

with *Meas*(), a given additive measure. In the discrete case, the differential function is replaced by a subtraction between two consecutive openings. By analysing these residues, we get the measure of all the structures that have been removed from the image at this scale. Therefore, the discrete pattern spectrum is defined as follows:

$$(PS(f))(\nu) = Meas(\gamma_\nu(f) - \gamma_{\nu-1}(f)), \quad \nu > 0. \tag{8}$$

Fig. 2 explains how a 1-D signal is decomposed. The pattern spectrum is saved into a discrete histogram, where each bin stores the contribution of the signal to its corresponding measure. Hereafter, the measurement used in equations 7 and 8 is the volume, and the family of openings are the openings with the length attribute, γ_λ. Hence, block c_3 is a 5 pixels long element having a volume of 15; this adds 15 to the 5^{th} bin. Elements c_4 and c_6 have a length of 1 pixel; therefore, they contribute to the first bin – and so on, until all the elements have been processed.

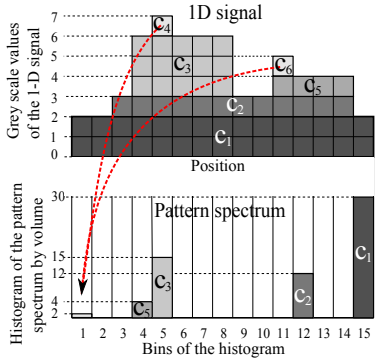


Fig. 2. Illustration of a grey scale pattern spectrum with the volume measurement on a one-dimensional signal.

Openings and granulometries are computed efficiently with the algorithm presented in the next section.

3 Algorithm for 1-D signals

This algorithm is able to compute granulometries, component trees and attribute openings by length for a 1-D signal. We first describe the decomposition used to get a minimal and complete representation of the signal. Then, we provide a detailed description of the algorithm.

3.1 Signal decomposition

Consider a 1-D signal $f : D \to V$, with V equal to $\mathcal{R}$ or $\mathcal{Z}$. We recall that $X^h = \{x \mid f(x) \geq h\}$ denotes the threshold of f at level h, and $\{X_i^h\}$ the set of connected components of X^h. Notice that one may obtain the same connected component for different h. We wish to obtain a representation of f by searching, for each X_i^h, for the maximum h allowing to extract it.

First, we will re-index $\{X_i^h\}$ into $\{X_j\}$. We will call *cord* a couple $c = (X_j, k)$ belonging to $\{X_j\} \times V$, where $k = \min_{x \in X_j} f(x)$ is the altitude of the cord. As X_j is an interval of $\mathcal{Z}$, we can write it $[sp, fp]$, where sp and fp denote its starting and end positions. Its length is $L(c) = fp - sp + 1$. Fig. 3(a) illustrates the decomposition of a 1-D signal into its cords.

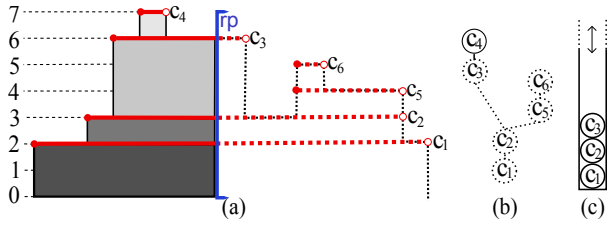


Fig. 3. (a) 1-D signal and the associated cords, (b) the component tree and (c) the current state of the stack (given the reading position rp). (The continuous/dotted line means already/not yet known elements.)

The length satisfies the inclusion property, since we have, for all cords $c_i = (X_i, k_i)$ and $c_j = (X_j, k_j)$ of f, such that $X_i \subset X_j$, $k_i > k_j$. We say that c_j is an *ancestor* of c_i and c_i is a *descendant* of c_j. The longest descendant of c_j (with respect to the cord length defined above) is its *child*. If c_i is the child of c_j, then c_j is the *parent* of c_i. With the parent-child relationship, we get a tree called component tree, or max-tree.

Given a cord $c_i = (X_i, k_i)$ and its parent $c_j = (X_j, k_j)$, the volume of cord c_i is defined as:

$$V(c_i) = (k_i - k_j)L(c_i). \tag{9}$$

The reconstruction of a signal f from its set of cords $C = \{(X_i, k_i)\}$ is straightforward:

$$f(x) = \max_{(X_i,k_i) \in C \,:\, x \in X_i} k_i. \tag{10}$$

Additionally, attribute openings by length and the pattern spectrum of f can be directly computed on C:

$$\gamma_\lambda(C) = \{c_i | L(c_i) \geq \lambda\}, \tag{11}$$

$$(PS(C))(\lambda) = \sum_{L(c_i)=\lambda} V(c_i). \tag{12}$$

An efficient decomposition of a function into its set of cords allows an efficient computation of openings, pattern spectra and component trees by using only logical or arithmetic operations. The following section presents the 1-D algorithm. Its efficiency stems from several facts: the signal is read sequentially; every cord is visited once, and only once, in the order child-parent. Later, we will see that all operations, including finding the maximum in Eq. 10, are done in $\mathcal{O}(1)$ per pixel.

3.2 Algorithm principle

By analysing Fig. 3, we immediately notice a couple of properties of the signal:

1) Every uprising edge is a starting point of at least one cord. Every downfalling edge is the end of at least one cord.
2) When we read f from the left to the right, the length of every cord is only known when the reading position reaches its end. In Fig. 3, the already known portion of each cord is represented with a continuous line - up to rp - and the still unknown portion with a dotted line.
3) Every cord can only be processed when the reading position reaches its end.
4) The incipient cords, waiting to be processed, can be stored in a Last-In-First-Out (LIFO) structure. The stored cords are necessarily ordered according to the inclusion relation (Fig. 3).

These properties are fundamental to get an efficient algorithm.

3.3 Algorithm pseudo code

Alg. 1 reads the input signal sequentially, from left to right (lines 4 and 5); rp denotes the current reading position.

Algorithm 1: (f_{out} or PS or CompTree) ← *ProcessSignal1D (f, λ, op)*

Input: $f : [1 \dots N] \to \mathcal{R}$ - input signal;
λ, parameter for the opening
op, selected operator

Result: $f_{out} = \gamma_\lambda f$ or PS or $CompTree$

```
1 Stack ← ∅;
2 CompTree ← ∅;
3 PS[1..N] ← 0;
4 for rp = 1..N do
5 |  (f_out or PS or CompTree)← ProcessPixel(f(rp), rp, λ, Stack, op)
6 Process Remaining Cords
```

The cords are coded by a couple (sp, k), where sp corresponds to the starting position, and k to the altitude. The pending cords (of yet unknown length) are stored in a LIFO-like $Stack$ supporting the following operations: $push()$, $pop()$ and queries $top()$ and $empty()$. Therefore, reading an attribute of the latest-stored cord is $Stack.top().att$ with att referring to k or sp. Inserting a new cord into the stack will be written: $Stack.push(k, sp)$ while removing a cord: $cordOut = Stack.pop()$. At the beginning $Stack$ and $CompTree$ are empty, and the pattern spectrum PS is filled with zeros (lines 1 to 3).

Algorithm 2: (f_{out} or PS or CompTree) ← *ProcessPixel* (k, rp, λ, Stack, op)

Input: $k = f(rp)$
rp, the reading position
λ, parameter for the opening
$Stack$, stack of cords (LIFO)
op, selected operator

Result: f_{out} or PS or $CompTree$ following op

```
1  if Stack.empty() or k > Stack.top().k then
2  |  Stack.push(k, rp, FALSE);
3  else
4  |  while k < Stack.top().k do
5  |  |  cordOut = Stack.pop();
6  |  |  if Op ==Size distribution then
7  |  |  |  Length = rp − cordOut.sp;
8  |  |  |  PS[Length]+ =
   |  |  |  Length × (cordOut.k − max(k, Stack.top().k))
9  |  |  if Op ==Opening then
10 |  |  |  if cordOut.Passed or rp − cordOut.sp ≥ λ
   |  |  |  then
11 |  |  |  |  f_out ← WriteCords(cordOut, Stack, rp);
12 |  |  |  |  Stack.push(k, rp, TRUE);
13 |  |  |  |  break
14 |  |  if Op ==Component tree then
15 |  |  |  if Stack.empty() or Stack.top().k < k then
16 |  |  |  |  currentNode.k = k;
17 |  |  |  |  currentNode.Children.push(nodeOut);
18 |  |  |  |  Stack.push(currentNode);
19 |  |  |  |  break
20 |  |  |  else
21 |  |  |  |  Stack.top().Children.push(nodeOut)
22 |  |  if Stack.empty() or k > Stack.top().k then
23 |  |  |  Stack.push(k, cordOut.sp, FALSE);
24 |  |  |  break
```

Each pixel rp is processed by Alg. 2, processing differently the rising and falling edges of the signal:

- Uprising edge (Alg. 2, line 1): is the beginning of at least one cord, we store its position and altitude in $Stack$ (line 2).
- Downfalling edge: is the end of, at least, one cord. The while cycle (lines 4 to 24) pops from $Stack$ all ending cords to process them one by one. At this point, the processing of the ending cords depends of the operator:
  - *Size distribution :* We compute the cord's length and add its contribution to the corresponding bin (indexed $Length$), lines 7 and 8. If the stack is empty, a query *top* to the stack will return 0.
  - *Opening :* We test the length of the cord (Eq. 11) to discard those shorter than λ, line 10. Whenever we find any cord longer than λ, we immediately

reconstruct (Eq. 10) the opening $f_{out} = \gamma_\lambda(f)$ up to the current reading position rp (lines 10 to 13). The principle stems from the reasoning that the length of every cord is only known when we reach its end. As soon as we find the first cord longer or equal to λ, from the inclusion property we know that all cords stored in the LIFO are (strictly) longer than λ. We do not need to wait until their end to reconstruct the output up to rp. We reconstruct the opening result f_{out} using the function $WriteCords()$. Thanks to the inclusion-ordered LIFO we ensure that every pixel is written only once. In the function $WriteCords()$, the stack is emptied while we write the cords (see Alg. 3). Hence, we add a flag $Passed$ to the cord structure to tell whether a cord is longer than λ. Finally, we push the current cord into the stack, with the flag $Passed$ set to $true$ (line 12). This flag is essential, as we will not be able to access its length later on.

– *Component tree :* We enrich the *cord* structure by a new attribute *Children*. It is a list of pointers on the cord structure. This attribute links every parent to its children. Every ending cord needs to be linked to its parent. Finding the correct parent component involves three possible situations:
  * If the stack is empty, we link $cordOut$ with $currentCord$ (lines 16, 17 and 18).
  * If the grey value of the new top-most node in the stack is lower than $currentCord$ grey value, we also link $cordOut$ with $currentCord$ (lines 16, 17 and 18).
  * Otherwise, we link $cordOut$ with the top-most cord in the stack (line 21).

At the end of the 1-D signal (Alg. 1, line 6), some cords may remain in the stack. We empty the stack and process all the remaining cords according to the operator op.

Algorithm 3: $f_{out} \leftarrow$WriteCords(cordOut, Stack, rp)

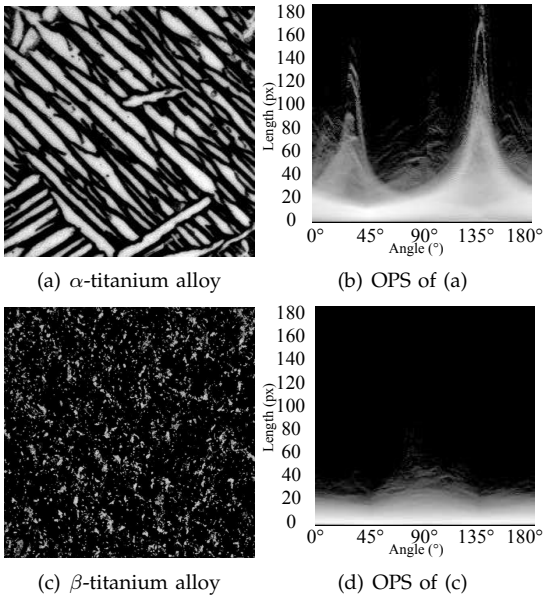
Input: $cordOut$, last cord popped
$Stack$, stack of cords (LIFO)
rp, reading position

Result: $f_{out} = \gamma_\lambda f$

1 $f_{out}[cordOut.sp : rp] = cordOut.k$;
2 **while not** $Stack.empty()$ **do**
3 $\quad end = cordOut.sp$;
4 $\quad cordOut = Stack.pop()$;
5 $\quad f_{out}[cordOut.sp : end] = cordOut.k$

We notice that this algorithm only needs comparison operations and subtractions between values. Therefore, it can handle a large variety of data types, including integer and floating point. In fact, from an algebraic point of view, the set of values needs only to have the structure of an ordered group.

The complexity is studied in section 5 and this algorithm is applied to 2-D images in the next section.



(a) α-titanium alloy (b) OPS of (a)

(c) β-titanium alloy (d) OPS of (c)

Fig. 4. Oriented pattern spectrum for α- and β- titanium alloys. See the text for explanation.

4 Application to 2-D images

It will be explained in this section how to apply the previous algorithm to 2-D images, by means of partitioning the image support into thin straight lines, at arbitrary orientations. This strategy is applied to the computation of Oriented Pattern Spectrum (OPS) [7]. Finally, some hints to compute the 2-D component tree are given.

In this section, $g : E \to V$ is a 2-D image, where E is rectangular domain of $\mathcal{Z}^2$ of the sort $[1, N_1] \times [1, N_2]$, and V, as previously, is equal to $\mathcal{Z}$ or $\mathcal{R}$.

4.1 2-D image scanning strategy

Alg. 2 takes one pixel as input, and is clearly independent of the orientation of the line. Hence, we can apply it to 2-D images, provided we have an appropriate image-scanning strategy. Soille et al. [25] described a way to go through all pixels of an image at a given orientation, by using Bresenham lines [26]. Moreover, they addressed the padding problems by adding constraints to avoid any overlaps between two translated lines. Hence, the logic behind the construction of these lines ensures that each pixel will be processed only once. This allows the algorithm to run in place. For openings in arbitrary orientation, we add a line buffer to store the index position of all the previous pixels of the line. Hence, we could easily write the result of the filter in the output image with no other extra computation.

Hereafter, image g is decomposed into a set of 1-D signals $\{g_{\alpha,k}\}_{k \in K}$, following direction α.

4.2 Oriented Pattern Spectrum

We have seen that the local orientation on a given pixel of a 2-D image can be measured by the supremum of linear openings. We may wish to additionally measure the pattern spectrum for each orientation, which leads to the Oriented Pattern Spectrum (OPS), initially introduced by Maragos [7]:

$$OPS(g)(\lambda,\alpha) = \sum_{k\in K} (PS(g_{\alpha,k}))(\lambda). \quad (13)$$

Computing the OPS can be very time consuming. Using the presented algorithm reduces its computation time. Fig. 4 illustrates the results of this operator. Oriented pattern spectra are represented as 2-D images, one column for each orientation. Fig. 4 (b) and (d) show the OPS of the (a) α- and (c) β- titanium alloys, respectively. We can see in (b) two peaks, giving evidence of an alignment in the image. The majority of the structures are oriented around 140° (measured anticlockwise from the horizontal line), with a second peak around 40°. The majority of the structures are up to 80 pixels long, with several individuals from 120 to 160 pixels long. The β-titanium alloy is rather isotropic, with only a slight alignment around 90°.

4.3 Border effects

Consider a stationary, random process ξ of arbitrarily placed, L-pixel long, non intersecting and non overlapping, straight lines, oriented in a constant direction ϑ. The $PS(\xi)$ in the direction of ϑ is $\delta(L)$, the Dirac function at L. Now, consider a bounded, discrete support $[1,N]^2 \subset \mathcal{Z}^2$, and the realisation of ξ on D, see Fig. 5. The intersection with a finite support introduces in the PS a bias (a.k.a. border effects) due to the truncation of the structures in ξ (see the red curve in Fig. 5).

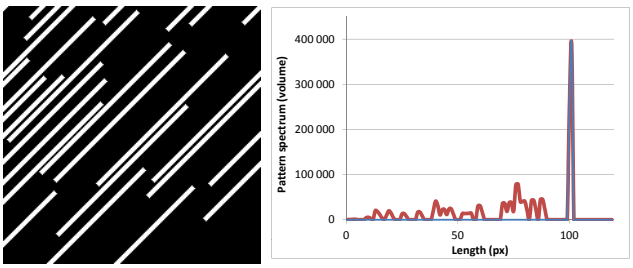


Fig. 5. Randomnly placed, 100-pixel long, straight lines and the pattern spectrum: blue - expected pattern spectrum, red - effect of truncation on a bounded support.

The border effect is an ubiquitous problem, differently handled in various applications. Van Droogenbroeck [19] proposes an interesting discussion. He recommends to add the maximal value of V (let us call it ∞) outside the image support. On the other hand, in the domain of connected filters, one often completes by adding the minimal value of V.

In granulometries one often uses padding by ∞. Indeed, ∞ is a recessive value that allows considering in Eq. 7 the truncated structures (of unknown length) as infinitely long, and makes them unaffected by γ_ν, for any $\nu < \infty$.

The proposed algorithm can easily handle both border management strategies. Considering the infinite extension, every cord touching the border is ignored. For openings, the flag *Passed* of the first cord pushed into the stack and the cords remaining in the stack at the end of the line, must be set to true. The timings stay unchanged with this strategy.

4.4 2-D component tree

If we compute a tree for each 1-D image $g_{\alpha,k}$ obtained from the 2-D image g, we get a set of trees called a forest. On its own, such a forest is not interesting, since it does not describe the 2 dimensional patterns of the images. However, Wilkinson et al. [27], and later Matas et al. [28], [29] have described a method to merge all these trees to get the 2-D component tree of the image. Hence, the proposed algorithm can be seen as a part of a new process to get the 2-D component tree in an efficient way.

5 Complexity

The computational complexity of this algorithm is evaluated focusing first on the 1-D algorithm. Then, we study the 2-D part.

5.1 1-D scan strategy

Consider a 1-D signal $f : [1 \dots N] \to V$, with $V = \mathcal{Z}$ or $\mathcal{R}$. The input signal is read sequentially from left to right (Alg. 1, lines 4 to 6) and it calls Alg. 2 once per every sample.

By analysing Alg. 2, we notice that every cord, where it starts, is pushed once and only once into the LIFO stack (line 2), and eventually retrieved (line 5), when it ends. The retrieval is done in a cycle *while* (line 4) since several cords may end simultaneously at one downfalling edge (as e.g. the cords c_2 and c_5 in Fig. 3). The cycle *while*, executes once per every cord. All remaining operators in the Alg 2 are $\mathcal{O}(1)$ operations, like tests or operations on the stack.

Hence, we may conclude, that Alg. 2 executes with the average complexity of $\mathcal{O}(1)$ per pixel. However, because of the conditions the execution time differs according to the content of the image. The time will decrease for smooth signals. The theoretical lower bound is reached with constant signals, containing only one cord.

Regarding memory consumption, the maximum number of cords pushed into the stack is bounded by the minimum between the number of grey levels and the number of pixels of the signal.

In the following paragraph, we will analyse the complexity for 2-D supports.

5.2 2-D scan strategy

We perform a complete scan of a 2-D rectangular image with a set of parallel, α-oriented Bresenham lines. With the Soille algorithm, this is done in $\mathcal{O}(1)$ per pixel. Furthermore, this set of lines can be computed in parallel since each line is independent from the others.

Therefore, the complexity does not increase with the size of the openings λ and every pixel is computed in a constant time. Next section is devoted to a comparison of the timings with the state of the art.

6 Timings and Comparison with Other Methods

Beside its adaptability, this algorithm is designed for speed. Thus, this section is devoted to compare this algorithm with the state of the art for openings, granulometries and oriented size distributions. All these experiments have been made on a laptop computer using only one thread (Intel Core 2 Duo T7700 CPU @2.4GHz).

6.1 Timings for openings

Four benchmarks on openings have been computed to test the speed of the proposed algorithm. First, we see the correlation between the computation time and the image content. Then, we compare openings in arbitrary orientations for different algorithms. Finally, we make a benchmark with reference to the orientation and with reference to different input data types.

6.1.1 Benchmark with the image content

By analysing Alg. 2, we note that the number of operations changes with the image content. Hence, we measure the average time for 1000 horizontal openings for different images of size 512×512 pixels: Goldhill, two other versions of Goldhill (where the number of grey levels has been set to 2 and 9, with no dithering), an uniformly distributed random noise image and, finally, a constant signal.

Fig. 7 collects the results. As expected, the constant image gives the smallest computation time. We also note that an image with uniform noise is computed faster than Goldhill image. A general rule for this algorithm is that timings are correlated to the mean number of pixels into the stack. A random signal will have, in average, fewer pixels in the stack than a natural image. Furthermore, the fewer cords there are, the closer you get to the theoretical lower bound.

6.1.2 Benchmark with other algorithms

For a comparison with the state of the art, we use five other algorithms. The first one is an algorithm by Van Herk, Gil and Werman [16], [17] (referred to as HGW algorithm hereafter). Then, we use Clienti et al.'s algorithm [18] (Clienti), Van Droogenbroeck et al.'s algorithm [16], [30] (Van Droogenbroeck), Bartovský et al.'s algorithm [20] (Bartovsky) and finally a naive implementation that follows the classical definition of an opening (Naive). All these algorithms have a $\mathcal{O}(1)$ complexity per pixel, excepted the naive algorithm $\mathcal{O}(\lambda)$. They have been integrated to the same platform, in C++, with exactly the same interface. Then, we average the computation time of 1000 realisations of openings with arbitrary orientations. The timings have been computed on Goldhill image (Fig. 8) but we note that the results are approximately the same with other images.

The difference between the naive implementation and others methods is huge. The naive implementation's complexity is independent on the image content but it does depend on the length of the openings. Our algorithm is very fast. However, Van Droogenbroeck's method outperforms our algorithm, especially for large values of λ. One reason can be pointed out to explain this difference; for our algorithm, every pixel of the output image is written exactly once. This can slow down our algorithm compared to an algorithm that only writes the modified pixels. We note, however, that Van Droogenbroeck's algorithm is not able to handle efficiently 16 bits or floating-point data images.

6.1.3 Benchmark with orientation

Openings in arbitrary orientation require an extraction of the lines. We use the same method for all the algorithms

(a) (b) (c) (d) (e)

Fig. 6. Image of size 512×512 used for Fig. 7. (a) goldhill, (b) goldhill with 9 grey levels, (c) goldhill with 2 grey levels, (d) random noise and (e) a constant image

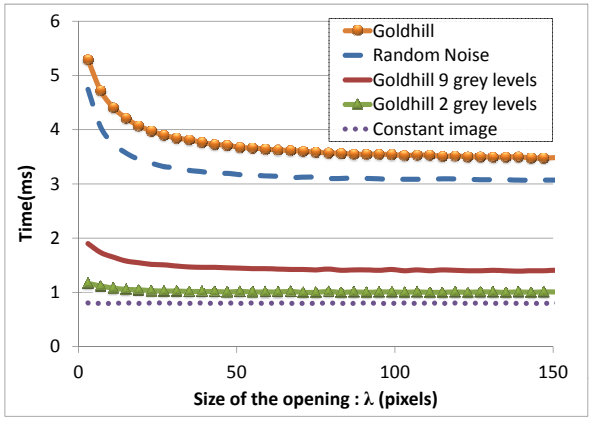


Fig. 7. Timings for horizontal openings of size λ for different images, using the proposed algorithm (512×512 pixels)

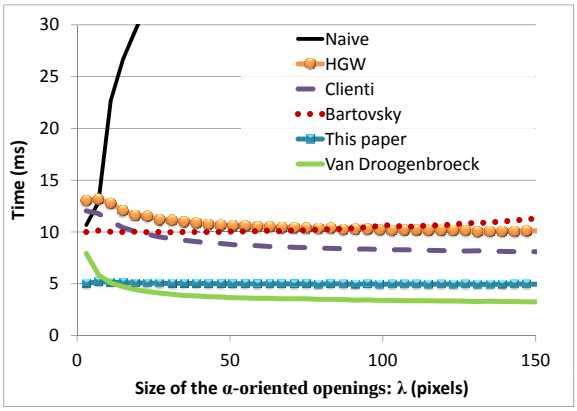


Fig. 8. Timings for openings in arbitrary orientation with regard to λ for different algorithms (Goldhill image)

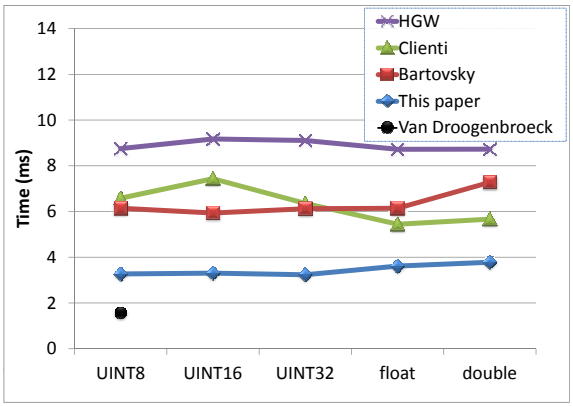


Fig. 10. Timings for openings with regard to the input data type (Uniform noise image 512×512 pixels)

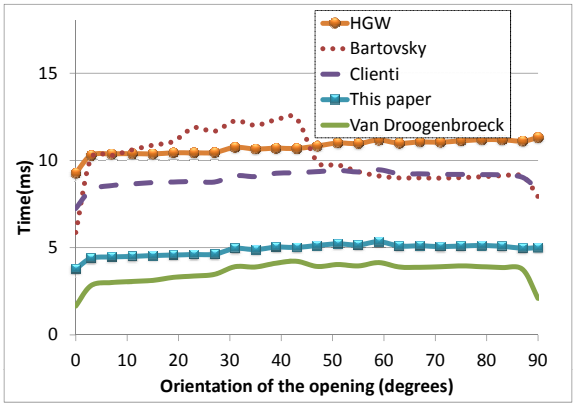


Fig. 9. Timings for openings with regard to the orientation for different algorithms (Goldhill image 512×512 pixels)

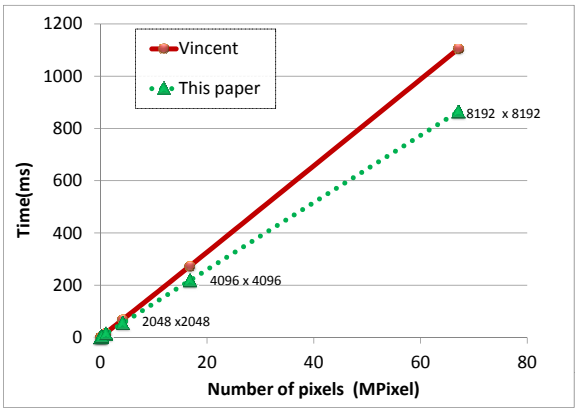


Fig. 11. Timings for horizontal granulometries with reference to the image size. The proposed algorithm outperforms the algorithm by Vincent (Uniform noise images).

[25] excepted for the Bartovský algorithm, which uses its own line extraction. We average the computation time of 1000 openings, for every orientation with $\lambda = 41$ pixels. Fig. 9 collects the results and we note that the computation times are almost independent of the orientation. The angle $\alpha = 0°$ and $\alpha = 90°$ are different since the extraction of the lines is straightforward. We note that the best situation for all the algorithms is $\alpha = 0°$ as expected. This is due to the row major organisation of our data, which minimises cache misses.

6.1.4 Benchmark with input data types

All the previous experiments have been computed with 8-bit images. This benchmark allows visualising the overhead introduced by using other input data type: 8-, 16-, 32-bit, floating point (single and double precision) for the computation of a horizontal opening. To avoid any bias introduced by the image content, we use a random uniform noise image of size 512×512 for our experiments. We compute the average time of 1000 openings for each input data type.

Note that the algorithm proposed by Van Droogenbroeck – the fastest for 8-bit images – does not support any other data type, hence, we have not included it into this benchmark. The results are depicted in Fig. 10. We note that our algorithm is the fastest.

6.2 Timings for granulometries

Compared to a naive implementation, the algorithm by Vincent for granulometries is highly efficient [21]. Even many years after its publication, it used to be the fastest algorithm for 1-D granulometries. We have compared these two algorithms and the timings are collected in Fig. 11, where we plot the average time needed to build a horizontal pattern spectrum with reference to the number of pixels of the signal. Our method is 21% faster than Vincent's one, which becomes useful when we compute the oriented size distribution. The OPS requires many linear granulometries in all orientations: we may need to compute 180^{n-1} granulometry for n-D images. The computation times for the OPS of images 4(a) (322x322) and 4(c) (625x625) are respectively equal to 0.37s and to 1.06s for 180 orientations.

7 Conclusions

This paper introduces a new, flexible and efficient algorithm for computing 1-D openings and granulometries. Its theoretical complexity is linear with respect to the number of image pixels, and constant with respect to the opening size. Moreover, it can be applied to a large set of image types; in fact, the image values only need to have the structure of an ordered group. Extensive benchmarks show that it is the fastest algorithm for computing 1-D openings on images whose data precision is higher than 8 bits (for 8-bit images, the algorithm by Van Droogenbroeck remains unvanquished). It can also be used to compute 1-D granulometries, running faster than the algorithm proposed by Vincent, which has led this category for many years. Moreover, 1-D component trees can also be efficiently computed with the same algorithm. From a software engineering point of view, it should be noted that having the same algorithm for computing different operators, with different data precisions, is very interesting. Futhermore, one can choose between two border extensions to adapt these operators to applications.

The proposed algorithm is applied to 2-D images in several ways: i) the classical linear openings for oriented filtering and/or enhancing of linear structures; ii) the collection of size distributions for all orientations gives the oriented pattern spectrum.

In the future, we shall focus on the computation of local granulometries for analysing non-stationary signals, or to segment textured images. A second extension is to introduce new scan strategies, beyond straight directions, and use this algorithm to efficiently compute path openings. Finally, small modifications of this algorithm are required to compute openings by reconstruction for 1-D signals.

References

[1] J. Serra, *Image analysis and mathematical morphology*. Academic Press, 1982, vol. 1.

[2] ——, *Image analysis and mathematical morphology*. Academic Press, 1988, vol. 2 Theoretical Advances.

[3] C. Heneghan, J. Flynn, M. O'Keefe, and M. Cahill, "Characterization of changes in blood vessel width and tortuosity in retinopathy of prematurity using image analysis," *Medical image analysis*, vol. 6, no. 4, pp. 407–429, 2002.

[4] B. Obara, "Identification of transcrystalline microcracks observed in microscope images of a dolomite structure using image analysis methods based on linear structuring element processing," *Computers & geosciences*, vol. 33, no. 2, pp. 151–158, 2007.

[5] G. Matheron, *Random sets and integral geometry*. Wiley New York, 1975, vol. 1.

[6] S. Batman, E. R. Dougherty, and F. Sand, "Heterogeneous morphological granulometries," *Pattern Recognition*, vol. 33, no. 6, pp. 1047–1057, 2000.

[7] P. Maragos, "Pattern spectrum and multiscale shape representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, no. 7, pp. 701–716, 1989.

[8] N. Theera-Umpon and P. D. Gader, "Counting white blood cells using morphological granulometries," *Journal of Electronic Imaging*, vol. 9, pp. 170–177, 2000.

[9] S. Outal, D. Jeulin, and J. Schleifer, "A new method for estimating the 3d size-distribution-curve of fragmented rocks out of 2d images," *Image Analysis and Stereology*, vol. 27, pp. 97–105, 2008.

[10] P. Salembier, A. Oliveras, and L. Garrido, "Antiextensive connected operators for image and sequence processing," *Image Processing, IEEE Transactions on*, vol. 7, no. 4, pp. 555–570, 1998.

[11] P. Salembier and J. Serra, "Flat zones filtering, connected operators, and filters by reconstruction," *Image Processing, IEEE Transactions on*, vol. 4, no. 8, pp. 1153–1160, 1995.

[12] R. Jones, "Connected filtering and segmentation using component trees," *Computer Vision and Image Understanding*, vol. 75, no. 3, pp. 215–228, 1999.

[13] B. Naegel and L. Wendling, "A document binarization method based on connected operators," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1251–1259, 2010.

[14] M. Couprie, L. Najman, and G. Bertrand, "Quasi-linear algorithms for the topological watershed," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2, pp. 231–249, 2005.

[15] J. Pecht, "Speeding-up successive minkowski operations with bit-plane computers," *Pattern Recognition Letters*, vol. 3, no. 2, pp. 113–117, 1985.

[16] M. Van Herk, "A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels," *Pattern Recognition Letters*, vol. 13, no. 7, pp. 517–521, 1992.

[17] J. Gil and M. Werman, "Computing 2-d min, median, and max filters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 5, pp. 504–507, 1993.

[18] C. Clienti, M. Bilodeau, and S. Beucher, "An efficient hardware architecture without line memories for morphological image processing," in *Advanced Concepts for Intelligent Vision Systems*. Springer, 2008, pp. 147–156.

[19] M. Van Droogenbroeck and M. Buckley, "Morphological erosions and openings: fast algorithms based on anchors," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2, pp. 121–142, 2005.

[20] J. Bartovský, P. Dokládal, E. Dokládalová, and M. Bilodeau, "Fast streaming algorithm for 1-D morphological opening and closing on 2-D support," in *Mathematical Morphology and Its Applications to Image and Signal Processing*, ser. LNCS, vol. 6671. Springer, 2011, pp. 296–305.

[21] L. Vincent, "Granulometries and opening trees," *Fundamenta Informaticae*, vol. 41, no. 1-2, pp. 57–90, 2000.

[22] L. Najman and M. Couprie, "Building the component tree in Quasi-Linear time," *Image Processing, IEEE Transactions on*, vol. 15, no. 11, pp. 3531–3539, 2006.

[23] D. Menotti, L. Najman, and A. de Albuquerque Araújo, "1d component tree in linear time and space and its application to gray-level image multithresholding," in *Proceedings of the 8th International Symposium on Mathematical Morphology*, 2007, pp. 437–448.

[24] V. Morard, P. Dokládal, and E. Decencière, "Linear openings in arbitrary orientation in o(1) per pixel," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*. IEEE, 2011, pp. 1457–1460.

[25] P. Soille, E. J. Breen, and R. Jones, "Recursive implementation of erosions and dilations along discrete lines at arbitrary angles," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 5, pp. 562–567, 1996.

[26] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems journal*, vol. 4, no. 1, pp. 25–30, 1965.

[27] M. H. F. Wilkinson, H. Gao, W. H. Hesselink, J. E. Jonker, and A. Meijster, "Concurrent computation of attribute filters on shared memory parallel machines," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 10, pp. 1800–1813, 2008.

[28] P. Matas, E. Dokládalová, M. Akil, T. Grandpierre, L. Najman, M. Poupa, and V. Georgiev, "Parallel algorithm for concurrent computation of connected component tree," in *Advanced Concepts for Intelligent Vision Systems*, ser. LNCS. Springer, 2008, vol. 5259, pp. 230–241.

[29] P. Matas, E. Dokládalová, M. Akil, V. Georgiev, and M. Poupa, "Parallel hardware implementation of connected component tree computation," in *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, 31 2010-sept. 2 2010, pp. 64 –69.

[30] R. Dardenne and M. Van Droogenbroeck, "libmorpho, http://www.ulg.ac.be/telecom/research.html." [Online]. Available: http://www2.ulg.ac.be/telecom/research/libmorpho.html

Vincent Morard is a PhD candidate at the Centre of Mathematical Morphology, the School of Mines in Paris, France. He is graduated from the engineering school CPE in Lyon, France, in 2009, as an engineer specialised in computed science and image processing. His research interests include mathematical morphology, noise reduction, pattern recognition and statistical learning.

Petr Dokladal is a research engineer at the Centre of Mathematical Morphology, the School of Mines in Paris, France. He graduated from the Technical University in Brno, Czech Republic, in 1994, as a telecommunication engineer and received his Ph.D. degree in 2000 from the University of Marne la Vallée, France, in general computer sciences, specialized in image processing.
His research interests include medical imaging, image segmentation, material control and pattern recognition.

Etienne Decencière received the engineering degree in 1994, the Ph.D. degree in mathematical morphology in 1997, both from MINES ParisTech, and the Habilitation in 2008 from the Jean Monnet University. He holds a research fellow position at the Centre for Mathematical Morphology of MINES ParisTech.
His main research interests are in mathematical morphology, image segmentation, non-destructive testing, and biomedical applications.

Binary Morphology With Spatially Variant Structuring Elements: Algorithm and Architecture

Hugo Hedberg, Petr Dokladal, and Viktor Öwall, *Member, IEEE*

***Abstract*—Mathematical morphology with spatially variant structuring elements outperforms translation-invariant structuring elements in various applications and has been studied in the literature over the years. However, supporting a variable structuring element shape imposes an overwhelming computational complexity, dramatically increasing with the size of the structuring element. Limiting the supported class of structuring elements to rectangles has allowed for a fast algorithm to be developed, which is efficient in terms of number of operations per pixel, has a low memory requirement, and a low latency. These properties make this algorithm useful in both software and hardware implementations, not only for spatially variant, but also translation-invariant morphology. This paper also presents a dedicated hardware architecture intended to be used as an accelerator in embedded system applications, with corresponding implementation results when targeted for both field programmable gate arrays and application specific integrated circuits.**

***Index Terms*—Hardware implementation, mathematical morphology, spatially variant structuring elements.**

I. Introduction

MATHEMATICAL morphology is a nonlinear image processing framework used to manipulate or analyze the shape of functions or objects, originally developed by Matheron and Serra in the late 1960s [1]. Mathematical morphology is set theory-based methods of image analysis and plays an important role in many digital image processing algorithms and applications, e.g., noise filtering, object extraction, analysis or pattern recognition. The methods, originally developed for binary images, were soon extended and now apply to many different image representations, e.g., grayscale, color or vector images, and more recently to matrices and tensor fields.

Real-time image processing systems have constraints on speed or hardware resources. In addition, in embedded or mobile applications, these systems require low power consumption and low memory requirements. An example of such a system may be found in [2], in which a real-time automated digital surveillance system with tracking capability is presented. The system is intended to be included in a self-contained network camera and the characteristics of the surveillance scene together with camera placement have a direct impact on system performance. By letting a locally adaptive morphological filter process the binary segmentation result, thereby exploring the depth information in the scene, a more accurate tracking may be observed. Therefore, due to the constraints and the performance increase in such applications, the need for efficient hardware (HW) architectures in terms of computational complexity and memory requirement with low power characteristics for this image representation becomes evident.

This paper is organized as follows: The remainder of this section addresses the motivation of using locally adaptive, spatially variant structuring elements (SV SEs), discusses the application context, and puts it into perspective by comparing to previously published work. Section II discusses basic morphological concepts and properties together with SV SEs in general and inferred restrictions. Section III details the algorithm and Section IV proposes a corresponding HW architecture. Section V presents implementation results of the architecture when targeted for both FPGA and ASIC. Section VI concludes the paper.

A. Application Context

Although translation-invariant structuring elements (TI SE) are sufficient in many image processing applications, SV SEs outperform them by their ability to adapt to local features. SE functions are studied and several examples are given by Serra [3, Ch. 2.2 and Ch. 4], and an early evaluation of the performance of SV SEs versus TI SEs can be found in Chen *et al.* [4].

Generally, there are two strategies to control the shape and size of the SE:

1) *image-exogenous information*, usually used to correct or to adapt to an image anamorphism;
2) *content dependent processing*, e.g., contour preserving filters, and image restoration.

1) Image Exogenous Information: Processing images deformed by anamorphism (such as perspective or wide-angle deformations) can be done in two ways: i) apply usual TI operations after a previous distortion correction or ii) as proposed here, use SV operators that adapt to the distortion. An example of an application that will benefit from anamorphism-aware processing is the road-traffic surveillance scene shown in Fig. 1(a), where the images are deformed by the perspective, see, e.g., Beucher [5]. Extracting individual vehicles from the motion mask, Fig. 1(b), may be done by an alternate morphological filter [6], starting with an opening to eliminate noise, followed

Manuscript received December 10, 2007; revised October 06, 2008. Current version published February 11, 2009. This work was supported in part by the Competence Center for Circuit Design at Lund University and in part by the Center of Mathematical Morphology at Mines Paris PARITECH. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Philippe Salembier.

H. Hedberg was with the Department of Electrical and Information Technology, Lund University, SE-22100 Lund, Sweden He is now with Prevas, Stockholm, Sweden (e-mail: hugo.hedberg@prevas.se).

P. Dokladal is with the Center of Mathematical Morphology (CMM), Mines Paris Paritech, 35, 77305 Fontainebleau Cedex, France (e-mail: petr.dokladal@mines-paritech.fr).

V. Öwall is with the Department of Electrical and Information Technology, Lund University, SE-22100 Lund, Sweden (e-mail: viktor.owall@eit.lth.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2008.2010108

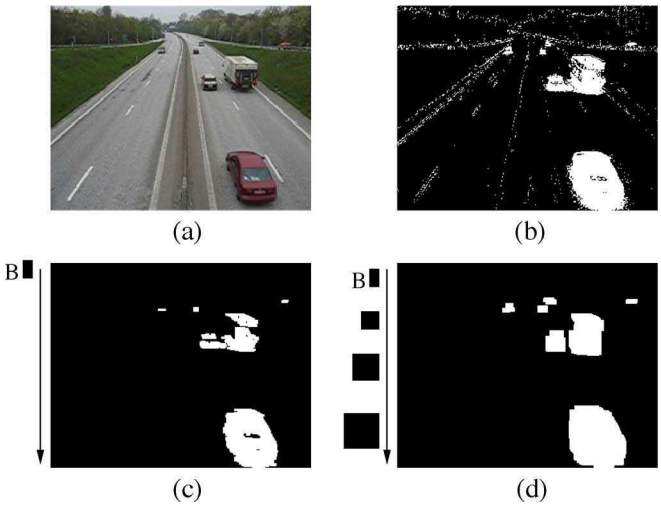


Fig. 1. (a) Typical road surveillance application input image, (b) binary motion mask, (c) and filtered motion mask using a TI SE. (d) Filtered motion mask using a SV SE, increasing in size from top to bottom.

by a closing to close holes and smooth the contours of the vehicle masks. One needs to use a SE sufficiently large to filter the noise, but small enough to preserve the individual vehicles. Due to perspective deformation, a TI SE will not produce a satisfactory result in all regions of the image, Fig. 1(c). However, using a SV SE, increasing in size from top to bottom of the image, eliminates noise and correctly extracts all vehicles, Fig. 1(d). Another application where a SV SE is useful is licence plate detection. Provided the resolution suffices, the SV SE may be used to compensate for the change of the apparent size of the licence plate. Furthermore, wide-angle-camera compensation may be used to correct anamorphism. This is used by Roerdink and Heijmans to measure forest density [7], where a progressively changing SE is used, different in the center and on the periphery of the image, to compensate for the distortion.

For ordinary cameras, the SE size may be set manually, proportional to the apparent size of the objects (as used here). Introducing other imaging techniques, such as range imagery, may give access to distance-to-camera information. Since this information relates to the apparent size of the objects, it allows direct control of the size of the SV SE [8].

2) Content Dependent Processing: In the second use case, the SE size is controlled by the content of the image. Note that there is no alternative approach to restore the image distortion and apply a TI operator as in the case of correction of anamorphism. There are several examples of such content dependent filtering, e.g., the reversible image coding and its restoration from skeleton [9], and also [1] and [10]–[13]. An example of a binary object is shown in Fig. 2(a), which is represented by its skeleton in 2(b). The skeleton is obtained by connecting the centers of maximal balls contained in the object in 2(a). One can associate weights to points on the skeleton which are equal to the Euclidean distance to the complement, i.e., the radius of the balls, and reconstruct the object by dilating the skeleton with the corresponding ball (radius equal to the distance to complement). During the restoration, the skeleton is dilated by large SV SEs, Fig. 2(c).

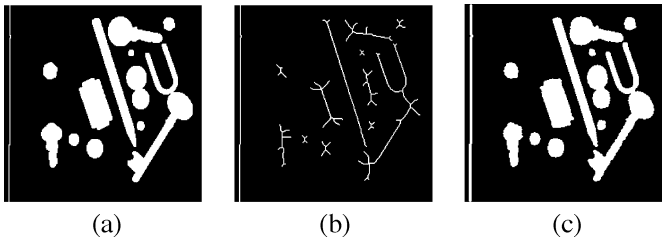


Fig. 2. Object reconstruction from skeleton. (a) Binary object "tools" (b) its skeleton, weighted by distance to the complement, (c) reconstruction from the skeleton.

Filters with adaptable pixel neighborhoods have been thoroughly investigated over the years. Illustrations may be found in the Nagao filter in Natsuyama [14], later in Wu and Maître [15], or more recently in Lerallut [16]. Such SV filters together with their pyramids and derived segmentation aspects are also studied by Debayle and Pinoli in [17] and [18]. They illustrate applications of image denoising, enhancement, filtering, and segmentation with SV SEs, which are compared with results obtained with TI SEs. The idea behind these filters is to define a SV SE that fits inside the objects to prevent blurring when the SE stretches across the boundaries of the objects. At the same time, the SE increases in size towards flat zones to obtain a stronger filtering effect.

SV morphology has also been investigated by Charif-Chefchaouni and Schonfeld [19], and more recently by Bouyanaya *et al.* in [20] and [21] for set-wise SV morphology and SV morphology on functions.

All these references propose application examples or theoretical advances but no efficient implementations. Therefore, the following sections will present an algorithm with a corresponding architecture that is suitable for such applications as discussed above.

B. Previous Optimization Efforts

In *naive* implementations of mathematical morphology operations, outputting one pixel requires the examination of its entire neighborhood defined by the SE at this position. Consequently, using large neighborhoods become particularly computationally intensive and efforts have been made to make these operations efficient.

Although considerable advances have been achieved in conception of fast algorithms and HW accelerators for TI morphology, little has been done for optimization of SV SE. Existing efforts group into different frameworks.

1) Fast Recursive Algorithms for Translation Invariant Structuring Elements: The implementations by Van Herk [22] and by Lemonier and Klein [23] support large linear SEs but need three and two scans, respectively, to complete each operation, requiring intermediate storage. The HW complexity is of $O(1)$ per pixel and memory requirement is of $O(n^2)$, where n is the width of a quadratic image. In addition, their extension to SV SEs in not possible.

Van Droogenbroeck and Talbot [24] propose an algorithm based on histograms. The histogram is updated as the SE slides over the image. The respective value for the needed rank filter (dilation, erosion, or median) is taken from the histogram. This

algorithm naturally extends to SV SEs. However, computing the histogram requires additional resources, and becomes cumbersome for finely quantized data and impossible for $\mathbb{R}$.

2) FFT-Based Algorithms for Translation Invariant Structuring Elements: There are also methods for fast computation of morphological operations with large structuring elements by thresholding convolutions computed as a product of Fourier transforms, see [25]. However, SV structuring elements cannot be written as a product of FFTs. Furthermore, even if there is no increase in computational complexity for large structuring elements, the computational complexity, and memory requirements of FFTs exceed the ones for recursive algorithms (item a), or the one in this work.

3) Efficient Hardware Implementations: Concerning efficient HW implementations, Klein and Peyrard [26] have designed a neighborhood processor for binary mathematical morphology, executing dilations/erosions, thinnings/thickenings, and geodesic operations (reconstructions).

Fejes and Vajda [27] and Velten and Kummert [28] both propose delay-line implementations. This classical approach supports arbitrary shaped SEs, but the computational complexity is of $O(n^2)$, where n is the width of a quadratic SE, and the memory requirements is of $O(n^2)$, where n is the width of a quadratic image. Therefore, this type of implementation becomes unsuitable for large SEs and high resolution applications. This is due to that each element in the SE increases the fan-in of the computations as well as the required amount of memory to delay the rows to extract the pixel neighborhood. In [29], an architecture is proposed based on the observation that many calculations between two adjacent pixels are redundant and can be reused, giving the architecture its name: partial-result-reuse (PRR). By this procedure, the computational complexity can be reduced to $O(2\lceil\log_2(n)\rceil)$, where n is the width of a quadratic SE ($\lceil . \rceil$ is the ceiling function). However, it uses the same type of delay-lines as in [27] and [28], thus resulting in the same memory requirement.

Hedberg *et al.* [30] propose a low-complexity (LC) and low memory requirement architecture. The complexity is reduced to $O(1)$ and memory requirement to $O(n)$, where n is the width of the input image, at the cost that the class of supported SEs is limited to flat rectangles of arbitrary size. Erosions and dilations are accomplished with only two summations and two comparisons independent of the structuring element size and resolution.

4) Spatially Variant Morphology: Recently, Cuisenaire [31] proposes a fast algorithm for binary spatially variant morphology based on thresholding the distance transform, widely used for efficient implementation of dilations and erosions [32], [33]. The class of allowed shapes is restricted to balls of various norms. Various algorithms exist for computing the distance map. They are either i) image scan operations [34], or ii) equidistant propagations from the sources, (see surveys [35] and [36] for overview and other citations). The former have high memory requirements since they use a large intermediate storage for partial results between the scans. The distance is computed on the entire image, penalizing the performance when small SEs are used. The latter, based on equidistant propagation from the sources do not necessarily compute the distance on the entire image and are more efficient. However, they use ordered structures and random memory accesses, penalizing performance on large data sets and are difficult to implement in HW, see Dejnozkova [37] for discussion.

C. Main Contribution

This work fits into the framework of binary Mathematical Morphology and represents the first step towards arbitrary shaped SV SE in efficient HW and SW implementations. The main contribution of this paper is twofold.

1) A new algorithm supporting a rectangular SV SE for binary mathematical morphology with very low computational complexity and memory requirements. An extension to a richer class of structuring elements is possible.
2) A corresponding HW architecture, suitable for embedded or mobile applications. The architecture has several important properties from a HW perspective, i.e., sequential pixel processing, low-computational complexity, and low memory requirement. Implementation results of the proposed architecture are presented in terms of resource utilization when targeted for both FPGA and ASIC.

The architecture proposed in this paper is a development from the one published in [30]. The new architecture allows changing the size of the rectangle within an image from pixel to pixel, and can thereby locally adapt its size. Although having mainly the same memory requirement, the SE flexibility comes at the cost of increased computational complexity from $O(1)$ to $O(n)$, n being the SE width.

II. Algorithmic Issues

Let I be an input image $I: D \rightarrow V$, with $D = \text{supp}\{I\} \subseteq \mathbb{Z}^2$ being the domain and V the set of values. In this paper, we place ourselves in the context of binary images $I : D \rightarrow \{0, 1\}$, where objects are represented by 1, i.e., the object X contained in a binary image I is $X = \{x | I(x) = 1\}$.

All morphological operations are based on logical or arithmetic calculations (for binary or valued images, respectively) on a local neighborhood of a pixel. The neighborhood is a subset of pixels defined by the shape of the *structuring element* $B \subset D$, which has a corresponding $origin \in B$, that determines the position of the calculated value in the output image. The translation of B by some $x \in D$ is often denoted by $B(x)$.

When using a SV SE, the fixed set $B \subset D$ is replaced by a flexible set given by $B : D \rightarrow \mathcal{P}(D)$ with $\mathcal{P}$ denoting the set of subsets. This means that instead of a fixed $B \subset D$ one uses $B : D \rightarrow \mathcal{C}$, where for every point $x \in D$, the mapping $B(x)$ is not a translation but chosen as an element from the class $\mathcal{C}$ of allowed shapes, used locally at x.

Spatially variant binary erosion and dilation are defined by means of Minkowski addition and subtraction (see Serra [3, pp. 41 and 42]) according to

$$\varepsilon_B X = \bigcap_{x \in X^c} [B(x)]^c \tag{1}$$

and

$$\delta_B X = \bigcup_{x \in X} B(x). \tag{2}$$

141

Alternatively, SV binary erosion and dilation may be defined based on set intersection and inclusion as

$$\varepsilon_{\widehat{B}}X = \{y|B(y) \subset X\} \tag{3}$$

and

$$\delta_{\widehat{B}}X = \{y|B(y) \cap X \neq \emptyset\} \tag{4}$$

where $\widehat{B}$ denotes the transposition of B, which may be defined as the set of ancestors of B according to

$$\widehat{B}(x) = \{y|x \in B(y)\} \tag{5}$$

with $y \in D$. This definition is non local, and cumbersome since the computation is done by exhaustive search. Notice the difference from the case of a TI SE where the transposition is a mere set reflection, i.e., $\widehat{B} = \{x|-x \in B\}$.

Adding arithmetic, (1)–(4) can be used to perform other operations and algorithms, e.g., morphological gradient $g = \delta - \varepsilon$ or morphological Laplacian $\mathcal{L}(I) = \delta I - 2I + \varepsilon I$. To form morphologic filters, e.g., opening $\gamma_B = \varepsilon_B \circ \delta_{\widehat{B}}$, closing $\varphi_B = \delta_B \circ \varepsilon_{\widehat{B}}$, or more complex filters, one generally has two options: i) combine adjoined definitions (1) and (4), or (2) and (3), or ii) use (5) to transpose B [3], [20].

The SE transposition (5), as well as the set inclusion/intersection versions of erosion/dilation, i.e., (3) and (4), are non local. This means that to compute the result at some point x, one needs to examine the input at unknown points y. Therefore, the result cannot be generated directly by the presented algorithm. To obtain adjunction and form filters, one needs to use the Minkowski addition/subtraction-based definitions in (1) and (2), together with precomputing the transposed SE according to (5).

A. Supported Structuring Elements

The structuring element B defines which pixel values in the input image I to include in the calculation of the output value. Whereas the geometric shape of a TI SE is constant throughout the input image, the shape of a SV SE may change from pixel to pixel in the generalized form. Restricting the set of allowed shapes $\mathcal{C}$ and the size distribution allows design of more efficient algorithms.

Shape: The algorithm is based on computation of distance function to object edges. Decomposing the computation of B into columns brings restriction to $\mathcal{L}_1$. Hence, in the HW realization presented below, the allowed SE shapes are restricted to rectangles (including $\mathcal{L}_1$-balls, squares). Note that Section V-A discusses an extension to a richer class of shapes.

Scan Order: Other restrictions are required if the algorithm is to be implemented in low complexity and low memory architectures with no intermediate storage. Usually, pixels arrive in a stream in raster scan order and output pixels are produced in a stream. Therefore, the output at location (i, j) cannot be produced until the entire neighborhood has been processed. Consequently, there is a latency L between the input and the output stream. For some pixel (i, j), the latency is given by

$$L(i,j) = N_d(i,j) \cdot I_w + N_r(i,j) \tag{6}$$

where I_w is the width of the image I, N_d and N_r are the coordinates of the origin offset from the bottom-right corner ($\mathrm{d} = \mathrm{down}$ and $\mathrm{r} = \mathrm{right}$), illustrated in Fig. 3.

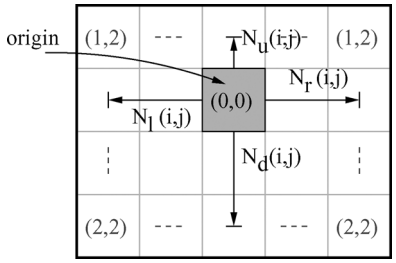


Fig. 3. Example of a 5×4 ($\mathrm{width} \times \mathrm{height}$) rectangular SE with $B(i,j) = (N_u, N_l, N_d, N_r) = (1, 2, 2, 2)$.

When using a TI SE for the entire image, i.e., $B(i,j) = B$, $\forall i, j$, the latency is constant and the raster scan order is maintained. However, if the structuring element changes from pixel to pixel, the latency varies. For unconstrained SE sizes, the output pixels will be produced in a different order with the necessity to store them in intermediate memory to retain raster scan order.

Under constraints, the intermediate storage may be dropped. From (6), $\Delta L/\Delta N_r = 1$ and $\Delta L/\Delta N_u = \Delta L/\Delta N_l = 0$ may be derived, which means that increasing/decreasing the size of the SE by one pixel to the right will increase/decrease the latency by one. Adding above and to the left has no impact on latency since these pixels have already been read.

Unitary changes of L from pixel to pixel, i.e., $|\Delta L/\Delta i, j| = 1$ can be handled with no additional memory, by stalling the input or output. Indeed, if N_r increases/decreases, the latency L increases/decreases, and the output/input is stalled. Stalling the output means that two input pixels are read before the next output value is calculated, whereas stalling the input means that two pixels are output before the next input pixel is read.

In order to avoid additional intermediate storage, for the rest of the paper, a restriction is placed on the class $\mathcal{C}$ of allowed shapes to be rectangles, not necessarily symmetric around the origin.[1] Therefore, $B(i,j)$ becomes a function $B : \mathbb{Z}^{+2} \rightarrow \mathbb{Z}^{+4}$, i.e., for every pair of coordinates (i, j). The function $B(i,j)$ yields a quadruple (N_u, N_l, N_d, N_r) defining the position of the origin with respect to the edges of the rectangle. These parameters are tied to the width and height of B by $B_w = N_l + N_r + 1$ and $B_h = N_u + N_d + 1$. The maximum width and height found in the collection of $B(i,j)$, $\forall i, j$, are denoted $\max(B_w)$ and $\max(B_h)$, respectively. Fig. 3 shows an example of a structuring element $B(i,j) = (1, 2, 2, 2)$, being a 5 by 4 ($\mathrm{width} \times \mathrm{height}$) rectangle with the origin offset by 2 rows and 2 columns from the lower-right corner.

From (6), $\Delta L/\Delta N_d = I_w$ means that increasing the size of the SE by adding one bottom row, will increase the latency L by the entire width I_w of the image. This substantial change of latency can not be handled without using an additional buffer. This means that from pixel to pixel, the rectangle can grow/diminish by one at all sides, except of adding/deleting one bottom row, authorized only between two image row

$$\left|\frac{\Delta N_{u,l,r}}{\Delta x, \Delta y}\right| \leq 1, \left|\frac{\Delta N_d}{\Delta y}\right| \leq 1 \text{ and } \left|\frac{\Delta N_d}{\Delta x}\right| = 0. \tag{7}$$

[1] An asymmetric origin is useful for even widths or when approximating eccentric amoebas by rectangles.

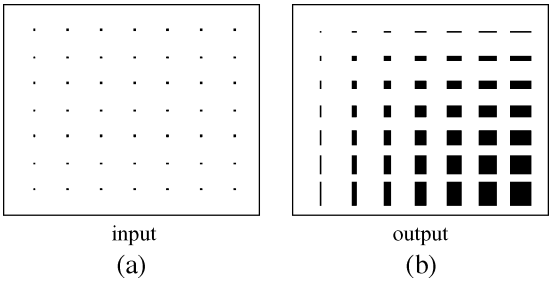


Fig. 4. (a) Synthetic test image containing black dots on a grid, corresponding to the foreground. (b) Dilation (2) obtained with similar SE as in Fig. 1, i.e., a rectangle increasing in size from the top left corner of the image downwards to the right.

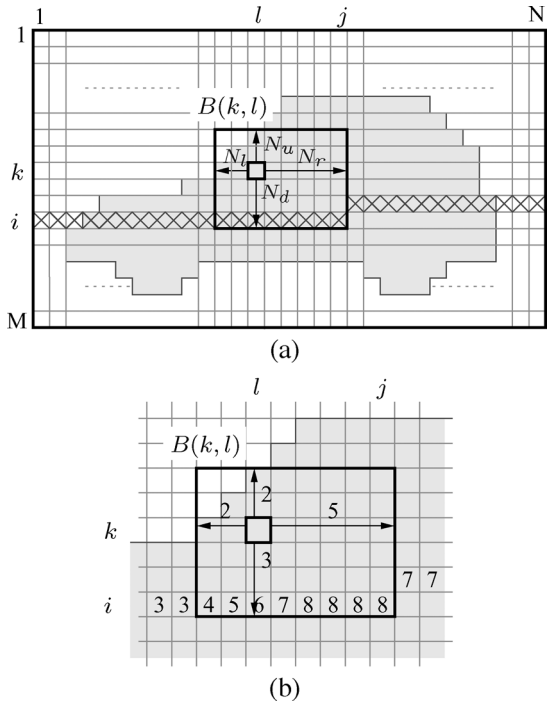


Fig. 5. (a) Synthetic input image when processing a pixel at location (k,l) with $B(k,l)$. (b) Zoom in of when processing the same pixel using $B(k,l) = (2,2,3,5)$ as structuring element. The numbers in bottom row of $B(k,l)$ show the current distance values, which saturates at the value 8.

An example of synthetic test data (640 × 480 pixels) is illustrated in Fig. 4(a). The image contains a set of black spots (on a uniform grid of 60 pixels). Applying a dilation to this input image will enlarge the black spots. Fig. 4(b) illustrates a dilation obtained with a rectangular SE progressively increasing in size from the top-left towards the bottom right corner of the image: $B(i,j) = (N_u, N_l, N_d, N_r) = (i/20, \mathrm{j}/20, \mathrm{i}/20, \mathrm{j}/20)$.

Note that these restrictions are not valid in applications where the SE size depends on the content of the image, e.g., contour filtering, object restoration (Fig. 2). However, as discussed previously, the constraints in (7) only concern the stream implementation capability, and can be relaxed if an intermediate storage is available, see Section V-A.

III. Algorithm Description

The algorithm reads the input image I and writes the output image O sequentially in raster scan order. Let (i,j) denote the current reading and (k,l) current writing position. Fig. 5(a) gives a synthetic example image $I(M,N)$ containing one object—a car. The object constitutes of pixels equal to 1, and the background constitutes of pixels equal to 0. Obviously, by causality, the reading position (i,j) precedes the writing position (k,l). The latency L between reading and writing the data depends on the size and location of the origin of the currently used structuring element $B(k,l)$, defined in (6). Since $B(k,l)$ varies for different coordinates, the latency L will also vary.

The SE shape function B is a parameter of the morphological operation and is also read in the raster scan order at the same rate and position as the output image O.

The reading (i,j) and writing (k,l) positions are bound by

$$\begin{aligned} i &= \max\left(1, \min\left(k + N_d(k,l), M\right)\right) \text{ and} \\ j &= \max\left(1, \min\left(l + N_r(k,l), N\right)\right). \end{aligned} \tag{8}$$

Suppose the currently processed pixel be (k,l) and that the corresponding structuring element $B(k,l)$—placed by its origin at (k,l)—has just been read. Recall the size of $B(k,l)$ is coded by (N_u, N_l, N_d, N_r), e.g., equal to (2, 2, 3, 5) in the example shown in Fig. 5. The input data need to be read to the bottom right position of $B(k,l)$, indicated as (i,j).

The algorithm proceeds by decomposing the erosion into columns. In each column $1,\ldots,N$ of the input image I, the algorithm keeps track of the distance $d(1,\ldots,N)$ from the currently processed line to the closest upward zero (background). For each column j, the distance $d(j)$ is updated as I is sequentially being scanned according to

$$d(j) = \begin{cases} 0 & \text{if } I(i,j) = 0 \\ d(j) + 1 & \text{if } I(i,j) = 1. \end{cases} \tag{9}$$

If $I(i,j) = 1$, i.e., belongs to the object, the distance $d(j)$ is incremented, otherwise the pixel belongs to the background, and $d(j)$ is reset to zero.

In Fig. 5(a), currently known distances are indicated by ×. Notice that for the currently processed pixel $O(k,l)$, the distances are calculated on a different row i. The corresponding distance values for this particular example are shown in Fig. 5(b). These distances are then compared, column-by-column, to the height of the currently used structuring element $B(k,l)$, given by $N_u + N_d + 1$. This evaluation, at position $O(k,l)$ in the output image, can be formalized as

$$\text{if the comparison} \quad d(j) \geq N_u + N_d + 1 \tag{10}$$

yields TRUE, for all $j \in [\max(1, l - N_l), \min(l + N_r, N)]$, then at position $O(k,l)$ write 1, else write 0. The whole algorithm can be written as follows.

Algorithm:

```
for k = 1...M
    for l = 1...N
        read B(k,l)
        read I up to (i,j)                    (8)
        update d up to j                      (9)
        O(k,l) = AND(d(j) ≥ N_u + N_d + 1)    (10)
        write O(k,l)
    end
end
```

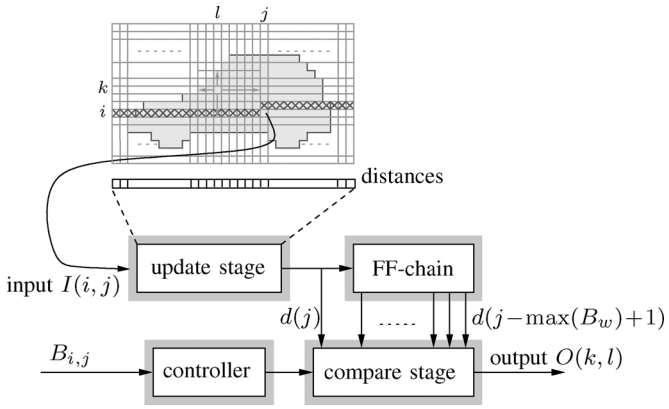


Fig. 6. Block diagram of the proposed algorithm.

AND means 1 if comparisons for all j such that $j \in [\max(1, l - N_l), \min(l + N_r, N)]$, yield TRUE, else 0 [see (10)]. For example, since the distances in the example shown in Fig. 5, i.e., $d(l-2) = 4$ and $d(l-1) = 5$, are smaller than the height $N_u + N_d + 1 = 6$ for the pixel located (k, l), the output at $O(k, l)$ is 0.

The distance calculation is an independent process of the morphological operation being performed, resulting in that the memory content is unrelated to the dimensions and origin of $B(i, j)$. This means that no information about a former B propagates in the algorithm. It is this algorithmic property that allows an adaptable SE, different for each individual pixel.

A. Block Diagram

A block diagram of the proposed algorithm is illustrated in Fig. 6. A controller is needed to stall the input and output depending on how parameters for the structuring element change. Based on these control signals, the distances to the closest upward zero, stored in the update stage, are updated. The output value from the update stage is always equal to the last calculated distance for the column of the current pixel according to (9). This distance is used as input to the compare stage and to the serially connected Flip-Flops (FF-chain), in order to let the distances propagate to be used in multiple calculations. The distances stored in the FF-chain (for the previous columns) are all used as inputs to the compare stage and the controller selects for each pixel which of these distances to include in the calculation, i.e., which distances that are to be used in (10). The selected distances are then compared to the height given by the current B. If all are greater or equal to this height, output 1 else 0 at the current location of the origin.

B. Software Implementation

Due to algorithmic properties such as the stream-like processing and in-place execution, the algorithm is applicable for software applications. As an example, if coded in C, the algorithm uses a small amount of memory (one image line) and runs very fast even for large images. Experiments on an Intel Centrino 2-GHz PC running Linux show that when eroding an image with a resolution of 1,000 × 1,000 using a SV rectangular SE of up to 100 × 100 pixels (similar to the one used in Fig. 4), takes ≈81 ms. Eroding an even larger image with a resolution of 10,000 × 1,000 image using the same SE takes 760 ms. The execution time scales linearly with the image size even for extremely large images, mainly coming from the stream-like memory access pattern.

IV. Architecture

A HW architecture for the proposed algorithm is illustrated in Fig. 7. The architecture is divided into three stages: update, FF-chain, and compare (refer to Fig. 6). In the update stage, a row memory $(\mathrm{mem}_{\mathrm{row}})$ stores the distances for each column in the input image and for each incoming pixel: if a 0 is encountered, the sum is reset to 0, else increased by 1. This is implemented as an incrementer and a multiplexer (placed in the middle of this stage in the figure). The input from the FIFO (First In First Out) is the control signal to the multiplexer, which outputs the reset or the increased sum for further processing. If the distance is equal to the maximum supported SE height $\max(B_h)$, the sum saturates at this value, which also is the initial value in order to leave the result unaffected at the image borders.

The FF-chain contains delay elements that stall the distances $d(j - \max(B_w) + 1)$ to $d(j)$, which may be used in the current calculation, i.e., may be evaluated against the columns in the current $B(i, j)$. The FF-chain has individual access to the entries (distances), and is implemented as a series of FFs that enables each distance to propagate as long as they are to be reused in a calculation. The block also includes multiplexers for initialization on a new row in the input image.

The compare stage compares stored distances to the height of the SE. The number of comparators equals the maximum supported SE width, $\max(B_w)$. The results from the comparators serve as input to the logic AND-operation. Notice that the fan-in to this unit increases linearly with $\max(B_w)$ and, thus, affects the critical path and is the major bottleneck of the architecture. Hence, for large SEs or high speed applications, a pipeline may be inferred. Using pipelining, one or several additional delays are required to synchronize the output with the data valid signal.

The CTR block in Fig. 7 manages all control signals in the architecture based on $B(i, j)$: the enable signal to decide the number of active comparators (enable), which operation to perform (ε/δ), and also border handling. By default, the architecture performs a logic AND-operation (minimum) on the selected distances, i.e., a subset of $d(j - \max(B_w) + 1)$ to $d(j)$, which in mathematical morphology corresponds to an erosion. To perform a dilation, simply calculate the distances to the closest upward one for each column and perform a logical OR-operation (maximum). This is due to the duality nature, i.e., $\varepsilon_B I = (\delta_B I')'$, where $'$ is the bit inverse. Therefore, the other way to obtain a dilation and still use the default operation is to simply invert the input and the output, accomplished in HW by placing a multiplexer and an inverter at the input and the output of the architecture, shown in Fig. 7.

A. Handling the Borders

Sliding the structuring element over the input image, some output values are based on evaluating neighborhoods that require pixels located outside the image borders. These pixels, or

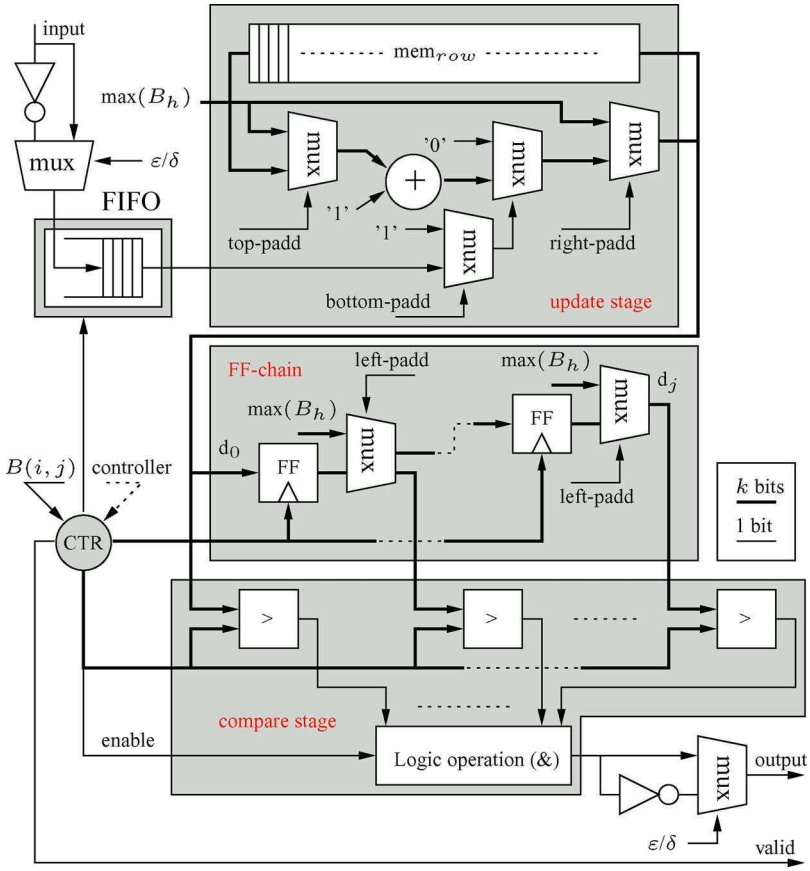


Fig. 7. Overview of the implemented architecture. Note that the data valid signal is generated by the CTR block.

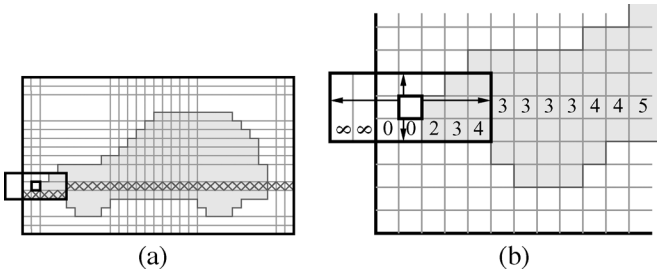


Fig. 8. (a) Illustration of when the structuring element stretches outside the image (b) and when the distances outside the image are assumed infinite; they will not affect the erosion.

in our algorithm distances, are referred to as padding. An example of a SE requiring left padding is shown in Fig. 8(a). The current architecture manages the padding pixels in one of two ways: precalculated initial values (top, right, and left padding) or pixels inserted into the data stream (bottom padding). The result is a less complex controller but with the drawbacks of requiring two clock domains and an input FIFO. The padding control is included in CTR in Fig. 7 with corresponding control signals, i.e., left-, top-, right, and bottom-padd.

Assume a rectangular SE, e.g., $B_{i,j} = 1, 3, 1, 3$, calculating the second output pixels of a new row is an example requiring left padding [Fig. 8(a)]. When starting at a new row, the distances to the left of the first column are assumed to be infinite, as illustrated in Fig. 8(b). This assumption is implemented as initial values equal to $\max(B_h)$, which are inserted simultaneously by using the multiplexers in the FF-chain stage in Fig. 7. This procedure causes the distances located beyond the image borders not to affect the calculation. When reaching and extending the structuring element beyond the right image border, the same initial value is inserted into the data stream and sent to the compare stage through the rightmost multiplexer in the update stage.

Using the same assumption as above when processing the first row in an image, the distances to the closest upward zero for the preceding row is assumed infinite. Again, this is implemented as initial values equal to $\max(B_h)$ (inserted into the adder through the leftmost multiplexer in the update stage in Fig. 7). The initial values are updated with the pixel value in the input image and the result is sent to both the compare stage and written back into the row memory.

Reaching the bottom segment of the input image, the structuring element can stretch outside the bottom border. Depending on the actual height of $B_{i,j}$, additional "1"s are inserted in the pixel stream (at most $\lceil \max(B_h)/2 \rceil$) through the lower multiplexer in the update stage. This insertion is necessary to handle the different latency that will occur in a video stream if different sizes of the structuring element are used at the end of one image to the beginning of the next. During the insertion of these extra pixels, the input data stream is stalled (requiring the FIFO on the input). Once the last pixel has been processed, the erosion operation is complete and starts over with the next frame.

B. Coding the Structuring Element Size

The structuring element size is controlled by the function $B(i,j)$ through the parameters N_u, N_l, N_d, N_r, defined in Section II-A. The parameters are generated outside the architecture and are sent as input in parallel with the input pixel stream to the controller in Fig. 7. Formally, $B(i,j)$ becomes $B(i,j,t)$ with $I(t)$ for video sequences and it is the user's responsibility to design the application-dependent $B(i,j,t)$ generation process, which must fulfill the conditions in (7).

In order to reduce the bandwidth of $B(i,j)$, one can use efficient coding. For example, the simplest coding scheme consist of coding the difference $\Delta N_{u,l,d,r}$ between two adjacent pixels on a line, instead of coding the size $N_{u,l,d,r}(i,j)$ directly. Limiting the difference to $|\Delta N_{u,l,r}/(\Delta i)| \leq 1$, the coding can be represented by using two bits, i.e., increase, decrease, no-change, reset, corresponding to a simple Δ-code. The reset value can be used to restore B to an initial setting at the beginning of each line. Thus, coding $B(i,j,t)$ will require $3 \times 2 = 6$ bits between two adjacent pixels on a line (since N_d is not allowed to change in the middle of a line), and two more bits in between two consecutive lines to represent N_d, ending up with a total number of 8 bits to code $B(i,j,t)$.

Virtually any appropriate coding system can be used, e.g., a run-length coding applied separately to each N_u, N_l, N_d and N_r will be useful if the size remains at least partially constant in some zones. Using an efficient coding will be profitable especially if more complex shapes are used (see Section V-A), since describing arbitrary shapes requires by far more information, especially for large SEs.

C. Memory Requirements

The row memory located in the update stage stores the distances for each column and is the largest single internal component in the architecture (excluding the input FIFO). The requirement is linearly proportional to the resolution according to

$$\mathrm{mem_{row}} = \lceil \log_2 (\max(B_h)) \rceil \cdot I_w \text{ bits} \tag{11}$$

where $\max(B_h)$ is the maximum supported SE height which determines the number of bits per stored value according to $k = \lceil \log_2(\max(B_h)) \rceil$. Additional registers in the FF-chain are needed to delay the stored distances ($\mathrm{mem_{row}}$ content) serving as input to the comparators, Fig. 7. The number of registers is proportional to the maximum allowed SE width. Since their content should be compared to the maximum SE height, the number of bits in these registers is

$$\mathrm{FF_{chain}} = k \cdot \max(B_w) \text{ bits.} \tag{12}$$

Combining (11) and (12), the total memory requirement for the algorithm is equal to

$$\mathrm{mem_{tot}} = \mathrm{mem_{row}} + \mathrm{FF_{chain}} = k\,(I_w + \max(B_w)) \text{ bits.} \tag{13}$$

D. Memory Organization

Concerning the implementation of $\mathrm{mem_{row}}$ in the update stage, ideally, a value should be read, updated, and written back to this memory in a single cycle. This require simultaneous read and write operations that are normally implemented using a dual-port memory. However, this type of memory introduces an area overhead mainly due to the dual address decoders. Another observation is that the memory access pattern is the same as in a FIFO, resulting in that the address generation becomes trivial and can be implemented as a simple modulo-counter. Based on these facts, $\mathrm{mem_{row}}$ can be advantageously implemented using a single-port memory of double width and half length, two registers, a multiplexer and a controller, running on the same clock domain as the input data. As an example, consider using a resolution of 640 × 480 (width × height), supporting a maximum structuring element of size 63 × 63. Normally, a memory of size $k \times I_w = 6 \times 640$ bits with dual-port functionality is required (11). Here, instead of using a dedicated dual-port memory, a double-width, half-length single-port memory with a size of $2k \times (I_w/2) = 12 \times 320$ can be used that reads and writes two samples every other clock cycle. The memory architecture is illustrated in Fig. 9 together with a simple controller that manages the FFs and the multiplexer.

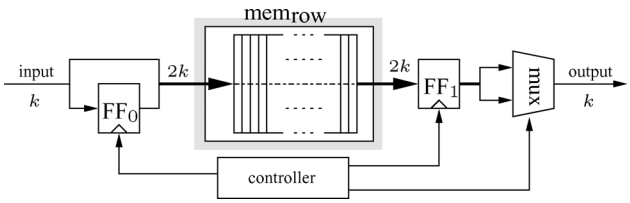


Fig. 9. Row memory implemented with one double-width single-port memory, which performs a read and write operation every other clock cycle. Note that the width of each bus is expressed in k.

The functionality of the k-bit flip-flop $\mathrm{FF_0}$ is to delay an input value in order to concatenate it with the following one. By this procedure, a bus is formed (of doubled width) constituting of two values that are written into the memory. The $2k$-bit flip-flop $\mathrm{FF_1}$ is used when reading from the memory. The multiplexer gives access to one of these two values stored on each position in the memory.

E. FIFO

In streaming data application environments, supporting a TI SE, the padding pixels (discussed in Section IV-A) may be addressed on a controller level by simply omitting them from the calculation without the need to stall the input data stream. However, supporting a SV SE requires the possibility to stall the input data stream since the latency can vary from one side of the image to the other (6). This requires two separate clock domains, separated by an asynchronous FIFO located at the input.

The size of this FIFO is a trade-off between operating frequency and memory resources. The size depends on many parameters, e.g., the relation between input data speed f_{in} and operating speed f_{op}, image size, and maximum supported structuring element size. It can optimized with respect to two objectives: i) low memory requirement, or ii) low power.

The total time it takes to stream a complete frame may be written as $t_{\mathrm{in}} = (1/f_{\mathrm{in}})(I_h I_w)$, where $I_h I_w$ is the image height × width. Furthermore, the total time t_{tot} required by the architecture to process a complete frame is determined by four factors: f_{op}, the image size, the size of the structuring element, and

the location of the origin. Assuming a centered origin, the total time for the architecture to process a complete frame (including padding) is equal to $t_{\rm tot} = (1/f_{\rm op})(I_h + \lfloor \max(B_h)/2 \rfloor)(I_w + \lfloor \max(B_w)/2 \rfloor)$, since half of the values can be inserted as initial values in the FF-chain, recall Section IV-A ($\lfloor . \rfloor$ is the floor function). The overall timing constraint for the architecture may be written as $t_{\rm in} \geq t_{\rm tot}$, or expressed in operating frequency as

$$f_{\rm op} \geq f_{\rm in} \frac{\left(I_h + \left\lfloor \frac{\max(B_h)}{2} \right\rfloor\right)\left(I_w + \left\lfloor \frac{\max(B_w)}{2} \right\rfloor\right)}{I_h \cdot I_w} \quad \text{MHz.} \tag{14}$$

Assuming that the maximum supported structuring element is small compared to the resolution, i.e., $\lfloor \max(B_h)/2 \rfloor \ll I_h$ and $\lfloor \max(B_w)/2 \rfloor \ll I_w$, then $f_{\rm op} \approx f_{\rm in}$ according to (14). Using this approximation, the architecture must at most stall $\lfloor \max(B_h)/2 \rfloor (I_w + \lfloor \max(B_w)/2 \rfloor)$ pixels during padding at the lower boundary of the image. A resolution of 640 × 480 at a frame rate of 25 fps, and supporting a maximum structuring element of 63 × 63, results in input data speed of $f_{\rm in} = 7.68$ MHz. Setting $f_{\rm op}$ to 10 MHz, the required FIFO capacity becomes 21 kb (dimensioned for the padding at the lower boundary). With this FIFO size and using (13), the total amount of memory for the complete architecture is ≈ 25 kb. When increasing $f_{\rm op}$ to 100 MHz, the architecture requires a FIFO with a capacity of ≈2 kb, reducing the total amount of memory to ≈ 6 kb.

The FIFO size has impact on the both the dynamic power consumption according to $P_{dyn} \propto f_{\rm op}$ [38], and the static power dissipation (area dependent). In practice, if minimizing the dynamic power is of high priority, this means operating at the lowest possible speed (for a given supply voltage), i.e., minimizing $f_{\rm op}$, resulting in a large FIFO. To summarize, the memory requirement is dependent on the operating speed $f_{\rm op}$ and memory resources can be traded for low power properties.

V. Implementation Results and Performance

Application: The algorithm runs optimally whenever the structuring element conforms to (7). This is verified in applications where $B(i, j)$ is generated by a continuous function such as in application Fig. 1, where B conforms to the image anamorphism given by the perspective.

The result in Fig. 1(d) has been obtained by applying both opening and closing operations on the motion mask in Fig. 1(b). If implemented by definition, i.e., $\varepsilon\circ\delta\circ\varepsilon$, using (1–4), it requires three image scans, storage of two intermediate images, random memory accesses, and a latency of three frames.

The sequential memory access of our algorithm allows composing cascade filters without intermediate storage. Hence, using our algorithm, the result in Fig. 1(d) can be obtained from 1(b) in one image scan, with very low computational complexity, low intermediate storage (three image lines), and low latency (several image lines).

Architecture: The architecture has been implemented in VHDL using a resolution of 640 × 480 and supporting a flexible structuring element up to 63 × 63. Indeed, in order to correctly filter the largest objects found in the image, we have chosen the largest B to be approximately 1/10 of the image width. In general, there are no algorithmic restrictions on the largest supported structuring element size, but k in (13) will increase accordingly.

TABLE I
Resource Utilization in a Xilinx Virtex II-PRO FPGA and in UMC 0.13 μm CMOS Process. Image 640 × 480 and SV Rectangular SE Up to 63 × 63

FPGA	used	available	**ASIC**	used
Slices	807	13696	**Area** $[mm^2]$	0.31
Block RAM	3	136	**Mem**$_{tot}$ $[kb]$	24.6
LUTs	1365	27392	**Gate count** $[k]$	60
Speed $[MHz]$	80	—	**Speed** $[MHz]$	260

The implementation has been targeted for both FPGA and ASIC: a Xilinx Virtex-II PRO FPGA (XC2VP30-7FF896) and the UMC 0.13 μm CMOS process, respectively. The most important implementation results and properties for both technologies are compiled in Table I, where the area is reported containing all memory blocks. This includes an asynchronous input FIFO of 21 kb, as discussed in Section IV-C (replaced by a dual-port memory of the same size in the ASIC implementation in order to support the simultaneous read and write functionality), resulting in that memory constitutes 86% of the total area in this particular implementation. The gate count is based on a 2-input NAND-gate (5.12 μm^2).

As mentioned in Section IV, the combinatorial critical path passes through the logical operation performed in the compare stage. Pipelining this operation will not necessarily increase the speed figures found in Table I since the bandwidth to the memory is the limiting factor.

In order to compare this work to the PRR and LC architectures discussed in Section I-B, important properties are compiled in Table II as a function of the resolution and the maximum supported SE. SE support refers to the class of supported structuring elements and SE flexibility to the ability to change the structuring element between two adjacent pixels. Naturally, this should be distinguished from the ability to change the structuring element in between frames which is supported by most architectures. The complexity refers to the number of operations per pixel, e.g., in the case of PRR, number of comparators; and in the case of LC, two summations and two additions. The memory requirement is basically the same as for the LC architecture but for the additional n delay elements found in the FF-chain. $\rm T_{exe}$ is reported in number of clock cycles (CC) to process a complete frame but does not include the latency, which is present in all architectures. Table II indicates that while still maintaining low memory requirements, the ability to support SV SEs comes at the cost of the complexity increase from 4 to n, found in the compare stage as an increased number of comparators, and multiplexers, making n proportional to the maximum supported SE width $\max(B_w)$.

A. Extensions

The present algorithm situates at the extreme end of optimization, imposing restrictions on the SE shape. For more demanding applications, there are two possible extensions that increase the applicability of the algorithm.

TABLE II
IMPORTANT PROPERTIES OF VARIOUS ARCHITECTURES, WHERE I_s AND B_s ARE THE SIDE IN PIXELS OF A SQUARE INPUT IMAGE AND STRUCTURING ELEMENT ($k = \lceil \log_2(B_s) \rceil$)

Design	**PRR**[29]	**LC**[30]	**This work**
SE support	Arbitrary	Rectangular	Rectangular
SE SV	No	No	Yes
Complexity	$2k$	4	B_s
mem [bits]	$(B_s - 1)\, I_s$	$k(I_s + 1)$	$k(I_s + B_s)$
$\mathbf{T}_{exe}$ [CC]	I_s^2	$I_s^2 + B_s\, I_s$	$I_s^2 + B_s\, I_s$

Extension to Richer Shapes: The currently supported class of shapes, noncentered rectangles, includes L_1-balls squares. The benefit of this restriction is a considerable reduction of the complexity (memory requirement, number of operations per pixel, and latency), far below other algorithms supporting SV SE. This shape restriction may be relaxed to include a richer class of shapes. Indeed, convex shapes may be supported by splitting them into two parts: above and below the origin, and applying these two halves sequentially in direct and reverse raster scan. For example, L_2 balls, disks can be implemented in two scans using two half disks, see Fig. 10.

Supporting richer shapes is paid by some increase in complexity. The number of operations per pixel becomes $2B_s$ comparisons, instead of B_s previously. The memory consumption becomes $N = I_w \times I_h$, since a complete frame needs to be stored, instead of previously one line I_w only. Applying two scans sequentially also increases the latency to more than one frame, roughly $I_h + B_h$ image lines.

Concerning the HW implementation aspects, a richer shape will require feeding the FF-chain stage of the architecture (see Fig. 7) with a different value for each column. This will require a more complex controller to manage all comparator inputs. Secondly, one will need a richer coding of the structuring element B. Having a richer shape will need additional resources just for reading—at every new pixel i, j of the input image—the exact shape $B(i, j)$. A better encoding, and possibly compression, of B will become very useful to reduce the memory bandwidth which can rapidly exceed the bandwidth of the input image.

Relaxing the Size Variability Constraints: As explained in Section V, this algorithm runs optimally when B is a continuous function and its most advantageous use case is anamorphism-aware filtering, allowing to obtain results in one raster scan. Besides that, it can also be used in other applications as discussed in the introduction. For example, image coding and restoration from skeletons in Fig. 2 belong to applications where the SE size depends on the image content, i.e., circular SV SEs. Since the radius of the circles are determined by the image content, such restriction as in (7) may not be maintained.

The restrictions in (7) concern only the streaming implementation of the algorithm and can be relaxed. Indeed, the only consequence of violating (7) is that the output pixels do not arrive in the raster scan order, and that the algorithm needs a memory to store the output image.

Hence, the result in Fig. 2(c) has been obtained in two scans by dilating the skeleton in Fig. 2(b) by two half circles (upper and lower halves) with memory requirements equal to one image size.

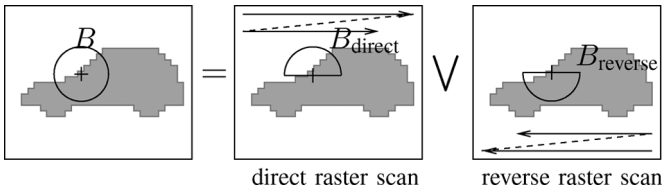


Fig. 10. Implementation of richer SE shape classes. Example: dilation by a disk, decomposed as sup of dilations by half-disks implemented in direct and reverse raster scans.

VI. CONCLUSION

This paper presents a novel algorithm for binary ε and δ supporting spatially variant, rectangular structuring elements. The memory data is decoupled from the structuring element size, which is the property that enables the structuring element flexibility. The complexity is far below other existing algorithms supporting a SV SE, which makes it compete with algorithms supporting only TI SEs. The sequential memory access pattern allows composing cascaded filters with low latency, and without intermediate storage. For more demanding applications there is an extension to support richer SE shapes (balls, diamonds) in two raster scans. Also, extending from binary to functional morphology is possible and is currently under investigation. The presented algorithm is interesting for various use cases: cascaded morphological filters running on systems under heavy time and space constraints such as embedded or communication systems or possibly also low-end user terminals.

A corresponding HW architecture of the algorithm is also presented, intended to be used as an accelerator in embedded systems. The memory requirement of the architecture is mainly proportional to the image width while the computational complexity is proportional to the maximum supported SE width. The image data is processed in raster scan order without storing the image in memory, which allows processing high resolution images on low memory systems. The architecture has been successfully verified on a Xilinx Virtex-II PRO FPGA and implemented as an ASIC in the UMC 0.13 μm CMOS process using a resolution of 640 $\times$ 480 and supporting maximum SE of 63 $\times$ 63 at 25 fps.

REFERENCES

[1] J. Serra, *Image Analysis and Mathematical Morpohology*. New York: Academic, 1982.
[2] F. Kristensen, H. Hedberg, H. Jiang, P. Nilsson, and V. Öwall, "An embedded real-time surveillance system: Implementation and evaluation," *J. Signal Process. Syst.*, vol. 52, no. 1, July 2008.
[3] J. Serra, "Structuring functions," in *Image Analysis and Mathematical Morphology*. New York: Academic, 1988, ch. 2.2, pp. 40–46.
[4] C.-S. Chen, J.-L. Wu, and Y.-P. Hung, "Statistical analysis of space-varying morphological openings withflat structuring elements," *IEEE Trans. Signal Process.*, vol. 44, no. 4, pp. 1010–1014, Apr. 1996.
[5] S. Beucher, J. Blosseville, and F. Lenoir, "Traffic spatial measurements using video image processing," presented at the SPIE Advances in Intelligent Robotics Systems, Cambridge Symp. Optical and Optoelectronic Engineering, Cambridge, MA, Nov. 1987.
[6] S. R. Sternberg, "Grayscale morphology," *Comput. Vis., Graph., Image Process.*, vol. 35, pp. 333–355, 1986.
[7] J. B. T. M. Roerdink and H. J. A. M. Heijmans, "Mathematical morphology for structures without translation symmetry," *Signal Process.*, vol. 15, no. 3, pp. 271–277, 1988.

[8] J. G. Verly and R. L. Delanoy, "Adaptive mathematical morphology for range imagery," *IEEE Trans. Image Process.*, vol. 2, no. 2, pp. 272–275, Feb. 1993.
[9] N. Bouaynaya, M. Charif-Chefchaouni, and D. Schonfeld, "Spatially-variant morphological restoration and skeleton representation," *IEEE Trans. Image Process.*, vol. 15, pp. 3579–3591, 2006.
[10] E. R. Dougherty and R. A. Lotufo*, Hands-on Morphological Image Processing,*. Bellingham, WA: SPIE, 2003.
[11] P. Maragos and R. Schafer, "Morphological skeleton representation and coding of binary images," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34, no. 5, pp. 1228–1244, Oct. 1986.
[12] D. Schonfeld and J. Goutsias, "Morphological representation of discrete and binary images," *IEEE Trans. Signal Process.*, vol. 39, no. 6, pp. 1369–1379, Jun. 1991.
[13] R. Kresch and D. Malah, "Morphological reduction of skeleton redundancy," *Signal Process.*, vol. 38, pp. 143–151, 1994.
[14] M. N. T. Natsuyama, "Edge preserving smoothing," *Comput. Graph. Image Process.*, vol. 9, pp. 394–407, 1979.
[15] Y. Wu and H. Maître, "Smoothing speckled synthetic aperture radar images by using maximum homogeneous region filters," *Opt. Eng.*, vol. 31, no. 8, pp. 1785–1792, 1992.
[16] R. Lerallut, E. Decencière, and F. Meyer, "Image filtering using morphological amoebas," *Image Vis. Comput.*, vol. 25, no. 4, pp. 395–404, 2007.
[17] J. Debayle and J. C. Pinoli, "General adaptive neighborhood image processing—Part I: Introduction and theoretical aspects," *J. Math. Imag. Vis.*, vol. 25, no. 2, pp. 245–266, 2006.
[18] J. Debayle and J. C. Pinoli, "General adaptive neighborhood image processing—Part II: Practical application examples," *J. Math. Imag. Vis.*, vol. 25, no. 2, pp. 267–284, 2006.
[19] M. Charif-Chefchaouni and D. Schonfeld, "Spatially-variant mathematical morphology," in *Proc. IEEE Int. Conf. Image Processing*, Austin, TX, Nov. 1994, pp. 13–16.
[20] N. Bouaynaya, M. Charif-Chefchaouni, and D. Schonfeld, "Theoretical foundations of spatially-variant mathematical morphology—Part I: Binary images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 823–836, May 2008.
[21] N. Bouaynaya, M. Charif-Chefchaouni, and D. Schonfeld, "Theoretical foundations of spatially-variant mathematical morphology—Part II: Gray-level images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 837–850, May 2008.
[22] M. van Herk, "A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels," *Pattern Recognit. Lett.*, vol. 13, no. 7, pp. 517–521, 1992.
[23] F. Lemonnier and J. Klein, "Fast dilation by large 1D structuring elements," in *Proc. Int. Workshop Nonlinear Signal and Image Processing*, Halkidiki, Greece, Jun. 1995, pp. 479–482.
[24] M. V. Droogenbroeck and H. Talbot, "Fast computation of morphological operations with arbitrary structuring elements," *Pattern Recognit. Lett.*, vol. 17, no. 14, pp. 1451–1460, 1996.
[25] B. Kisačanin and D. Schonfeld, "A fast thresholded linear convolution representation of morphological operations," *IEEE Trans. Image Process.*, vol. 3, no. 4, pp. 455–457, Apr. 1994.
[26] J. C. Klein and R. Peyrard, "Pimm^1, an image processing ASIC based on mathematical morphology," in *Proc. 2nd Annu. IEEE ASIC Seminar and Exhibit*, Rochester, NY, Sep. 25–28, 1989, pp. P7 1.1–P7 1.4.
[27] S. Fejes and F. Vajda, "A data-driven algorithm and systolic architecture for image morphology," in *Proc. IEEE Int. Conf. Image Process.*, Austin, TX, Nov. 13-16, 1994, vol. 2, pp. 550–554.
[28] J. Velten and A. Kummert, "FPGA-based implementation of variable sized structuring elements for 2D binary morphological operations," in *Proc. IEEE 1st Int. Workshop Electronic Design, Test, and Applications*, Jan. 29-31, 2002, pp. 309–312.
[29] S. Y. Chien, S. Y. Ma, and L. G. Chen, "Partial-result-reuse architecture and its design technique for morphological operations with flat structuring element," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 9, pp. 344–371, Sep. 2005.
[30] H. Hedberg, F. Kristensen, and V. Öwall, "Low-complexity binary morphology architectures with flat rectangular structuring elements," *IEEE Trans. Circuits Syst. I*, vol. 55, pp. 2216–2225, 2008.
[31] O. Cuisenaire, "Locally adaptable mathematical morphology using distance transformations," *Pattern Recognit., J. Pattern Recognit. Soc.*, vol. 39, no. 3, pp. 405–416, 2006.
[32] *Mathematical Morphology in Image Processing*, E. R. Dougherty, Ed. New York: Marcel Dekker, 1992, ch. 8.
[33] I. Ragnelmam, "Fast erosion and dilation by contour processing and thresholding of distance map," *Pattern Recognit. Lett.*, vol. 13, pp. 161–166, 1992.
[34] P. E. Danielsson, "Euclidean distance mapping," *Comput. Graph. Image Process.*, vol. 14, pp. 227–248, 1980.
[35] G. Borgefors, "Distance transformations in digital images," *Comput. Vis., Graph., Image Process.*, vol. 34, no. 3, pp. 344–371, 1986.
[36] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2D euclidean distance transform algorithms: A comparative survey," *ACM Comput. Surv.*, vol. 40, no. 1, Feb. 2008.
[37] E. Dejnozkova, "Architecture dédiée au traitement d'image basé sur les équations aux dérivées partielles," Ph.D. dissertation, School of Mines, Mines, France, 2004.
[38] J. M. Rabaey, A. Chandrakasan, and B. Nikolić*, Digital Integrated Circuit*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2003.

Hugo Hedberg received the M.S.E.E. and Ph.D. degrees in electrical engineering from Lund University, Lund, Sweden, in 2001 and 2008, respectively. His doctoral thesis addressed hardware accelerators for automated digital surveillance systems.

His main research area is hardware implementations of image processing algorithms targeted for real-time embedded systems with a special interest in developing low-complexity architectures for morphological operations. He is currently with Prevas, Stockholm, Sweden.

Petr Dokladal graduated from the Technical University, Brno, Czech Republic, in 1994, as a telecommunication engineer and received the Ph.D. degree in 2000 from the University of Marne la Vallée, France, in general computer sciences, specialized in image processing.

He is a research engineer at the Centre of Mathematical Morphology, School of Mines, Paris, France. His research interests include medical imaging, image segmentation, object tracking, and pattern recognition.

Viktor Öwall, (M'90) received the M.Sc. and Ph.D. degrees in electrical engineering from Lund University, Lund, Sweden, in 1988 and 1994, respectively.

During 1995 to 1996, he joined the Electrical Engineering Department, University of California, Los Angeles, as a Postdoctorate, where he mainly worked in the field of multimedia simulations. Since 1996, he has been with the Department of Electrical and Information Technology, Lund University. His main research interest is in the field of digital hardware implementation, especially algorithms and architectures for wireless communication, image processing, and biomedical applications. Current research projects include combining theoretical research with hardware implementation aspects in the areas of pacemakers, channel coding, video processing, and digital holography.

Dr. Öwall was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING from 2000–2002 and is currently an Associate Editor of the TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS.

Parallel implementation of sequential morphological filters

Jan Bartovský, Petr Dokládal, Eva Dokládalová & Vjačeslav Georgiev

Journal of Real-Time Image Processing

ISSN 1861-8200

J Real-Time Image Proc
DOI 10.1007/s11554-011-0226-5

Springer

J Real-Time Image Proc
DOI 10.1007/s11554-011-0226-5

ORIGINAL RESEARCH PAPER

Parallel implementation of sequential morphological filters

Jan Bartovský · Petr Dokládal · Eva Dokládalová · Vjačeslav Georgiev

Received: 17 June 2011 / Accepted: 29 September 2011

Abstract Many useful morphological filters are built as more or less long concatenations of erosions and dilations: openings, closings, size distributions, sequential filters, etc. An efficient implementation of these concatenations would allow all the sequentially concatenated operators run simultaneously, on the time-delayed data. A recent algorithm (see below) for the morphological dilation/erosion allows such inter-operator parallelism. This paper introduces an additional, intra-operator level of parallelism in this dilation/erosion algorithm. Realized in a dedicated hardware, for rectangular structuring elements with programmable size, such an implementation allows obtaining previously unachievable, real-time performances for these traditionally costly operators. Low latency and memory requirements are the main benefits when the performance is not deteriorated even for long concatenations or high-resolution images.

J. Bartovský (✉) · E. Dokládalová (✉)
Laboratoire d'Informatique Gaspard-Monge,
Equipe A3SI, Université Paris-Est, ESIEE Paris,
93162 Noisy-le-Grand Cedex, France
e-mail: bartovsj@esiee.fr

E. Dokládalová
e-mail: dokladae@esiee.fr

P. Dokládal
Center of Mathematical Morphology (CMM),
Mines-ParisTech, 77305 Fontainebleau Cedex, France
e-mail: petr.dokladal@mines-paristech.fr

V. Georgiev
Faculty of Electrical Engineering, University of West Bohemia,
30614 Pilsen, Czech Republic
e-mail: georg@kae.zcu.cz

Keywords Mathematical morphology · Serial filters · Real-time implementation · Dedicated hardware

1 Introduction

Mathematical morphology is very popular, self-contained, image processing framework providing a complete set of tools from filtering, multi-scale image analysis to pattern recognition. It has been used in a number of applications, including the biomedical and medical imaging, video surveillance, industrial control, video compression, stereology, or remote sensing [8, 16, 18, 19].

In image-interpretation applications requiring a high correct-decision liability, one often use robust multi-criteria and/or multi-scale analysis. It generally consists of a serial concatenation of alternating atomic operators dilation and erosion with a progressively increasing computing window, the so-called structuring element (SE). Its examples include:

- Alternate sequential filters (ASF)—that are concatenations of openings and closings with a progressively increasing structuring element, useful for multi-scale analysis [19, 20].
- Size distributions—(aka granulometries) are concatenations of openings allowing measuring the size distribution in a population of objects [14, 17, 28].
- Statistical learning—a selected set of morphological operators ζ_i can be separately applied to an image f. Then for every pixel $f(x, y)$, the vector of values $\zeta_i(f)(x, y)$ can serve as a vector of descriptors for the pixel-wise learning and classification [4].

Although built from basic blocks (the dilation and erosion), these operators are costly due to the number of

Published online: 21 October 2011

iterations. The real-time capabilities (i.e., low latency) are even more difficult to achieve due to the sequential data dependence and high memory requirements.

The recently introduced algorithm for the dilation and erosion [7] shows how to handle efficiently the implementation of such concatenations. It enables an inter-operator level of parallelism where all the sequentially concatenated operators can run simultaneously, on time-delayed data. Obviously, it is fully exploited only if the algorithm is implemented in an adequate hardware (HW).

In this paper, we propose a HW implementation of the original algorithm and we introduce an additional, intra-operator level of parallelism. Such an implementation allows obtaining previously unachievable, real-time performances for these traditionally costly operators. Section 2 discusses the state of the art of existing dilation/erosion algorithms and concludes by the novelties presented in this paper. Sections 3 and 4 recall the definitions and algorithmic principles and Sect. 5 illustrates the functional scheme of a sequential HW implementation. Then, Sect. 6 introduces the parallel implementation allowing obtaining an additional performance increase. Finally, Sect. 7 presents results obtained on FPGA.

2 Fast implementations of morphological filters

During the last decades, propositions of optimized implementations concentrated on the efficiency of computing the dilation and erosion. The majority of authors measures this efficiency as a number of comparisons per pixel. Nevertheless, the minimization of comparisons can result in high memory requirements. It can even penalize the execution time since the overall latency issues are neglected.

For the following, we define as *operator latency* the latency introduced by the dependence of the result on future data samples. For example the max filter $y_i = \max(x_{i-2}, x_{i-1}, \ldots, x_{i+2})$ has operator latency 2. We define as *algorithm latency* any additional latency introduced by the algorithm, e.g., the necessity to perform a reverse scan on data. The latency is a time-less measure expressed in a number of data samples.

Note that there is also an additional delay, called *computing latency*, induced by the time needed to compute the result after all data are available. It is a platform dependent measure, independent of two previous latency definitions. In the example above, the polyadic max can either be executed sequentially on sequential machines, or in parallel on the dedicated hardware.

Then the overall latency of the system is the sum of these three terms.

2.1 Algorithmic advances

The most efficient dilation/erosion algorithms are based on the SE decomposition to a set of basic, more easily optimized shapes, see [22, 30, 31]. A special attention is paid to 1-D algorithms obtaining a significant gain in the overall performance.

The most popular 1-D algorithm is called HGW (published by van Herk [26], and Gill and Werman [9]). The computation complexity per pixel is $\mathcal{O}(1)$, i.e., is independent of the SE size. Nonetheless, the algorithm requires two scans: forward and reverse. Lemonnier [12] proposes to identify local extrema and propagate their values as long as it is covered by the SE. Again, forward and reverse scans are needed. Notice that in 2-D the reverse scan of the vertical component multiplies the algorithm latency by a factor of the image width.

Lemire [11] proposes a fast, stream-processing algorithm for causal linear SE. It runs also on floating-point data, has low memory requirements and zero algorithm latency. However, an intermediate storage of local maxima results in a random access to the input data. This problem is solved in Dokladal and Dokladalova [7] using the strictly sequential access to the data. It allows the real on-the-fly computing and has zero algorithm latency.

A different approach represents the algorithm proposed by Buckley and Van Droogenbroeck [25]. It detects the anchors—the portions of the signal unaffected by the operator—and updates only the parts to be modified by the operation. It has zero algorithm latency. However, the algorithm uses a histogram which makes memory requirements dependent on the number of gray levels.

Recently, Urbach and Wilkinson [24] propose an algorithm for arbitrary shaped 2-D flat SEs based on the computation of multiple horizontal linear SEs for every pixel and storing them in a look-up table. The result is then computed by taking the maximum from the intermediate values (stored in the look-up table) corresponding to the shape of the SE. The horizontal linear SE can be computed with one of the above mentioned 1-D algorithms.

2.2 Implementations

In the beginning of the 1970s, Klein and Serra [10] propose a texture analyzer for linear and rectangular SE by decomposition based on the delay-line concept. More recently, Velten and Kummert [27] propose also a delay-line based architecture supporting arbitrary-shaped SEs. However, the complexity being quadratic $\mathcal{O}(W^2)$ (W denotes the length of the SE), it becomes penalizing for large SEs. In Chien et al. [2], the authors show how to reduce the number of redundant comparisons within large

SEs by merging adjacent smaller SEs. The complexity becomes $\mathcal{O}(\lceil \log_2(W) \rceil)$ with identical memory requirements.

Clienti et al. [3] proposes a highly parallel morphological system-on-chip. It is based on a set of neighbourhood processors optimized for 3×3 SE, interconnected in a partially configurable pipeline. Larger SE is obtained by homothecy (see Basic Notions below) requiring to instantiate a deep pipe of these processors or multiple image scans.

A similar approach has been published by Deforges et al. [5]. Based on Xu's [30] decomposition combined with a stream implementation, the authors propose a pipeline architecture composed of the elementary parametrizable blocks. It handles an arbitrary convex shape of structuring element in only one scan of the input image. However, using large SE will require the proportional increase of the atomic HW resources, concatenated in a deep pipe. The principal limitation comes from a limited programmability of such a pipe.

To complete this brief survey, we can also cite the systolic architectures proposed by Diamantaras and Kung [6], Malamas et al. [13] or Shih et al. [21] for gray-scale or binary morphology. Their common inconvenience is the need of an intermediate storage for 2-D structuring element and a long response time of the system.

2.3 Novelty of this paper

All previous algorithms concentrate on the optimization of the dilation/erosion algorithm, rather than on the optimization of the entire operator chain. The performance will inevitably decrease for more complex applications with long loops (iterations, idempotence) or concatenations.

Consider some serial morphological filter $\zeta = \delta\varepsilon \ldots \delta\varepsilon$ (with δ and ε standing for dilation and erosion, see Sect. 3 below for details). If the atomic operators δ and ε use sequential access to data then the entire ζ can run on pipelined, time-delayed data. If the atomic algorithms δ and ε—in addition—have zero algorithm latency, then the entire chain ζ inherits the same properties: sequential data access and zero algorithm latency. This is an interesting property, since computing ζ suddenly becomes very efficient: in stream, with only the (further irreducible) operator latency of ζ. In comparison with the preceding state of the art, the Dokladal [7] algorithm extends the possibility to implement erosion/dilation filters with the arbitrarily large, 2-D SE in only one scan over the image, with the minimal algorithm latency and memory requirements. If implemented in a dedicated hardware, we can obtain the same features even for the implementation of long concatenations ζ.

This paper starts from the sequential HW implementation of the Dokladal algorithm (published in Bartovský et al. [1]). It describes more deeply the implementation features and optimization techniques. It shows how to exploit the inter-operator parallelism in ζ. Additionally, it introduces another intra-operator parallelism in the computation of the 2-D erosion/dilation. The 2-D erosion/dilation is implemented as a run-time programmable block. The operation (erosion or dilation), the size and the origin of the SE can be modified on-the-fly between two frames. Several such blocks concatenated in a pipeline allow obtaining previously unachievable, real-time performances for operators in the form of ζ. We can reach almost 100 Hz HDTV 1080p performance, independent of the length of ζ.

3 Basic principles

Let $\delta_B, \varepsilon_B : \mathbb{Z}^2 \to \mathbb{R}$ be a dilation and an erosion on grey-scale images, parameterized by a structuring element B, assumed rectangular, flat (i.e., $B \subset \mathbb{Z}^2$) and translation-invariant, defined as

$$\delta_B(f) = \vee_{b \in B} f_b \tag{1}$$

$$\varepsilon_B(f) = \wedge_{b \in \widehat{B}} f_b \tag{2}$$

The hat $\widehat{\ }$ denotes the transposition of the structuring element, equal to the set reflection $\widehat{B} = \{x \mid -x \in B\}$, and f_b denotes the translation of the function f by some vector b. The SE B is equipped with an origin $x \in B$. Below, $B(x)$ denotes B placed with its origin at x.

The implementation of (1) and (2) consists of searching the extremum of f within the scope of B

$$[\delta_B(f)](x) = \max_{b \in B} [f(x - b)] \tag{3}$$

$$[\varepsilon_B(f)](x) = \min_{b \in B} [f(x + b)] \tag{4}$$

The dilation and erosion by convex structuring elements verify the homothecy. Let B be some convex structuring element, and rB the change of scale of B, with $r > 1, r \in \mathbb{Z}$. Then for the dilation we have

$$\delta_{rB} f = \underbrace{\delta_B \ldots \delta_B}_{r-\text{times}} f. \tag{5}$$

The homothecy allows obtaining large-size dilations by repeating several times the dilation by a small SE.

Combinations of dilations and erosions form other operators. The basic concatenation products are opening $\gamma_B = \delta_B \varepsilon_B$ and closing $\varphi_B = \varepsilon_B \delta_B$. From here we can form the alternating filters obtained as $\gamma\varphi$, $\varphi\gamma$, $\gamma\varphi\gamma$ and $\varphi\gamma\varphi$. The number of combinations obtained from two filters is rather limited. Other filters can be obtained by combining two *families* of filters. This leads to morphological alternate sequential filters (ASF), originally proposed by Sternberg [23], and studied in Serra [19], Chapter 10. In general, it is

a family of operators parameterized by some $\lambda \in \mathbb{Z}^+$, obtained by the alternating concatenation of two families of increasing and decreasing filters $\{\xi_i\}$ and $\{\psi_i\}$, respectively, such that $\psi_n \leq \ldots \leq \psi_1 \leq \xi_1 \leq \ldots \leq \xi_n$.

The most known ASF are those based on openings and closings, obtained by taking $\psi = \gamma$ and $\xi = \varphi$:

$$\mathrm{ASF}^{\lambda} = \gamma^{\lambda}\varphi^{\lambda}\ldots\gamma^{1}\varphi^{1} \tag{6}$$

starting with a closing, and

$$\mathrm{ASF}^{\lambda} = \varphi^{\lambda}\gamma^{\lambda}\ldots\varphi^{1}\gamma^{1} \tag{7}$$

starting with an opening.

The second application example is the size distribution of a population of objects [14, 17, 28]. One way to compute them is the residue from a sequence of openings

$$\mathrm{sd}(\lambda) = \|\gamma^{\lambda} - \gamma^{\lambda-1}\| \tag{8}$$

The following section briefly recalls the principles of the used algorithm, [7]. It starts by the 1-D dilation algorithm, followed by the principle of separation of the n-D dilation into perpendicular 1-D computations, preserving the stream aspects at all levels.

4 Algorithm description

4.1 1-D dilation algorithm

The algorithm principles and properties have been originally published in [7]. We briefly recall the main important principles for HW implementation.

For some 1-D input signal $f : 1\ldots N \rightarrow \mathbb{R}$, the algorithm[1] computes the value $\delta_B f(wp) = f(rp)$. The SE $B, B \subset \mathbb{Z}$, is a line segment, containing its origin, and not necessarily symmetric. Consequently, the size of B is given by the span from the centre to the left and to the right, SE1 and SE2. The length of B is SE1 + SE2 + 1. The coordinates *wp* and *rp* stand for the current *writing* and *reading* positions.

The algorithm uses a FIFO queue Q. The queue supports operations *push, pop* and *dequeue* (modifying the FIFO's content) and queries *front* and *back*. The input signal f is read sequentially. A newly read value $f := f(rp)$ is inserted in the FIFO queue as a pair $\{f, rp\}$, the sample f and reading position *rp* (code line 3). In this pair, one can independently access either the value or the position by indexing. For example the last stored element's value can be accessed (without dequeing it) by a query Q.back()[1].

The algorithm does not store non decreasing intervals (see [7] for details and proof). The values that appear to belong to increasing or constant intervals are dequeued (code lines 1–2). Consequently, the values stored in the queue are always ordered in a decreasing order.

The old values, uncovered by the SE, are retrieved from the queue (code lines 4–5). The result of the dilation $\delta_B f(x)$ is read at the front of the queue (code line 7). The result becomes available as soon as enough input data have been read, otherwise the output is empty (code line 9).

4.2 2-D dilation algorithm

The separability of n-D morphological dilation into lower dimensions is a well known property. For example, a rectangular SE R decomposes as $R = H \oplus V$ where H and V are horizontal and vertical segments and $\oplus$ is the Minkowski addition. Then the dilation by a rectangle R can be computed by concatenation of two perpendicular 1-D dilations

$$\delta_R = \delta_V \delta_H. \tag{9}$$

The sequential access to the data in 1-D makes that two perpendicular 1-D computations can be assembled into 2-D with sequential access at both levels, 2-D and 1-D, for both input and output data. There is no additional latency and no intermediate storage (the data are pipelined).

See the example of dilation by a rectangle $R = H \oplus V$ of an $N \times M$ image f, Fig. 1. The image is sequentially read in the raster-scan mode, line by line from left to right. The various indices *rp* and *wp* denote *reading* and *writing positions*, respectively, for the segments H and V, and the rectangle R. The computation is illustrated for column i and line j, i.e., the result $\delta_R f(i, j)$ is to be written at wp_R.

The computation of $\delta_R = \delta_V\, \delta_H$ decomposes as follows: the current reading position of δ_R coincides with rp_H, that is $rp_R = rp_H$. The result of the horizontal dilation, at wp_H, is immediately read by the vertical dilation in the respective column, that is $wp_H = rp_V$. The result of the vertical dilation δ_V is written at the writing position wp_R, i.e., $\delta_V = wp_R$.

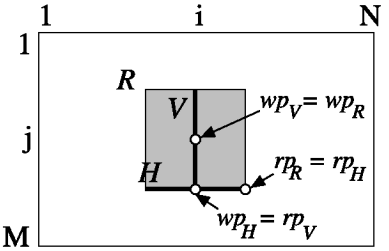


Fig. 1 Decomposition of dilation by a rectangle R into two 1-D dilations by segments H and V, see (9)

[1] See Appendix for the 1-D dilation pseudocode.

Notes:

- The rp_R and wp_R run over the image in the raster scan mode. The distance between rp_R and wp_R is the actual, further irreducible, operator latency.
- There is one instance of the horizontal dilation running at the current line j, and N instances of vertical dilation, i.e., one per each column.

5 Sequential hardware implementation

In this section, we firstly describe in details the implementation of the 1-D dilation in a basic block that (thanks to the separability) can be used as a building brick in any dimensional system. We illustrate this below on a 2-D dilation or erosion. Secondly, we show how the intra-operator parallelism can be introduced to increase the performance.

5.1 1-D algorithm implementation

The 1-D algorithm presented in Sect. 4 is a system with sequential behavior. It contains a while loop that can not be unrolled (uncertain number of iterations). The common way to implement such a system is the Mealy finite-state machine (FSM, see [15]). The FSM issues all the necessary operations over the memory as well as it controls the input and output data-flow. It consists of two states {*S*1, *S*2} (Fig. 2).

The *S*1 state *Dequeues all useless values*. It is a data dependent stage of the algorithm as it dequeues an a priori unknown amount of pixels. This is represented in the code by the while statement (code line 1). Consequently, its computation time varies from 1 to the SE size clock cycles in the worst case when all the previously stored pixels are unnecessary.

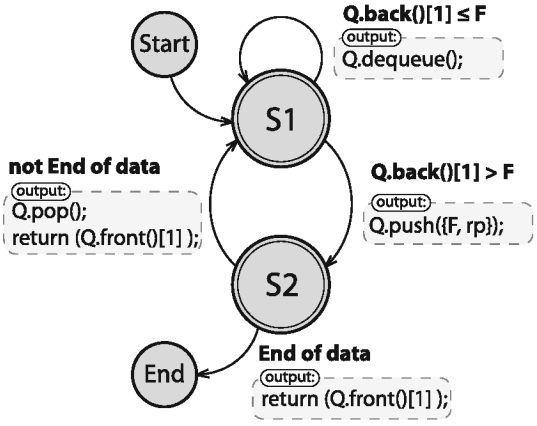


Fig. 2 State diagram of the 1-D algorithm FSM. State transition conditions are typeset in *bold*; the output signals are given in a *shadow bounding box*

The *Enqueue current sample* signal (code line 3) is issued upon the transition from *S*1 to *S*2.

The *S*2 state handles the code lines 4 and 5, *Delete too old values*, and the lines 6–9 *Return valid value* or *Return empty*. These instructions are independent and executed in parallel. Consequently, the execution of *S*2 takes only one clock cycle.

5.2 1-D block architecture

The HW implementation can be separated into two areas (Fig. 3), the FSM part and the memory part. The FSM manages entire computing procedure and temporarily stores values in the memory part. The memory instantiates one FIFO queue in the case of horizontal direction (horizontal scan) and N FIFO queues in the vertical case (N is the image width). The queues are addressed by a modulo N Page counter (active in the case of vertical direction).

The control unit is a sequential circuit that manages the state transitions. It increments the *rp*, *wp* and manages the page and position stamp counter appropriately. The Control unit also performs the queue memory operations and handles the backward full flags used for data-flow control.

5.2.1 Principle

In the beginning of *S*1, the last queued pixel is invoked by Back() operation from the queue and fetched to the Comparator 1. The pixel value is compared with the value of the current sample. Notice that the comparator evaluates all three possible relations (>, <, =) at the time, for both dilation and erosion. The Control unit decides on the basis of comparison results and selected morphological function (dilation or erosion) whether the enqueued pixel is to be dequeued (lines 1–2). Otherwise, the current pixel is extended with the reading position stamp and enqueued (line 3).

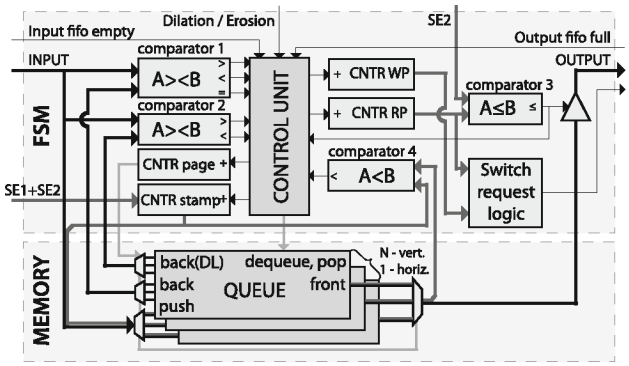


Fig. 3 Overview of implemented 1-D architecture. The FSM part manages computation, memory part contains the data storage queues

The *S2* invokes the oldest queued pair {pixel, stamp} by Front() operation. The read pixel is a correct result if *rp* has already reached or exceeded the *SE2* parameter. This output allowing condition (line 6) is checked by Comparator 3. The deletion of outdated values is performed by comparing the current value of the reading-position stamp with the *rp* value of the oldest pair. Notice that the deletion has no impact on the output dilation value because Pop() operation (lines 4 and 5) issued by the Control unit has an effect only with the next clock edge.

The switch request logic is used only in the parallelized version of the architecture, see Sect. 6. It is a simple block containing several comparators which generate a signal with the last output value of each parallel segment. Its purpose is to inform the switch connected to the output that the end of the segment has been reached and the following segment is to be processed.

The entire set of parameters, i.e., SE dimensions and selection of the morphological function, is run-time programmable at the beginning of the line for 1-D, and of the frame for the 2-D implementation, respectively. In addition, no further controller is needed; the internal behavior is driven only by the regular scan order data-flow.

5.2.2 *Reducing the impact of data-dependency*

Hereafter, we briefly describe two techniques brought to the system for higher throughput and lesser area occupation.

5.2.2.1 Number of dequeue steps The data-dependent number of dequeue steps (below denoted by *Steps*) has an unpleasant consequence on the HW design: longer balancing FIFOs (see Fig. 4), lower data throughput. For HW design it is important to minimize the worst case upper bound $Steps_{\text{orig}} = \text{SE} - 1$.

The number of stored pixels is within [1, SE[. Suppose that we are to dequeue D pixels. We know that the pixels are queued in a strictly decreasing order. Thus, if the DL-th pixel ($DL<D$) can be dequeued then also all previous pixels can be dequeued. This can be done at the same time. Hence, the worst-case number of dequeue steps is

$$Steps = \max_{D<SE}(D\text{div}DL + D\text{mod}DL) \tag{10}$$

where D denotes the number of pixels to be dequeued and div and mod the integer division and the remainder operations. D can be regarded as a uniformly distributed, random variable $D \in [1, \text{SE}[$. Then we need to find the optimal DL that minimizes *Steps* (Eq. 10) for all D such as

$$DL_{optim} = \arg\min_{DL<D} \max_{D<SE}(D\text{div}DL + D\text{mod}DL) \tag{11}$$

The optimal DL_{optim} brings us the minimal number of dequeue steps $Steps_{\text{optim}}$

$$Steps_{\text{optim}} = \min_{DL<D} \max_{D<SE}(D\text{div}DL + D\text{mod}DL) \tag{12}$$

Table 1 exemplifies, for some *SE* widths, the original and reduced number of dequeue Steps, obtained with optimal *DL*. Notice that more than one optimal *DL* can exist.

The SE is user programmable. DL_{optim} also is programmable, though it is useless to make it accessible to the user; it can instead be read from a LUT for every given user-specified SE.

5.2.2.2 Pixel addressing The absolute pixel addressing in the queues can be advantageously replaced in the HW by using the modulo addressing. Instead of the absolute reading position *rp*, we use the relative modulo position $stamp = rp \mod SE$. The pixels are enqueued by *Q.push*(*f*, *stamp*) (code line 3).

The delete condition of line 4 changes accordingly. Using the modulo addressing, a stored pixel becomes outdated whenever its modulo address equals the current pixels' one ($stamp = Q.front()[2]$).

The advantage of the modulo addressing is a smaller data width. It fits into $\lceil \log_2(\text{SE} - 1) \rceil$ bits, whereas the absolute addressing requires $\lceil \log_2(N - 1) \rceil$ bits. This is mainly advantageous for vertical orientation using N queues for a unit.

5.3 2-D dilation implementation

Recall that dilation is separable into lower dimensions, Eq. 9. The dilation by a rectangle can be implemented using two 1-D dilation blocks, Fig. 4.

The computing latency of the dilation varies per each pixel. In order to preserve the input/output stream flow, one needs to compensate the different latencies by insertion of balancing FIFOs. The FIFO fills when the preceding block outputs data faster than the subsequent block can read. The depth of this FIFO directly defines the upper bound of the system latency of the 2-D block.

Obviously, the FIFOs should be as small as possible. The necessary depth infers from the dequeuing worst case

$$F_{\text{input}} = \frac{Steps_{hor} + 2}{StreamRate} - 1 \tag{13}$$

$$F_{balance} = N\left(\frac{Steps_{ver} + 2}{StreamRate} - 1\right) \tag{14}$$

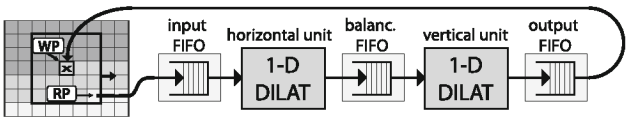


Fig. 4 2-D implementation is composed of 1-D blocks for respective directions

Table 1 Optimal dequeue length *DL*, original and reduced number of dequeue steps for selected SE widths

SE width	3	11	21	31	41
$Steps_{orig}$	2	10	20	30	40
DL_{optim}	2	3, 4	4, 5, 6	5, 6, 7	6, 7
$Steps_{optim}$	2	4	7	9	10

where $Steps_{hor}$ and $Steps_{ver}$ are numbers of the dequeue steps in horizontal and vertical directions (12).

The output FIFO ensures a permanent stream delay in all circumstances. Its maximal size is a sum of both FIFOs (input and balancing). The instantaneous filling of output FIFO is complementary to the filling of both FIFOs combined. The overall delay does not change. If more 2-D blocks are pipelined to form compound operators (e.g., opening, closing, ASF), only one output FIFO at the end is necessary.

The output and balancing FIFOs can be merged (see Fig. 5) into one memory thanks to the following properties: (1) the vertical unit reads exactly one pixel from the balancing FIFO for each pixel written to the output FIFO. Consequently, filling of these two FIFOs is complementary; the occupied memory spaces can not collide with each other, (2) the read/write activity is at most 1 access per 2 clock cycles. Hence, reading ports of both FIFOs can use one memory port and the writing ports can use the other memory port (without overloading it). Merging both FIFOs reduces the memory to approximately one half. The result memory (see Fig. 5) has two pairs of standard FIFO ports, but it contains only one dual-port RAM.

5.4 Clock rate

The overall average clock rate stays in the interval from 2 clock cycles per pixel in the best case, up to 3 clock cycles per pixel in the worst case. The current rate between 2 and 3 clock cycles per pixel is data dependent.

A temporarily worst case arrives whenever a monotonously decreasing signal is followed by a high value. This makes a number of samples to be dequeued at the time (code lines 1–2, and the S1 state of the FSM), and the computing latency temporarily increases. However, the average computing latency remains unchanged, compensated by the fact that during the entire monotonous decrease of the signal no values have been dequeued. Therefore, the average clock cycles per pixel rate remains constant.

5.5 Memory requirements

The memory requirements of the 2-D architecture consist of horizontal and vertical computation-involved memories and two balancing FIFOs, defined by (13) and (14).

In the vertical case, the algorithm uses a several queues. Instantiating N separated memories would be resource inefficient because the FPGA RAM blocks could not be exploited. Instead, these queues are gathered in a single dual-port memory (see Fig. 6) since only one queue is accessed at the time (the others are idle). A single memory block also allows using an off-chip memory.

Every queue has a related pair of front and back pointers which must be retained throughout the entire computation process. The appropriate pair is always read before the particular queue is used and the modified pointers are stored back after the computation left the queue. These pointers are stored in a separated pointer memory. The queues are efficiently packed into RAM blocks resulting in a small memory extension.

Let $W \times H$ denote the width $\times$ height of the rectangular SE, and bpp bits per pixel. The memory contribution per 2-D unit is given by:

$$M_{hor} = W(\text{bpp} + \lceil \log_2(W-1) \rceil) \quad \text{(bits)} \tag{15}$$

$$M_{ver} = N(H(\text{bpp} + \lceil \log_2(H-1) \rceil) + +2\lceil \log_2(H-1) \rceil) \quad \text{(bits)} \tag{16}$$

The following example illustrates the very low memory consumption achieved thanks to the stream processing. Neither the input, nor the output or any working image are buffered.

Example: Consider a dilation of 8 bpp, SVGA image (i.e., $800 \times 600 = N \times M$) by a square, 31×31 SE.

The computation (the queues) requires (15) and (16)

$$M_{hor} = 31(8+5) = 403 \text{ bits}$$

and

$$M_{ver} = 800(31(8+5) + 2 \times 5) = 330.4 \text{ kbits}$$

resulting in a total of 331 kbits for the 2-D dilation.

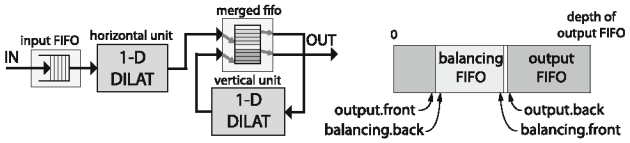


Fig. 5 Merged FIFO replaces the balancing and output FIFOs to reduce memory requirements

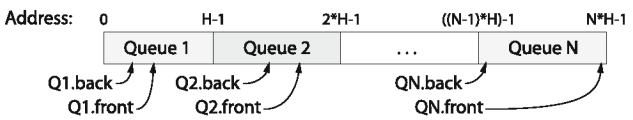


Fig. 6 Vertical queues are mapped into linear memory space side by side. The front and back pointers are stored at separated memory

The input and the balancing FIFOs require (13) and (14)

$$F_{\text{input}} + F_{\text{balance}} = (N+1)\left(\frac{Steps+2}{StreamRate} - 1\right) 8 \text{ bpp}$$
$$= (800+1)\left(\frac{9+2}{3} - 1\right) 8 \text{ bpp} \approx 17 \text{ kbits}$$

The total memory needs to implement the 2-D dilation are $331 + 17 = 349$ kbits. This is far below the mere size of the image itself $800 \times 600 \times 8$ bpp $= 3.84$ Mbits which, at any moment, does not need to be stored.

6 Parallel hardware implementation

This section develops and implements the concept of the previously mentioned *intra-operator parallelism* in the dilation/erosion operator. Its main objective is to increase the throughput while maintaining the beneficial properties of the proposed algorithm, namely the sequential data access and minimal algorithm latency as much as possible.

The principle is based on utilization of concurrently working units that process different parts of the image simultaneously. The number of units used in parallel for horizontal and vertical directions defines the parallelism degree (PD). Considering that the input data are fetched line by line, we propose a solution minimizing the waiting-for-data periods of all units.

The image partition for 2-D dilation conforms to the intersection of two horizontal and vertical partitions (Fig. 7). Its granularity is determined by the PD. The horizontal partition (partition of image among horizontal units) is interleaved, whereas the vertical units use the partition into compact blocks.

During the parallel processing the computation runs simultaneously at multiple segments of the image, see Fig. 8. These segments must belong to different columns and lines, i.e., must be placed on a diagonal.

The input data rate can be theoretically *PD*-times faster than the computational throughput of one unit. Therefore, each image line needs to be buffered in a line buffer. The line buffers are filled at the external (fast) pixel rate and read by the internal *PD*-times slower rate.

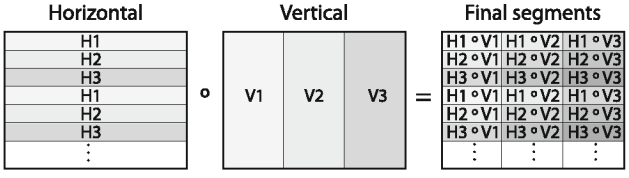


Fig. 7 Example of image partition for $PD = 3$: image is divided horizontally line by line and into *PD* equal stripes in a vertical direction. The final image partition is obtained by intersection

Figure 8 gives an example for $PD = 3$. We have three horizontal (H1…H3), and tree vertical (V1…V3) processing units. As soon as the line buffer receives the first pixel, the first horizontal unit H1 starts the processing and feeds results to the first vertical unit V1. Its output is fed to the first output line, see Fig. 8a. After N received pixels, the output of H1 is connected to V2 which belongs to output line 1. Since the H1 left V1 and line 2 is read, the H2 can start processing second line feeding V1 connected to output line 2, see Fig. 8b. When the 2N input pixel is received, the H1 connects to V3, H2 connects to V2 and H3 connects to V1, see Fig. 8c, and so on.

6.1 Architecture

The parallel architecture depicted in Fig. 9 contains four separable generic parts scalable by $n \equiv PD$: input buffer, horizontal and vertical parts and output buffer. The input buffer is mainly composed of the 1-to-n multiplexer and n line buffers (we omitted the control logic). It divides the fast input stream into n (n-times slower) streams processed by computational units as described above. The output buffer composes n slow streams of the processed data into a single, fast, output stream respecting the image horizontal scan order. The operator blocks can be concatenated into more complex functions (opening, closing, ASF, etc.). The buffers are used only at the beginning and at the end of the chain.

Both horizontal and vertical parts instantiate n balancing FIFOs, n horizontal or vertical units, and one switch that manages the interconnection. Each horizontal unit along with the front-end FIFO conforms to Sect. 5.2.

The width of the processing area proportionally affects both vertical memories, see (14) and (16). The area of every horizontal unit remains unchanged, since every unit processes the entire line. The overall memory of the horizontal part is a factor of n. Contrarily, the memory requirements of every vertical part is divided by n because it processes only a fraction of the original image width. The area of the FSM of vertical units increases linearly with n.

6.2 Switching

The routing of the computation units is handled by a switch block. Every switch contains n input ports from previous units and the same number of output ports linked to the subsequent units. The purpose of the switch is to manage up to n interconnection channels. Notice that they are bidirectional: forward data and backward FIFO full flag. As described in Fig. 8, the output switching of all input ports is circular, i.e., V1, V2 … Vn, V1, V2,… and so forth. This

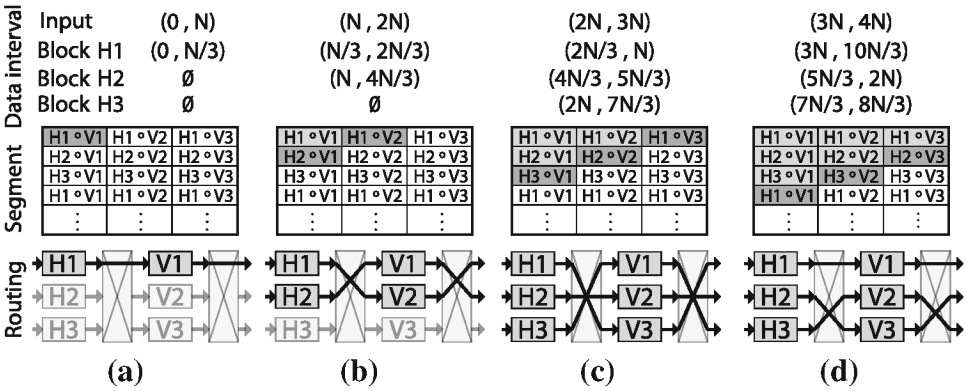


Fig. 8 Image partitioning and switch routing in parallel processing for $PD = 3$. Decomposed in time **a** beginning of processing, **b–d** after kN pixels, $k = 1\ldots3$. The *shading* denotes the state. *Dark gray* being computed. *Light gray* already computed. *White* waiting

property makes the switching easier because the only condition to evaluate is when to switch and whether the requested output unit is available.

The moment when to switch a given port is provided by the preceding unit's *Switch Request* logic. It generates a request every time it crosses the border of adjacent segments. If the desired unit is free, the switch reconnects the channel. If not, the switch sets high the FIFO full flag of requesting unit to stall it until the desired destination unit is freed and the channel can be established. All the channels are switched independently so stalling one unit does not affect the others.

Figure 10 depicts the basic unit of the switch for one pair of input/output ports referred to as A. For n pairs of ports this circuitry is instantiated n-times. Each input port possesses a related control unit block that manages all channel transitions considering the availability of the requested partition. If this is still occupied, the requesting computation unit is stalled by holding its FIFO full flag active.

7 Experimental results

The proposed 2-D stream processing architectures have been implemented in VHDL, and targeted to the Xilinx Virtex5 FPGA (XC5VSX95T-2) using the XST synthesis tool. The processing clock frequency is 100 MHz. Notice that the queues are gathered in a block RAM memory, and thus its access time augments the critical path delay. The measured performance for non-parallel architectures (PD=1) in terms of overall latency, clock cycles per pixel and FPGA area are given by Tables 2 and 3.

One can observe that the overall latency is factor of the SE size, the image width (both caused by operator latency) and the pixel rate (computing latency). The average pixel rate (AR) remains constant (Table 2). The average pixel rate can be expressed by (17) and the stream frame-per-second (FPS) ratio by (18). T_{proc} is overall time consumed by processing and f_{clk} = 100 MHz is clock frequency of computation units.

$$\text{AR} = \frac{T_{\text{proc}} - 2\text{SE}_2(N + M + \text{SE}_2)/PD}{NM} \quad \text{(clk/px)} \qquad (17)$$

$$\text{FPS} = \frac{f_{\text{clk}} PD}{ARNM + 2\text{SE}_2(N + M + \text{SE}_2)} \quad \text{(fr/s)} \qquad (18)$$

M, N denote the width and height of the image, SE_2 denotes the width of the structuring element from the origin rightwards.

Concerning the area occupation (see the Xilinx documentation [29]), the number of registers is quasi-constant; the number of LUTs and BRAM blocks increases linearly with the SE and image sizes (Table 3). Although the vertical memory [size is given by (16)] is packed into the RAM block, the amount of the used memory always

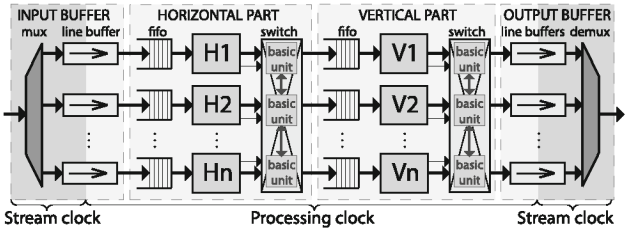


Fig. 9 Overview parallel 2-D architecture. The horizontal and vertical stages can be instantiated several times between input/output buffers to create compound operators

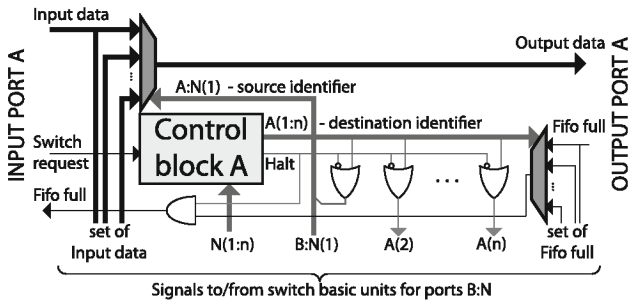


Fig. 10 Basic unit of the switch. Every switch contains n basic units for a correct routing between n input/output ports

Table 2 Timing and area versus SE, SVGA image size, $PD = 1$

Size of SE (sq.)	3 × 3	11 × 11	21 × 21	31 × 31	41 × 41
Latency (clk)	1,908	9,474	18,888	28,351	37,969
Av. rate (clk/px)	2.344	2.356	2.360	2.361	2.361
Registers	212	232	242	242	252
LUTs	584	761	859	859	953
Block RAMs	2	6	13	13	28

Table 3 Timing, frame rate and area w.r.t. image, SE = 31 × 31 square, $PD = 1$

Size of image	CIF	VGA	SVGA	XGA	1080p
Latency (clk)	12,826	23,465	28,351	37,472	69,548
Av. rate (clk/px)	2.371	2.376	2.361	2.383	2.368
Experimental FPS	384	130	85	51	20.5
Worst-case FPS	319	106	68	41	16
Registers	231	237	242	242	253
LUTs	761	853	859	859	1,057
Block RAMs	7	13	13	13	26

exceeds the theoretical value. It is caused by a different memory organizations; e.g., the required word is 13 bits whereas available memories are of width 36 bits and its fractions.

The experimental frames-per-second (FPS) rate is obtained on a natural test image (see Fig. 11). The worst-case FPS is a theoretical worst-case performance of the system expected on the synthetic saw-shaped data.

Table 4 presents the relative speed-up of the parallel architecture versus the intra-operator parallelism *PD*. In terms of overall latency and average processing rate, the processing domain clock cycle is considered as a reference unit. Note that the latencies of parallel versions are merely fractions (divided by *PD*) of non-parallel values.

The FPGA area results, Table 5, are separated into two groups: the area of computing parts and buffers. The area of input and output buffers is linear w.r.t. both N and PD since their essential components are PD line buffers (FIFO memories with independent ports of N elements). The area of the operator units in terms of Slice registers and LUTs is proportional to PD as well because n independent circuits are instantiated in a parallel manner. Although the overall vertical memory requirements remain unaffected by PD, practically the number of occupied RAM blocks slightly increases. It is caused by a different memory organization.

The ultimate timing results ($PD = 6$) versus the image size are listed in Table 6. It illustrates the real performance of the architecture. It allows to achieve at least 96 fps with 1080p image size (full HD TV image size).

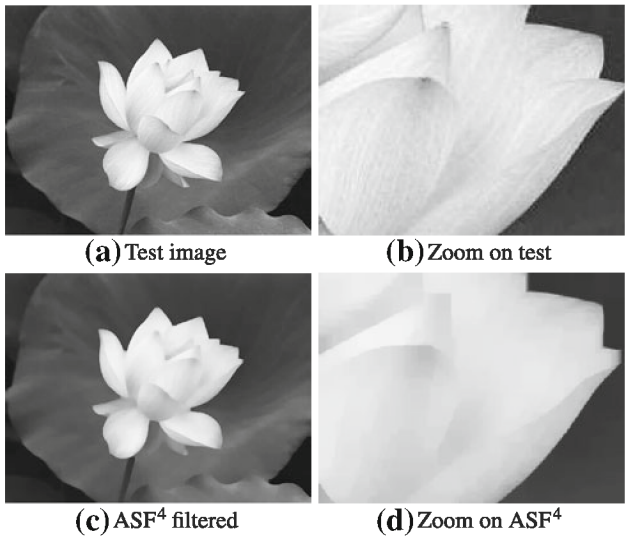


Fig. 11 **a** Experimental 800 × 600 lotus image. **b** Zoom on the fine veinous texture disadvantageous for the algorithm. **c** Result of ASF^4 filter, see Eq. 6 or 7. **d** Zoom on the ASF^4 filtered image

The worst case occurs on artificial saw-shaped image with no constant plateaus. Such an image infers the maximal number of algorithm's while-loop iterations. The best case fps (not mentioned in the table) is obtained with a constant image. A real, unfiltered image containing textures or random noise achieves performance somewhere between best and worst cases. For instance at 1080p, the worst case is 96 fps, best case 140 fps, achieved experimental performance is 113 fps.

This frame rate remains constant for any morphological serial filter (such as ASF). Obviously, the FPGA area increases accordingly to the size of the ASF. The implementation is eased by the fact that one can use an off-chip memory.

7.1 Comparison with existing HW implementations

Table 7 presents a comparison with other recent architectures. The table is divided into three sections. The processing unit section presents the features of a single 2-D computational unit. The second part the HW specifications, and the third part the performance on a given application, an ASF filter.

One can see that Clienti [3] offers a high throughput for small 3 × 3 rectangular SEs. Similarly, the Chien ASIC chip [2] provides very reasonable performance on small SEs. On the other hand, Déforges [5] directly offers large, non-rectangular, convex SE, but with a lower processing rate. The programmability is not mentioned, namely, the possibility to control the SE shape after the synthesis is not clear.

Table 4 Timing versus degree of intra-operator parallelism *PD*

PD	2	3	4	5	6
Latency (clk)	14,243	9,561	7,244	5,818	4,893
Av. rate (clk/px)	1.220	0.824	0.625	0.505	0.426
Exp. speed up	1.938	2.869	3.785	4.682	5.554

SVGA image, SE = 31 × 31 square

Table 5 Area versus degree of intra-operator parallelism *PD*

PD	2	3	4	5	6
Registers	650	978	1,280	1,605	1,938
LUTs	2,138	3,227	3,862	4,875	6,054
Block RAMs	13	14	14	18	21
Reg. buf	661	969	1,279	1,587	1,896
LUTs buf	1,408	2,086	2,776	3,459	4,135

SVGA image, SE = 31 × 31 square

Table 6 Timing and frame rate versus image size, *PD* = 6, SE = 31 × 31 square

Size of image	CIF	VGA	SVGA	XGA	SXGA	1080p
Latency (clk)	2,208	3,996	4,893	6,390	7,391	11,641
Av. rate (clk/px)	0.443	0.431	0.426	0.426	0.427	0.418
Experimental FPS	2,075	724	472	290	174	113
Worst-case FPS	1,915	640	411	246	151	96

Although all these solutions are efficient for small SE sizes or short concatenations, they become more or less penalized for longer filters. This issue is illustrated in an example application, Table 7. It estimates the performances on a five-stage $\mathrm{ASF}^5 = \varphi_{11\times11}\gamma_{11\times11}\cdots\varphi_{3\times3}\gamma_{3\times3}$. Decomposed into a sequence of dilations and erosions, it can be realized as $\mathrm{ASF}^5 = \varepsilon_{11\times11}\delta_{21\times21}\ldots\varepsilon_{5\times5}\delta_{3\times3}$. Notice that it makes use of a progressively increasing SE. On neighborhood processors, large SE can be obtained using the homothecy Eq. 5. The Clienti SPOC instantiates 16 of 3 × 3 processing units. Hence, the ASF^5 will require five image scans with the entire image necessarily buffered in the memory. Chien also uses the homothecy. This deteriorates the throughput.

One could immediately figure out to instantiate a longer pipe in order to reduce the number of image scans. Alas, a long, fixed-length pipe lacks the flexibility. Consider another application for the illustration of the problem: the size distributions, exemplified by Fig. 12. Contrarily to ASF, the size distributions are often sampled sparsely, the SE increments by more than one and, at the same time, one often goes to much larger SE sizes. Every opening $\{\gamma_{B\_i}\}$ in (8) needs to be output and stored in the memory to compute the subtraction. For small sizes, a long pipeline is underused and the workload of the processing units unbalanced, whereas for large λ one may still need several image scans.

For example, for sizes λ = 5, 10, 15, 20, 25, as in Fig. 12, the Clienti SPOC will require seven image scans. The 16 processor pipe is underused for λ = 5, 10, 15, whereas it will require 2 scans for λ = 20, 25.

Our processing unit with programmable SE size avoids using the homothecy. This allows optimal workload distribution over the entire pipe, so important for processing large images in real-time systems.

8 Conclusions

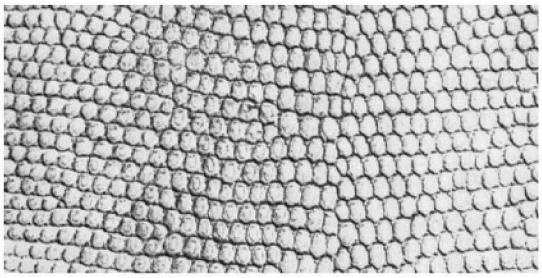
This paper describes an efficient implementation of serial morphological filters with flat, rectangular structuring elements of arbitrary size. The efficiency is obtained through the following properties:

- The computational complexity is linear w.r.t. the image size and independent of the SE size.
- The overall latency is mostly equal to the latency of the operator, inferred by the size of the used structuring element.

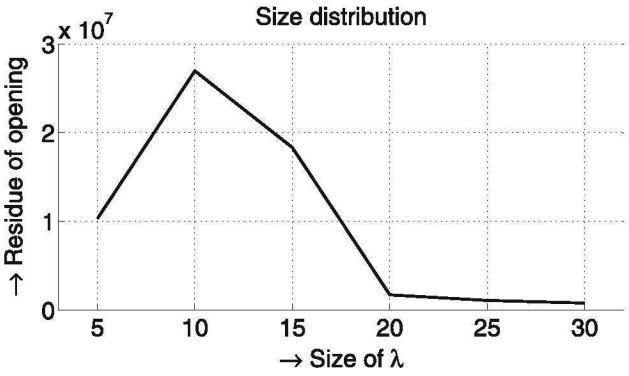
Table 7 Comparison of several FPGA and ASIC architectures concerning morphological dilation and erosion

	Processing unit					HW system		Example application ASF^5	
	Parallel degree	Supported SE	Throughput (Mpx/s)	f_{max} (MHz)	Clock rate (clk/px)	Number of units	Supported image	Image scans	FPS
Clienti [3]	4	Arb. 3 × 3	403	100	0.25	16[a]	1,024 × 1,024	5	80
Chien [2]	1	Disk 5 × 5	190	200	1.052	1	720 × 480	27	21.5
Déforges [5]	1	Arb. convex	50	50	1	1[a]	512 × 512	11	17.2
This paper	6	Rectangles	234	100	0.426	11[a]	1,920 × 1,080	1	113

[a] Number of available stages varies with size of used FPGA



(a) Example of a texture



(b) Size distribution sd(λ)

Fig. 12 The size distribution of the texture grain

- It uses strictly sequential access to the data at all algorithm levels.
- Low memory consumptions (far below the size of the image) allow embedding on a single chip complex operators able to process large images.
- Two levels of parallelism: (1) the inter-operator parallelism in serial concatenations $\zeta = \delta\varepsilon \ldots \delta\varepsilon$, allow running all these atomic δ and ε operators simultaneously, and (2) the intra-operator parallelism in every atomic dilation/erosion. The intra-operator parallelism is scalable (tested up to six) and allows the decomposition of fast streams into several slower streams processed in parallel without altering the streaming property of the system.

The architecture serves as a basic building block to be used for construction of more complex operators such as ASF, granulometries, etc., with the same properties and performance. The performances obtained on an FPGA are approaching the 100 Hz HDTV 1080p standard. These performances are far above what has been reported in the literature. These performances allied to the programmability are extremely interesting. They open the accessibility of advanced morphological operators in industrial systems running under severe time constraints. The number of examples includes the on-line production control, aging material defectoscopy, etc., wherever one requires processing of high resolution images and low latency.

Appendix: The 1-D dilation pseudocode

Algorithm 1: df←1D_DILATION (rp, wp, f, SE1, SE2, N)

Input: rp, wp - reading/writing position; f - input signal value $f(rp)$; SE1, SE2 - SE size towards left and right; N - length of the signal; Q - a FIFO-like queue

Result: output signal value $\delta_B f(wp)$

```
1 while Q.back()[1] ≤ f do
2 |   Q.dequeue() ;            // Dequeue useless valuess
3 Q.push({f, rp}) ;            // Enqueue the current sample
4 if wp - SE1 > Q.front()[2] then
5 |   Q.pop() ;                // Delete too old value
6 if rp = min (N, wp + SE2) then
7 |   return (Q.front()[1] ) ;  // Return valid value
8 else
9 |   return ({}) ;             // Return empty
```

References

1. Bartovský, J., Dokládalová, E., Dokládal, P., Georgiev, V.: Pipeline architecture for compound morphological operators. In: ICIP10 (2010)
2. Chien, S.-Y., Ma, S-.Y., Chen, L.-G. Partial-result-reuse architecture and its design technique for morphological operations with flat structuring elements. In: IEEE Transactions on Circuits and Systems for Video Technology. **15**(9):1156–1169 (2005)
3. Clienti, Ch., Beucher, S., Bilodeau, M.: A system on chip dedicated to pipeline neighborhood processing for mathematical morphology. In: EURASIP (ed) EUSIPCO 2008, Lausanne (2008)
4. Cord, A., Jeulin, D., Bach, F.: Segmentation of random textures by morphological and linear operators. In: 8th ISMM, pp. 387–398 (2007)
5. Déforges, O., Normand, N., Babel, M.: Fast recursive grayscale morphology operators: from the algorithm to the pipeline architecture. J. Real Time Image Process., pp. 1–10 (2010). doi: 10.1007/s11554-010-0171-8
6. Diamantaras, K.I., Kung, S.Y.: A linear systolic array for real-time morphological image processing. J VLSI Signal Process. Syst. **17**(1):43–55 (1997)
7. Dokládal, P., Dokládalová, E.: Computationally efficient, one-pass algorithm for morphological filters. J. Visual Commun. Image Represent. **22**(5):411–420 (2011)
8. Dougherty, E.R.: Mathematical morphology in image processing. Taylor and Francis Inc., London (1992)
9. Gil, J., Werman, M.: Computing 2-d min, median, and max filters. IEEE Trans. Pattern Anal. Mach. Intell. **15**(5):504–507 (1993)
10. Klein, J.-C., Serra, J.: The texture analyser. J Microsc. **95**:349–356 (1972)
11. Lemire, D.: Streaming maximum–minimum filter using no more than three comparisons per element. CoRR, abs/cs/0610046 (2006)
12. Lemonnier, F., Klein, J.-C.: Fast dilation by large 1D structuring elements. In: Proceedings of the International Workshop on

Nonlinear Signal and Image Processing, Greece, pp. 479–482 (1995)
13. Malamas, E.N., Malamos, A.G., Varvarigou, T.A. (2000) Fast implementation of binary morphological operations on hardware-efficient systolic architectures. J VLSI Signal Process. Syst. **25**(1):79–93 (2000)
14. Maragos, P.: Pattern spectrum and multiscale shape representation. IEEE Trans. Pattern Anal. Mach. Intell. **11**(7):701–716 (1989)
15. Mealy, G.H.: A method for synthesizing sequential circuits. Bell Syst. Tech. J. **34**:1045–1079 (1955)
16. Najman, L., Talbot, H. (eds.): Mathematical morphology: from theory to applications. ISTE Ltd, Wiley (2010)
17. Sabourin, R., Genest, G., Prêteux, F.: Off-line signature verification by local granulometric size distributions. IEEE Trans. Pattern Anal. Mach. Intell. **19**(9):976–988 (1997)
18. Serra, J.: Image Analysis and Mathematical Morphology, vol. 1. Academic Press, New York (1982)
19. Serra, J. Image Analysis and Mathematical Morphology, vol. 2. Academic Press, NY (1988)
20. Serra, J., Vincent, L.: An overview of morphological filtering. Circuits Syst. Signal Process. **11**(1):47–108 (1992)
21. Shih, F.Y., Chung, T.K., Pu, C.C.: Pipeline architectures for recursive morphological operations. IEEE Trans. Image Process. **4**(1):11–18 (1995)
22. Soille, P., Breen, E., Jones, R.: Recursive implementation of erosions and dilations along discrete lines at arbitrary angles. IEEE Trans. Pattern Anal. Mach. Intell. **18**(5):562–567 (1996)
23. Sternberg, S.: Grayscale morphology. Comput Vision Graph Image Process. **35**(3):333–355 (1986)
24. Urbach, E.R., Wilkinson, M.H.F.: Efficient 2-D grayscale morphological transformations with arbitrary flat structuring elements. IEEE Trans. Image Process. **17**(1):1–8 (2008)
25. Van Droogenbroeck, M., Buckley, M.J. Morphological erosions and openings: fast algorithms based on anchors. J Math. Imaging Vis. **22**(2-3):121–142 (2005)
26. van Herk, M.: A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. Pattern Recognit. Lett. **13**(7):517–521 (1992)
27. Velten, J., Kummert, A.: Implementation of a high-performance hardware architecture for binary morphological image processing operations. In: The 2004 47th Midwest Symposium on circuits and systems, 2004. MWSCAS '04. vol. 2, pp. II-241–II-244 (2004)
28. Vincent, L.: Granulometries and opening trees. Fundam. Inf. **41**(1–2):57–90 (2000)
29. Xilinx.: Virtex-5 family documentation. Available at: http://www.xilinx.com/support/documentation/virtex-5.htm (2009)
30. Xu, J., Decomposition of convex polygonal morphological structuring elements into neighborhood subsets. IEEE Trans. Pattern Anal. Mach. Intell. **13**(2):153–162 (1991)
31. Zhuang, X., Haralick, R.M.: Morphological structuring element decomposition. Comput. Vis. Graph. Image Process. **35**(3):370–382 (1986)

Journal of Real-Time Image Processing manuscript No.
(will be inserted by the editor)

Jan Bartovský · Petr Dokládal · Eva Dokládalová · Michel Bilodeau · Mohamed Akil

Real-Time Implementation of Morphological Filters with Polygonal Structuring Elements

Received: date / Revised: date

Abstract In mathematical morphology, the circular structuring elements (SE) are used whenever one needs angular isotropy. Difficult to implement efficiently, the circles are often approximated by convex, symmetric polygons that decompose under the Minkowski addition to 1-D inclined segments.

In this paper, we show how to perform this decomposition efficiently, in stream with almost optimal latency to compute gray-scale erosion and dilation by flat regular polygons. We further increase the performance by introducing a spatial parallelism while maintaining the sequential access to data.

We implement these principles in a dedicated architecture that can be concatenated to efficiently compute sequential filters, or granulometries in one scan. With a configurable image size, programmable SE size, this architecture is usable in high-end, real-time industrial applications. We show on an example that it conforms to the real-time requirements of the 100Hz 1080p FullHD TV standard, even for serial morphological filters using large hexagons or octagons.

Keywords Mathematical Morphology, Hardware Implementation, Alternating Sequential Filter, Parallel Computation, Polygonal Structuring Element

J. Bartovský
Faculty of Electrical Engineering, University of West Bohemia, Pilsen, Czech Republic.
Computer Science Laboratory Gaspard Monge, ESIEE Paris, University Paris-Est, Noisy-le-Grand, France.
E-mail: j.bartovsky@esiee.fr

E. Dokládalová, and M. Akil
Computer Science Laboratory Gaspard Monge, ESIEE Paris, University Paris-Est, Noisy-le-Grand, France.
E-mail: {e.dokladalova, m.akil}@esiee.fr

P. Dokládal and M. Bilodeau
Centre for Mathematical Morphology, Mines ParisTech, Fontainebleau, France.
E-mail: {petr.dokladal, michel.bilodeau}@mines-paritech.fr

1 Introduction

Since its first introduction in late 1960's, the mathematical morphology has settled in the field of image processing as a useful tool for analysis of the shape or the form of spatial structures [14, 16, 17]. Over time it has found its application as a widely-used image processing technique [7, 15].

Thanks to the recent technological development of sensors, the resolution of images increased to tens of megapixels. Certain morphological operations, e.g., top-hat transformation, ultimate openings, granulometry, alternating sequential filters (ASF) [18], etc., on such large images require a large structuring element (SE), since its size shall be proportional to the size of the image and its contents.

Existing hardware implementations either support rectangles using SE the decomposition (fast computation, but angular anisotropic), or support arbitrarily-shaped SEs at the cost of significant performance decrease. Our work supports polygonal SEs at the performance rate of the rectangular SEs.

The paper is organized as follows: Section 2 brings a small survey of existing morphological algorithms and architectures. Section 3 outlines the basic aspects of morphological dilation and erosion, and how to decompose the polygons into a set of lines. Section 4 describes the algorithm, and its use to decompose polygons while preserving the sequential access to data, minimal memory consumption and latency. Section 5 gives the functional implementation of the algorithm. The principal result, a parallel version using two levels of parallelism (temporal and spatial) is presented in Section 6. Finally, Section 7 presents experimental results achieved on an FPGA.

2 State of the art

In this section we briefly discuss the state of the art of algorithms for dilation and erosion followed by the introduction of recently proposed morphological hardware implementations.

Most efficient dilation/erosion algorithms using 2-D SEs are based on the SE decomposition [1, 19, 23]. These algorithms are highly optimized within a reduced (usually 1-D) scope achieving a significant performance gain. The composition of the optimized 1-D algorithms accelerates the n-D computation as well.

The first and still one of the the most popular 1-D algorithms is van Herk [21] (proposed also by Gil and Werman [10]). Although the computation complexity is constant, it requires two image scans: forward and backward. Lemonnier [13] proposed an approach of localizing local extrema and propagating their values as long as they are covered by the SE. Again the forward and backward image scans are required. In [12], Lemire proposed a fast, stream-processing algorithm for causal linear SE, but an intermediate storage of local maxima results in a random access to the input data. This problem is solved in Dokladal [9] who proposed a new algorithm with sequential access to the data, zero latency, and very low memory requirements. It allows a real "on-the-fly" computing, and minimizes the amount of working memory.

Soille [20] uses polygons approximated by periodic lines. The complete dilation by a polygon requires several iterations over the image. Each iteration is then obtained by the fast 1-D van Herk algorithm oriented by desired angle.

2.1 Hardware implementations

Velten and Kummert [22] proposed a naive, delay-line based architecture supporting arbitrarily-shaped SEs. The complexity is quadratic and the memory requirements are proportional to the SE and to the width of the image due to the need to store all the pixels to be reused. It has been partially improved by Chien *et al.* [5]. They removed a relevant number of redundant comparisons within the large SEs by merging several adjacent smaller SEs together. Even though the presented architecture supports a small 5×5 disc SE only, the proposed principle allows SEs to be extended to larger polygons at the cost of hardware resources.

Ikenaga and Ogura [11] developed a Content Addressable Memory based architecture (CAM) with a large processing element array (up to hundred thousand elements). The processing speed of this architecture for small 5×5 disc SEs is very high (approx. 30 μs for 512×512 px image), but the cost of CAM memories and their power consumption become limiting for large images. This architecture uses the homothecy that is iterative usage of small SE to obtain a larger SE. The iterativity significantly decreases performances because multiple passes over the image must be done.

Clienti *et al.* [6] published a highly parallel system called Several neighborhood Processors-On-Chip (SPoC). It is based on a set of neighborhood processors optimized for 3×3 SE, interconnected in a configurable pipeline. Larger (and polygonal) SEs are obtained by homothecy that requires instantiating a deep pipeline of these processors or multiple passes over the image.

Recently, Deforges *et al.* [8] designed a morphological architecture supporting arbitrary convex polygons as SEs. They decompose the SE into a set of causal two-pixel SEs, which are applied in a sequence. In the case of more complex SEs, however, the need of hardware resources significantly increases.

Prior to this paper, Bartovsky *et al.* [2] reported an efficient parallel design based on the 1-D dilation algorithm [9]. However, it supports rectangular SEs only.

From the previous paragraphs we can see that there are few hardware architectures capable of supporting polygonal SEs, and none of them is optimized for polygons. These architectures are usually suitable for small SEs but lose efficiency for large SEs. In this paper we propose an architecture primarily dedicated to large polygonal SEs.

2.2 Novelty

We present an original implementation of morphological operations dilation and erosion with a polygon-shaped SE (namely hexagon and octagon). We chiefly focus on the following:

1) We discus the issue of polygon SE. It is closely related to the SE decomposition – a generalization of the original 1-D algorithm for operating under different angles, and border handling.

2) We introduce a polyvalent hardware implementation of 1-D dilation/erosion processing unit for large, different angle oriented 1-D SE, so-called Line Unit (LU). Computation is carried out by a recently proposed 1-D dilation algorithm (Dokládal [9]) with a stream-processing capability. The LU retains the original properties of this algorithm allowing its easy concatenation into a pipeline.

3) We place 4 LUs in a sequence to create the Polygon Unit (PU) supporting multiple types of SEs: lines, rectangles, hexagons, and octagons. This block allows a run-time programming of the morphological function (dilation/erosion), SE features (size and position of the origin), and image dimensions. Later, several PUs instantiated in the Parallel Polygon Unit (PPU) allow to meet the requirements of high-end industrial applications.

4) On a choosen application, an ASF, we demonstrate how a whole processing chain can be realized using the proposed PPU in a single image raster scan.

The three last points represent the main contribution of this paper, the morphological hardware architecture for large polygonal SEs. Such a solution remains efficient even for large SEs that overwhelm the capabilities of other architectures, as already discussed in Section 2.1.

3 Basic Notions

Let δ_B, ε_B: $\mathbb{Z}^2 \rightarrow \mathbb{R}$ be a dilation and an erosion on gray-scale images, parameterized by a structuring element B, as-

sumed flat (i.e., $B \subset \mathbb{Z}^2$) and translation-invariant, defined as

$$\delta_B(f) = \bigvee_{b \in B} f_b \; ; \qquad \varepsilon_B(f) = \bigwedge_{b \in \widehat{B}} f_b \tag{1}$$

The hat $\widehat{\ }$ denotes the transposition of the SE, equal to the set reflection $\widehat{B} = \{x \mid -x \in B\}$, and f_b denotes the translation of the function f by some scalar b. The SE B is equipped with an origin $x \in B$.

The basic concatenation of the dilation and erosion forms other morphological operators. The closing and opening on gray-scale images, φ_B, γ_B: $\mathbb{Z}^2 \to \mathbb{R}$, parameterized by a structuring element B, are defined as

$$\varphi_B(f) = \varepsilon_B[\delta_B(f)] \; ; \qquad \gamma_B(f) = \delta_B[\varepsilon_B(f)] \tag{2}$$

Furthermore, the concatenation of the closing and opening forms sequential filters, e.g., ASF. The λ-order ASF is composed of the sequence of λ closings and λ openings with a progressively increasing SE. It starts with either the closing or opening

$$\text{ASF}^\lambda = \varphi^\lambda \gamma^\lambda \varphi^{\lambda-1} \gamma^{\lambda-1} \dots \varphi^1 \gamma^1 \tag{3}$$

$$\text{ASF}^\lambda = \gamma^\lambda \varphi^\lambda \gamma^{\lambda-1} \varphi^{\lambda-1} \dots \gamma^1 \varphi^1 \tag{4}$$

3.1 SE Decomposition

The separability of n-D morphological dilation into lower dimensions is a well-known property. The decomposed dilations are then applied in a sequence according to the following equation

$$\delta_R(f) = \delta_{H \oplus V}(f) = \delta_H(\delta_V(f)) \tag{5}$$

where R denotes a rectangle, H and V horizontal and vertical line segments, respectively. This decomposition applies, in general, to convex shapes.

In order to suppress the angular anisotropy of rectangles (note the difference between a side length and a diagonal length), one prefers using circles. Regarding the implementation aspects, circles are often approximated by regular polygons (all sides have the same length) that are easily decomposable, originally described in [1, 23].

A $2n$-top ($n \in \mathbb{N}$) regular polygon SE P_{2n} can be decomposed into a set of n line SEs L_{α_i}

$$P_{2n} = \underbrace{L_{\alpha_1} \oplus \dots \oplus L_{\alpha_n}}_{n \text{ times}} \tag{6}$$

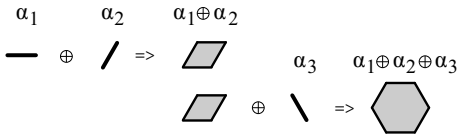
oriented by angle α_i, such as

$$\alpha_i = (i-1)\frac{180^\circ}{n} \;\; [^\circ] \;\; ; i \in \mathbb{N}, i \le n \tag{7}$$

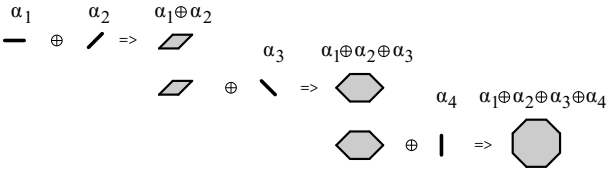
The length of all L_{α_i} is equal to the side of the desired polygon and can be computed from the circumcircle radius R as

$$\|L_{\alpha_i}\| = 2R \sin\left(\frac{180^\circ}{2n}\right) \tag{8}$$

For example, a hexagon can be obtained by three L_{α_i} oriented in $\alpha_i = \{\, 0^\circ, 60^\circ, 120^\circ\}$ on a 6-connected grid, and an octagon by four L_{α_i}, $\alpha_i = \{\, 0^\circ, 45^\circ, 90^\circ, 135^\circ\}$ using an 8-connected grid, see Fig. 1.



(a) Hexagon, $\alpha_i = \{\, 0^\circ, 60^\circ, 120^\circ\}$



(b) Octagon, $\alpha_i = \{\, 0^\circ, 45^\circ, 135^\circ, 90^\circ\,\}$

Fig. 1 Polygon SE composition of line SEs. (a) hexagon is composed of 3 segments, (b) octagon is composed of 4 segments. $\oplus$ operator stands for the Minkowski addition; α_i stands for L_{α_i}.

Hence, from (5) and (6) a 2-D dilation by a $2n$-top polygon $\delta_{P_{2n}}$ of some function $f : \mathbb{R}^2 \to \mathbb{R}$ can be obtained by n consecutive 1-D dilations $\delta_{L\alpha_i}$ by line segments oriented by α_i

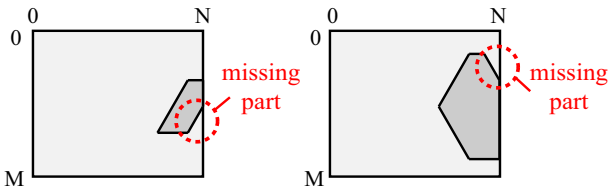
$$\delta_{P_{2n}}(f) = \underbrace{\delta_{L\alpha_1}\,(\,\dots\;\delta_{L\alpha_n}}_{n \text{ times}}\,(f)). \tag{9}$$

The aforementioned decomposition holds true for the unbounded support $\mathbb{Z}^2$. However, when using real images with a bounded support $D \subset \mathbb{Z}^2$, $D = [1..M] \times [1..N]$, decomposition boundary effects appear if at least one $L_{\alpha_i} \neq \{0^\circ, 90^\circ\}$ is used. The cause is that the Minkowski addition of all decomposed line segments of (6), which are cropped by image boundaries after every L_{α_i} of that concatenation, does not necessarily correspond to P_{2n} cropped by image boundaries just once as desired. It is expressed by the following expression where $D\cap$ represents intersection with the image support D

$$\begin{aligned} &D \cap (L_{\alpha_1} \oplus \dots \oplus L_{\alpha_n}) \neq D \cap (L_{\alpha_n} \oplus \dots \\ &\dots \oplus D \cap (L_{\alpha_2} \oplus D \cap (L_{\alpha_1}))). \end{aligned} \tag{10}$$

The illustrative example of such boundary effects with a hexagonal SE is depicted in Fig. 2. We can see that the composition $\alpha_1 \oplus \alpha_2$ is incomplete compared to the desired one in Fig. 1; a small part of the SE is missing. It holds true even for the entire hexagon, the composition $\alpha_1 \oplus \alpha_2 \oplus \alpha_3$ is also incomplete. It is caused by the right boundary cropping not only the final P_{2n}, but also all intermediate results. The cropped values are later missing to form an appropriate polygon section.

This issue is solved by adding a padding to the image. The section of P_{2n} contained inside the image support is then complete, the missing part of P_{2n} is located in the



(a) $\alpha_1 \oplus \alpha_2$ incomplete (b) $\alpha_1 \oplus \alpha_2 \oplus \alpha_3$ incomplete

Fig. 2 Polygon SE composition without padding. The desired SEs presented in Fig. 1 are incomplete, a small triangle is missing.

padding area. The added padding contains recessive values, i.e., values that do not affect the computation of particular morphological operator (for $f{:}D \to V$, $\wedge V$ for dilation, $\vee V$ for erosion). The thickness of padding is different in horizontal an vertical direction and is determined by the size of oblique segments, particularly by the half of vertical and horizontal projection

$$B_{\mathrm{H}} = \|L_{\alpha_i}\| \cos(\alpha_2)/2 \quad \text{[pixels]} \tag{11}$$
$$B_{\mathrm{V}} = \|L_{\alpha_i}\| \sin(\alpha_2)/2 \quad \text{[pixels]}. \tag{12}$$

4 Algorithm Description

This section explains the algorithmic principles involved in this paper. First, we expose the 1-D algorithm used for the dilation by line segments with arbitrary orientation. Next, we show how to combine these 1-D computations in order to obtain polygons running in stream. The issue of boundary handling is addressed afterwards.

4.1 1-D Dilation Algorithm

The implementation of (1) consists of searching the extremum of f within the scope of SE B

$$[\delta_B(f)](x) = \max_{b \in B} [f(x-b)] \tag{13}$$
$$[\varepsilon_B(f)](x) = \min_{b \in B} [f(x+b)] \tag{14}$$

The algorithm used below is based on the property [9] that for some $B(x)$ (which contains its origin) the computation of the dilation $\delta_B f(x)$ needs only those values of $f(x_i)$ that can "be seen" from x when looking over the topographic profile of f, see Fig. 3. The values shadowed by the mountains - that can not be maxima - are immediately excluded from the computation.

From the implementation point of view, assuming a sequential access to the input data f, the dilation $\delta_B f(x)$ depends on points read after x. We say that B is non causal. One can transform a non causal SE to a causal one by utilizing the property that dilation commutes with translation

$$\delta_{B+t} f(x) = \delta_B f(x-t), \qquad \forall t \in D \tag{15}$$

These two principles are used by Alg. 1. For each pixel of some input signal $f : \mathbb{Z} \to R$, the algorithm reads one

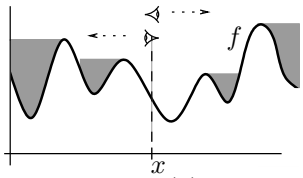


Fig. 3 Computing the dilation $\delta_B f(x)$: Values in valleys shadowed by mountains when looking from x over the topographic relief of f are useless.

pixel $F = f(rp)$ at the so-called *reading position rp* and writes back one result pixel $dF = \delta_B f(wp)$ at the current *writing position wp*, such as $rp > wp$. The wp coordinate coincides with the origin of the SE, rp conforms to the most recent input pixel of B. Indeed, the reading position rp is its right-hand side end, which conforms to the intuitive necessity to have read all the samples covered by the SE before computing the dilation.

Alg. 1 is to be called from an outer loop iterating over the writing position in $\delta_B f$, such as *while wp<N*. The writing position wp is to be incremented whenever Alg. 1 outputs a valid value. We give below the pseudocode, for detailed description see [2].

Algorithm 1: dF ← 1D_DIL (rp, wp, F, SE1, SE2, N)

Input: F - input signal sample $f(rp)$; rp, wp - reading/writing position; *SE1*, *SE2* - SE size towards left and right; N - length of the signal
Result: dF - dilated signal sample $\delta_B f(wp)$
Data: Q - Queue (first in, first out)

```
while Q.back()[1] ≤ F do
    Q.dequeue() ;                          // Dequeue useless valuess
Q.push({F, rp}) ;                          // Enqueue the current sample
if wp − SE1 > Q.front()[2] then
    Q.pop() ;                              // Delete too old value
if rp = min (N, wp + SE2) then
    return (Q.front()[1] ) ;               // Return valid value
else
    return ({}) ;                          // Return empty
```

4.2 Stream-Preserving Decomposition of Polygons

The Alg. 1 can be used to compute the dilation by L_{α_i} segments in a stream. Its properties make it suitable for composing concatenated operators, namely the sequential access to input and output data, and minimal latency. Therefore, when the input image is read in a horizontal raster scan mode, i.e., line by line, and every line from the left to the right, the output of Alg. 1 instance conforms to the very same scan order, delayed by some latency defined by the distance between reading rp and writing wp positions. It allows a direct connection of several Alg. 1 instances in a sequence without any need of coupling elements. The resulting 2-D SE is then obtained with minimal latency, that is as soon as all necessary data have been read.

The example of decomposition of a hexagon into three L_{α_i} is depicted in Fig. 4. The image is sequentially read by horizontal L_{0° at the reading position of the polygon (a). The result of the horizontal segment is immediately provided as an input to the first oblique L_{60° at (b) so that the reading position of L_{60° coincides with the writing position of L_{0°. By the very same rule, the result of the L_{60° is brought as input data to the second oblique L_{120° at (c), the writing position of which is the writing position of the complete polygon (d). The total latency is then defined by distance between the reading (a) and the writing position (d) of the polygon.

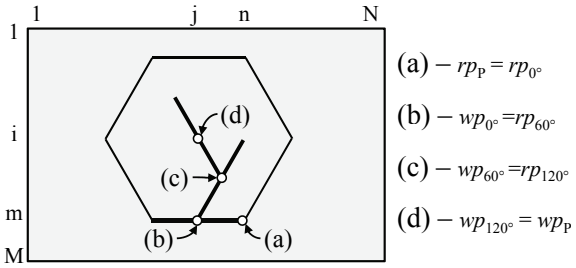


Fig. 4 Stream concatenation of three L_{α_i} into hexagonal SE P; rp/wp - reading/writing position.

The computational complexity of (9) remains almost constant w.r.t. the SE size (except the padding)

$$\mathcal{O}((N + 2B_{\mathrm{H}})(M + 2B_{\mathrm{V}})) \tag{16}$$

for an $N \times M$ image, and B_{H}, B_{V} padding sizes. Provided that of $B_{\mathrm{H}} \ll N$ and $B_{\mathrm{V}} \ll M$, it reaches the complexity of rectangles $\mathcal{O}(NM)$, see [2].

4.3 Discrete Inclined 1-D Segments

The oblique segments included in a hexagon and an octagon, i.e., L_{α_i}, $\alpha_i = 45, 60, 120, 135^\circ$, need appropriate addressing to determine the pixels to process. Note that all inclinations verify $\alpha_i \geq 45^\circ$, and the coefficients k_i verify $k_i = \tan\alpha_i \geq 1$. If we use the 8-connectivity for $k_{45^\circ,135^\circ} = \pm 1$, and the 6-connectivity for $k_{60^\circ,120^\circ} = \pm 2$, we can very easily generate the pixel adressing - for every inclination - by only modifying the original column index col by an additive constant $line/k_i$ such as

$$col_{\mathrm{shift}} = (col + line/k_i) \bmod (N + 2B_{\mathrm{H}}), \tag{17}$$

where the inclination deviation from the vertical direction $line/k_i$ is called *offset* and changes only with a new image line.

5 Hardware Implementation

In this section, we present the hardware implementation (called line unit LU) of Alg. 1 for the dilation by L_{α_i} with emphasize on the inclined segments computation. Then, we chain several elementary LU units into a pipeline to form the polygonal processing unit PU. Finally we propose the parallel polygonal unit PPU.

5.1 1-D Algorithm Implementation

Alg. 1 along with the col_{shift} addressing feature is seen as a simple Mealy finite state machine (FSM). This FSM controls all algorithm operations over the queue, rp and wp pointers etc. The state diagram (in Fig. 5) of the algorithm behavior consists especially of 2 main states $\{S1,S2\}$ and one auxiliary state EOL. The basic operation of the direct algorithm implementation can be found in [3,4]. The $S1$ state manages the dequeuing loop and pushing of a new value, code lines 1–3; the $S2$ state handles the deletion of outdated values and returns the result, code lines 4–9.

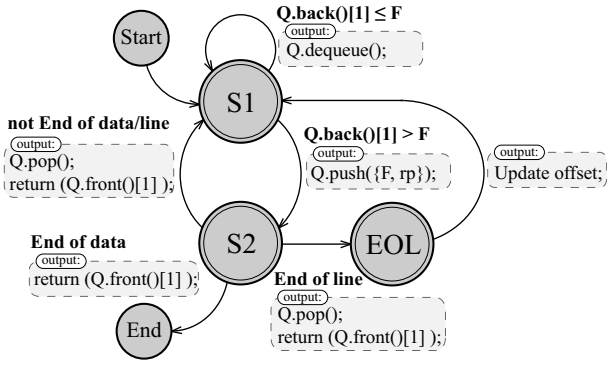


Fig. 5 State diagram of Alg. 1. Conditions of state transitions are typed in bold, output actions are located in gray rectangles.

The auxiliary state EOL is entered only at the end of every image line. Its main purpose is to update the offset value, to determine the shifted column addressing. The generation of the necessary inclination is extremely easy since requires only elementary operations like incrementing, decrementing or stalling.

5.2 1-D Line Unit Architecture

The architecture of the LU unit capable of dilation by different line segments is shown in Fig. 6. The basic description of the preceding version supporting only horizontal and vertical orientation can be found in [4]. Several modifications have been applied to the former version to allow inclined L_{α_i}. We have mainly added the Slope control unit that is highlighted in Fig. 6.

The LU comprises two parts: an FSM part and a memory part containing a collection of double-ended queues. The FSM manages the whole computing procedure and temporarily stores values in the memory part. The memory instantiates one queue in the case of horizontal segment,

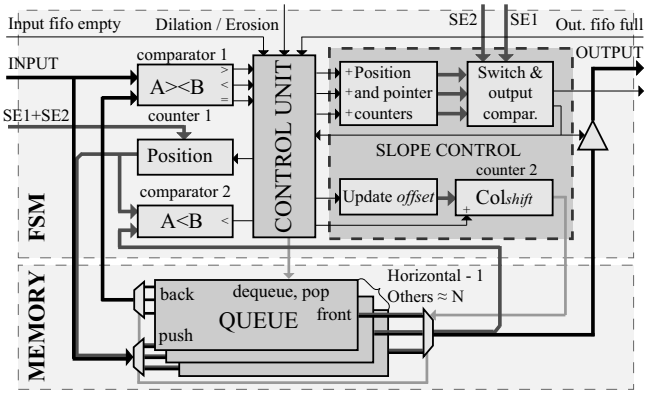


Fig. 6 Overview of the LU architecture. The FSM part manages computation, the memory part contains data storage–queues.

N queues in the vertical case (N is the image width), or $N + 2B_{\mathrm{H}}$ queues in the oblique case. Input and output ports are multiplexed; hence a multiplexor select signal can easily address one queue to work with. The shifted column address (col_{shift}) is used as the select signal.

The processing of one pixel proceeds as follows: In the beginning of $S1$, the last queued pixel is invoked by the Back() operation from the queue and fetched to Comparator 1 where it is compared with the current sample, and dequeued if necessary (code lines 1–2). The current pixel is simply extended with the value of position counter 1 and enqueued (line 3).

The $S2$ invokes the oldest queued pair $\{F, rp\}$ by the Front() operation. This read pixel is output as a correct result if the set of output conditions (code line 6) is fulfilled. The deleting of an outdated value is managed by comparing the stored position value with the current one in Comparator 2.

The purpose of the Slope control is to select the corresponding queue memory which is currently used by Alg. 1. The queues are addressed by the *Col*$_{\mathrm{shift}}$ counter, which is incremented with every pixel of the input image and reset at the end of image line. The initial reset value of the *col*$_{\mathrm{shift}}$ counter is *offset* (see Section 4.3). The *offset* is updated at the end of every image line (state EOL); its value is incremented or decremented either every line or every other line according to k_i.

5.3 Polygon Unit Architecture

The previously described LU units can be arranged in a sequence to form a 2-D Polygon Unit (PU). The overall architecture of the PU unit (see Fig. 7) is composed of three different-purpose parts: computation part, controller, and padding part.

The computation part mainly contains four LUs for distinct L_{α_i} orientations. There are the horizontal unit ($\alpha_1 = 0°$), the first inclined unit ($\alpha_2 = 45°$ or $60°$), the second inclined unit ($\alpha_3 = 135°$ or $120°$), and the vertical unit ($\alpha_4 = 90°$) connected in a simple pipeline; the output of each unit is read by the successive unit which processes the image by further L_{α_i}. The computation part is able to operate either with a hexagon or octagon SE. In the case of the hexagon SE, the vertical unit is bypassed.

Note that the output of every computation unit is an intermediate result image, which can be brought out for another purpose, e.g., a multi-scale analysis descriptor. Then the dilation by line, rectangle, and octagon SEs (all centered) can be obtained during a single image scan (considering units reordering). Only the Remove padding block is to be copied several times for each output data stream.

According to the boundary effects mentioned in Section 4, the inclined units need padding to extend the original image before the processing. The padding is removed after the last 1-D unit. It is carried out by a pair of dual padding blocks at the beginning and the end of the computation part.

The controller ensures the correct global system behavior. It accepts the SE diameter and the shape select signal, then it determines the particular SE sizes for every LU and padding from them, and initiates the computation. The entire set of parameters, i.e., the image width and height, SE features, and the morphological function select, is run-time programmable at the beginning of the frame.

To enable processing a uniform input stream, one needs to handle unequal processing rates of LUs. It is caused by variable algorithm latency to compute a dilation for one pixel. Therefore, the balancing FIFO memories are inserted in front of each 1-D unit, and to the input and output ports. The depth of input and output FIFOs depends on the timing of input data stream (possibility of stalling, synchronization, etc.).

5.4 Memory Requirements

The most significant memory demand is made by the set of queues. Although the algorithm works with separated queues, the queues within each LU are merged into a single dual-port memory, mapped side by side in a linear memory space. Every queue has a related pair of front and back pointers which must be retained throughout the entire computation process in the pointer memory. This approach leads to more efficient implementation.

The LUs have the following memory requirements (considering N×M image including padding, L_{α_i} with bounding boxes $W_x \times H_x$, and bpp bits per pixel):

$$M_{\mathrm{hor}} = W_{\mathrm{H}}(bpp + \lceil \log_2(W_{\mathrm{H}} - 1) \rceil) \quad \text{[bits]} \tag{18}$$

$$M_{\mathrm{ver}} = N((H_{\mathrm{V}} - 1)(bpp + \lceil \log_2(H_{\mathrm{V}} - 1) \rceil) + 2\lceil \log_2(H_{\mathrm{V}} - 1) \rceil) \quad \text{[bits]} \tag{19}$$

$$M_{\mathrm{slope}} = (N + W_{\mathrm{S}})((H_{\mathrm{S}} - 1)(bpp + \lceil \log_2(H_{\mathrm{S}} - 1) \rceil) + 2\lceil \log_2(H_{\mathrm{S}} - 1) \rceil) \quad \text{[bits]} \tag{20}$$

Example: Consider a dilation of 8-bit, SVGA image (i.e., 800×600=$N \times M$) by a hexagon with radius 41 px. Such a SE is decomposed into horizontal SE 21 px wide, and 2 slope

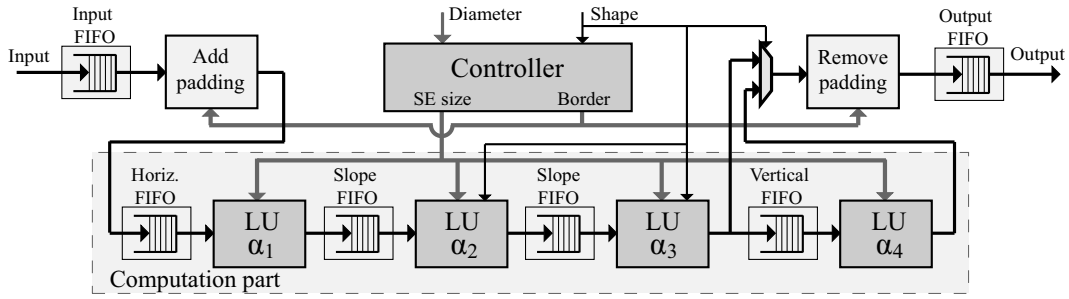


Fig. 7 Overall architecture of the polygonal PU unit. It contains one LU for each $\delta^{\mathrm{L}}_{\alpha_i}$ of (9), control, and padding units.

SE each 11 px wide and 19 px tall (hexagon SE bounding box is 41×37 px).

The computation memory (the queues) requires (18–20)

$$\begin{aligned} M_{\text{hor}} &= 21(8+5) = 273 \quad \text{[bits]} \\ M_{\text{slope}} &= (811+11)((19-1)(8+5)+10) = \\ &= 200'568 \quad \text{[bits]} \end{aligned}$$

resulting in a total consumption of $M_{\text{all}} = M_{\text{hor}} + 2M_{\text{slope}} \cong$ 392 kbits for the 2-D dilation by hexagon. This is far below the mere size of the image itself $M_{\text{image}} = 800 \times 600 \times 8bpp \cong 3.66$ Mbits which does not need to be stored at any moment.

6 Parallel Implementation

This section describes the Parallel Polygon Unit (PPU) that aims at increasing the computational performance while maintaining as much as possible the beneficial streaming properties of the proposed algorithm.

6.1 Partition of the Image

The parallelism is achieved throught utilization of concurrently working units that simultaneously process different parts of the image (spatial parallelism). The number of instantiated units defines the parallelism degree (*PD*). Since the processing runs in stream, we propose a solution that transforms the input stream into a set of *PD* streams in a way to minimize the waiting-for-data periods of all units. For the sake of clarity, we use *PD*=2 in the description hereafter. A similar method has proven to be useful in [2].

The partition of the input image is twofold, see Fig. 8: an interleaved line-by-line partition for the horizontal α_1 units, and vertical stripes for the vertical and inclined $\alpha_2, \alpha_3, \alpha_4$ units. The final image partition of 2-D image is the intersection of both.

Intuitively, the streams have to be transformed from one type to the other between α_1–α_2, and α_4–*output* in the PU. The transformation is done by simple circular stream switching when a partition edge is encountered. With the beginning of the image, it starts with the H1∘V1 segment at the

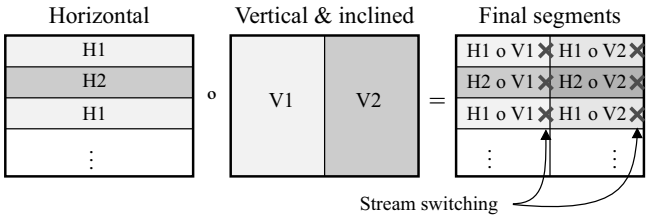


Fig. 8 Example of image partition for *PD*=2: line by line for horizontal orientation; vertical stripes for non-horizontal orientation.

first line. When the end of this segment is reached, the streams are switched such that segments H1∘V2 (1st line) and H2∘V1 (2nd line) are processed at the same time. Later, it proceeds to segments H2∘V2 (2nd line) and H1∘V1 (3rd line) and so forth. In general *PD* segments located on a backward diagonal run simultaneously throughout the image (note that the streams are mutually delayed by N/PD pixels).

Processing the partition segments separately introduces undesired border effects on each partition edge. A common solution – similar to padding at image borders – is to introduce an overlap. Contrarily to the padding that adds recessive values, the overlap extends a partition by a portion of the neighboring partition. The width of the overlap depends on the size of the SE, and is equal to the width of the horizontal padding B_{H}. Intuitively, the overlap introduces redundant computation, and slightly degrades the performance and minimal latency.

6.2 Parallel architecture

At this point, all the previously mentioned principles are brought together to form the Parallel Polygon Unit (PPU). The PPU (see Fig. 9) is scalable with respect to *PD*, the number of parallel streams it can process at the time. Each stream needs one pipeline of four LUs ($\alpha_i, i = 1..4$, just like the PU), two *add overlap* blocks in front of inclined LUs, two *remove overlap* blocks behind inclined LUs, *add padding* at the front end, and *remove padding* at the back end. The PPU also contains a pair of switches to transform the streams from one type to the other, and a controller (ommited from Fig. 9).

Figure 10 shows the introduction of the overlap in a course of the i-th image line. As we know, this line is split

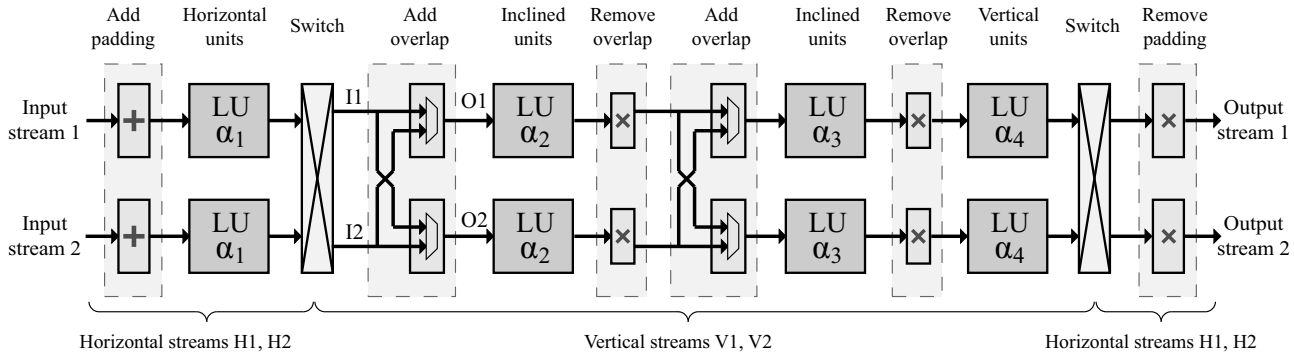


Fig. 9 Overall architecture of the parallel polygonal unit PPU for *PD*=2. The controller and balancing FIFOs are ommited.

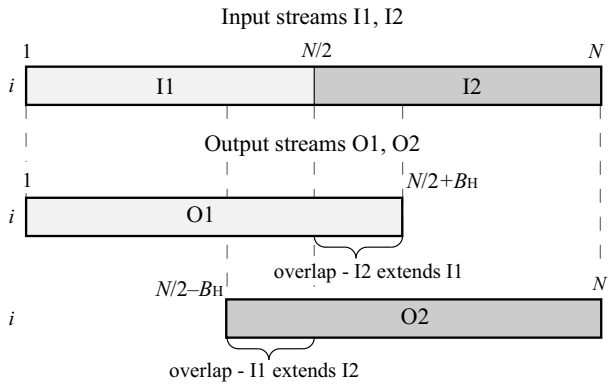


Fig. 10 Addition of overlap on one image line

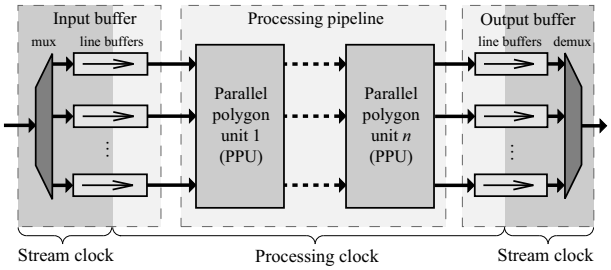


Fig. 11 Overall architecture of parallel ASF application.

into two streams. The streams are labelled I1, I2 before the addition and O1, O2 after (refer also to Fig. 9). The entire I1 stream plus B_{H} pixels of I2 form O1 output stream with overlap, and last B_{H} pixels of I1 and the whole I2 stream form O2.

During the overlap sections, either I1 or I2 stream is mapped to both output streams at the same time. This data duplication temporarily disallows parallel processing of both streams and may result in stalling of either stream. However, the effect of the overlap is negligible as long as $B_{\mathrm{H}} \ll N$.

Two important properties are to be noted: (i) input and output streams are mutually delayed by N/PD (ensured by stream switching); (ii) several PPUs can be chained into a pipe. The result schematic of some application, e.g., ASF, may look like in Fig. 11. At the front end there is an input buffer transforming the input stream (which is *PD*-times faster than each of *PD* processing streams) into *PD* processing streams Hi ($i = 1..PD$). The transformation only needs i-th image line to be stored in $\{i \bmod PD\}$-th line buffer. In this manner, the processing streams are properly delayed by N/PD pixels. The output buffer transforms *PD* processing streams into one fast stream in the opposite way. One can place as many PPUs as desired between these two buffers in a pipeline or other topology.

The PPU involves the following limitations on the programmability of parameters: the image size is set before synthesis, and padding sizes $B_{\mathrm{H}}, B_{\mathrm{V}}$ are computed for the maximal allowed SE, specified before the synthesis. The reason is that handling the varying SE and image sizes would introduce an unreasonable hardware overhead of image partition, padding, and overlap features. The SE remains fully programmable.

7 Experimental results

Hereafter we discuss the results of the proposed implementation. First, we discuss the results of a single 2-D PU and PPU unit, followed by their performance in application to an ASF filter. We conclude by comparison with other architectures.

The proposed PU and PPU architecture has been implemented in VHDL and targeted to the Xilinx FPGA Virtex-6 device (XC6VLX240T). The ultimate specification conforms to the following: 8-bit gray-scale images of size up to 1080p (1920×1080 px), height of L_{α_i} up to 31 px (thus, hexagon SE up to 61 px, and octagon SE up to 91 px), and support of uniform stream processing. Notice that all three previous factors affect the memory requirements that (in contrast to the PC), have significant influence on the clock frequency. Our specification implies the clock frequency of 100 MHz.

The timing with respect to (shortly w.r.t.) the image size and the size of the SE (Table 1 and 2) have been evaluated on a natural photo image. We report several measures.

1) The pixel rate gives the average number of clock ticks to process one pixel. It is given by the overall number of clock ticks divided by the image size. One can observe that

Table 1 Timing of PU and PPU w.r.t. image size (SE size = 51 px, *PD*=6).

Image Size	VGA	SVGA	XGA	1080p
Pixel Rate (PU) [clk/px]	2.61	2.53	2.53	2.44
Latency [image line]	25	25	25	25
FPS (PU) [frame/s]	125	82	50	19
FPS (PPU) [frame/s]	599	406	257	105
Speed-up PPU vs. PU [-]	4.79	4.94	5.1	5.34

Table 2 Timing of PU w.r.t. SE size (SVGA image)

SE size [px]	21	31	41	51	61
Rate [clk/px]	2.42	2.46	2.49	2.53	2.58
FPS [frame/s]	85	84	83	82	81
Latency [image line]	10	15	20	25	30

the rate is almost constant w.r.t. both the size of the image and the size of the SE. The slight variation is caused by the size of the SE which affects the size of the padding and overlap, increasing the number of effectively processed pixels, see (16).

2) The latency is expressed in a number of image lines. Observe that it is strictly a half of the SE size. This is a further irreducible factor corresponding to the dependency of the output on the input. This corresponds to the half-height of the SE that we need to wait to have read enough data to compute the dilation.

3) The last measure is the throughput in terms of the number of frames per second (FPS). The ultimate result we obtain is 105 fps for the 1080p resolution, allowing the 100Hz 1080 FullHD TV standard to be processed in real time.

The speed-up PPU vs PU measures the acceleration obtained from the parallelization. The difference from the ideal upper limit (*PD*=6) is due to the overlap. With increasing image size the acceleration converges towards 6 because the SE size (and consequently the overlap) becomes negligible with regard to the image size.

Table 3 outlines efficiency of the scalability (that is the parallelism degree *PD*) in terms of the FPS and speed-up. One can observe that the real speed-up is somewhat lower than the PD. The difference is due to two factors: (i) the overlap, which demands redundant computation, and (ii) the stream switching that needs inter-stream synchronization which may introduce wait cycles.

Table 4 reveals the cost of parallelization on FPGA resources in terms of registers, LUTs, and BRAMs of the PPU and the pair of input and output buffer as shown in Fig. 11.

7.1 Alternating Sequential Filter

The ASF filter is an essential method of morphological filtering, see example Fig. 12. According to (4) the λ-order ASF (referred to as ASF$^\lambda$) is composed of the sequence of λ closings and λ openings with the increasing SE, such as the

Table 3 Speed-up of PPU w.r.t. *PD* (SVGA image, SE size = 31 px).

Parallelism degree *PD*	2	3	4	5	6
FPS [frame/s]	162	234	306	376	441
Speed-up [-]	1.92	2.77	3.62	4.44	5.22

Table 4 FPGA resources w.r.t. *PD* (SVGA image, SE size = 91 px).

PD	1	2	3	4	5	6
Registers (P)PU	787	1,644	2,469	3,215	4,019	4,850
LUTs (P)PU	2,656	4,831	7,330	9,301	11,540	14,221
Block RAM (P)PU	39	39	59	42	53	63
Registers buf	0	251	355	466	590	671
LUTs buf	0	1,296	1,929	2,545	3,158	3,748

s^{th} stage ($s \in \mathbb{N}; s \leq \lambda$) has octagonal SE of width $2s+1$.

$$\begin{aligned}\text{ASF}^\lambda &= \gamma^\lambda\varphi^\lambda\gamma^{\lambda-1}\varphi^{\lambda-1}\dots\gamma^1\varphi^1 \\ &= \delta_{B_\lambda}\varepsilon_{B_\lambda}\varepsilon_{B_\lambda}\delta_{B_\lambda}\delta_{B_{\lambda-1}}\varepsilon_{B_{\lambda-1}}\dots\varepsilon_{B_1}\delta_{B_1}\end{aligned}$$

The initial number of morphological operators 4λ can be reduced using the associativity of dilations and erosions. Hence, every two consecutive dilations or erosions may be merged into one to obtain only $2\lambda+1$ operators, such as

$$\text{ASF}^\lambda = \delta_{B_\lambda}\varepsilon_{B_\lambda\oplus B_\lambda}\delta_{B_\lambda\oplus B_{\lambda-1}}\dots\varepsilon_{B_1\oplus B_1}\delta_{B_1}. \qquad (21)$$

Since the ASF is applied as a sequence of alternating dilations and erosions with a changing SE, it can be advantageously implemented by chaining instances of the proposed architecture into a pipeline structure. The output of each operator is immediately processed by a subsequent operator to achieve the following beneficial properties: (i) all the operators are being applied in parallel (temporal parallelism), (ii) the image is filtered with minimal latency inferred by the Minkowski addition of all SEs.

Table 6 illustrates the performance of ASF$^\lambda$ in terms of the experimentally achieved FPS and the inferred latency. Note that the frame rate of the whole ASF decreases with respect to the order λ since larger SE implies larger padding and overlap. However, the performance of the filters is comparable with the rate of a single unit in Table 1.

(a) Original (b) Filtered

Fig. 12 Example of ASF filtering. A zoom into original and the ASF^3 filtered "Mountain" image.

Table 5 Comparison of several FPGA and ASIC architectures concerning morphological dilation and erosion. N, M stand for the image width and height of respective architectures.

	Processing unit				Hardware System		Application Example ASF^6		
	Technology	Supported SE	Throughput [Mpx/s]	f_{max} [MHz]	Number of units	Supported image	Image scans	FPS [frame/s]	Latency [px]
Clienti [6]	FPGA	arbitrary 3×3	403	100	16	1024×1024	6	66.7	$5NM + 84N$
Chien [5]	ASIC	disc 5×5	190	200	1	720×480	45	12.2	$44NM + 84N$
Déforges [8]	FPGA	arbitrary convex	50	50	1	512×512	13	14.7	$12NM + 84N$
This paper	FPGA	regular polygon	195	100	13	1024×1024	1	185	$84N$

Table 6 Timing of ASF$^\lambda$; SVGA image size, *PD*=6.

Order of ASF λ	1	2	3	4	5	6
Number of δ, ε	3	5	7	9	11	13
Size of max. SE [px]	5	9	13	17	21	25
FPS by PUs [frame/s]	88	87	86	86	86	85
FPS by PPUs [frame/s]	491	483	466	440	415	387
Latency [image line]	4	12	24	40	60	84

7.2 Architecture Comparison

The implementation and performance comparison of our architecture with a few others is presented in Table 5. At first, we take into account single 2-D units only. Clienti [6] yields a high throughput for an elementary SE 3×3. The Chien [5] ASIC chip achieves a reasonable throughput with a small 5×5 diamond SE. Both architectures uses homothecy to obtain larger SEs. On the other hand, Déforges [8] directly offers large convex SEs. The maximal SE size and the programmability is not mentioned, namely, the possibility to control the SE shape after the synthesis is not clear.

All these solutions are efficient for small SE sizes or short concatenations. They become more or less penalized for longer concatenations, such as serial filters. As an application example, consider ASF$^6 = \delta_{13\times13}\varepsilon_{25\times25}\ldots\varepsilon_{5\times5}\delta_{3\times3}$ that consists of 13 morphology operations. One Clienti's system instantiates 16 elementary 3×3 processing units. Hence, it will require 6 image scans (the entire image must be stored in the memory). Chien also uses the homothecy, therefore, as much as 45 scans are to be done. In the case of Déforges, neither FPGA occupation nor possibility of using multiple instances in a single chip was communicated. Therefore, we consider one image scan for each operator.

Obviously, between two consecutive scans the data are read/written from/into the memory that degrades performance and significantly increases latency to orders of several image scans. The dense memory traffic might also jam the data bus. Our architecture processes an image in a single image scan with minimal latency (84 image lines for ASF^6) and memory requirements. These features allows a temporal-parallel execution of all atomic operators that is essential for achieving the real-time performance for high demanding applications. In addition, the low memory requirements facilitate embedding a several instances of proposed computation unit in a single FPGA circuit.

8 Conclusions

It is widely known that processing data in stream allows reducing the latency, the memory consumption and increases the system throughput. Until recently, computing morphological dilations or erosions in stream was only possible for small, limited neighborhoods [5,6,8], or large rectangles [2]. Dilations by large polygons were computed iteratively, by using the homothecy. This required an external memory for intermediate data, limited the flexibility, and drastically increased the system latency.

This paper opens the possibility of stream execution to morphological dilation with large polygons. Although the decomposition of polygons into the Minkowski addition of inclined lines is known since years [1], we bring several propositions that—combined together—allow the execution in stream.

We show how to implement the dilation by inclined linear segments with a sequential access to input and output data.We show how to handle the border effects, and recall (since this is less known) that it requires large padding. Further, we show how to partition an image to introduce efficient spatial parallelism while maintaining the sequential access to data at all levels. This avoids increasing the system clock by dividing a fast data stream into several slower streams to process at a slower rate. We show how to efficiently handle the border effects on the partition.

The proposed polygon decomposition uses the sequential access to both input and output data. This allows for the temporal parallelism, where in concatenations like $\ldots\delta\varepsilon\delta\ldots$ all these operators run simultaneously on the time-delayed data. We attain a very low (nearly optimal) latency, which has beneficial impacts on the memory consumption. No external memory is used even for large SEs and large images. All these aspects brought together allow a considerable data throughput for sequential morphological filters. We have implemented a programmable IP block, usable in industrial systems running under heavy timing constraints satisfying upto the 100Hz 1080p FullHD TV requirements.

References

1. R. Adams. Radial decomposition of discs and spheres. *CVGIP Graphical models and image processing*, 55(5):325–332, 1993.
2. J. Bartovský, P. Dokládal, E. Dokládalová, and V. Georgiev. Parallel implementation of sequential morphological filters. *Journal of*

Real-Time Image Processing, pages 1–13. 10.1007/s11554-011-0226-5.
3. J. Bartovský, P. Dokládal, E. Dokládalová, and V. Georgiev. Stream implementation of serial morphological filters with approximated polygons. In *17th IEEE ICECS*, pages 706–709, Dec. 2010.
4. J. Bartovský, E. Dokládalová, P Dokládal, and V. Georgiev. Pipeline architecture for compound morphological operators. In *IEEE ICIP'10*, pages 3765–3768, Sept. 2010.
5. S.-Y. Chien, S.-Y. Ma, and L.-G. Chen. Partial-result-reuse architecture and its design technique for morphological operations with flat structuring elements. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(9):1156 – 1169, Sept. 2005.
6. Ch. Clienti, S. Beucher, and M. Bilodeau. A system on chip dedicated to pipeline neighborhood processing for mathematical morphology. In *EUSIPCO 2008*, Lausanne, Aug. 2008.
7. C. Coster and J.-L. Chermant. Image analysis and mathematical morphology for civil engineering materials. *Cement and Concrete Composites*, 23(2-3):133 – 151, 2001.
8. O. Déforges, N. Normand, and M. Babel. Fast recursive grayscale morphology operators: from the algorithm to the pipeline architecture. *Journal of Real-Time Image Processing*, pages 1–10, 2010.
9. P. Dokládal and E. Dokládalová. Computationally efficient, one-pass algorithm for morphological filters. *Journal of Visual Communication and Image Representation*, 22(5):411–420, 2011.
10. J. Gil and M. Werman. Computing 2-D min, median, and max filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):504–507, 1993.
11. T. Ikenaga and T. Ogura. Real-time morphology processing using highly parallel 2-D cellular automata CAM2. *Image Processing, IEEE Transactions on*, 9(12):2018 – 2026, Dec. 2000.
12. D. Lemire. Streaming maximum-minimum filter using no more than three comparisons per element. *CoRR*, abs/cs/0610046, 2006.
13. F. Lemonnier and J. Klein. Fast dilation by large 1D structuring elements. In *Proc. Int. Workshop Nonlinear Signal and Img. Proc.*, pages 479–482, Greece, Jun. 1995.
14. G. Matheron. *Random sets and integral geometry*. Wiley New York, 1974.
15. L. Najman and H. Talbot, editors. *Mathematical Morphology: From Theory to Applications*. ISTE Ltd and John Wiley & Sons Inc, 2010.
16. J. Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, New York, 1982.
17. J. Serra. *Image Analysis and Mathematical Morphology*, volume 2. Academic Press, New York, 1988.
18. J. Serra and L. Vincent. An overview of morphological filtering. *Circuits Syst. Signal Process.*, 11(1):47–108, 1992.
19. P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
20. P. Soille, E. J. Breen, and R. Jones. Recursive implementation of erosions and dilations along discrete lines at arbitrary angles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(5):562–567, 1996.
21. M. van Herk. A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. *Pattern Recogn. Lett.*, 13(7):517–521, 1992.
22. J. Velten and A. Kummert. Implementation of a high-performance hardware architecture for binary morphological image processing operations. In *MWSCAS '04*, volume 2, pages 25–28, 2004.
23. J. Xu. Decomposition of convex polygonal morphological structuring elements into neighborhood subsets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(2):153–162, 1991.

JRTIP manuscript No.
(will be inserted by the editor)

Pavel Karas · Vincent Morard · Jan Bartovský · Thierry Grandpierre · Eva Dokládalová · Petr Matula · Petr Dokládal

GPU Implementation of Linear Morphological Openings with Arbitrary Angle

Received: date / Revised: date

Abstract Linear morphological openings and closings are important non-linear operators from mathematical morphology. In practical applications, many different orientations of digital line segments must typically be considered. In this paper, we (1) review efficient sequential as well as parallel algorithms for the computation of linear openings and closings, (2) compare the performance of CPU implementations of four state-of-the-art algorithms, (3) describe GPU implementations of two recent efficient algorithms allowing arbitrary orientation of the line segments, (4) propose, as the main contribution, an efficient and optimized GPU implementation of linear openings, and (5) compare the performance of all implementations on real images from various applications. From our experimental results, it turned out that the proposed GPU implementation is suitable for applications with large, industrial images, running under severe timing constraints.

Keywords morphology, opening, closing, linear, 1-D SE, parallel, efficient, algorithm, implementation, GPU

1 Introduction

Openings and closings are non-linear image operators of mathematical morphology [1]. They are at the basis of statistical measures called granulometries [2–5] and of a class of non-linear morphological filters called Alternate Sequential Filters (ASF) [6, 7].

In practical applications, the granulometries allow estimation of a priori unknown geometrical characteristics of objects in the image. For illustration, we can cite (1) medical imaging applications e.g., blood cell classification [8], (2) automated document analysis [9], or (3) industrial control [10]. The role of the ASF filters is to reduce the noise while preserving the principal features in the image. They represent the principal element of numerous applications e.g., texture analysis [11] or remote sensing [12]. Even the morphological openings themselves are useful for their filtering properties in some industrial applications such as [13].

Generally speaking, to obtain the desired result–size distribution or filtering effect–we have to use a sequence of openings and/or closings with varying parameters of the applied computing window, so-called structuring element (SE). For a given shape of the SE, these variable parameters are the progressively increasing size of SE and rotation angle. In order to ensure the exhaustivity of the result, applications often require computing of an enormous number of iterations with greater SE, often approaching hundreds of pixels. Considering continually increasing image resolution used in industrial applications, one can intuitively feel that it results in overwhelming requirements on the computing power. This is true even despite recent efficient algorithms [14, 15].

In this context, we study how to efficiently implement the above mentioned operators on graphics cards with the objective to reduce these computing requirements on the system. Initially, graphics cards were designed for graphics purposes only and were not programmable. Based on numerous parallel processors they were very powerful compared to their price. Current GPUs have passed the one Tera FLOPS barrier, and there is no need to use dedicated graphics languages any more since several frameworks have been developed for GPGPU[1] purposes: CUDA [16] by nVidia and OpenCL [17] by the Khronos Group are today

P. Karas
Centre for Biomedical Image Analysis, Faculty of Informatics, Masaryk University, Botanicka 68a, 60200 Brno, Czech Republic. E-mail: xkaras1@fi.muni.cz

V. Morard · P. Matula · P. Dokládal
Centre of Mathematical Morphology, Department Mathematics and Systems, Mines ParisTech, 35, rue St. Honoré, 77300 Fontainebleau Cedex, France. E-mail: {vincent.morard, petr.matula, petr.dokladal}@mines-paristech.fr

P. Matula
University of Heidelberg, BIOQUANT, IPMB, and German Cancer Research Center, Dept. Bioinformatics and Functional Genomics.

J. Bartovský · T. Grandpierre · E. Dokládalová
Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI, Université Paris-Est, ESIEE Paris, 93162 Noisy-le-Grand Cedex, France. E-mail: {j.bartovsky, t.grandpierre, e.dokladalova}@esiee.fr

[1] General-purpose computing on graphics processing units

the most popular in the GPGPU community. Both are based on C language extensions. Notice that in this paper, we use the CUDA language for all presented implementations on GPU. Nevertheless, the principles remain the same when moving to a different language.

In our study, we focus on the morphological openings using linear SE under arbitrary angle, essential in a wide range of practical applications dealing with linear image structures such as fingerprint analysis, geosciences [18], and vessel segmentation [19].

The main novelty of this paper consists of an efficient and optimized GPU implementation of the algorithm by Bartovsky et al. [15], which turned out to be the most suitable for a parallel implementation on GPU. We also present performance comparisons and experimental results of all implementations on real data. It turned out that the proposed GPU implementation can compute linear openings for 180 directions of an image of size 640×640 pixels within 60 ms.We also show that the proposed implementation is significantly faster than the state-of-the-art implementation in the OpenCV_GPU library [20], especially for larger SEs.

In the remainder of this paper we start by the introduction of the mathematical background in Section 2. Section 3 reviews and compares the state of the art of efficient implementations for computing linear morphological openings of different orientations. Section 4 presents the first GPU implementation of two candidate algorithms by Bartovsky [15] and Morard [14]. Afterwards, the Section 5 describes the optimized implementation of Bartovsky algorithm, including the parallelism enhancement discussion. Section 6 illustrates the use of this efficient implementation in practical applications. It includes also the discussion of overall experimental results. Finally, the conclusions recall the main contributions of our work and briefly introduces its perspectives.

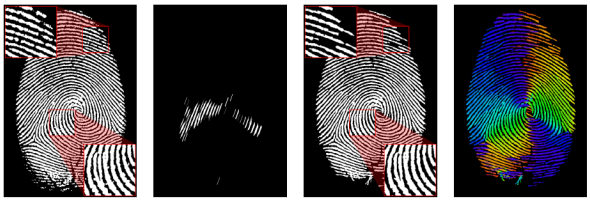
2 Basic Notions

Before defining morphological openings and closings, we define erosions and dilations for gray-scale images. Let a mapping f: $\mathbb{Z}^2 \to \mathbb{R}$ be a gray-scale discrete image, and $\mathscr{F}$ be the collection of all such images. The support used throughout this paper shall be a rectangular subset domain of $\mathbb{Z}^2$.

The erosion and dilation are mappings $\varepsilon_B, \delta_B : \mathscr{F} \to \mathscr{F}$, parameterized by the so-called structuring element (SE) B, $B \subset \mathbb{Z}^2$. Here, we restrict the family of SE to flat and translation-invariant. Hence, the dilation and erosion are, as usually, defined by

$$\delta_B(f) = \bigvee_{b \in B} f_b \,; \qquad \varepsilon_B(f) = \bigwedge_{b \in \widehat{B}} f_b \tag{1}$$

where the hat $\widehat{\ }$ denotes the transposition of the structuring element, i.e., $\widehat{B} = \{x \mid -x \in B\}$, and f_b denotes the translation of the function f by some vector b, and $\bigvee$ and $\bigwedge$ denote the supremum and infimum on a collection of functions.



(a) Input (b) γ_λ^{70} (c) $\vee\gamma_\lambda$ (d) ζ_λ

Fig. 1 Example application of linear openings. **(a)** input image, **(b)** linear opening with a digital line segment of length $\lambda = 41$ pixels and orientation 70 degrees, **(c)** enhancement of linear structures, and **(d)** color coded local orientation of linear structures calculated using definitions in Eq. (3). In **(a)** and **(c)**, two zoomed regions are shown for a better comparison.

Similarly, the morphological openings and closings are mappings γ_B, φ_B: $\mathscr{F} \to \mathscr{F}$, also parameterized by B. The transposition $\widehat{B}$ used in the definition of erosion ensures that the dilation and erosion form a so-called adjunction pair. This allows us to obtain the morphological opening and closing by concatenation of the dilation and erosion:

$$\gamma_B(f) = \delta_B[\varepsilon_B(f)] \,; \qquad \varphi_B(f) = \varepsilon_B[\delta_B(f)]. \tag{2}$$

Erosions and dilations are dual under complementation, i.e. for functions $\delta_B = -\varepsilon_B(-f)$, see e.g. [21]. Consequently, γ_B and φ_B are also dual and all algorithms developed for openings can easily be used also for closings.

In the sequel, we focus on linear morphological openings and closings obtained with SE in a form of a 1-D digital line segment of the length λ and orientation α, denoted by γ_λ^α and φ_λ^α, respectively.

Further operators can be built based on linear openings. The first operator, $\vee\gamma_\lambda(f)$, is computed by taking the supremum of the openings by digital line segments in all orientations; the second operator, $\zeta_\lambda(f)$, can extract the local orientation of linear structures:

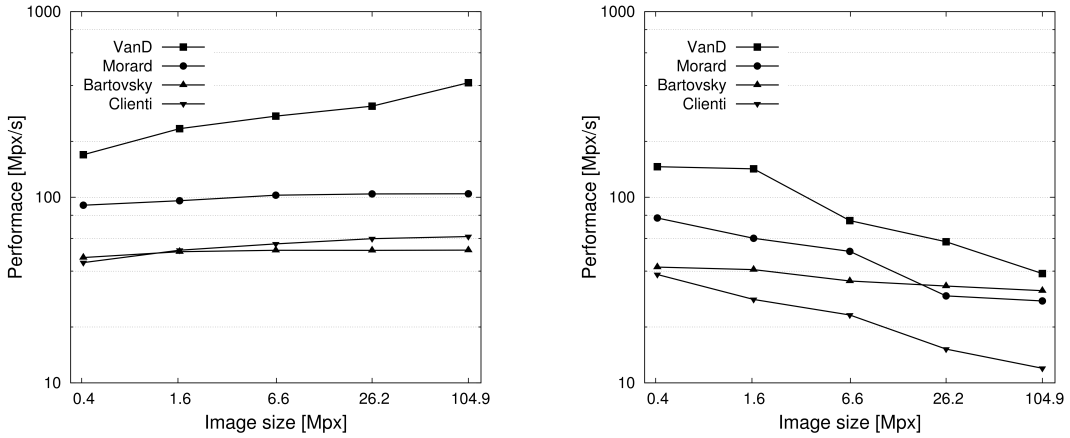
$$\vee\gamma_\lambda(f) = \bigvee_{\alpha \in [0,180]} \gamma_\lambda^\alpha(f) \,; \qquad \zeta_\lambda(f) = \arg \bigvee_{\alpha \in [0,180]} \gamma_\lambda^\alpha(f). \tag{3}$$

As an example, the effect of these operators on a real image is presented in Fig. 1.

3 Opening Algorithms

Opening algorithms can be divided into three classes: erosion and dilation chaining (a two-stage algorithm), direct computation, and algorithms based on connected component tree building.

Two-stage algorithm: computed by using Eq. (2), it is the simplest and historically earlier approach used to compute openings. Noting N the number of pixels of the image



(a) Horizontal SE

(b) Vertical SE

Fig. 2 Comparison of CPU implementations for horizontal and vertical SEs of size approx. 5% of the image width.

and L the size of the SE, the complexity of the naïve approach is $O(N \times L)$. The computational complexity was improved throughout the years to make the algorithm feasible for practical applications. In 1985, Pecht [22] defined a logarithmic decomposition of the SE. This decomposition, which removes most of the redundancy, was further extended by Coltuc [23], reducing the complexity to $O(N \times \log L)$. Later, the complexity was further reduced to linear $O(N)$, hence independent of the size of the structuring element, by Van Herk, and Gil and Werman [24, 25]. The linear algorithm will be referred to as the *HGW algorithm* hereafter. In [26], Soille et al. extended this work to arbitrary-oriented openings. In [27], Clienti et al. improved the HGW algorithm by removing the image backward scanning to reduce latency.

Direct Computation: for the family of openings with 1-D SE, new algorithms were introduced recently to directly compute openings in only one scan of the entire image, preserving the complexity of $O(N)$. In [28], Van Droogenbroeck and Buckley introduced an algorithm based on anchors, allowing very fast computation of the linear openings. The anchors are points, which are not affected by the opening. Nevertheless, the algorithm uses a histogram. The main drawback of using a histogram is the increasing memory consumption for finely quantized data. Even though the memory is no longer an issue for PC architectures, it becomes a penalizing factor for parallelized implementations on other architectures with limited memory like GPU. Secondly, search over a long histogram becomes costly and finely, for floating point accuracy, the histogram can not be used at all.

Later, Morard et al. [14] introduced a very simple algorithm based on an ordered stack of *cords* where a cord refers to a continuous set of pixels of intensity greater than or equal to a certain value I. With the inclusion relation between cords and by analyzing the length of each cord, the computation of linear openings and of linear granulometries is straightforward. Finally, Bartovsky et al. [15] also developed an algorithm to build linear openings. It sequentially scans the signal and erases every peak narrower than SE. Further explanations on these algorithms are given in section 4.

Connected component tree: such approach was described in [29]. The approach is based on building connected components, hence it can be adopted for more complex tasks such as watershed segmentation [30]. The drawback of the algorithm is that the complexity depends on the number of gray levels in the image and requires random access data, consequently it is not adapted for applications running under strict time constraints.

3.1 Parallel Implementations

There are several implementations of HGW algorithm in the literature since it can be easily parallelized. Brambor [31] described a parallel implementation of the HGW algorithm on SIMD architectures. Their implementation was tested on an Intel CPU with the SIMD-SSE2 instruction set. Clienti [27] improved the HGW algorithm and implemented[2] it on a SIMD architecture as well. Domanski et al. [33] used CUDA to implement the HGW algorithm on GPU, achieving 13–33 $\times$ speedup. There are several drawbacks of the algorithm as it computes openings and closings by dilation-erosion chaining, which requires more computations than direct approaches. It also has larger memory requirements [34].

On the contrary, there are few parallel implementations of the component tree algorithms, among which we can cite Wilkinson [35], Menotti-Gomes [36] on multicore, and also Matas [30] on ccNUMA 4-core. They are effectively

[2] available in Fulguro image processing library [32]

so complex that it is difficult to exhibit some parallelism in these complex algorithms.

Finally, in order to improve the computing efficiency by parallel implementation, direct computation algorithms seem to be the best candidates compared to HGW and component-tree algorithms. This is why we focus on this class of algorithms in the remainder of this paper.

3.2 Selection of a Direct Linear Opening Algorithm

We need to select the best sequential algorithm candidate for a parallel implementation leading to the most efficient execution on a GPU platform. As explained above, such an algorithm has to allow arbitrary angles computing using direct linear opening for lower computation and complexity requirements. Hence, there are three algorithms available to benchmark and compare: (1) Van Droogenbroeck [28] referred to as *VanD*, (2) Morard et al. [14] referred to as *Morard* and (3) Bartovsky et al. [15] referred to as *Bartovsky*. To this list, we will add Clienti algorithm [27] although it is not a direct computation based algorithm. Effectively, this is one of the fastest HGW implementation that can consequently give a useful comparative point. It will be referred to as *Clienti*.

For a fair comparison, all these algorithms were implemented using the same image-processing library with the same interface and with all the optimization flags turned on. We used only one core of an Intel Core i7-870 2.93 GHz CPU for this benchmark. These algorithms have been applied on the texture images shown in Fig. 12(a), (b).

Execution performances of the four algorithms for both horizontal and vertical openings are presented in Fig. 2. The performance P is computed as $P = N/t$, where N is the image size and t is the computation time. This measure is consequently independent of the image size. Note, however, that the performance actually does depend on the image size (see e.g. Fig. 2(b)). Each marked value in the plots represents the performance computed from the mean computation of 100 openings. For each image the length of a structuring element was chosen to be equal to 5 % of the image width/height because, in most applications, the length of SE is considerably smaller than the image size.

From the benchmarks we see that for vertical SEs and for large images, the performances significantly decreases for all algorithms, except for the Bartovsky. This is explained by the fact that this algorithm accesses the data sequentially even for vertical structuring elements. It is clear that VanD algorithm is the fastest, followed by Morard, Bartovsky, and Clienti, for both vertical and horizontal orientation. While VanD achieves the best performance, it is nevertheless unsuitable for parallelization on a GPU because of its high memory requirements especially for higher bit depths and arbitrary oriented SEs. In contrast, Morard and Bartovsky algorithms are able to compute openings and closings efficiently regardless the orientation of the SE or the data type precision with minimum memory requirements. Therefore, we selected these two algorithms for parallelization and implementation onto the targeted GPU architecture.

Algorithm 1: Morard algorithm: $G \leftarrow \text{Open1D}(F, \lambda, \text{S})$

```
Input: F – input 1-D signal, λ – size of SE, S – pointer to
       LIFO stack
Result: G – output 1-D signal
Data: S – a stack of triplets (value, position, flag)
initialize S;
for rp ← 0 to |F| − 1 do
    if S.empty() or F[rp] > S.top(value) then
        S.push({F[rp], rp, false});
    else
        while F[rp] < S.top(value) do
            fz ← S.pop();
            if fz(passed) or rp − fz(position) ≥ λ then
                WriteFlatZones(F, G, rp, S, fz);
            process S;
process all zones remaining in S;
```

4 Basic Implementation on GPU

4.1 Morard and Bartovsky Algorithms

In this section, we briefly introduce the Morard and Bartovsky algorithms for computation of morphological openings and closings with a linear SE. The detailed description can be found in [14,15]. Both algorithms are able to compute the operation in O(N) time with respect to the image size and O(1) with respect to the size of SE. The Bartovsky algorithm was originally designed for streaming architectures such as FPGA and hence performs scanning of the input data in sequential order. Nevertheless, it can be simply modified to perform scanning along lines according to the orientation of SE, much like the Morard algorithm. During the scan, intensity and position of each pixel is stored in an auxiliary data structure if necessary. Whereas the Morard algorithm uses the LIFO stack, the Bartovsky algorithm uses the FIFO queue. For arbitrary directions, both algorithms use the Bresenham's lines, as described in [26].

In the Morard algorithm, the image line is scanned for pixels where the intensity changes. Whenever the current pixel intensity is higher than the preceding, the current pixel is pushed to the stack. In the opposite case, the stack is being emptied while necessary. The algorithm needs to store an extra bit for a boolean flag indicating the status of a pixel. The size of the stack is limited only by the size and the bit depth of the image. After processing the stack, the output is written. The outputs are irregular. A pseudo-code is shown in Alg. 1.

In the Bartovsky algorithm, the image line is scanned for so-called peaks. According to a peak configuration, either a pixel is pushed to the queue or the queue is being emptied. The size of the queue is limited by the size of SE. After

Algorithm 2: Bartovsky algorithm: $G \leftarrow \text{Open1D}(F, \lambda, \text{Q})$

Input: F – input 1-D signal, λ – size of SE, Q – pointer to FIFO queue
Result: G – output 1-D signal
Data: Q – a double-end queue of pairs (*value*, *position*)
initialize Q;
for $rp \leftarrow 0$ **to** $|F|-1$ **do**
  while $F[rp] \leq$ Q.back(*value*) **do**
    process Q;
  Q.push($\{F[rp], rp\}$);
  if $rp =$ Q.front(*position*) $+ \lambda$ **then**
    Q.pop();
  if $rp \geq \lambda$ **then**
    $G[rp-\lambda] \leftarrow$ Q.front(*value*);

processing the queue, the output is written. The writes are regular and follow the reads with the well-defined latency corresponding to the size of SE. A pseudo-code is shown in Alg. 2.

4.2 GPU Parallel Architecture

Before we describe basic GPU implementations of the algorithms, we briefly review some properties of the GPU architecture, as shown in Fig. 3.

From the hardware point of view, a GPU consists of hundreds of so-called *streaming multiprocessors (SM)*. Each includes a number of *shader processor (SP)* cores, a number of *registers*, a small *shared memory* providing communication between SPs, and fast *ALU units* for hardware acceleration of transcendental functions. One *global memory* is shared by all SMs and provides a capacity in order of GB and the memory bandwidth in order of 100 GB/s. There are also two additional read-only cached memory spaces accessible by all threads: the *constant* and *texture* memory spaces. They can help programmers to improve the performance of their implementations [37,38]. In some recent GPU architectures, such as FERMI by nVidia [39], the global memory is cached as well.

From the programmer's point of view, every program consists of two parts, a *host code* for CPU, and a *kernel code* for GPU. Before executing a kernel, the host program allocates a memory on GPU and transfers data if necessary. Then a kernel is configured and executed. The configuration defines the number of threads allocated for kernel execution. Generally, the number of threads should be much higher than number of processing units of GPU. This approach allows (1) the proper scaling on various hardware configurations, and (2) hiding the memory latencies. The threads form groups called *blocks* (as in CUDA [37]) or *work-groups* (as in OpenCL [40]), following the hierarchy of SMs and SPs. Synchronization of threads and data sharing is possible within the group only. The threads are executed concurrently in *warps*, usually of 32 threads each.

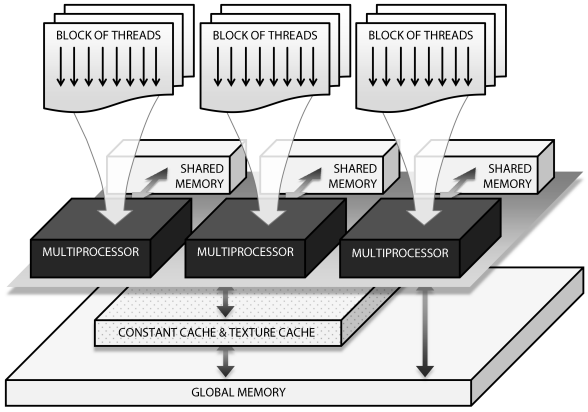


Fig. 3 Thread mapping and memory hierarchy in the GPU architecture.

4.3 GPU Implementation of Morard and Bartovsky Algorithms

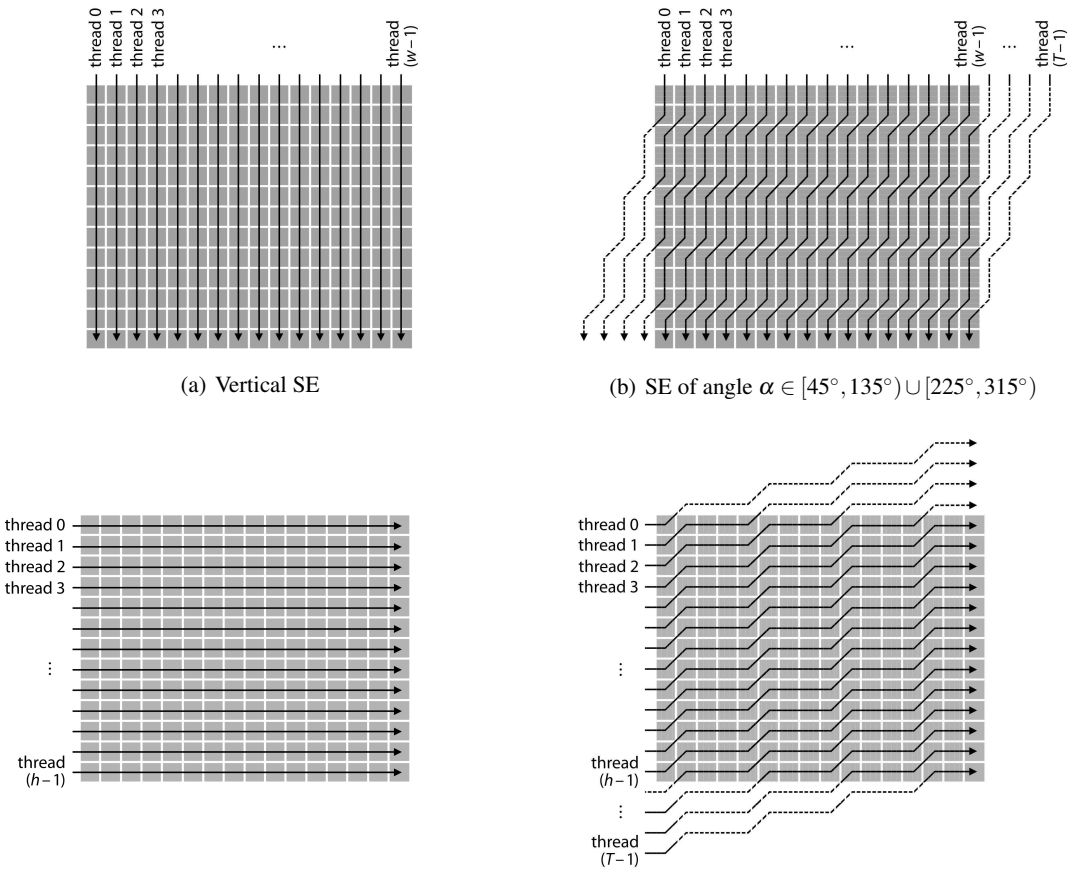
Regarding the design of both algorithms, the input image is scanned in the sequential manner. However, all lines of the image can be scanned concurrently, as shown in Fig. 4. Thus, the parallelism can be introduced by binding individual threads to individual lines of the image. Each image line is processed independently using its own algorithm-dependent stack or queue, respectively. This requires the GPU memory to be large enough to contain all auxiliary data structures. Here, the Bartovsky algorithm is favorable, since the sizes of the queues used are limited to the size of SE whereas the stacks used by Morard algorithm are generally limited by the size of the image.

The mapping of threads for all SE orientations is described in Fig. 4. For an arbitrary angle α, the overall number of threads can be simply computed as follows:

$$T = w + \lceil h|\cot\alpha|\rceil,\ \alpha \in [45^\circ, 135^\circ) \cup [225^\circ, 315^\circ), \quad (4a)$$
$$T = h + \lceil w|\tan\alpha|\rceil,\ \alpha \in [-45^\circ, 45^\circ) \cup [135^\circ, 225^\circ), \quad (4b)$$

where w and h are the width and the height of the input image, respectively. Thus, generally $T > w$ or $T > h$, respectively. Some of threads spend a part of the computation time outside the image domain where they do not compute anything. Thanks to thread locality, this brings little overhead only because in the GPU thread scheduler, thread warps that fall outside the image domain are quickly replaced by those that fall inside.

It should be noted that the processing of both the stack and the queue is data-dependent, thus data accesses are irregular. Therefore, during this part of the computation, threads are divergent. This limits the overall performance of the parallel implementation. In the Bartovsky algorithm, the data accesses to both input and output images are regular.



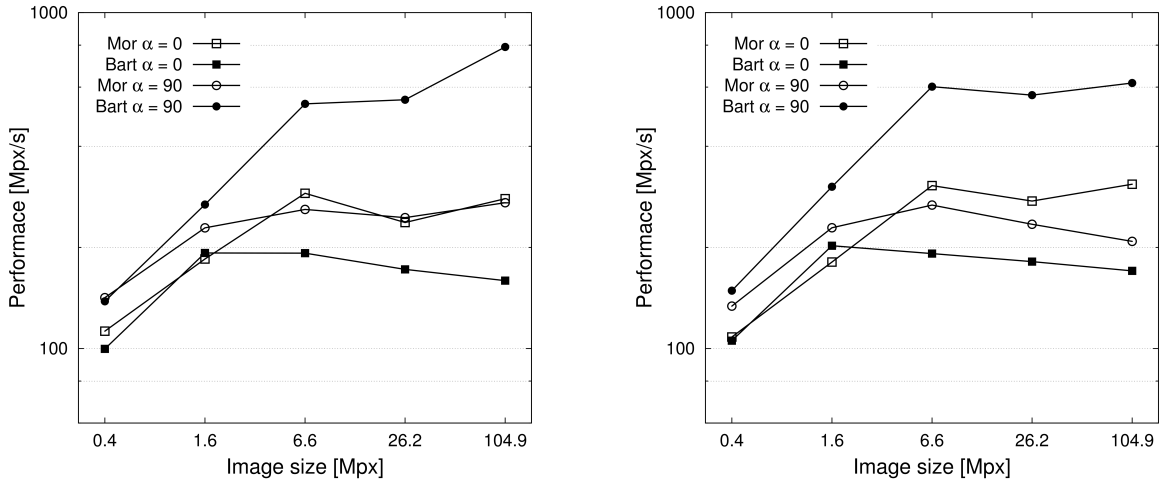
(a) Vertical SE

(b) SE of angle $\alpha \in [45°, 135°) \cup [225°, 315°)$

(c) Horizontal SE

(d) SE of angle $\alpha \in [-45°, 45°) \cup [135°, 225°)$

Fig. 4 The mapping of threads to the 2-D image grid in the GPU implementation of the algorithm for computing 1-D morphological openings/closings. Each thread, denoted by its ID, is mapped to an individual image column or row, respectively.



(a) Image D15

(b) Image D47

Fig. 5 Comparison of basic GPU implementations of Morard and Bartovsky algorithms. Opening was computed with SE of both horizontal and vertical direction and size approx. 5 % of image width.

4.4 Experimental Results

The basic GPU implementations, described in the previous section, were compared on nVidia Tesla C2050 GPU with 14 MPs at 1.15 GHz and 3 GB RAM. Again, all experiments were made with the texture images shown in Fig. 12(a), (b). Results are shown in Fig. 5. The presented performance does not include the times needed for the data transfer between CPU and GPU.

Contrarily to the performance on CPU, the Bartovsky algorithm achieves better performance than the Morard algorithm. However, its performance is highly sensitive to the SE orientation, as shown in Fig. 5. For a horizontal SE, the performance is significantly lower due to extensive L1 cache misses, as detected by Nsight Visual Profiler [41]. As the L1 cache is stored in memory banks and memory accesses are synchronized, most of cache queries are directed to the same memory bank leaving other memory banks unused. This is not the case of the Morard algorithm since it generally keeps the memory accesses irregular.

5 Efficient Implementation of the Bartovsky Algorithm

The choice of the algorithm to optimize is guided by the analysis of the Bartovsky and Morard algorithms in sections 4.3 and 4.4. Despite higher performances of the Morard algorithm on CPU, the Bartovsky algorithm has several advantages:

(1) Both the data accesses to the input and output image are regular. This helps to make the thread execution synchronous and reduces the thread divergence.

(2) The maximum length of the FIFO queue is limited by the length of SE. This strongly limits the memory requirements. Recall that we are to bind one thread per image line (Fig. 4). For large images, with potentionally many threads, it is important to have a small memory footprint of each thread.

In the basic implementation on GPU (contrarily to CPU), the Bartovsky algorithm is sensitive to the SE orientation. Hence, we shall introduce several optimization steps not only to increase the overall performance but, particularly, to keep the performance stable for all orientations. These steps are described in the following sections.

To prove the choice of the Bartovsky algorithm, we applied the optimization steps also on the Morard algorithm and performed tests to compare both optimized GPU implementations. The results are shown in Section 6.

5.1 Parallelism Enhancement

In the basic implementation, the parallelism was introduced by mapping GPU threads to individual image rows or columns, creating the grid of h or w threads, respectively (where h and w are height and width of a 2-D input image, respectively). However, if an input image is not large enough, the GPU's MPs are not fully occupied. Therefore, we introduce more parallelism by splitting the image into two or more parts. In the following text, we refer to them as *zones*. As each pixel can be affected by $(2\lambda-2)$ neighboring pixels (where λ is the length of SE), the zones need to overlap by $2(\lambda-1)$ pixels. An example of the image split for a vertical SE is shown in Fig. 6.

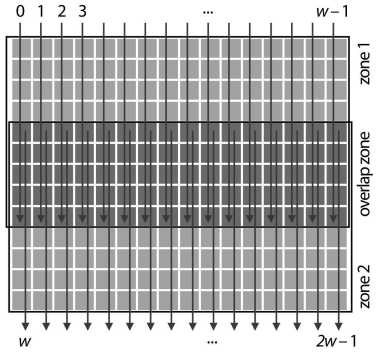


Fig. 6 Image split applied for opening/closing with vertical SE of size 2. Image is split into 2 zones, introducing twice more threads. The threads are denoted by vertical arrows, analogously to Fig. 4.

It is evident that the theoretical speed-up s that can be achieved depends on the SE length λ, the image size and the number of zones Z. For vertical SE we get the following:

$$s = Z\left[1 - \frac{(Z-1)(2\lambda-1)}{h}\right], \tag{5}$$

where h is the image height. For small λ we get $s \approx Z$, while for $\lambda \approx h/(2Z)$ we get $s \approx 1$. It should be noted that for large input images it is not necessary to introduce more parallelism by splitting the image, it can even decrease the performance. The optimal choice of the parameter Z is discussed in section 5.4.

5.2 Optimization of Data Accesses

To reduce the memory latency, the usage of the global memory should be avoided where possible. Alternative memory spaces can be used for both the input image and the FIFO queues. The input image can beneficially be stored in the read-only cached texture memory. This is true also for the recent FERMI architecture, which introduced L1 cache [39, 42], because the texture memory helps significantly improve the performance for horizontal SEs where L1 cache fails due to bank conflicts. The FIFOs can be stored in the shared memory. As the amount of the available shared memory is limited, the maximum length l_{max} of the FIFO is limited to $l_{max} = S/(S_\mathrm{b} \times d)$, where S is the amount of the shared memory available, S_b is the number of threads per block (workgroup) sharing the memory, and d is the size of the data type. Hence, the most recent values of the FIFO are stored in the shared memory in a circular buffer so the position of the first element varies during the computation. The rest of the FIFO

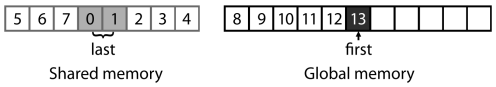


Fig. 7 FIFO storage in the shared and the global memory. The numbers indicate the order of pushing the element into the FIFO. The marked elements (the first one and the last two) are also cached in registers.

is stored in the global memory. The FIFO storage is shown in Fig. 7.

The constraint on the FIFO length mentioned above can be considered as a hard limit. However, using as much shared memory as possible can lead to a performance decrease because in that case, the shared memory is allocated for one thread block per MP only. Since the MPs are capable of execution of more thread blocks, the optimal FIFO length needs to be chosen carefully, as discussed further.

To conclude our choice of memory spaces, using L1 and L2 caches for input data turned out to be inefficient for some SE orientations, as explained in Section 4.4. The texture memory is read-only, cached, and can be allocated up to the size of the device memory. Therefore, it is suitable for the input image. The size of the texture cache is relatively small (8 kB per multiprocessor) but sufficient, thanks to the spatial locality. The shared memory is rewritable, relatively small (up to 48 kB per MP), and fast. In terms of latency, it is comparable with registers, provided that block conflicts are avoided—which is guaranteed in our implementation. Thus, it is suitable for FIFO caching.

5.3 Consideration of Arbitrary Orientations

As noted in section 4.4, the performance of Bartovsky algorithm on GPU is sensitive to SE orientation. Thus, a careful attention should be paid to this issue when computing openings in arbitrary directions. By using the texture memory, the input image is cached and the latency of memory reads is reduced. The final step is to optimize the memory writes to the output image. Experiments proved that openings with SEs of orientation $\alpha \in (-45°, 45°)$ are computed faster when writing the output image transposed. The output image is subsequently corrected using the modified *transpose* kernel from [43].

5.4 Configuration of Performance Parameters

In the optimized GPU implementation, we have introduced several parameters that influence the performance: the block size (work-group size) S_b, the number of zones Z, and the optimal length l of the FIFO buffer allocated in the shared memory. They all depend on these properties: the image dimensions (w or h), the SE length λ, the number of the MPs N_MP available on the used device, the shared memory size per MP S, the warp size W and the number of blocks (work-groups) N_b that can be executed on a single MP. For optimal configuration, the following set of rules should be satisfied:

1. S_b should be multiple of W,
2. $Z \geq (S_\mathrm{b} \times N_\mathrm{b} \times N_\mathrm{MP})/T$ where T is defined in Eq. (4),
3. $Z \leq h/(2\lambda)$ or $Z \leq w/(2\lambda)$ following Eq. (5),
4. $l \leq S/(S_\mathrm{b} \times d \times N_\mathrm{b})$,
5. l should be a power of two allowing the bitwise operator "&" to be used instead of the costly modulo operator for addressing the FIFO queue items.

It should be noted that some of the rules cannot be satisfied in some cases. In particular, rules (2) and (3) can be conflicting for small input images. Performances for some parameter configurations are presented in the following section. CUDA programmers are advised to use a tool called "CUDA Occupancy Calculator" which can help to compute the optimal kernel configuration [43].

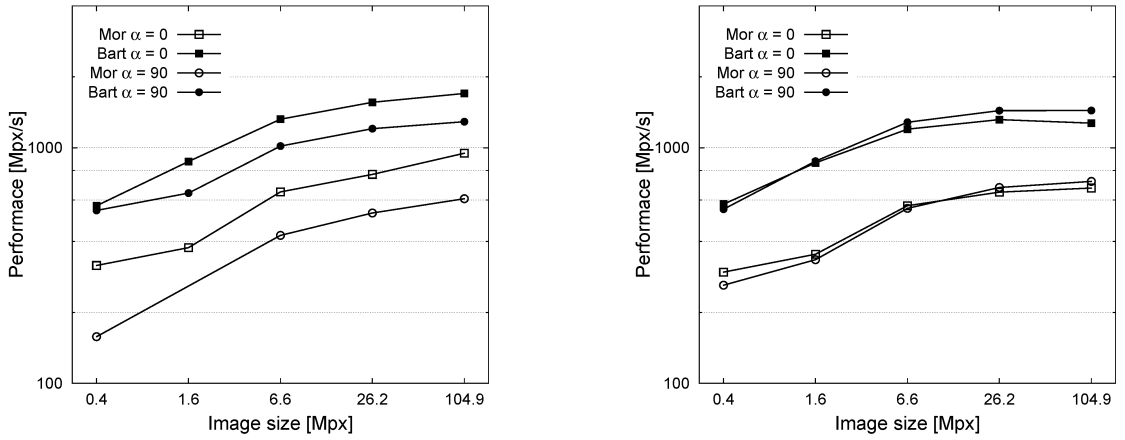
6 Experimental Results

We made the performance analysis of the optimized GPU implementations, based on images with clear linear structures (see Fig. 12), and the results of our comparisons are shown in Fig. 8, 9, and 10.

In the first experiment (Fig. 8), the comparison of optimized GPU implementations of Bartovsky and Morard algorithms is shown, in analogy to Fig. 5. We assumed that the former is more suitable for the GPU architecture than the latter. This assumption was confirmed by the experiments. Thus, in the following experiments, we used the more successful implementation.

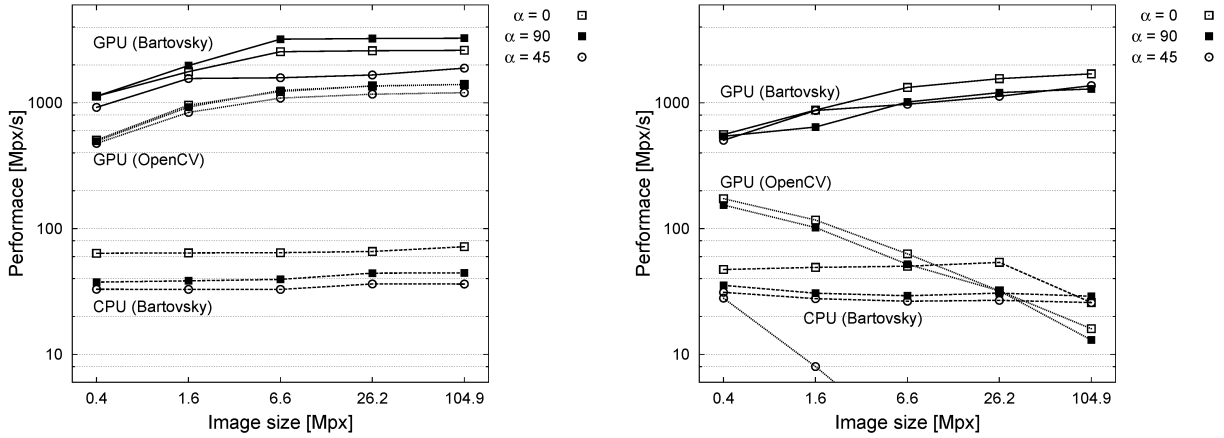
In the second experiment (Fig. 9 and 10), we compared the performance of our GPU implementation, referred to as "GPU (Bartovsky)", to the corresponding CPU implementation, referred to as "CPU (Bartovsky)", and also to the state-of-the-art implementation in the OpenCV library with the GPU support (so-called OpenCV_GPU) [20], referred to as "GPU (OpenCV)". It turns out that our GPU implementation is approximately 10–50 $\times$ faster than the CPU implementation, depending on the input data size and the length of the SE. Despite the fact that the Bartovsky algorithm itself is sequential so the parallelism introduced in its GPU implementation is limited, our implementation is faster than the OpenCV_GPU in every case. Whereas for small SEs the difference between the two GPU implementations is negligible, for larger horizontal and vertical SEs the speedup is up to 50 $\times$. For diagonal (and arbitrarily oriented) SEs, the performance of the OpenCV_GPU implementation falls down very quickly. This is because this implementation uses the NVIDIA Parallel Primitives (NPP) library [44] which supports only simple SE shapes, therefore the line SE has to be represented by a corresponding 2-D rectangle, i.e. a matrix with elements correctly set to 0 or 1.

The most important performance limit of our implementation is the number of threads that can be executed. If the



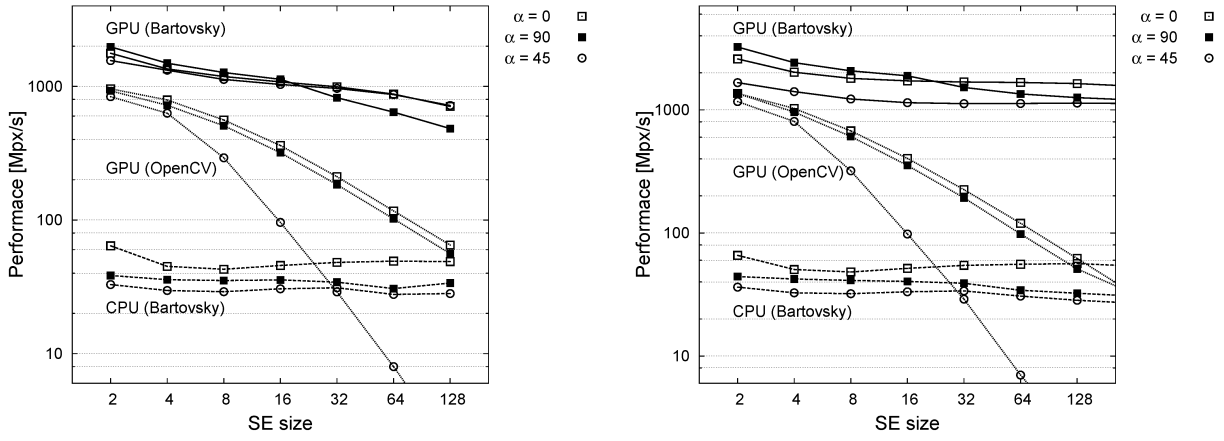
(a) Image D15 (b) Image D47

Fig. 8 Comparison of optimized GPU implementations of Morard and Bartovsky algorithms. For comparison with the basic GPU implementation, refer to Fig. 5.



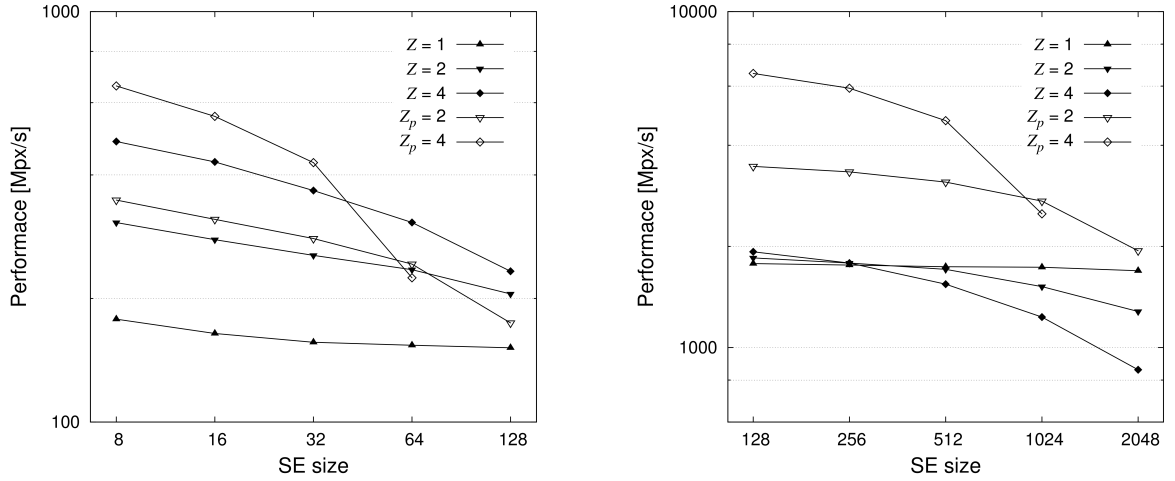
(a) Image D15, SE size = 2 (b) Image D15, SE size = 5 % of image width

Fig. 9 Comparison of CPU and GPU implementations for various image sizes, various SE orientations, and fixed SE sizes.



(a) Image D15, 1280 × 1280 px (b) Image D15, 5120 × 5120 px

Fig. 10 Comparison of CPU and GPU implementations for fixed image sizes, various SE orientations, and various SE sizes.



(a) Image D15, 640×640 px

(b) Image D15, 10240×10240 px

Fig. 11 The contribution of the image split into performance for various image sizes: $Z = 1$ means no split, $Z = 2$, $Z = 4$ means split into 2 and 4 zones, respectively, as described in 5.1; $Z_p = 2$, $Z_p = 4$ means theoretical speedup, as predicted in Eq. (5).

Table 1 Optimal kernel configuration for nVidia graphics cards and for a vertical SE. Choice of the optimal block size S_b, the Z parameter, and the size of the shared memory l depend on many parameters, as described in section 5.4. The Z parameter should be decreased if the SE size is too large. Compute capability refers to a core architecture of nVidia graphics cards [45].

Compute capability 1.x				Compute capability 2.x			
S_b	Image width w	Z	l	S_b	Image width w	Z	l
32	$< 64N_\mathrm{MP}$	4	16	32	$< 64N_\mathrm{MP}$	4	32
64	$< 128N_\mathrm{MP}$	4	8	96	$< 192N_\mathrm{MP}$	4	16
64	$< 256N_\mathrm{MP}$	2	8	96	$< 384N_\mathrm{MP}$	2	16
64	$\geq 256N_\mathrm{MP}$	1	8	96	$\geq 384N_\mathrm{MP}$	1	16

input image is too small, there is not enough threads and the GPU's is underused. This can be avoided by splitting the image in zones (refer to Fig. 6). Adjacent zones need to overlap to avoid border effects. For large SE sizes, the overlap becomes large, with the consequence that the performances decrease, see Fig. 10(a). For larger image sizes, the decrease is proportionally lesser, see Fig. 10(b).

Thanks to the optimizations described in section 5.3 we achieved stable performance for all SE orientations. The performance was tested for all $\alpha \in [0^\circ, 180^\circ)$, although for the sake of simplicity, the graphs show only three orientations. To conclude, our GPU implementation can be used for an arbitrary SE length and direction, achieving the performance more than 1000 Mpx/s. This allows computing one opening on a 40 Mpx image with any 1-D SE in any orientation.

The contribution of splitting the image into zones is shown in Fig. 11 and compared with a theoretical speedup, as computed by Eq. (5). For smaller images, the optimization increases the performance as expected. Note that for large SE sizes, the performance does not decrease as quickly as theoretically predicted. It is because the further parallelism introduced by the split not only occupies more MPs but it also helps to hide memory latencies. For larger images, the number of threads is large enough to occupy MPs, hence the image split does not introduce a further speedup.

The optimal choice of parameters for nVidia graphics cards and for a vertical SE is shown in Table 1. For AMD Radeon graphics cards, the optimum values may differ. For other orientations, the optimal parameters are analogous.

The performance values do not include the data transfers between CPU and GPU. According to our measurements, the time needed to transfer data is comparable with the time needed for the computation of a single opening, hence the overall speedup is half. Here, the GPU implementation is favorable for images larger than 6 Mpx. In the computation of multiple openings, the data transfer overhead is negligible. The performance values for our GPU implementation were obtained on top-class Tesla C2050 GPU (current price 1500 EUR), but we did several test also on $10\times$ cheaper GeForce GTX 470 GPU, and the results were comparable.

6.1 Practical Applications

In practice, linear openings and closings can be used for the detection of either local or global orientations of linear structures. Hence, we tested and compared two CPU implementations (Bartovsky, Morard) and our GPU implementation of linear openings and closings based on images from three different application domains, namely fingerprint analysis, texture characterization, and document analysis. In all cases, we computed a set of linear openings allowing massive parallelism on GPU simply by computing linear openings in all

Table 2 Comparison of CPU and GPU implementations for computation of an angular spectrum. The speedup is computed by comparing the GPU implementation with the best CPU (Morard) implementation.

Image	Image size	No. of angles	Time [ms] CPU (Bart)	CPU (Mor)	GPU (Bart)	Speedup GPU/CPU
Fingerprint	784×1133	180	3768.7	3129.2	115.9	27.0
D15	640×640	180	2425.8	2014.0	58.0	34.7
D47	640×640	180	2397.7	2014.0	55.1	36.6
Music1	1000×1411	81	5830.0	4840.7	129.3	37.4
Music2	1000×1301	81	7297.1	4958.8	136.5	36.3

directions concurrently avoiding the problem with insufficient MPs occupancy. We will show that this leads to a significant speed-up even for small input images. The benchmark results for all applications are shown in Table 2.

6.1.1 Local Orientation of Linear Structures in Fingerprint Images

The first application was the computation of local orientation of linear structures in a fingerprint image (Fig. 1). The operator $\zeta_\lambda(f)$ was computed according to Eq. (3). The GPU implementation achieves a significant speedup (approximately $27\times$) even for small input images (0.9 megapixels).

6.1.2 Angular Spectrum of Texture Images

The second application was the computation of the angular spectrum of texture images. The spectrum was computed in order to find the most important orientation(s) of linear structures in the image. Two test images along with their spectra are shown in Fig. 12. A spectrum $\sigma_\lambda(\alpha)$ of an image f is computed as follows:

$$\sigma_\lambda(\alpha) = \sum_{x \in \Omega(f)} \left[\gamma_\lambda^\alpha(f)\right](x). \tag{6}$$

Again, the GPU implementation achieves a significant speedup (approximately $35\times$) for input images of 0.4 megapixels.

6.1.3 Rotation Detection of Music Sheet Scans

In the third practical application, we detected the rotation of music sheet scans. The music sheets were scanned and stored in an electronic archive. In the process of scanning, a random rotation could occur due to imperfect insertion of the paper to the scanner. The rotation was detected by computing linear closings with large SEs ($\lambda = 250$) of 81 different orientations within the angular range from -10° to 10° with the step of 0.25°. The angular spectrum was computed according to Eq. (6). Two test images along with their spectra are shown in Fig. 13. In this case, the achieved speed-up was approximately $37\times$.

7 Perspectives - Extention to 2-D

The 2-D opening is not separable into two orthogonal 1-D openings as is the dilation. Hence, one cannot directly combine two orthogonal 1-D openings to obtain a 2-D opening.

It is known, that efficient GPU accelerations can only be obtained with simple, regular threads, using as low memory as possible. Hence, the separability principle is useful. Following this idea, one can form 2-D openings by concatenating 2-D erosion and 2-D dilation which are separable. A 1-D dilation algorithm with alike properties as the Bartovsky algorithm has been published in [34].

Assume a 2-D rectangular B in Eq. (1). It can be decomposed into two (horizontal and vertical) 1-D dilations and two 1-D erosions. Similarly, for hexagons we need to compute three erosions and three dilations, and for octagons four erosions and four dilations.

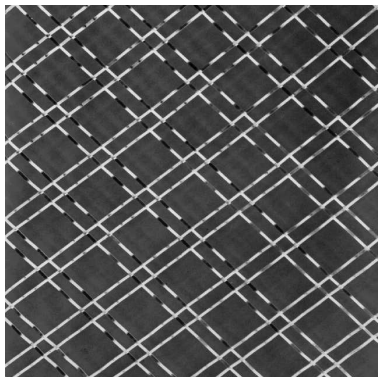
All these orthogonal operators need to be computed sequentially. One cannot expect to obtain the same performances in 2-D as with 1-D openings, since the execution times of the orthogonal operators are added together.

8 Conclusions

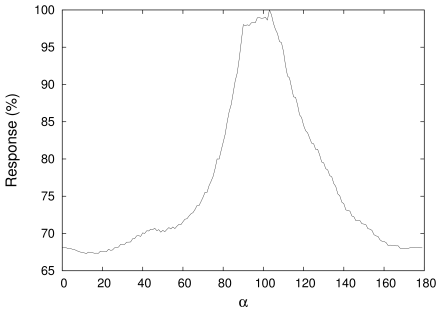
We have reviewed and compared the most efficient linear morphological opening/closing algorithms. At present, the fastest approaches (Van Droogenbroeck and Buckley, Morard et al., and Bartovsky et al.) compute the opening within a single image scan. The algorithm of Van Droogenbroeck and Buckley is the fastest one on CPU, however, it is efficient for 8-bit gray-scale images and for vertical and horizontal linear openings only. Morard and Bartovsky algorithms are applicable to any data accuracy (including floating point).

As described in the paper, both Morard and Bartovsky algorithms themselves are sequential. Hence, the only possibility of introducing more parallelism is on the thread execution level. We explain the choice of the algorithm (Bartovsky) to implement with regard to the GPU architecture (little memory, synchronous execution of threads). We have used various optimization techniques to speed up the code. Mapping various types of data (input, output and FIFOs) to various memory spaces is a crucial aspect. The choice of

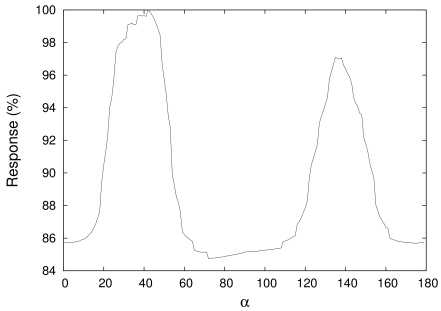
(a) Image D15



(b) Image D47

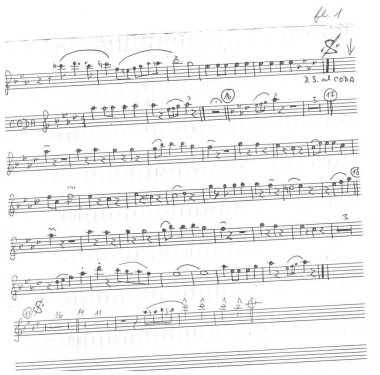


(c) D15 Spectrum $\sigma_{40}(\alpha)$

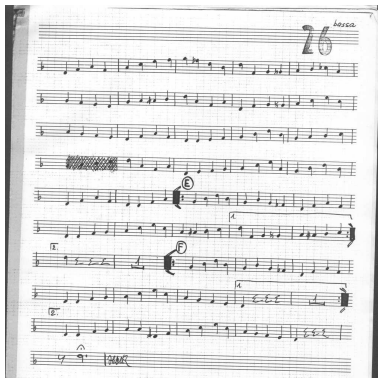


(d) D47 Spectrum $\sigma_{20}(\alpha)$

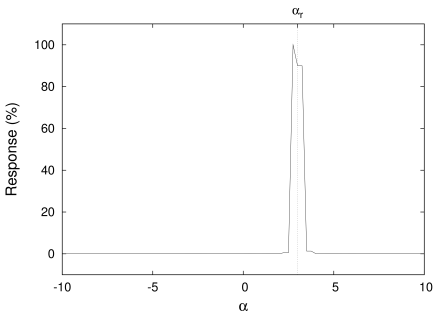
Fig. 12 Texture images [46] and their angular spectra.



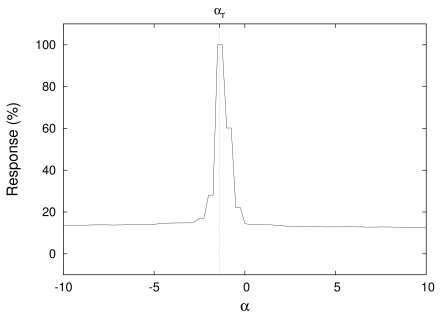
(a) Image Music1



(b) Image Music2



(c) Music1 Spectrum $\sigma_{250}(\alpha)$



(d) Music2 Spectrum $\sigma_{250}(\alpha)$

Fig. 13 Music sheet scans (courtesy of Josef Pilný, Big Band Lanškroun) and their angular spectra. The real (manually measured) rotation angle is denoted by α_r.

memory spaces is described in detail in Section 5.2. We also observed performance limits on small images, hence we try to introduce more spatial parallelism by splitting small images. Finally, we give rules of optimal configuration regarding input image size and the features of the available GPU.

We provide comparisons with the fastest implementations on CPU and on GPU. The current state-of-the-art standard, OpenCV_GPU, is suitable for GPU architecture but has time complexity dependent on the SE size. The proposed implementation obtained stable performance over all orientations and sizes of the structuring element. For a single opening of a large image, we have measured up to 50 $\times$ speedup compared with CPU. For small images, the gain is significant (up to 37 $\times$ speedup) if one computes a set of openings in multiple directions. For example, within 60 ms, the GPU implementation is capable of computing a single opening at arbitrary angle of a 60 Mpx image, or a set of openings in 180 directions of a 640$\times$640 px image.

To conclude, this solution is suitable for applications with large, industrial images, running under severe timing constraints, such as production control in e.g. metallurgy or textile industry. A typical such application requiring using high-resolution images, and running under severe time constraints is the surface control. Thin (often μm) cracks in large surfaces require using high resolution images, and the timing is given by the industrial cycle.

Source codes of the CPU and GPU implementations of the Bartovsky algorithm are publicly accessible under GNU GPL license [47].

Acknowledgements

This work has been done during ERASMUS stage of Pavel Karas at ESIEE Paris. The presented work is the result of collaboration between ESIEE Paris (A3SI team), Mines-ParisTech (CMM) and Masaryk University in Brno (CBIA).

This stay period of Pavel Karas at ESIEE Paris has been also supported by the Ministry of Education, Youth and Sport of the Czech Republic (Projects No. MSM-0021622419 and No. LC535). This research was also supported by the Grant Agency of the Czech Republic (Grant No. P302/12/G157).

References

1. J. Serra. Morphological filtering: An overview. *Signal Processing*, 38:3–11, 1994.
2. L. Vincent. Fast opening functions and morphological granulometries. In *SPIE*, volume 2300, pages 253–267, 1994.
3. S. Batman, E. R. Dougherty, and F. Sand. Heterogeneous morphological granulometries. *Pattern Recognition*, 33(6):1047 – 1057, 2000.
4. L. Vincent. Granulometries and opening trees. *Fundam. Inf.*, 41:57–90, January 2000.
5. E. R. Urbach, J. B. T. M. Roerdink, and M. H. F. Wilkinson. Connected rotation-invariant size-shape granulometries. *Pattern Recognition, International Conference on*, 1:688–691, 2004.
6. J. Serra and L. Vincent. An overview of morphological filtering. *Circuits, Systems and Signal Processing*, 11(1):47–108, 1992.
7. H. Heijmans. A new class of alternating sequential filters. In *I of Proceedings of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, pages 3–0, 1995.
8. N. Theera-Umpon and P. D. Gader. Counting white blood cells using morphological granulometries. *J. Electronic Imaging*, pages 170–177, 2000.
9. A. Bagdanov and M. Worring. Granulometric analysis of document images. In *Proceedings of the International Conference on Pattern Recognition*, volume I, pages 478 – 481, 2003.
10. S. Outal, D. Jeulin, and J. Schleifer. A new method for estimating the 3d size-distribution curve of fragmented rocks out of 2d images. *Image Analysis and Stereology*, 27(2), 2011.
11. D. Talukdar and R. Acharya. Estimation of fractal dimension using alternating sequential filters. In *Proceedings of the 1995 International Conference on Image Processing (Vol. 1)*, ICIP '95, Washington, DC, USA, 1995. IEEE Computer Society.
12. S. O. Sigurjonsson, J. A. Benediktsson, and J. R. Sveinsson. Street tracking based on sar data from urban areas. In *International Geoscience and Remote Sensing Symposium*, pages 1273–1276, 2005.
13. M. Kowalczyk, P. Koza, P. Kupidura, and J. Marciniak. Application of mathematical morphology operations for simplification and improvement of correlation of images in close-range photogrammetry. In *ISPRS08*, page B5: 153 ff, 2008.
14. V. Morard, P. Dokládal, and E. Decencière. Linear openings in arbitrary orientation in O(1) per pixel. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1457–1460, May 2011.
15. J. Bartovský, P. Dokládal, E. Dokládalová, and M. Bilodeau. Fast streaming algorithm for 1-d morphological opening and closing on 2-d support. In P. Soille, M. Pesaresi, , and G.K. Ouzounis, editors, *ISMM 2011*, volume 6671 of *LNCS*, pages 296–305. Springer, July 2011.
16. nVidia Corporation. NVIDIA GPU Computing Developer Home Page. `http://developer.nvidia.com/category/zone/cuda-zone`, Jun 2011.
17. Khronos Group. OpenCL. `http://www.khronos.org/opencl/`, 2011.
18. B. Obara. Identification of transcrystalline microcracks observed in microscope images of a dolomite structure using image analysis methods based on linear structuring element processing. *Computers and Geosciences*, 33:151–158, 2007.
19. F. Zana and J.-C. Klein. Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation. *IEEE Transactions on Image Processing*, 10:1010–1019, 2001.
20. Willow Garage. OpenCV_GPU. `http://opencv.willowgarage.com/wiki/OpenCV_GPU`, Oct 2011.
21. P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
22. J. Pecht. Speeding-up successive minkowski operations with bit-plane computers. *Pattern Recognition Letters*, 3(2):113–117, 1985.
23. D. Coltuc and I. Pitas. On fast running max-min filtering. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 44(8):660–663, 1997.
24. M. van Herk. A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. *Pattern Recognition Letters*, 13(7):517–521, 1992.
25. J. Gil and M. Werman. Computing 2-d min, median, and max filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(5):504–507, 1993.
26. P. Soille, E.J. Breen, and R. Jones. Recursive implementation of erosions and dilations along discrete lines at arbitrary angles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):562–567, 1996.
27. C. Clienti, M. Bilodeau, and S. Beucher. An efficient hardware architecture without line memories for morphological image processing. In *Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems*, ACIVS '08, pages 147–156, Berlin, Heidelberg, 2008. Springer-Verlag.

28. M. Van Droogenbroeck and M.J. Buckley. Morphological erosions and openings: fast algorithms based on anchors. *Journal of Mathematical Imaging and Vision*, 22(2):121–142, 2005.
29. L. Garrido, P. Salembier, and D. Garcia. Extensive operators in partition lattices for image sequence analysis. In *Signal Processing*, pages 157–180, 1998.
30. P. Matas, E. Dokládalová, M. Akil, T. Grandpierre, L. Najman, M. Poupa, and V. Georgiev. Parallel algorithm for concurrent computation of connected component tree. In *Advanced Concepts for Intelligent Vision Systems*, pages 230–241. Springer, 2008.
31. J. Brambor. *Algorithmes de la Morphologie Mathématique pour les architectures orietées flux*. PhD thesis, École des Mines de Paris, 2006.
32. Ch. Clienti. Fulguro image processing library. `http://sourceforge.net/projects/fulguro/`, 2011.
33. L. Domanski, P. Vallotton, and D. Wang. Parallel van Herk/Gil-Werman image morphology on GPUs using CUDA. GTC 2009 Conference posters, 2009.
34. P. Dokládal and E. Dokládalová. Computationally efficient, one-pass algorithm for morphological filters. *Journal of Visual Communication and Image Representation*, 22:411–420, 2011.
35. M. H. F. Wilkinson, H. Gao, W. H. Hesselink, J.-E. Jonker, and A. Meijster. Concurrent computation of attribute filters on shared memory parallel machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1800–1813, 2008.
36. D. Menotti-Gomes, L. Najman, and A. de Albuquerque Araújo. 1D Component tree in linear time and space and its application to gray-level image multithresholding. In *International Symposium on Mathematical Morphology'07*, volume 1, pages 437–448. INPE, 2007.
37. nVidia Corporation. *CUDA Toolkit Reference Manual*. 2701 San Tomas Expressway, Santa Clara, CA95050, August 2010.
38. J. Kong, M. Dimitrov, Y. Yang, J. Liyanage, L. Cao, J. Staples, M. Mantor, and H. Zhou. Accelerating MATLAB Image Processing Toolbox functions on GPUs. In *GPGPU '10: Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, pages 75–85, New York, NY, USA, 2010. ACM.
39. nVidia Corporation. FERMI Tuning Guide. `http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/Fermi_Tuning_Guide.pdf`, Aug 2010.
40. AMD Corporation. OpenCL Course: Introduction to OpenCL Programming. `http://developer.amd.com/zones/OpenCLZone/courses/pages/Introduction-OpenCL-Programming-May-2010.aspx`, May 2010.
41. nVidia Corporation. NVIDIA Parallel Nsight. `http://developer.nvidia.com/nvidia-parallel-nsight`, 2011.
42. J. Nickolls and W.J. Dally. The GPU Computing Era. *IEEE Micro*, 30:56–69, March 2010.
43. nVidia Corporation. CUDA SDK Code Samples. `http://developer.nvidia.com/cuda-toolkit-32-downloads`, Aug 2010.
44. nVidia Corporation. NVIDIA Performance Primitives (NPP) Library User Guide. `http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/NPP_Library.pdf`, Feb 2011.
45. nVidia Corporation. CUDA C Programming Guide. `http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/CUDA_C_Programming_Guide.pdf`, Sep 2010.
46. T. Randen. Brodatz Textures. `http://www.ux.uis.no/~tranden/brodatz.html`.
47. Pavel Karas and Jan Bartovsky. CUDA-based Linear Openings. `http://sourceforge.net/p/linearopenings/`, Jan 2012.

Image Anal Stereol 2003;22:121-132
Original Research Paper

A PARALLEL ARCHITECTURE FOR CURVE-EVOLUTION PARTIAL DIFFERENTIAL EQUATIONS

EVA DEJNOŽKOVÁ AND PETR DOKLÁDAL

School of Mines of Paris, Center of Mathematical Morphology, 35, Rue Saint Honoré, 77 300 Fontainebleau, France
e-mail: {dejnozke,dokladal}@cmm.ensmp.fr
(Accepted May 20, 2003)

ABSTRACT

The computation of the distance function is a crucial and limiting element in many applications of image processing. This is particularly true for the PDE-based methods, where the distance is used to compute various geometric properties of the travelling curve. Massive Marching[a] is a parallel algorithm computing the distance function by propagating the solution from the sources and permitting simultaneous spreading of component labels in the influence zones. Its hardware implementation is conceivable as no sorted data structures are used. The feasibility is demonstrated here on a set of parallely-operating Processing Units arranged in a linear array. The text concludes by a study of the accuracy and the implementation cost.

Keywords: distance function, hardware for image processing, partial differential equations, parallel computing.

[a]Short preliminary study published in: Dejnožková and Dokládal (2003b) (algorithm) and Dejnožková and Dokládal (2003a) (architecture)

INTRODUCTION

Recently, the image processing methods based on Partial Differential Equations (PDE) have gotten an ever increasing attention. The examples of application can be found in numerous domains such as filtering (non-linear diffusion), active contours used for segmentation of either static images (Voronoï graph, watershed, shortest path, object detection) or sequences (object tracking) or more recent methods as shape from shading. The implementation of the PDE-based method requires the computation of non-linear functions. They are often solved by iterative and recursive algorithms characterized by a high computational cost. Therefore, only a limited number of real-time applications exists. The most common existing custom chips implement non-linear filtering, *i.e.* the non-linear diffusion (Perona and Malik, 1988; Gijbels *et al.*, 1994). One can find some experiments with super-computers (Sethian, 1996) or some examples of PDE-based segmentation using the level-sets implemented on graphic hardware (Rumpf and Strzodka, 2001).

The design of a custom chip becomes meaningful not only for the acceleration but also for the implementability on embedded systems (Suri *et al.*, 2002). Many authors put forth a considerable effort to reformulate existing sequential algorithms in a parallel form or to speed up the convergence (Weickert *et al.*, 1998). However the number of necessary iterations remains excessively high.

The motivation of our work is to define a general type of parallel architecture fitting the needs of the above-mentioned applications. There are several reasons why few architecture propositions have been made for interface (curve-based) algorithms. The curve traveling in a continuous space $\mathbb{R}^n$ is implicitly described in a discrete space $\mathbb{Z}^n$ by the distance to the curve, as proposed first in Osher and Sethian (1988). The distance function is the computation support of the level-set methods. Its zero-level set represents the traveling interface. Its accurate computation is very important because its geometrical properties (e.g. the curvature) are then used to describe its evolution in the time. The interface evolution imposes frequent and random memory accesses. Also, the numerical solution of a classical variational formulation leads to deformations of the implicit description of the curve (Kimmel, 1995) imposing more or less frequent reinitializations. In the work of Zhao *et al.* (1996) and Gomez and Faugeras (1992) can be found propositions of algorithm without re-initialization paid by the necessity to search the propagation speed on the zero-level set even for the points situated elsewhere. Because of the complexity and high computational cost, the classical re-initialization approach is still leading (Paragios, 2000). Repetitive re-initialization alternated with another type of processing increases the requirements on the implementing architecture.

Other applications, where the distance function is the result and not only a support, are the reconstruction

of 3D surfaces, minimal path search (Kimmel and Sethian, 2001) or continuous watershed (Meyer and Maragos, 1999).

In Gijbels *et al.* (1994) has been proposed a linear-array, SIMD architecture for PDE-based filtering by linear diffusion. An efficient hardware (parallel) implementation of a (weighted) distance function preserving the accuracy and the necessary sub-pixel precision for the continuous interface evolution is still needed. The goal of this paper is to show that the same architecture type can also be used for interface evolution algorithms.

This paper is organized as follows. After a brief review of existing distance computing algorithms, we analyse the principles of the Massive Marching. Section "Hardware implementation" presents its hardware implementation. Experiment results state the achieved implementation parameters such as the surface requirements, clock rate and necessary fixed-point precision for a given error.

REVIEW OF EXISTING ALGORITHMS

The sub-pixel accuracy is one of the essential features required from algorithms computing the distance function for the level-set methods. The initial condition, a closed curve $\mathscr{C}_0$ placed arbitrarily in $\mathbb{R}^n$, represents the zero-level set of the searched distance function $u(x,y)$. For $n=2$ we have:

$$\mathscr{C}_0 = \left\{(x,y) \in \mathbb{R}^2 \mid u(x,y) = 0\right\} . \qquad (1)$$

Recall that u is discrete, *i.e.* only $u(i,j),\ i,j \in \mathbb{Z}^2$ is known. Other necessary characteristics required from the distance computing algorithms are the preservation of the accuracy, computation on a narrow band for the active contours and simultaneous propagation of components labels.

Current algorithms used to find u given $\mathscr{C}_0$ are from the hardware implementation aspect optimal only for a particular type of applications. Two types of algorithms exist: the first type proceeds by scanning the entire image and the second type propagates a narrow-band solution from the initial interface.

The algorithms using successive scannings are simple to implement. However, they suffer from a serious drawback that the next scan cannot begin before the previous one ends. Therefore, the scanning-based methods are optimal only for those applications where the solution on the entire image is required. A classical example is the algorithm of Danielsson (1980) which is based on two coordinates description and its overall complexity is $\mathscr{O}(N)$, where N is the number of points in the image. It yields the square of the distance function, which imposes to compute the square roots before the distance can be used for other computation, e.g. the curvature. Moreover, this algorithm is not conceived to operate with a sub-pixel precision. This problem is resolved in Tsai (2000) which proposes a *sweeping* algorithm (inspired from Boué and Dupuis (1999)). By using a new numerical scheme, it yields the exact distance and not the square and it reduces the numerical error of the classical Godunov scheme. The complexity of this algorithm is $\mathscr{O}(MN)$ where M is the number of data points and N is the number of grid points. However it is not possible to calculate the influence zones of different sources.

The algorithms operating in the narrow band are more appropriate for the propagation of labels. Their implementation is complicated by using sophisticated data structures and the complexity is $\mathscr{O}(Nlog(N))$. The Fast Marching introduced by Sethian (1996) is the most often used propagation technique in combination with the PDE-based methods. The algorithm allows to compute the distance function by realizing the principle of Huygens, as it is introduced in Verbeek and Verwer (1990), by propagating equidistant waves. For this, the Fast Marching algorithm needs to use an *ordered data structures* with a *real-number priority*. In every iteration can be processed only the point with the highest priority. The maximum priority represents a *global information* and makes this algorithm sequential. Moreover, the hardware implementation of data structures using a real-number priority is difficult because of operations like insertion, reading and re-positioning of elements.

This is not the case of the USP algorithm (Eggers, 1997) which does not use any sorted data structures. However, the result it yields is the square of the distance and it requires to memorize the current iteration number. It does not operate in sub-pixel accuracy and the complexity is $\mathscr{O}(n^3)$ for images of $n \times n$ points.

IMPLEMENTATION ISSUES

The filtering as well as the segmentation algorithms respond to some function of geometrical properties of the level-set function u such as gradient or curvature. These geometrical properties are obtained directly from the values of u or its derivatives (Sethian, 1996; Sapiro, 2000). The computation of the derivatives is an elementary operation. For every point concerned, these operations are performed on the nearest neighborhood and are independent each of the other. Hence they can be executed in parallel (Dejnožková, 2002).

Since the PDE-based algorithms principally consist in repetitive computation of the elementary

operations, the SIMD (Single Instruction Multiple Data Stream) architecture is the natural choice to reduce the processing time. We propose a divide-and-conquer approach in order to obtain a more balanced processors' activity and to limit the space on the chip. Thus one can benefit from a quasi parallel implementation by dividing the input data into blocks.

Recall that the SIMD architecture consists in an array of processors with an interconnection network for the communication. Each processor has its private non-shared memory. A single controller broadcasts instructions to all the processors. The processors then execute the instructions simultaneously at a given time.

The choice of the algorithm to implement and the architecture type come together. Compared to the filtering, other techniques as the continuous watershed or active contours require computation of a (weighted) distance to the given markers. These algorithms use sophisticated ordered data structures (such as hierarchical queues or a sorted heap) which can penalize the execution time on the SIMD architectures. The processing of such data structures introduces a sequential approach. Only one point (with maximal priority) can be processed at a time. Another reason why the SIMD architecture would be less efficient for this type of algorithms is the random access to the memory.

In the next section we show the implementation of the Massive Marching algorithm used for the computation of a distance function. This algorithm is fully parallel. Hence the execution time on an architecture with P processors is $t_{\text{parallel}} = t_{\text{sequential}}/P$.

MASSIVE MARCHING ALGORITHM

Throughout this paper we use the following notations. Let $p = [x_p, y_p]$ be a point of an isotropic, rectangular and unit grid. $V(p)$ denotes the 4-neighborhood of p defined as $V(p) = \{[x_p, y_p \pm 1], [x_p \pm 1, y_p]\}$. The point q is a neighbor of p if $q \in V(p)$, $u(p)$ denotes the value of the distance function in p.

NUMERICAL SCHEME

Numerical schemes allow to obtain the value of the distance function in a point according to the values of the neighbors. The numerical scheme is a discretization of the eikonal equation:

$$|\nabla u| = \mathscr{F}\,, \tag{2}$$

where $\mathscr{F}$ is the weight for a weighted distance. Some propositions of numerical schemes can be found in Sapiro (2000) or Tsai (2000). The most often used scheme is, in the domain of the Level Set, the scheme proposed by Godunov (Sethian, 1996).

$$\left[\max\left\{u(p) - u\left([x_p \pm 1, y_p]\right), 0\right\}^2 - \max\left\{u(p) - u\left([x_p, y_p \pm 1]\right), 0\right\}^2\right]^{\frac{1}{2}} = \mathscr{F}(p)\,. \tag{3}$$

In order to obtain the maximum values of the terms, we have to consider the neighbors with minimum values of u. The Godunov scheme requires to determine the maximal real solution of a quadratic equation (3).

Suppose that the distance u in the point p can be expressed by the following function of the neighborhood $V(p)$ of the point p and the weight $\mathscr{F}(p)$

$$u(p) = u_{\min}(p) + f_{\text{diff}}(V(p), \mathscr{F}(p))\,. \tag{4}$$

$u_{\min}(p)$ is the distance value of the minimal neighbor: $u_{\min}(p) = \min_{q_i \in V(p)}\{u(q_i)\}$. The formulation of f_{diff} depends on the choice of the numerical scheme. The Godunov scheme can be rewritten in the form of Eq. (4) where f_{diff} reads as

$$f_{\text{diff}} = \frac{|u_x(p) - u_y(p)|}{2} + \sqrt{\frac{\mathscr{F}(p)^2}{2} - \left(\frac{u_x(p) - u_y(p)}{2}\right)^2}\,, \tag{5}$$

and where $u_x(p) = \min\{u([x_p \pm 1, y_p])\}$ and $u_y(p) = \min\{u([x_p, y_p \pm 1])\}$

In Eq. (5) the minimum real solution is considered. If there is no real solution then the distance value is computed only from the minimal neighbor and $f_{\text{diff}} = \mathscr{F}(p)$.

INITIALIZATION

$\mathscr{C}_0$ is a closed curve which generally lies between the points of the grid $\mathbb{Z}^2$. Its accurate inter-pixel location is identified by the distance map u to $\mathscr{C}_0$. However, if it is to be described implicitly on a discrete support $\mathbb{Z}^2$, the curve $\mathscr{C}_0$ may not be placed arbitrarily. Its location is determined by the switching function $\varphi(p)$ where $\text{sign}(\varphi(p))$ indicates whether a given point p lies inside or outside $\mathscr{C}_0$ (as introduced in Sethian (1996)). Hence, $\mathscr{C}_0$ is located between adjacent points for which $\text{sign}(\varphi(.))$ differs. The exact distance u of these points to $\mathscr{C}_0$ is obtained by some interpolation method.

The choice of the interpolation depends on the requirements of the application. One can use either a constant value or a bilinear or a more sophisticated interpolation method allowing to detect more or

less complicated forms (for examples see Osher and Shu (1991); Siddiqi *et al.* (1997)). The majority of practical applications use a linear approximation. In the following, consider a linear interpolation and $|\varphi| = const.$ for all $p \in \mathbb{Z}^2$. Since u can only have a finite number of constant values for all points adjacent to $\mathscr{C}_0$, the initialization of the distance function reduces to $u : \mathbb{Z}^2 \to \{c_1, c_2, \ldots, c_n\}$, where all $c_i \in \mathbb{R}$. The number of the constants depends on the number of neighbors from which the interpolation is calculated. The Fig. 1 shows all the possible 4-neighborhood configurations for the linear interpolation.

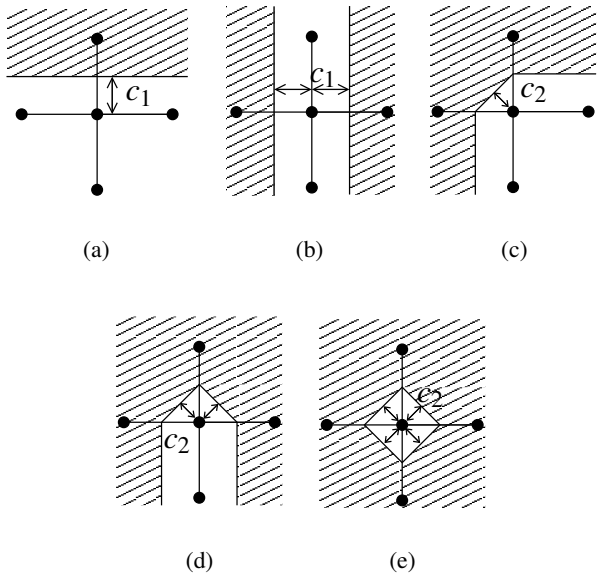


Fig. 1. *Configurations of the 4-neighborhood for the initialization.*

If the central point lies on the object's edge, *i.e.* the sign of neighbors changes with respect to the central (interpolated) point only in one direction (see Fig. 1(a) or 1(b)), the interpolation is computed only from $\varphi(.)$ of these neighbors. If the sign changes in both directions (corner or isolated point) the resulting value is computed from $\varphi(.)$ of the two neighbors (Fig. 1(c) to 1(e)).

Hence the initialization of $u(p)$ can be realized efficiently as a logical function of $\mathrm{sign}(\varphi(p))$, $\mathrm{sign}(\varphi(q_1))$, ..., $\mathrm{sign}(\varphi(q_4))$ and the result of the function is used to retrieve the corresponding value from a look-up-table containing the constants c_i.

PROPAGATION

We use the following sets to define the algorithm: $\mathscr{A}$ is the set of points initialized by the interpolation, $\mathscr{Q}$ is the set of points marked as active $\mathscr{Q} = \{q_i \mid q_i \notin \mathscr{A}$ and $V(q_i) \cap \mathscr{A} \neq \emptyset\}$. The algorithm reads as follows:

Initialization

- Initialize the neighborhood of the curve with a signed distance (set $\mathscr{A}$)
- Initialize the distance value u of the other points to ∞
- Mark the neighbors of $\mathscr{A}$ as *active* (set $\mathscr{Q}$)

Propagation

while $\mathscr{Q} \neq \{\}$, do in parallel for all $p \in \mathscr{Q}$:

{

⋆ *Compute new distance value*:

- *Jacobi step:*

$$u^{n+1}(p) = u^n_{\min}(p) + \min\{f_{\mathrm{diff}}(V(p), \mathscr{F}(p)), \mathscr{F}(p)\} \quad (6)$$

- *Gauss-Seidel step:*

$$u^{n+1}(p) = u^{n+1}_{\min}(p) + \min\{f_{\mathrm{diff}}(V(p), \mathscr{F}(p)), \mathscr{F}(p)\} \quad (7)$$

⋆ *Activation of new points to process:*

- delete p from $\mathscr{Q}$, insert p in $\mathscr{A}$
- if $u(p) < \mathrm{NB}_{\mathrm{width}}$ then for all q_i, $q_i \in V(p)$ such that $u^{n+1}(q_i) - u^{n+1}(p) > \varepsilon(q_i)$ (8)

  insert $q_i \to \mathscr{Q}$

}

where $\mathrm{NB}_{\mathrm{width}}$ is the desired width of the narrow band[1]. (The values of unprocessed points in t_n are automatically carried over to the next iteration and are noted as values at t_{n+1}.)

At each iteration, the value is calculated for the *active* points. The algorithm does not use any kind of sorted processing. Consequently, the front of the propagation is not equidistant to the initial curve. Two situations exist where the points that are currently being calculated will have to be reactivated later:

1. The value of the point is calculated on an incomplete neighborhood which imposes a *two-step algorithm*
2. The points are activated by a propagation front coming from a source which is not necessarily the closest one which is detected by *activation rule*.

[1] To obtain u on the entire image let $\mathrm{NB}_{\mathrm{width}} = \infty$

Two-step based algorithm

We say that the point value is computed on an incomplete neighborhood since adjacent (mutually dependent) points may be processed simultaneously. The calculation is therefore performed in two steps. The steps are named after their similarity with the algorithm of Markov chain approximation by PDE as introduced in Boué and Dupuis (1999). The first one, the *Jacobi step*, calculates the value of the distance function at t_{n+1} given only the values obtained at t_n. The second one, the *Gauss-Seidel step*, recalculates the distance value at t_{n+1} by using also the values obtained at t_{n+1}.

The algorithm computes the value $u(p)$ from the least neighbor. Hence, the infinite values are not considered for the computation. As mentioned above, in the Jacobi iteration, the first estimation of $u^{n+1}(p)$ is obtained by using the values of neighbors from the previous iteration which is recomputed once more in the Gauss-Seidel step.

Activation rule

The activation rule has two important roles: the supervision of the propagation end and the detection of the overlapping of the propagation waves (see Fig. 2).

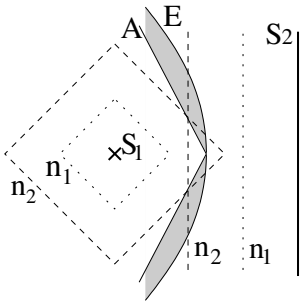


Fig. 2. *Example of an overlapping of the propagating waves and zone of reactivation (in gray). S_1, S_2 are the propagation sources. The dashed lines represent the propagation waves after n_1 and n_2 iterations.*

We calculate the distance function $u(x) = \min[\mathrm{dist}(x, S_1), \mathrm{dist}(x, S_2)]$. Let E denote the set of points equally distant from S_1 and S_2, $E = \{x | \mathrm{dist}(x, S_1) = \mathrm{dist}(x, S_1)\}$. The points to the left (resp. right) from E are closer to S_1 (resp. S_2). Note that in this example E is a parabola. A denotes the set of points activated simultaneously by the two fronts coming from the two sources. Since the propagation front of Massive Marching is not equidistant to the propagation source, the sets A and E do not coincide. The zone delimited by A and E contains points that were activated from S_2 whereas they are closer to S_1. These points will be reactivated again by the front coming from S_1 in order to lower their value from $\mathrm{dist}(x, S_2)$ to $\mathrm{dist}(x, S_1)$.

Suppose that $u^n(p)$ has just been calculated and p is deactivated. We search for an estimator of $u^{n+1}(q_i)$ to know whether the neighbor q_i of p should be activated in order to compute or to re-compute its value. Since the goal is to obtain the minimal solution the main idea is to test whether the value $u(q_i)$ could be brought down by considering p as the least neighbor of q_i. Suppose that p is the least neighbor of q_i. Then in the next iteration q_i will receive its value from p. From Eq. (4), $f_{\mathrm{diff}}(p)$ is the difference between the distance value $u(p)$ of a given point p and the least of the neighbors $u_{\min}(p)$. The new value $u(q_i)$ will satisfy $u(q_i) \geq u(p) + \inf f_{\mathrm{diff}}$.

Let $K_{\min}$ be the lower bound of f_{diff}:

$$K_{\min}(p) = \inf f_{\mathrm{diff}}(V(p), \mathscr{F}(p)) \,. \tag{9}$$

$\mathscr{F}(p)$ is an arbitrary but time invariant function. $K_{\min}$ is a predictor of the least increment of u in one iteration. All neighbors q_i of p such that $u(q_i) - u(p) > K_{\min}(p)$ should therefore be (re-)activated and (re-)calculated since the new value $u(p)$ may affect $u(q_i)$ in the next iteration. Hence, ε in Eq. (8) must satisfy:

$$\varepsilon(p) \geq K_{\min}(p) > 0 \,.$$

Remark: The lower bound of f_{diff} of the Godunov scheme (from the section 2.1) is

$$K_{\min}(p) = \sqrt{\frac{\mathscr{F}(p)^2}{2}} \,. \tag{10}$$

Note that $K_{\min}$ is constant whenever $\mathscr{F}$ is constant in (2) and becomes a function of $\mathscr{F}$ whenever $\mathscr{F}$ varies over the image.

Setting $\varepsilon < K_{\min}$ is useless because it would authorize the activation of points that will not be updated and the propagation could go backwards. By letting $\varepsilon > K_{\min}$ one can authorize fewer reactivations (lower execution time) paid by some error (proportional to $\varepsilon - K_{\min}$) in the result (Dejnožková, 2002).

LABEL PROPAGATION

The propagation of the region labels can be realized simultaneously with the computation of the distance function to obtain the influence zones for Voronoï tesselation or continuous watershed. Suppose that the region labels are initialized during the Massive Marching initialization stage. The algorithm modifies as the distance must be computed from neighbors having the same label.

- *Jacobi step*
  if $u_x(p)$ and $u_y(p)$ have the same label then use Eq. (6)
  else use $u^{n+1}(p) = u^n_{\min}(p) + F(p)$
- *Gauss-Seidel step*
  1. if $u_x(p)$ and u_y have the same label use Eq. (6)
     else use $u^{n+1}(p) = u^{n+1}_{\min}(p) + F(p)$
  2. p receives the label of $u^{n+1}(p)$

COMPUTATION ERROR

Two types of error exist: numerical scheme error and incomplete neighborhood error.

Numerical scheme error

As the majority of first order schemes also the Godunov scheme suffers from "shortsightedness" as it uses only the nearest neighborhood. Moreover, Tsai (2000) explains that when the propagation starts from isolated points it creates diamonds instead of circles. In Sethian (1999) Sethian proposes a switching mechanism between the first and second order scheme in order to improve the accuracy. Nevertheless, in ∞ the Godunov scheme converges to the exact solution.

Incomplete neighborhood error

This error is caused by the computation on an incomplete neighborhood (see section "Two-step based algorithm"). Massive Marching calculates simultaneously the values of adjacent points, *i.e.* values depending each on the others. Therefore the calculation is performed in two steps.

Also all the methods referenced in the introduction allow to recalculate the points several times in order to obtain more accurate solution to Eq. (2). The scanning-based methods recalculate during each scanning all the points of the image. Successive scannings have to be repeated until the convergence. Methods for the narrow band use a variable number of recalculations, implemented by using a sorted heap, depending locally on the neighborhood of every particular point. At every recalculation the point receives a new value of the distance according to the new values of the neighbors. Massive Marching authorizes to reactivate the neighbor q_i if the point p receives a new value $u(p)$ inferior to $u(q_i) - \varepsilon$ (see condition Eq. 8).

An additive error (typically at the fourth decimal place) may still appear in some special cases (as corners etc.), see Dejnožková (2002). The experiments have shown that the two-step calculation gives sufficient accuracy for most practical applications (see section "Experimental results"). Should more accurate results be required then the Gauss-Seidel step can be repeated.

COMPUTATION COMPLEXITY

In order to obtain the calculation complexity of Massive Marching we first assume that $\mathscr{F} = const.$ over the entire image.

The value of an active point p is obtained in a constant time $\mathscr{O}(1)$ after which the point deactivates itself. The point activates all its neighbors that verify the condition (8). The function (4) is strictly positive, no point can therefore reactivate the neighbor from which it has received the activation. If the propagation starts from one point representing the source, the algorithm complexity is $\mathscr{O}(N)$, with N be the number of points in the image.

For sources having more complicated geometrical forms or more than one source the complexity may exceed $\mathscr{O}(N)$ since some points may be activated more than once. We show that the number of reactivations is bounded. Consider two isolated points a and b such that there is a point c, $c \in V(a)$, $c \in V(b)$ and $a \notin V(b)$. Suppose that the two propagation fronts arrive respectively via a and b and meet in c. The two fronts have different speed and in the iteration n the distance values in a and b verify $|u^n(a) - u^n(b)| \geq 2K_{min}$. The faster front will stop in c whereas the slower one will continue. It can be shown that its propagation will stop after i iterations, where

$$i < \frac{u^n(b) - u^n(a)}{2K_{min}} - 1 \,.$$

In images with $\mathscr{F} = const.$, the waves propagate with unit increment of u in one iteration in vertical and horizontal direction, whereas in the diagonal directions the increment is obtained only after two iterations. See illustration at Fig. 2. The waves arriving from S_1 and S_2 meet first on the intersection of A and E since both waves have the same speed on the horizontal direction. Later, see the iteration n_2 for example, the waves meet outside the skeleton since the wave arriving from S_1 arrives diagonally and is therefore slower. The slower wave will continue its propagation up to the skeleton of the distance E where it stops.

For images with bounded support, the term $|u^n(a) - u^n(b)|$ is upper bounded and hence the number of reactivations also. For images where $\mathscr{F} \neq const.$, this term is also limited and depends on $\mathscr{F}$.

HARDWARE IMPLEMENTATION

The main implementation issues are outlined in the section "Implementation issues". In this section we present the implementation details and we discuss the possible extensions of the proposed architecture.

GLOBAL ARCHITECTURE

For the simulation and validation, we have chosen the division of the input image into the columns, *i.e.* one processing unit per column of the image. In a given moment the processing units process in parallel all the points in a row. We give below a description of the proposed (and tested) access to the neighbors. The processing units are controlled by the single Global Control block. It reads high-level instructions from the Code Memory. Each high level instruction indicates the action executed on the entire image in one scan and the code does not have to be decomposed in the elementary operations. The algorithm for computing a distance function on the entire image (given in section "Propagation") resumes to these high-level instructions:

interpolate
loop (if any point is active)
{ *Jacobi_Step*
Gauss-Seidel_Step }

The condition *any point is active* is the OR operation over all the activation flags.

The Global Control block not only controls the execution of the algorithm but also allows to change the values of the approximation registers inside each processing unit. Thus we can implement the approximation of almost any non-linear function.

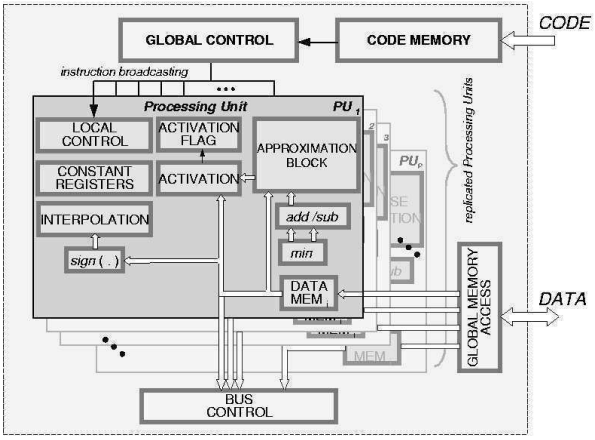


Fig. 3. *Global architecture with replicated Processing Units (PUs).*

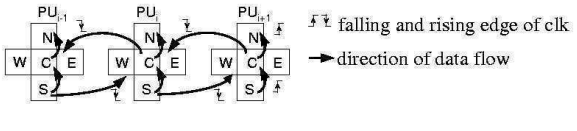
PROCESSING UNIT

The Processing Unit (see Fig. 3) contains specific blocks implementing different stages of the algorithm: the INTERPOLATION BLOCK ensures the initialization of the algorithm, and the APPROXIMATION BLOCK computes new pixel values. Each PU has a register containing the ACTIVATION FLAGs for its part of the image. The activation flag is used as a mask controlling the PU activity. All the Processing Units, whose currently processed point is active (the activation flag is set) execute the instructions, otherwise they are idle, except of sending their values to the east- and west-side neighbors.

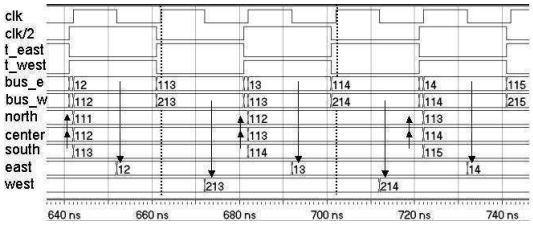
Note that the input data are stored in a non-shared data memory. Therefore, all the units can access to its data simultaneously at the given time. The memory is a double-port memory; before the execution of the algorithm, the data are uploaded by using a global access port (not given in the schematics) and read after. Recall that the Massive Marching uses the 4-neighborhood. In order to reduce the number of interconnections, we use bi-directional buses for the communication with the adjacent PUs. The bus direction is controlled by the signals t_east and t_west derived from $clk/2$.

Neighborhood retrieval

The complete neighborhood of a point (cf. Fig. 4(b)) is obtained in two clock cycles in the following way (Fig. 4(a)). With a rising edge of the clock a new SOUTH value is read from the local data memory whereas the old values SOUTH and CENTER are shifted upwards (e.g. 112, 113, 114). The values EAST and WEST are read from the bus on the falling clock edges. First, the WEST value is read from the west-side adjacent PU while the SOUTH point value is being sent to the east-side adjacent PU. On the next falling edge is read the EAST point while the CENTER point is being sent to the west-side adjacent PU. The complete neighborhood is ready immediately after the reading of the EAST point (indicated by the dashed line). The same communication protocol also applies to filtering.



(a)



(b)

Fig. 4. *Timing diagram of reading of the point neighborhood.*

Note that the choice of the image division affects only the communication procedure between the adjacent processing units and not their internal architecture.

Approximation block

Instead of computing the exact value of Eq. (5), requiring computation of a square and square root, we propose to use an approximation. The approximation allows to preserve the necessary sub-pixel precision while reducing the implementation cost. Two types of approximations have been tested, the piecewise linearization and look-up-table (LUT). The functional scheme of the tested approximation blocks is given by Fig. 5. The Eq. (5) is rewritten in the form

$$u(p) = \frac{u_x(p) + u_y(p)}{2} + g_{\text{diff}}\left(|u_x(p) - u_y(p)|\right) \quad (11)$$

which is, for the hardware implementation, approximated by either a linear approximation or a look-up-table

$$\hat{u}_{\text{Lin.Approx}}(p) = \frac{u_x(p) + u_y(p)}{2} + a_i(|u_x - u_y|) + b_i \quad (12)$$

$$\hat{u}_{\text{LUT}}(p) = \frac{u_x(p) + u_y(p)}{2} + a_i \quad (13)$$

The number of operations to obtain $\hat{u}_{\text{Lin.Approx}}(p)$ reduces to three additions, one subtraction and two multiplications and for $\hat{u}_{\text{LUT}}(p)$ only two additions and one multiplication (paid by higher memory requirements for comparable accuracy). The computation is done with a fixed-point precision. A comparative study of the implementation cost is presented below.

The implementation of the approximation is given by Fig. 5. The input signals are $|u_x - u_y|$ and $u_x + u_y$. (The terms u_x and u_y are obtained by two comparators in the MIN block (Fig. 3)). The former enters also in the Interval Search block generating the address (interval number) of the register containing the corresponding approximation constants a_i and b_i (cf. Fig. 6(a)). The SELECTION CONTROL block is testing whether the values u_x, u_y are finite. If both values are finite (pixels have already been activated) then the distance is computed by using Eq. (11). If only one of the values u_x, u_y is finite then the distance computation reduces to addition of one to the finite value: the approximation constants are replaced in order to add 1 to the minimal neighbor. Recall that both values cannot be infinite in the same time since such a point would not be activated.

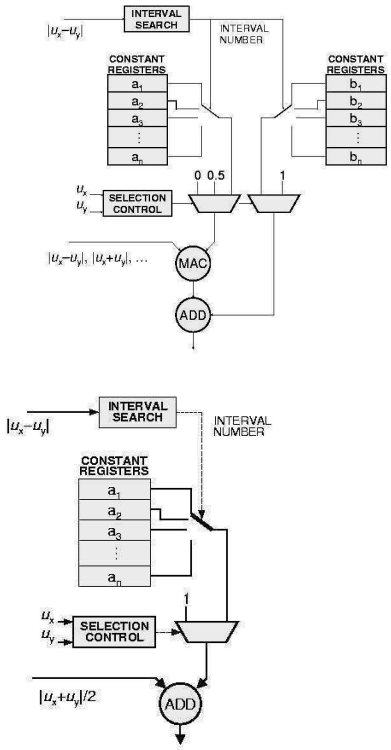


Fig. 5. *Internal block architecture of the Approximation Block; (a) Piecewise linearization, (b) Look-Up-Table.*

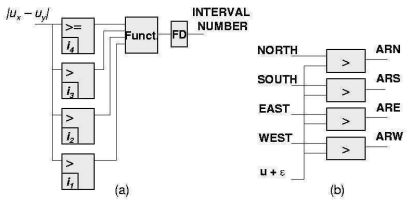


Fig. 6. *Interval Search and Activation Request block.*

The computation process is completely pipelined. After an initial latency of 10 clock periods, the result is obtained in one clock period. The approximation block as it is given here can calculate the distance in one clock cycle only for $\mathscr{F} = 1$. For $\mathscr{F} \neq 1$, the multiplier must perform two additional multiplications. The overall bandwidth of the architecture will be lower unless two additional multipliers (or approximations) are used.

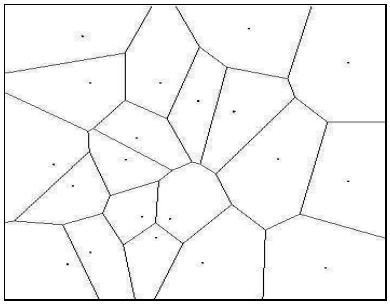
Pixel activation

Each pixel has its own flag controlling the activity status of the processing unit. It indicates whether the new value is to be computed or not. The activation flag of the point x gets active whenever the condition Eq. (8) is verified. The active points are testing their activity for the next iteration by using the condition Eq. (8) and may also activate their inactive neighbors by sending them an activation request (see Fig. 6(b) for

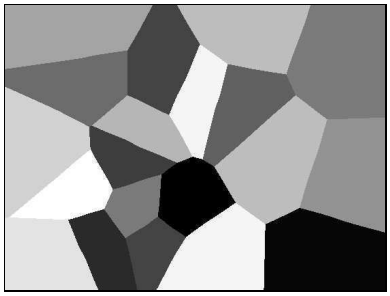
signals ARN, ARS, ARE, ARW - Activation Request to North, South, etc.). As soon as all the flags are inactive the algorithm ends.

EXPERIMENTAL RESULTS

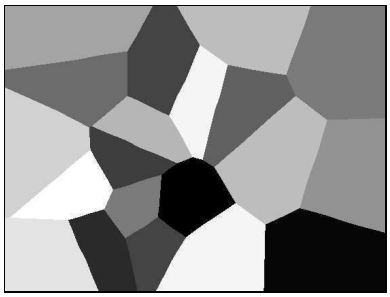
In order to prove the validity of the algorithm, we illustrate the behaviour of Massive Marching on computation of the Voronoï tesselation for a given set of points in a 2D euclidian space. In this case, we consider $\mathscr{F}(p) = 1$ for $\forall p \in \mathscr{P}$ (see Fig. 7). Note that if needed, the propagation of labels can be done simultaneously with the propagation of the distance.



(a) Exact, computed on Delaunay triangulation



(b) Massive Marching

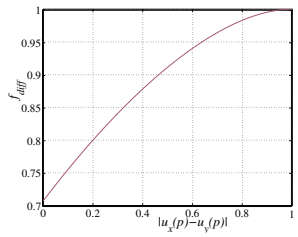


(c) Fast Marching

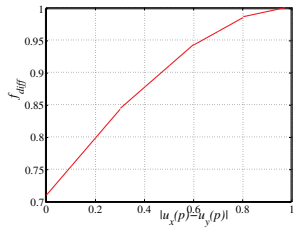
Fig. 7. *The Voronoï tessellation obtained by Massive Marching, compared to Fast Marching.*

The result achieved by Massive Marching is compared to the exact result and to the result obtained by the sorted heap algorithm (Sethian, 1996). Slight differerence is due to i) an error of the Fast Marching induced by the direction of the scanning and ii) an approximation error of Massive Marching.

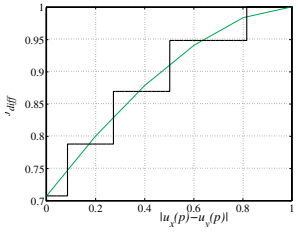
We have tested the accuracy of the result with respect to two factors: i) the approximation type of the Godunov numerical scheme and ii) the number of fractional bits of the fixed-point implementation of the approximation block. We have observed the error in ℓ_∞ in the result with respect to the exact solution simulated with "double" precision in C. Recall that the norm ℓ_∞ corresponds to the maximum of the vector elements. Here, it represents the upper bound of the error.



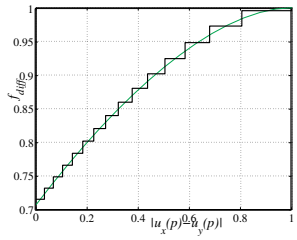
(a) Original function of the Godunov scheme



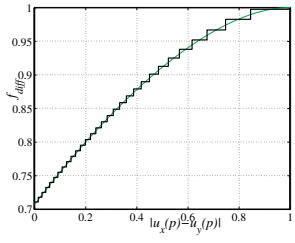
(b) Linear approximation with 4 intervals



(c) Inequally spaced LUT approximation with 5 intervals



(d) Inequally spaced LUT approximation with 15 intervals



(e) Inequally spaced LUT approximation with 30 intervals

Fig. 8. *Different approximations of* f_{diff}.

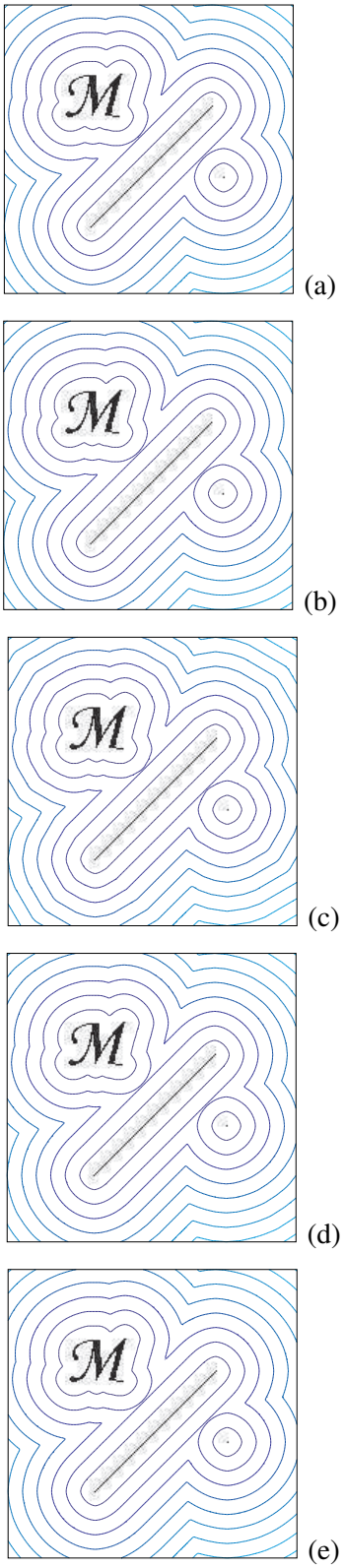


Fig. 9. *Iso-distance lines obtained with various approximations (8+8 bits of precision); (a) Original function of the Godunov scheme, (b) Linear approximation with 4 intervals, (c) LUT approximation with 5 intervals, (d) LUT approximation with 15 intervals, (e) LUT approximation with 30 intervals.*

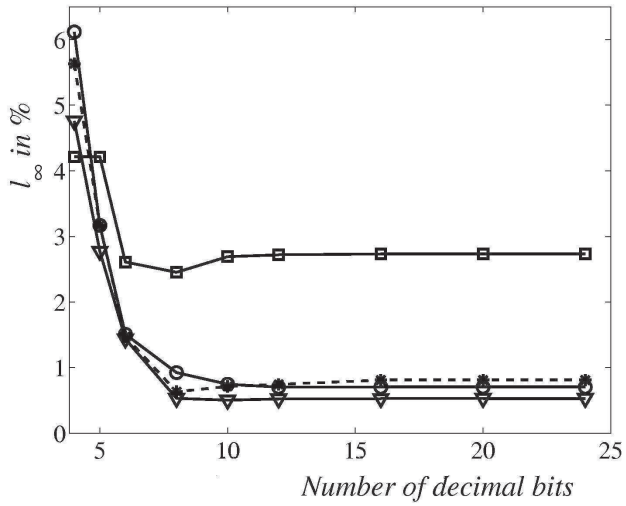


Fig. 10. *Error (in ℓ_∞) of the look-up-table approximation compared to the exact solution depending on the fractional part width. Rectangles: LUT 5 intervals, circles: LUT 15 intervals, triangles: LUT 30 intervals; asterisks: piecewise linearization.*

Two approximation types of the function g_{diff} were used: a *piecewise linearization* with four intervals and *look-up-table* approximation with five, fifteen and thirty steps, see Fig. 8. The first and the last elements in the look-up-tables are exact (equal to $1/\sqrt{2}$ and 1) in order to minimize the error in the left, right, up and down and diagonal directions. The other values are obtained as to distribute the error evenly over the entire interval $]0, 1[$. The distance results can be visually assessed in Fig. 9 on the iso-distance lines given in the same figure for 8+8 bit precision. The test image contains three sources: a letter M, straight line and a point. The approximation error (in ℓ_∞) with respect to the exact result is given in Table 1.

Table 1. *Error of approximation of the Godunov scheme given by Fig. 9.*

Approx. type (interval no.)	Linear. 4	LUT 5	LUT 15	LUT 30
Error ℓ_∞ (%)	0.3	3.9	1.2	0.9

The implementation of the approximation block (cf. Fig. 5) was tested in fixed-point precision with 8 bits for the integer part and 4, 6, 8, 10, 12, 16, 20 and 24 bits for the fractional part. Note that the eight-bit integer part limits the distance to 0 to 255 and has to be increased if needed more. The overall (approximation plus rounding) error is given by Table 2.

Table 2. *Overall (approximation plus rounding) error in ℓ_∞ of the result in Fig. 9(b, c, d and e) to the exact result Fig. 9(a).*

Approx. type (interval no.)	Linear. 4	LUT 5	LUT 15	LUT 30
Error ℓ_∞ 8 bits (%)	0.64	2.45	0.92	0.54
Error ℓ_∞ 16 bits (%)	0.81	2.74	0.70	0.53

Simultaneously with the error introduced by the approximation and rounding error, see Fig. 10, we have observed the implementation cost in terms of the number of equivalent NAND gates in the netlist, reported by the compilator, before the optimization and routing on a specific chip (only for precision 8+8 and 8+16 bit integer+fractional part). For the surface estimation cf. Tables 3 and 4 below.

Table 3. *Surface estimation of the approximation block 8+8 bits of precision (integer+fractional part).*

Approx. type (interval no.)	Piecewise lin. 4	LUT 5	LUT 15	LUT 30
Surface after optimization (NANDs)	10132	4031	6920	13856
Memory bits	128	80	240	480

Table 4. *Surface estimation of the approximation block 8+16 bits of precision (integer+fractional part).*

Approx. type (interval no.)	Piecewise lin. 4	LUT 5	LUT 15	LUT 30
Surface after optimization (NANDs)	20044	5743	9056	16592
Memory bits	192	120	360	720

Note that when using the piecewise linearization the surface requirements increase considerably (about twice of equivalent NANDs) if the accuracy increases from 8+8 to 8+16 bits (integer plus fractional part). On the other hand the surface occupation grows linearly when the look-up-table is used: increase by some 20% to 40% of equivalent NANDs and by one third of memory bits. The nonlinear increase of the implementation cost between the 8+8 and the 8+16 accuracy is due to the use of a highly optimized multiplier/accumulator.

CONCLUSIONS

This paper proposes a SIMD-type architecture for curve-evolution PDEs. This architecture has already been used for filtering by linear diffusion, see Gijbels *et al.* (1994). In this paper is shown that the same architecture type can also be used for the narrow-band like algorithms.

Obviously, the activity of Processing Units arranged in a linear array is unbalanced for narrow-band like algorithms. The activity distribution depends on the geometric form of the objects. This inconvenience is the price paid for the advantage that without major modifications this architecture can run algorithms consisting of several stages, e.g. filtering followed by watershed or voronoï tessellation computation. The only modification consists in reconfiguration of the approximation blocks by uploading new values in the look-up-tables or the linear approximation registers. The algorithm is then executed by broadcasting corresponding high level instructions to the Processing Units.

For implementation on a FPGA, the maximum clock frequency we have obtained is 150MHz. One point is processed in two clock cycles (Jacobi plus Gauss-Seidel) which gives a theoretical bandwidth of one Processing Unit 75×10^6 points^{-1}. The worst execution time estimation for the QCIF format (176 pixels wide by 144 high) with the distance source in a corner is 400 µs.

We have observed that even if implemented sequentially in some situations (for denoised filtered images) the Massive Marching outperforms algorithms using ordered structures. For heavily noised input images, the Massive Marching performance remains comparable to other algorithms despite frequent reactivations.

Future work: Extension of Massive Marching to 3D seems promising. The execution of other

algorithms on big 3D images is penalized by excessive memory requirements of large, real-number-priority ordered structures. The use of Massive Marching may be advantageous because of the absence of any ordered waiting structures.

A better activity distribution would be achieved with processing units retrieving points waiting in a FIFO-like queue. Suppose a completely pipelined, random-access capable prefetch so that the neighborhood is retrieved in one clock cycle. The theoretical execution time will be $\mathcal{O}(N)/P$ cycles, where P is the number of pipelined processing units. The surface activity will be more balanced. An efficient neighborhood prefetch therefore needs to be found so that several pipelined processing units could be used.

REFERENCES

Boué M, Dupuis P (1999). Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control. SIAM J Numer Anal 36:667–95.

Danielsson P (1980). Euclidean distance mapping. Comput Vision Graph 14:227–48.

Dejnožková E (2002). Massive marching : A parallel computation of distance function for PDE-based applications. Tech. Rep. N-17/02/MM, ENSMP, Center of Mathematical Morphology.

Dejnožková E, Dokládal P (2003a). A multiprocessor architecture for PDE-based applications. Visual Information Engineering, VIE 2003. Proceedings.

Dejnožková E, Dokládal P (2003b). A parallel algorithm for solving eikonal equation. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP. Proceedings.

Eggers H (1997). Fast parallel euclidian distance transformation in $\mathbf{Z^n}$. SPIE Proceedings 3168:33–40.

Gijbels T, Six P, Gool LV, Catthoor F, Man HD, Oosterlinck A (1994). A VLSI architecture for parallel non-linear diffusion with applications in vision. IEEE Workshop on VLSI Signal Processing .

Gomez J, Faugeras O (1992). Reconciling distance functions and level sets. Tech Rep No: 3666, INRIA.

Kimmel R (1995). Curve Evolution on Surfaces. Ph.D. thesis, Technion Israel Institute of Technology.

Kimmel R, Sethian JA (2001). Optimal algorithm for shape from shading and path planning. J Math Imaging Vis 14:237–44.

Meyer F, Maragos P (1999). Multiscale morphological segmentations based on watershed, flooding, and eikonal PDE. In: Nielsen M, Johansen P, Olsen O, Weickert J, eds., Scale-Space Theories in Computer Vision, no. 1682 in Lecture Notes in Computer Science. Springer-Verlag, 351–62.

Osher S, Sethian J (1988). Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. J Comput Phys 79:12–49.

Osher S, Shu CW (1991). High-order Essentially Non-oscillatory schemes for Hamilton-Jacobi equations. SIAM J Numer Anal 28:907–22.

Paragios N (2000). Geodesic Active Regions and Level Set Methods : Contributions and Applications in Artificial Vision. Ph.D. thesis, Université de Sophia Antipolis.

Perona P, Malik J (1988). A network for multiscale segmentation. Proceedings IEEE International Symposium Circuits and Systems CISCAC88 :2565–8.

Rumpf M, Strzodka R (2001). Level set segmentation in graphics hardware. In: Proceedings ICIP 2001.

Sapiro G (2000). Geometric Partial Differential Equations and Image Analysis. Cambridge: Cambridge University Press.

Sethian JA (1996). Level Set Methods. Cambridge: Cambridge University Press.

Sethian JA (1999). Fast marching methods. SIAM Review 41:199–235.

Siddiqi K, Kimia B, Shu CW (1997). Geometric shock-capturing ENO schemes for subpixel interpolation, computation and curve evolution. Graph Model Im Proc 59:278–301.

Suri J, Singh S, Reden L (2002). Computer vision and pattern recognition techniques for 2-D and 3-D MR cerebral cortical segmentation: A state-of-the-art review. Int J Patt Anal Ap 5:46–76.

Tsai Y (2000). Rapid and accurate computation of the distance function using grids. Tech. Rep. 17, Department of Mathematics, University of California, Los Angeles.

Verbeek P, Verwer B (1990). Shading from shape, the eikonal equation solved by grayweighted distance transform. Patt Recogn Lett 11:681–90.

Weickert J, ter Haar Romeny BM, Viergver MA (1998). Efficient and reliable schemes for nonlinear diffusion filtering. In: IEEE Trans Image Proc, vol. 7. 398–410.

Zhao H, Chan T, Merriman B, Osher S (1996). A variational level set approach to multiphase motion. J Comput Phys 127:179–95.

EURASIP Journal on Applied Signal Processing 2005:17, 1–16

Embedded Real-Time Architecture for Level-Set-Based Active Contours

Eva Dejnožková
Centre of Mathematical Morphology, School of Mines of Paris, 35 Rue Saint Honoré, 77305 Fontainebleau Cedex, France
Email: dejnozke@cmm.ensmp.fr

Petr Dokládal
Centre of Mathematical Morphology, School of Mines of Paris, 35 Rue Saint Honoré, 77305 Fontainebleau Cedex, France
Email: dokladal@cmm.ensmp.fr

Received 14 June 2004; Revised 5 April 2005; Recommended for Publication by Luciano da F. Costa

Methods described by partial differential equations have gained a considerable interest because of undoubtful advantages such as an easy mathematical description of the underlying physics phenomena, subpixel precision, isotropy, or direct extension to higher dimensions. Though their implementation within the level set framework offers other interesting advantages, their vast industrial deployment on embedded systems is slowed down by their considerable computational effort. This paper exploits the high parallelization potential of the operators from the level set framework and proposes a scalable, asynchronous, multiprocessor platform suitable for system-on-chip solutions. We concentrate on obtaining real-time execution capabilities. The performance is evaluated on a continuous watershed and an object-tracking application based on a simple gradient-based attraction force driving the active countour. The proposed architecture can be realized on commercially available FPGAs. It is built around general-purpose processor cores, and can run code developed with usual tools.

Keywords and phrases: level set, partial differential equations, object tracking, real-time execution, embedded platforms.

1. INTRODUCTION

The level set was proposed in 1988 in [1] as a simple method to modelize or analyze the motion of a travelling interface. It offers a convenient and stable framework to implement a large variety of methods where images are seen as sets of curves. Since then, its applications have been extended to other image processing fields such as the restoration (filtering or contrast enhancement), segmentation (active contours, watershed) to the form analysis (shortest path, shape-from-shading). See [2] or textbooks [3, 4] for applications and a general overview.

From the implementational point of view, the methods can be divided into two groups: (i) filtering-like methods operating on a set of constant-level curves describing the entire image and (ii) methods that act on a single (or several) contour(s), representing one (or several) object(s) present in the image. Below, we reference these algorithms according to their computation scope, the filtering-like methods as *global-scope*-type and the active contours methods as *narrowband* type.

1.1. Scope and objectives

The objective of this paper is to open the world of hand-held, mobile devices such as PDAs, still picture or movie cameras or mobile phones to powerful image processing methods from the level set framework. The novelty of this paper resides in the presentation of a reusable architecture capable to run optimally various algorithm types from the level set family. This architecture corresponds well to the system-on-chip concept, and verifies the needs of hand-held devices concerning their energy and implementational limitations. We particularly concentrate, among other aspects, on the execution on multiprocessor, parallel, scalable architectures which is an important aspect permitting to reduce the energic consumption.

The rest of this paper is organized as follow. After reviewing the state of the art of existing implementations and accelleration attempts, analyzing the family of the level-set-based algorithms (Section 2), we present the architecture (in Section 3) that best verifies the algorithmic needs and remains efficient with respect to the HW implementation issues listed above. Its efficiency is demonstrated on a contour-tracking algorithm proposed in Section 4.2. The text concludes by presenting some benchmark results and general conclusions.

1.2. State of the art and technological difficulties

The implicit representation of the travelling interface by using the level set increases the computational effort by one

order of magnitude. A faster implementation obtained by narrowbanding the computations around the travelling interface (originally called the *tube method*) was proposed by Adalsteinsson *et al.* [5] and Malladi *et al.* [6]. Despite the narrowbanding which reduces considerably the number of points to process, the level set methods remain computationally expensive, because of (i) using nonlinear functions, and (ii) a high number of iterations. The computational complexity has unpleasant consequences on both the execution time and the power consumption. If the execution time can be reduced by parallel execution (provided that the algorithm is parallelizable), the overall energy budget (following from the number of necessary operations multiplied by the energy to perform one basic operation) remains constant.

The numerous attempts to speed up the implementation of PDE-based methods made in the past were done in various axes.

(i) Algorithmic: as, for example, the implementation alternative to the level set, using spline-based modeling of the contours. Precioso and Barlaud [7] have obtained a fast execution on Pentium-based machines with spline-based active contours. A special care must be done to handle the topology changes. Cserey et al. study in [8] the implementation of linear and nonlinear diffusions on neural networks. In general, the algorithmic modifications are often applicable to only one type of algorithms.

(ii) Mathematical: proposing a faster convergence either in another space, or using another integration scheme. Weickert et al. propose in [9] the semi-implicit integration scheme, and the AOS scheme with arbitrarily large integration step for filters which can be written in a specific form as in [10]. The semi-implicit scheme increases the integration speed without affecting the numerical stability; it deteriorates only the numerical accuracy. Later, Goldenberg *et al.* [11] and Smereka [12] use a semi-implicit scheme for the active contours. However, the mathematical modifications are often applicable to only a restricted family of algorithms.

(iii) Hardware-based implementations are of three types.
  (a) *Supercomputers*: Holmgren and Wallin [13] use a self-optimizing nonuniform memory access (NUMA) supercomputer implementing a high-accuracy solver for several integration kernels. Sethian [14] has studied study flame propagation models on a CM-2 machine with 65K processors. The author reports a true massively parallel calculation with one processor per grid node.
  (b) *Graphic hardware*: Rumpf and Strzodka benefit from a high memory bandwidth and implement a nonlinear diffusion [15], and a level set segmentation [16] on a graphic card. Cates *et al.* [17] implement an active-contours-based segmentation tool on a graphic hardware to increase the interactivity when a number of parameters must be tuned to obtain a correct segmentation results. Sigg *et al.* implement a signed-distance function transform on a graphic hardware [18].
  (c) *Specific HW accelerators*: Hwang *et al.* [19] propose an orthogonal architecture designed for numerical solution of PDEs, not inevitably related to the image processing. It is built around n processing units and n^2 memory blocks. Each processor is connected to the memories by buses dedicated to only one processor, equipped with a memory access controller. The drawback of this design is that the number of interconnexions and buses increases with the square of the number of processors.
  Gijbels *et al.* [20] propose a VLSI architecture for nonlinear diffusion conceived for image improvement on image sequences. The authors use an SIMD[1] architecture with distributed memory for parallel nonlinear diffusion (i.e., the global-scope-type) used in some vision application. The estimated performances are some 100 iterations on a 256×256 image every 0.25 seconds, whereas the processing units themselves are clocked at 20 MHz.

This paper focuses on the HW-based implementation issues of the level set techniques on embedded, *one-chip devices* that will be easily (i) *scalable*, to adapt their computational power to the requirements of the chosen application, (ii) *programmable* with conventional programming tools, (iii) by far *less energy consuming* than Pentium-based desktop machines with comparable computational power, and (iv) as *small sized* as possible. The surface occupation is important because it has a direct impact on the price of both the chip itself and the embedding system (such as personal vehicles, hand-held devices, etc.). These contraints exclude both the graphic hardware and supercomputer implementations, since they do not match the objectives of one-chip devices, as well as the SIMD architecture, presented in [20], which cannot be used either because of its considerable number of used processing units (one unit per image column).

Generally speaking, it is essentially due to the algorithmic complexity that no embedded platforms have so far been proposed for the narrowband-type algorithms. The issues to handle include the following.

(i) *Nonlinear computations* employed in the integration step, *numerous iterations* necessary to obtain the convergence, and often required *floating-point accuracy* impose using fast ALUs. Their considerable surface occupation and energy consumption exclude their replication in a great number on one chip.

(ii) The *distance function* computation represents another difficulty of parallelization of the narrowband applications. Dejnožková and Dokládal [21] present a detailed analysis of existing algorithms (namely fast march-

[1] Single instruction multiple data.

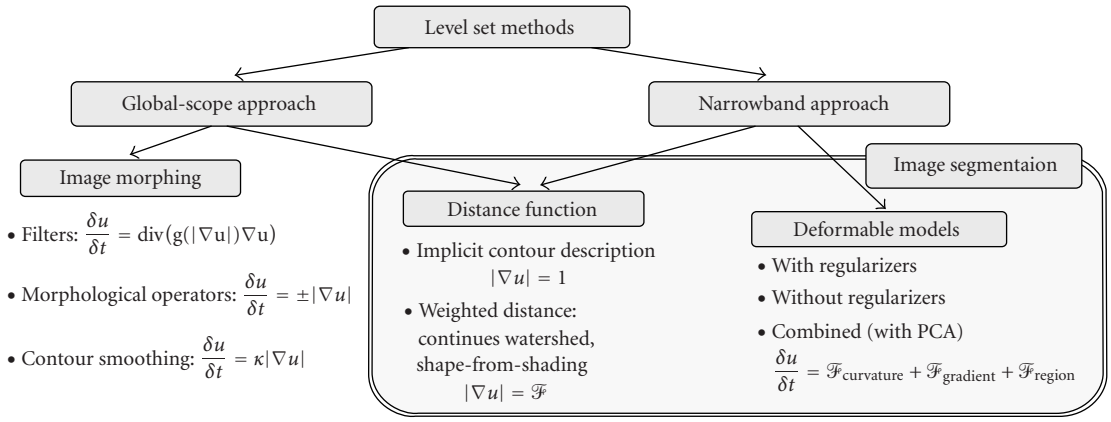


Figure 1: PDE-based algorithms overview.

ing). It shows that they are sequential and ordered (explained below). The authors propose to remove the bottleneck with the introduction of massive marching [21]. It is a fully parallel algorithm, making use of a nonequidistant propagation front.

Since we aim the entire algorithm family, whose common denominator is the level set implementation, the architecture has to be maximally flexible and scalable, and maximally using the occupied silicon surface. The recently emerging dynamic reconfiguration represents an alternative solution to the tradeoff between the functional flexibility of complex systems and the occupied surface and energy consumption, see for example [22, 23]. In some cases, the advantages of the dynamic reconfiguration may however be outmatched by the drawbacks that constitute a lenghty and difficult design, the need for special design tools [24], and external circuits controlling the chip reconfiguration.

Our study demonstrates that for the level set domain, the satisfying tradeoff between the flexibility and size can also be obtained by the programmability, offered by on-chip embedded processor cores and some DSP functions.

The following section presents the analysis of the architectural choices, including the computational resources, memory consistency model, and communication management. The resulting system has been synthesized for commercially available FPGAs.[2] Their performance becomes almost comparable to the ASICs.[3] Though the ASICs still outperform the FPGAs in the energy consumption (a key feature in mobile devices), the FPGAs remain a useful prototyping platform, and a possible intermediate development step towards an ASIC.

2. ALGORITHM ANALYSIS

This section discusses hardware implementation issues of several algorithm types from the level set context. All the types consist of two basic steps: an *initialization* step that differs according to the method used, and the *evolution* step, which makes the contour(s) travel in space and/or time according to the given partial differential equation (PDE). Usually it makes use of some local integration kernel, and is repeated until stability. In general, only the use of a local information is easily parallelizable. If the image is considered as a continuous signal, then the PDEs can be seen as an iteration of a local filter operating on the neighborhood [4].

Typically, the evolution proceeds by deforming one or several curves (propagation front) or surface with a given PDE. The PDEs methods can be classified into the following categories (cf. Figure 1).

(1) Surface propagation includes *diffusion filters* [25], [26], or [27] for a more comprehensive survey, *geometric smoothing* [10, 28, 29], *denoising*, and *morphological operators* [30], [31], [32] or [33] characterized by the evolution equation $\partial u/\partial t = \mathcal{F}(u)|\nabla u|$, where u represents the evolving image. The input image represents the initial conditions u_0. All points in the image are processed in every iteration. The temporal evolution is based on the local neighborhood and generates the evolution of the level sets in the space [4]. The evolution stops as soon the convergence or the given iteration number is reached.

(2) Wave propagation includes algorithms of *weighted distance*, *continuous watershed* [34], *Voronoi tesselations* [35], or *shape-from-shading* [36] that are controlled by the Eikonal equation $|\nabla u| = \mathcal{F}$. This steady-state solution is propagated from the given sources (that may be obtained from the initial image by other means) on the entire image according to the defined speed $\mathcal{F}$. The algorithm operates locally, only on the narrowband of the evolving front. The solution is propagated in waves equidistant to the sources by using ordered data structures. This technique is being referred to as marching methods, proposed by Sethian [37], as a special case of the Dijkstra shortest-path algorithm.

(3) Deformable models. An important breakthrough in the deformable models represents the introduction of

[2] Field-programmable gate array.
[3] Application-specific integrated circuit.

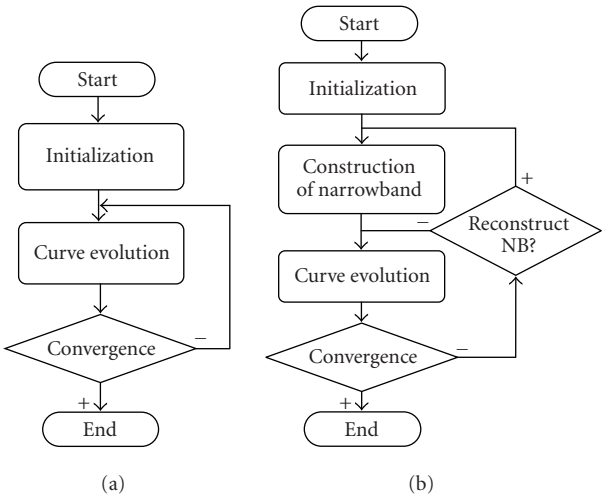


FIGURE 2: Different stages of the level set family algorithms. (a) *Global-scope*-type algorithms. (b) *Narrowband*-type algorithms.

active contours (or snakes) proposed by Kass *et al.* [38] in 1987, and deformable surfaces by Terzopoulos *et al.* [39] one year later. Another early example of deformable models represents the ballons by Cohen [40]. Implicit representation of the interface as a constant level set of another function was studied simultaneously and independently in 1993 by Caselles *et al.* [41] and Malladi *et al.* [42], and later by Malladi *et al.* [43, 44, 45]. The geodesic active contours were proposed meanwhile in [46, 47]. Another model was proposed later in [48].

We distinguish the types *with regularizers* (controlled by statistical information of regions), *without regularizers* [27], or *combined* with other techniques (e.g., principal component analysis [49]). The evolution equation writes in the form $\partial u/\partial t = \mathcal{F}_{curvature}(u) + \mathcal{F}_{grad}(u) + \mathcal{F}_{region}(u)$. The algorithms proceed by deforming a given initial contour (given by $u_0 = 0$). The deformation is controlled by internal and external forces obtained at each iteration from (i) the contour itself and (ii) the geometrical (curvature, gradient) or statistical characteristics (mean value of the region intensity) found in the image [4].

(4) Optical flow is controlled by the equations $\partial u/\partial t = f(\nabla u, I_1) + g((\partial I_2/\partial x), h)$, $\partial v/\partial t = f(\nabla v, I_1) + g((\partial I_2/\partial y), h)$. The motion vector is obtained by solving some system of the above-given equations at each point in the image (I_1, I_2 are the successive sequence images, h is the searched motion vector field) [50].

Since the nature of the optical flow algorithms differs from the temporal curve evolution principle of the three first groups, the proposed architecture does not address this type of algorithms. On the other hand, the optical flow often serves as a support for the three other types.

All the computation steps of the first three categories can be unified in two following iteration types, see Figure 2. (1) *Global-scope iteration type* includes the surface evolution. It operates *sequentially* on the entire image. (2) *Narrowband iteration type* includes the wave propagation and deformable models (curve evolution).

Indeed, applying narrowbanding to the curve evolution algorithms changes the computational aspects. The points to recalculate in every iteration are now taken from some subset of the image. This set is commonly called narrowband, and contains points situated closely (up to some chosen distance) to the current position of the travelling interface. Two types of operations are commonly applied on the narrowband: (i) the curve motion scheme itself, and (ii) the (re-)construction of the narrowband. The (re-)construction differs substantially from the other algorithm types. Indeed, all HW implementations of the active contours, cited in Section 1, use fast marching; a progressive, equidistant construction of the distance function. Fast marching itself belongs to the *wave propagation* algorithm group. It requires ordered data structures based on the priority of points [3]. From the algorithmical point of view, the ordering introduces a great data dependency, reducing the parallelization potential. From the HW implementation point of view, algorithmic ordering of the points to process introduces *random* accessing to the memory.

Parallelize the wave propagation is a tough issue, calling attention of many researches for a long time, compare a survey by Roerdink and Meijster in [51]. The recent introduction of massive marching opens the possibility to parallelize also the computation of the distance function (cf. [52] or [21]). Massive marching is similar to the fast marching method (cf. [53] or [54]) and uses the same entropy-satisfying upwind scheme. It differs from fast marching by the fact that it eliminates its sorted propagation of the solution and makes the implementation fully parallelizable, with a small grain and low data dependency.

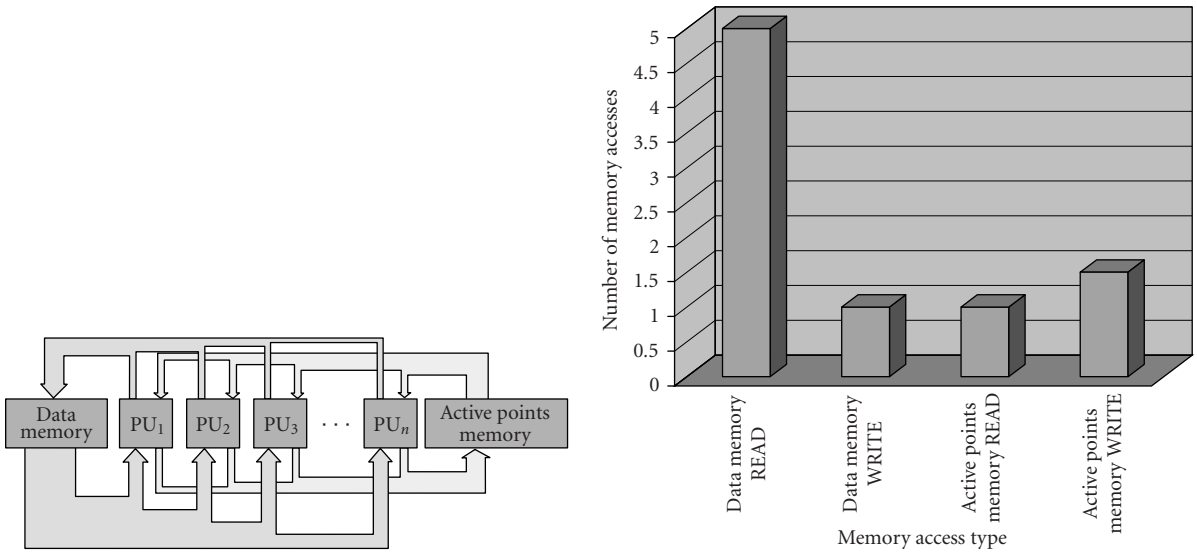
For completeness, we mention another parallelization strategy, called group marching, developed by Kim in [55]. Group marching identifies on the front groups of points that are processed parallely in the same time. It requires nonetheless to maintain a global variable making a truly parallel implementation difficult.

The next section analyzes the execution of the different algorithm steps by considering the use of massive marching for the narrowband construction. Note that the curve evolution (both algorithm types) as well as the narrowband construction (narrowband type) are time-critical. Many iterations may be needed to obtain the convergence and the narrowband has to be reconstructed repetitively during the evolution to preserve the required properties of the implicit curve description.

2.1. Data-flow analysis of different algorithm steps

In the following, we assume that, except the methods where regularizers[4] are used, the new values that the points re-

[4] Statistical information, like colour for example, represents global variables.



(a) (b)

FIGURE 3: Curve evolution implemented by using several processing units operating in parallel: (a) data-flow chart, and (b) corresponding memory accesses.

ceive are results of operations only on local neighborhood.

Limiting the calculations to a narrowband around the travelling contour corresponds formally to operating on sparse matrices. Practically, in order to obtain the correct evolution, all active points need to be recalculated in one iteration, before the next iteration starts. Therefore, unless one uses one processing unit per point, the values u^{n+1} need to be stored separately from u^n, until all the points are updated. If this condition is verified, then the processing order is not important, and both the iteration types can be unified under the following form. The set $\mathcal{A}$, respectively, represents either the entire image or the narrowband set:

for all $p_i \in \mathcal{A}$ do *(in parallel)*
{ *Retrieve_Neighborhood* $u^n(N(p_i))$ and $u^n(p_i)$;
Calculate_Value $u^{n+1}(p_i)$;
Update_Value $u^{n+1}(p_i)$;
Activate_New_Points (insertion in $\mathcal{A}$); }

Since our constraints exclude the massive parallelism (for production cost's reasons), we adopt a semiparallel approach instead. The data-flow chart corresponding to a semiparallel execution of this code on several processing units is given by Figure 3a. The data u are stored in the data memory block (two pages for u^n and u^{n+1}). The active points memory block stores the set $\mathcal{A}$, that is, the coordinates of the points to process (not used for the global scope-type algorithms). The active points are read and processed by several independently operating processing units. Both the memory blocks are organized in two pages, for the present one and the next iteration.

The width of the paths corresponds to the volumes of transferred data. The most intensive data traffic is on the shared blocks. The READ data memory flow is five times larger than the WRITE data memory flow because the complete four-neighborhood is read to update the central value (cf. Figure 3b). Similarly, since one processed point may activate several of its neighbors, the mean WRITE active points memory flow is slightly higher than READ active points memory.

The narrowbanding of active contours techniques impose random memory access to the data memory block. These aspects will be taken into account in Section 3.

2.2. *Timing analysis*

To optimize the data flow, limit simultaneous accesses to the shared blocks, and obtain a balanced activity of all the used blocks, it is necessary to consider also the timing of the algorithm execution.

The global-scope type operates on the entire image, that is, each point in the image is active and the set $\mathcal{A} = \text{supp}(I)$, $I =$ image. The narrowband type operates on $\mathcal{A} = \{p \mid |\text{dist}(p)| < NB_{\text{width}}/2\}$, where NB_{width} is the width of the narrowband around the contour. For massive marching, the definition of $\mathcal{A}$ slightly differs (see [21]).

This code has two major features.

(1) The retrieval of the point's and its neighbors' values $u^n(p_i)$ and $u^n(N_4(p_i))$ requires five memory readings and is usually faster than the following calculation of $u^{n+1}(p_i)$, which usually involves nonlinear functions. During the calculation of $u^{n+1}(p_i)$, the memory block is idle.

(2) The execution of same parts of the code can have different length due to *IF*-conditions and various input values.

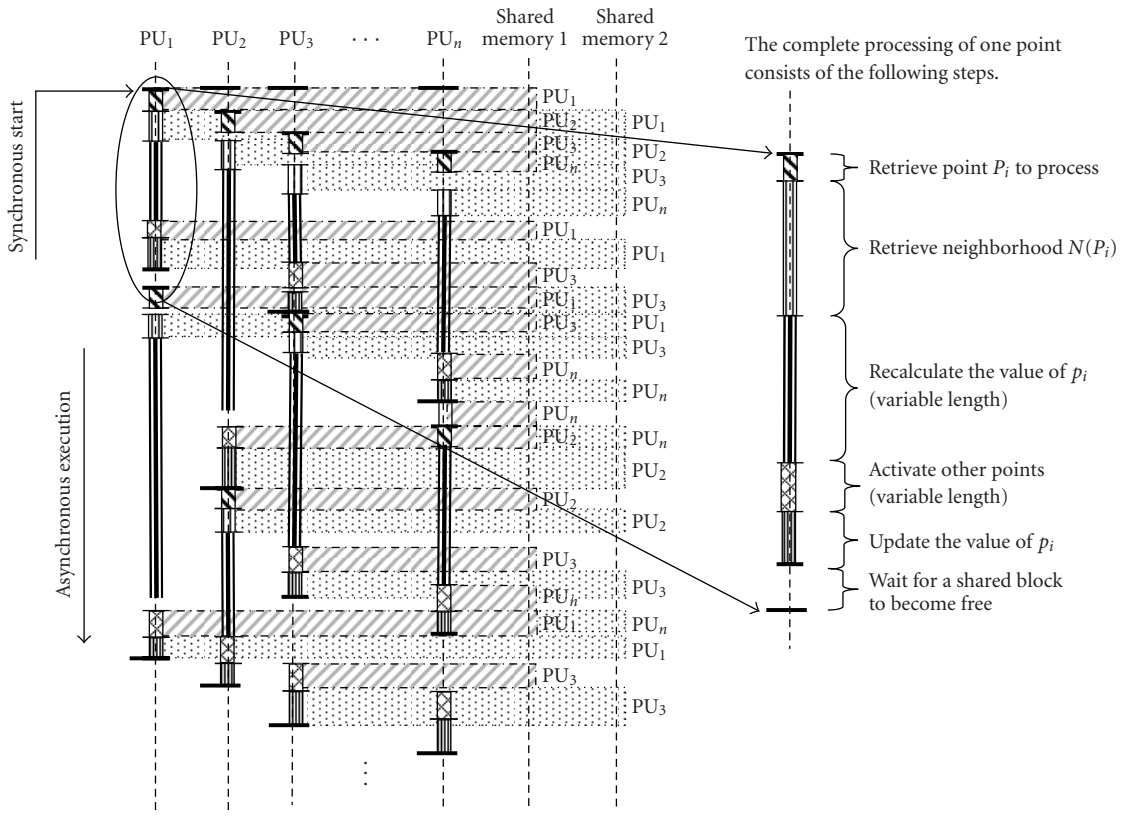


FIGURE 4: Asynchronous execution of the code on four processing units (PU_1 to PU_4) and accesses (in grey) to a shared block.

Whenever the memory is idle, it can (and should) be used to retrieve other data to process. The fact that the algorithms operate locally (using only the information from the neighborhood) characterizes this algorithm family by a fine granularity. On the other hand, the nonlinear functions categorize these algorithms rather into the medium granularity group because, in most cases, a fully functional ALU is necessary to implement the computation. These considerations impose the choice of an MIMD architecture. The processors operate in the SPMD[5] mode, corresponding best to the data flow diagram given by Figure 3.

After a synchronous start of all the processing units, the variable length of some portions of the code gives birth to an asynchronous execution, (cf. Figure 4). The asynchronous execution is advantageous for parallelizable algorithms with numerous *IF*-conditions, because it randomizes the access to the shared blocks. Simultaneous accesses become rare and their HW management is easier. After the analysis of various algorithms, it becomes clear that the choice of asynchronous execution of the code on several PUs is a natural choice for the level family.

Note, that despite the asynchronous execution, the PDE-based algorithms have one or more synchronization points: the end of one iteration. This is indicated by either (i) emptiness of one of the active points memory pages (narrowband-type algorithms), or (ii) end of the raster scan of the image (global scope-type algorithms). The end of the algorithm is indicated by either (i) emptiness of both active points memory pages (for the narrowband-type algorithms), or (ii) the number of necessary iterations (both algorithm types), or (iii) the convergence (both algorithm types).

3. ARCHITECTURE

The image processing domain is known for various algorithm granularity and data dependency. Indeed, the data dependency and granularity are two factors that have major influence on the choice of parallel implementations. Historically, the fundamental model of parallel architectures has been introduced by Flynn [56]. The further effort has been concentrated, besides the computation resources, on the efficiency of the communication configurations (see Cypher and Sanz [57]).

The massive parallelism is efficient for regular algorithms with fine granularity (cf. Gibbons and Rytter [58], Broggi *et al.* [59] or artificial retine by Manzanera [60]). On the other hand, if it used for random memory access implementations, the chip activity versus occupied surface will become poor. The same arguments are valid in the case of SIMD-type architectures (see Cypher and Sanz [57] or e.g., survey in [61]).

[5]Single program multiple data—the processors execute asynchronously the same program.

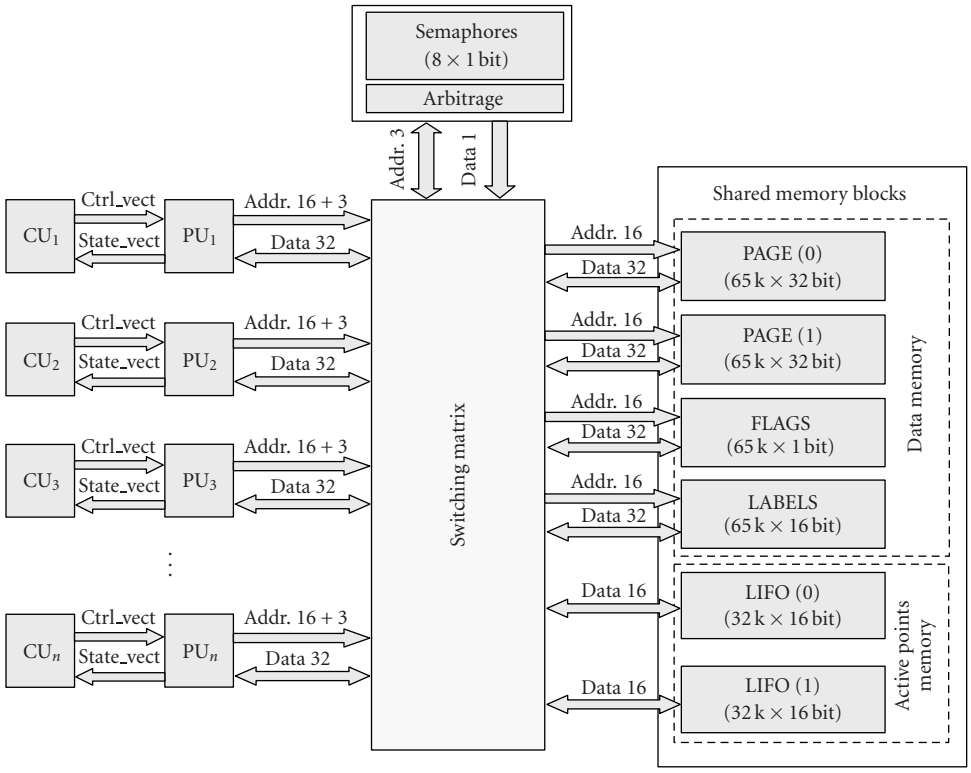


FIGURE 5: Global overview of the architecture.

Hence, the image analysis community started to consider, as a possible execution platform for mean and high granularity algorithms, in the late 1980s, programmable, multiprocessor, one-chip architectures. See for example [62], [63] or the survey of multiprocessor architectures with shared and distributed memory [64]. For another example and additional references, see a motion estimation on a set of video signal processors by De Greef *et al.* [65], or watershed segmentation in Moga *et al.* [66], Noguet [67], or Bieniek [68].

The data flow of the active contours, analyzed in the previous section, makes them correspond better to "weaker" parallelism models where the design effort concentrates on the task and data dependency decomposition, task scheduling, and efficient management of accessing to shared resources. The architecture template, presented in this section by Figure 5, is derived from the data-flow analysis given by Figure 3. In the following, we detail the description of the individual blocks.

Processing and control units

The computation of the propagation speed $\mathcal{F}$, which is generally a nonlinear function, is a challenge for an efficient implementation. It seems necessary to use a fully functional arithmetic logic unit (ALU).

The processing units were realized in VHDL and HandelC as a model of a RISC processor. They are equipped with a set of registers. The used data word width is 32 bits to store a fixed-point data (24 + 8 the integer and fractional part).

Every processing unit is controlled by a control unit (CU). The execution of the algorithm was simulated by coding the algorithm in HandelC. The advantage of this approach is that the functional model can be replaced by another processor model or by an embedded core available on some FPGAs.

Switching matrix

The medium granularity combined with intensive random accesses to the data memory shows that no optimum *fixed* interconnection network can be found for the level set algorithm family. Rather than using a fixed network, one can use a switching matrix which, coupled with semaphores and arbitrage, permits to any processing unit access to any shared block, provided that it is not currently being used by another processing unit. Several PUs can access simultaneously to different shared blocks.

The address buses are 16 + 3 bits (16 bits for the data addressing and 3 bits to address eight semaphores for the following eight shared blocks), four data memory blocks (data PAGE(0), PAGE(1), FLAGS, and LABELS). The active points memory is divided into two blocks, each equipped with a bidirectional reading/writing channel since each of the stacks is in one iteration either read or written but not both.

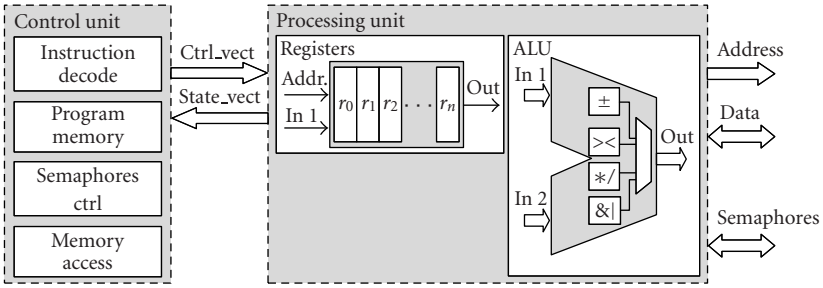


FIGURE 6: Internal architecture of the processing and control units.

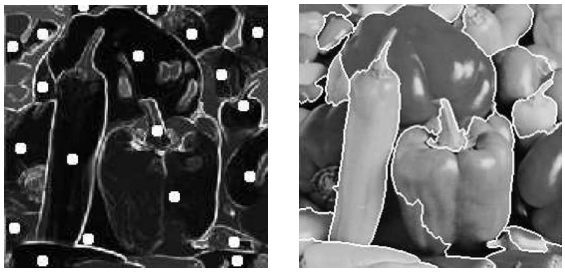


FIGURE 7: The "peppers" image: (a) gradient and manually placed markers; (b) continuous watershed obtained with massive marching on four processing units.

Semaphores and arbitrage

Every operation asking to access to a shared block uses the semaphores (block semaphores ctrl at Figure 6). The code that performs the semaphore-controlled access must respect the following:

```
loop :
test semaphore x          // test and lock immediately if free
if x is not free jump loop // repeat otherwise
read/write                // access to the memory
release semaphore x       // release the semaphore
```

Whenever a semaphore is tested, it is (by the same instruction) immediately locked, provided that it was free. If not, the test is repeated as long as the semaphore can be allocated to the asking processor. After the reading/writing, the semaphore is released. The semaphores are invisible to the user provided that the compiler generates the corresponding code.

Whenever a simultaneous access to a shared block occurs, an arbitrage is used to prevent conflicts. The arbitrage is a standard block that makes part of most modern multi-processor platforms. The ideal arbitrage, usually done on the *first-come-first-served* basis, and often realized as a finite state machine, is quite costly in terms of the silicium surface. We can benefit from the randomness of the asynchronous execution, limiting the likelihood of simultaneous accesses, and saving the space by using a simple arbitrage assigning the processors an uneven priority. Obviously, this is only possible up to a certain number of processors however. In this paper, we have evaluated the feasibility by measuring the activity up to four processors (see Figure 8 showing the activity distribution).

Data memory

A low data dependency that characterizes the level set family algorithms permits to use a simple global shared memory management, being referred to in the literature as *weak consistency* model, introduced in [69]. The weak consistency is characterized by three conditions (cf. [70]). (i) Before a READ or WRITE access for any processor is allowed, all synchronizations must be achieved. (ii) Before a synchronization access is allowed, all previous READ or WRITE accesses must be achieved. (iii) Synchronization accesses are sequentially consistent with respect to each other. Note that no condition concerns the order in which the accesses are performed. See [70] for details and comparison with other consistency models.

The synchronization points are imposed by the iterative nature of the algorithms. All active points must be processed (in arbitrary order) in one iteration, before the following iteration can start. This is ensured on this architecture by the fact that the data to process are read from one memory page, and the results are written to the other. As soon as all the points in one iteration are processed (all READ and WRITE accesses are achieved), the roles of the pages PAGEs(i), i = 0,1, switch. Switching the roles of the memory pages represents the synchronization.

This architecture is conceived as scalable. According to the computational power required by a given application, one can use more or fewer processing units. It follows from Figure 3 that the highest data traffic concentrates on the shared memory blocks. Thanks to the nature of the code, the reading and writing directions on both data and active points memory blocks are separated into two one-directional channels. The results of the previous iteration (values u^{n-1}) are read from one page and the new values (u^n) are written to the other. This corresponds perfectly to the weak consistency model.

Active points memory

The READ and WRITE accesses to perform on the image data are controlled by data stored in the active points memory. It is organized in two pages. One page contains points

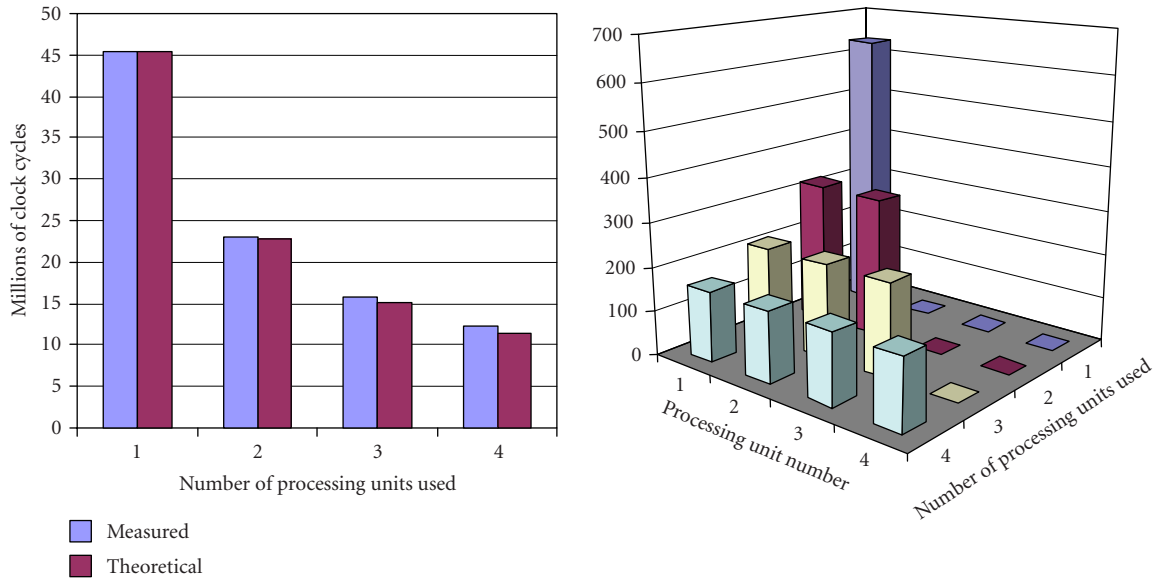


FIGURE 8: (a) The execution time of the algorithm in function of the number of parallely working processing units. (b) The activity load distributed over several processing units, number of points processed by every processing unit (X thousands).

to process in the current iteration. This page is progressively emptied as the data are processed. The second page contains points to process in the next iteration. It is progressively fed with data. The emptiness of one page represents the synchronization point. The roles of the pages (for both data and active points memory) switch. The emptiness of both active points pages represents the end of the algorithm.

The reading/writing direction to the data and active points memory blocks is controlled by using a boolean variable *switch* which commutes at the end of every iteration. For the sake of universality, it is left to the programmer's responsibility to control the reading.

Thanks to the fact that the processing order is indifferent, this memory can be implemented by using two LIFOs. Compared to a FIFO, using LIFO eliminates the transport delay.

For most applications, the reading should always be done on data page(*switch*) and LIFO(*switch*) and writing on page($\overline{switch}$) and LIFO($\overline{switch}$). The binary *switch* value can be derived from the zero bit of the iteration number n.

Flags

The labels and flags are similar to the data memory with a smaller word size. The labels and flags are available to the programmer for an additional algorithm control and region propagation.

4. PERFORMANCE EVALUATION

The performance of this architecture has been tested by running two different types of PDE-based algorithms: a continuous watershed and an object-tracking application.

The objective of the watershed computation is to justify the choice to use an MIMD architecture by testing whether the overall computational effort is uniformly distributed over all the processors used. The objective of the tracking application (cf. Section 4.2), is to evaluate the overall bandwidth of the architecture, and the capability to run a computationally expensive application in real time.

4.1. Evaluation test 1: A continuous watershed implementation

Recall that, in terms of PDEs, watersheds can be obtained by calculating a weighted distance function to a given set of sources, corresponding to the markers [34], while propagating simultaneously the labels

$$\|\nabla u(x, y)\| = \frac{1}{\|\nabla I\|}. \tag{1}$$

Recall that the set of sources must be identical with the set of local minima in the image, as shown in [71]. The distance function was computed in a semiparallel way, on four parallely operating processing units, from a manually placed set of markers, see Figure 7.

Figure 8b shows the execution time (in terms of total clock cycles against the number N of processing units operating in parallel). The obtained number of clock cycles corresponds to the theoretical number of clock cycles calculated as $\text{clk}_N = \text{clk}_1/N$. The measured execution time (expressed in terms of clock cycles) slightly exceeds the theoretical value because of the access to the shared blocks (memory, LIFO), controlled by a semaphore. Figure **??** gives the computational load distributed over the processing units in function of the number of processing units used. The computational load is expressed in terms of number of points processed by every processing unit. If only one unit is used, the total computational load is covered by this unit. If more processing units operate in parallel, the load is uniformly distributed.

Table 1: The obtained bandwidth for watershed computation versus other platforms.

Platform	Frequency	Bandwidth (10^3points/s)
Proposed MIMD architecture Four RISC processors	120 MHz FPGA	2 610
PC with P4 Win 2000	1.6 GHz	827
IPAQ with Xscale WinCE	400 MHz	120
IPAQ with Strong ARM WinCE	200 MHz	50

Table 1 compares the bandwidth of weighted distance computation with simultaneous propagation of source labels, obtained by using massive marching implemented on various platforms. The bandwidth is computed as the number of points in the image divided by the execution time. The execution time of the proposed MIMD architecture was obtained by counting the clock cycles during simulation (HandelC code). The execution time obtained on a PC/P4, IPAQ/Xscale and StrongARM corresponds to the processor time spent in the process (programmed in C).

Note that the bandwidth of every given architecture is somewhat lesser than the theoretical bandwidth because some points are activated several times. The computation complexity of massive marching is roughly $\mathcal{O}(N)$, with N being the number of points in the image. It exceeds N by the number of reactivated points because of using a nonequidistant propagation front.

4.2. Evaluation test 2: Object-tracking application

To test the performance of this architecture, we use a model-free, gradient-based object-tracking algorithm proposed in [72].

4.2.1. A gradient-based attraction field

Consider an image $\mathcal{I}$ and some gradient of $\mathcal{I}$, $g = \nabla \mathcal{I}$. Let

$$g_K = g * K, \tag{2}$$

where K is some triangular window $\mathbb{Z}^2 \to \mathbb{R}^+$, such that

$$K(x,y) = \begin{cases} 1 - \alpha\left(x^2+y^2\right)^{1/2} & \text{if } \left(x^2+y^2\right)^{1/2} < \dfrac{1}{\alpha}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Note that in the signal processing domain, convoluting with such a window is a frequency filter. However, filtering is not the objective here.

∇g_K represents a gradient-dependent integrator with interesting properties. Generally, the evolution of a curve $\mathcal{C}$ writes

$$\frac{\partial \mathcal{C}}{\partial t} = \mathcal{F}\vec{n}, \tag{4}$$

where $\vec{n}$ is the normal vector to $\mathcal{C}$, and $\mathcal{F}$ represents the motion speed. For the contour-based tracking, we propose

$$\mathcal{F} = \nabla g_K. \tag{5}$$

It can be shown (by approximating g in (2) by a Dirac impulse δ, and computing $\mathcal{F}$ in (5) in a discrete form) that ∇g_K is a bidirectional integrator pointing towards the crest of the gradient g from both sides.

The advantage of using a bidirectional integrator is twofold: (i) it allows the contour to converge towards the gradient maximum from both sides, and (ii) it eliminates the necessity to use a constant one-directional attraction force there, where the data is zero. This fact eliminates the problem of local breaches in the gradient, often introducing leakage in object reconstruction. Attempts to alleviate this problem were made in [73] introducing a viscous watershed capable to slow down the propagation in such narrow openings. Although the leakage could probably be alleviated by using curvature, the leakage problem does not occur when using ∇g_K, since on zero gradient the contour does not move.

Let ϕ represent some feature of the object to track. Supposing that this feature is unstable in time, or perturbed by external phenomena, one may need to employ an additional cue to enhance the stability. Natural gesture speed is one of the possible cues to track individuals. This fact is also used in defining the capture range of the contours. Suppose that the maximum interframe displacement of the object is bounded by D. This information should be taken into account by letting $\operatorname{supp}\{(x,y) \mid K(x,y) > 0\}$ be a circle of radius D, generating a nonzero attraction field in a narrow zone around the contour. Hence, a convenient value of α in (3) is $a = 1/D$.

Indeed, as the attraction force stops on the zero crossing of the gradient, its principle is similar to the Haralick [74] edge detector, which detects edges on zero crossing of the second derivative of $\mathcal{I}$ in the gradient direction. Kimmel and Bruckstein in [75] reformulate the Haralick edge detector in terms of the level set framework and shows how it can be combined with additive constraints to segment images. As stated before, our objective is the contour-based object tracking. Whereas various motion predictors can be used to predict the displacement direction according to the past, arbitrary deformations of the object give birth to a displacement field with locally varying direction. Any contour-based tracking must therefore be able to handle both partially forward and backward displacements of the contour. A good overview of other existing attraction vector fields can be found in [76].

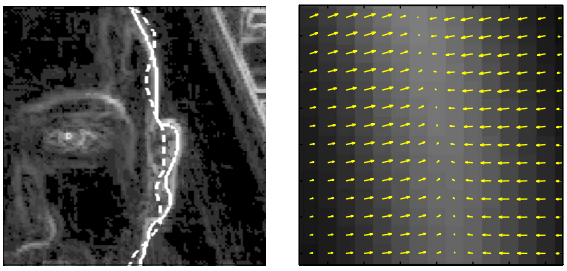


Figure 9: (a) The initial (dashed) and final (solid line) position of the contour, and (b) zoom on the attraction force field F.

4.2.2. Application

By integrating (4), the current contour C^n of the object is obtained by using the attraction field g_K^n generated by the current frame $\mathcal{I}^n$, and the contour C^{n-1} in the previous frame (cf. Figure 9):

$$C^n = \lim_{T\to\infty} \int_0^T \nabla g_K^n(C)\vec{n}\, dt + C^{n-1}, \tag{6}$$

$$\text{with } C(t=0) = C^{n-1}, \tag{7}$$

where $g_K^n = g * K$,

$$g(p) = \frac{\nabla_{\text{Lab}} I(p)}{1 + d_{\Omega|\phi}(I(p))}. \tag{8}$$

The ∇_{Lab} denotes the gradient on the Lab colour space. The particularity of the Lab space is that it is perceptually uniform, and ∇_{Lab} is locally Euclidean. The $d_{\Omega|\phi}$ denotes the distance to a given feature. We use a feature based on the skin chroma. We take $\Omega' \equiv$ HLS, and $\phi = \{x \in \text{HLS} | x_H \in [-20^o, 50^o]\}$. This feature is only related to hue, thus the distance $d_{\text{HLS}|\phi}$ is the angular distance d^α to the skin chroma ϕ. The size of the triangular window K is ten pixels, that is, $\alpha = 0.1$, calculated from a natural gesture speed as seen by our camera.

Initialization

The description of the initialization of the tracking is outside the scope of this paper. It can be successfully done by combining several features, see for example [77], using the face colour and shape or [78] combining the colour and motion (in a car application, no perturbing motion is present in the background before the car runs).

4.2.3. Implementation

In the following, we outline the details concerning the implementation of the object tracking on the proposed architecture.

This architecture has been simulated using the HandelC programming language. The control units have been replaced by a pipelined model controlling each processing unit, equipped with a fully functional ALU realizing the basic arithmetic/logic operations in fixed-point precision, and equipped with a set of registers. The algorithms have been hardcoded in the control units in HandelC instructions. Note that every HandelC instruction is executed in one clock cycle.

Table 2: Frame parameters.

Frame size (X × Y)	324 × 428
Number of points in the frame	138 672
Frames per second	15
Data flow (points per second)	2 080 080

Application parameters

The video stream contains 15 frames per second, each 324 × 428 pixels, giving total data flow $2.08 \cdot 10^6$ pixels per second (cf. Table 2).

The narrowband width has been set to 20 points (ten to each side of the contour) and the mean length of the contour of the face (cf. Figure 10) to track is approximately 600 points, giving in average 12 000 active points to update per iteration, see Table 3.

The above given face tracking application requires 25 iterations in every frame for the contour to adapt itself to the new position of the face. (We consider that natural gesture speed, camera resolution, and distance to the face limit the interframe displacement of the drivers face to approximately 10 pixels.) Every five iterations, the narrowband needs to be reinitialized (cf. Table 4).

Instruction count for various algorithm steps

The construction of the attraction force field requires one convolution (cf. (2)). An $N \times N$ fast 2D convolution can be efficiently implemented by a serie of $2N$ 1D FFT applied to the columns and rows, N^2 multiplications, and a series of $2N$ 1D IFFT. Efficient algorithms exist to perform FFT/IFFT in place, see for example [79], and modern DSPs are equipped with efficient, highly optimized blocks calculating fast the FFT, for example [80].

We suppose that the convolution is computed on a companion chip. In the following, we focus on the implementation of the level-set-based part of the application, that is, the (i) initialization and construction of the narrowband, (ii) contour evolution.

The gradient can be calculated with two additions and two divisions (if central differences are used). The attraction force ∇g_K calculated on the entire frame requires 277, 344 additions and as many multiplications (cf. Table 5). The construction of the narrowband, by using massive marching, requires two steps: (i) the *interpolation* to initialize the contour can be done with 4 additions per point and (ii) the *propagation* of the distance function requires 5 additions and 6 multiplications per point. Performed twice (Jacobi and Gauss-Seidel steps) on 12 000 points (narrowband size from Table 3) gives 216 000 additions and 144 000 multiplication required to construct the narrowband. The narrowband is reconstructed five times per frame, giving the level set inherent computational effort of 1 080 000 additions and 720 000 multiplications per image frame.

Figure 10: Contour tracking applied to driver's face extraction, using the weighted gradient (skin chroma being the feature of interest). Randomly chosen images from a video sequence.

Table 3: Narrowband parameters.

Narrow bandwidth (points)	20
Approximate mean contour length (points)	600
Number of points in the narrowband	12 000

Table 4: Object-tracking application parameters.

Number of iterations before reinitialization	5
Reinitializations per frame	5
Number of iterations per frame	25

The actual curve evolution involves several steps: (i) the evolution speed $\mathcal{F}$ requires 3 addition and 4 multiplications (including the gradient of the distance function U), (ii) the integration is done in one additions and one multiplication, giving in total 4 additions and 5 multiplications per point. Multiplied by 12 000 points in the narrowband (48 000 additions and 60 000 multiplications) and by 25 iterations per frame gives $1.2 \cdot 10^6$ additions and $1.5 \cdot 10^6$ multiplications per frame. The total application effort is $2.56 \cdot 10^6$ additions and $2.50 \cdot 10^6$ multiplications per frame, representing in total 75.8 MFLOPS to run in real time.

Table 6 presents the lower limits of the bandwidth obtained for different steps of the object-tracking application. The computation of the gradients ∇g_K and ∇u requires the same elementary operations (differences and extrema computation on the neighborhood), and presents obviously the same bandwidth $19.3 \cdot 10^6$. The limiting factor in this case is the neighborhood extraction from the input image. We have obtained the same bandwidth estimation for the integration step. The integration does not read the neighborhood (already stored in the registers) but only writes the integration result. Its performance can sometimes be limited by the bandwidth of the foregoing step.

The bandwidth $2.61 \cdot 10^6$ points/s, obtained for the narrowband construction, includes the detection of the initial contour position by interpolation and the propagation of the distance function.

We evaluate the performance of the architecture by computing the processing time of the each algorithm stage as a function of the number of processed points and these measured worst-case bandwidths. The processing time of all the steps is obtained by multiplying the worst-case bandwidth, the number of iterations, and the number of the points to process.

The sum of the processing times of individual steps gives the frame-to-frame processing time $6.18 \cdot 10^{-2}$ seconds, corresponding to 16.3 processed frames per second.

The performance, outlined in Table 7, compares the execution time of one iteration of the above-detailed object-tracking application on this architecture compared to similar results obtained on other platforms reported in the literature.

The nVIDIA GeForce2 graphic card, see [16], operates in integer accuracy, and is therefore less useful for algorithms requiring multiple iterations. The application running on PC P4, see [81], was implemented by using the additive operator splitting (AOS) scheme, permitting greater integration step, and requiring thus fewer iterations.

4.3. Power assessment

As the silicium surface on FPGAs continues to grow (to become comparable to ASICs), the computational power is no longer a limiting factor for the design. Instead, the preoccupations concern more and more the energy dissipation and the system autonomy.

The energy budget of some algorithm can be characterized by the energy necessary to execute the elementary operation multiplied by the number this operation is executed. Suppose that this algorithm is to be executed in a limited time. A parallel execution (provided that the algorithm is parallelizable) will allow to reduce the clock frequency (compared to the clock frequency of the sequential implementation) and reduce the energy budget of the elementary operation.

Though it is important to take into account the energy considerations as soon as possible during the design, at this development stage, it is still difficult to estimate precisely the power consumption. The execution of the algorithms was simulated by using a general-purpose RISC processor model. The power consumption was then estimated by using the consumption reported by various soft-core processors manufacturers: for Microblazer (Xilinx), see [82]; for ARM 9 family see [83]; and compared with typical-to-maximum thermal dissipation reported for Pentium 4 at 1.6 GHz (see [84]), compare Table 8.

Table 5: Instruction count for various steps.

Instruction count for various steps	*Additions*	*Multiplications*
Preprocessing		
∇g_K (operations per point)	2	2
Total per frame (additions, multiplication)	277 344	277 344
Construction of the narrowband		
Interpolation (additions, multiplications per point)	4	0
Propagation (additions, multiplications per point)	5	6
Total per initialization (additions, multiplication)	216 000	144 000
Total level-set-inherent computational effort	1 080 000	720 000
Curve evolution		
Evolution speed $\mathcal{F} = \nabla g_K \cdot \nabla U$	3	4
Integration (additions, multiplications per point) $U = U - (\mathcal{F}\, dt)$	1	1
Curve evolution per point (additions, multiplications)	4	5
Curve evolution per iteration (additions, multiplication)	48 000	60 000
Total curve evolution per frame	1 200 000	1 500 000
Total application per frame (curve evolution + level set inherent)	2 557 344	2 497 344
Overall real-time computational effort (FLOPS)	$75.8 \cdot 10^6$	

Table 6: The Execution Time of the Object Tracking Application.

Algorithm step	Estimated bandwidth (point/s)	Number of iterations	Number of points	Processing time (s)
Initialization				
Gradient ∇g_K	$19.3 \cdot 10^6$	1	138 672	$7.19 \cdot 10^{-3}$
Narrowband construction	$2.61 \cdot 10^6$	5	12 000	$2.30 \cdot 10^{-2}$
Evolution				
Gradient ∇u	$19.3 \cdot 10^6$	25	12 000	$1.56 \cdot 10^{-2}$
Integration u^{n+1}	$19.3 \cdot 10^6$	25	12 000	$1.56 \cdot 10^{-2}$
Total execution time (per frame)				$6.13 \cdot 10^{-2}$
Application frame processing rate (frame/s)				16.3

Table 7: The execution time of one iteration, compared to similar algorithms on other platforms.

Platform	Frequency	Execution time for one iteration (ms)
Proposed MIMD architecture Four RISC processors	120 MHz/FPGA	1.25
Graphic hardware nVIDIA GeForce2	250 MHz	4
PC with P4/Win 2000	1.6 GHz	19.1

Table 8: Comparison of power consumption.

Processor	Power consumption (W)
Microblazer / Xilinx	0.11
ARM9 / ARM	0.14
Pentium4 (1.6 GHz)/ Intel	60–75

5. CONCLUSIONS

In this paper, we present an embedded architecture for real-time image processing using level-set-based active contours. The contribution of this paper is twofold. In its first part, the text proposes a unifying insight into the level set framework from the system design point of view, to propose a unique iteration type with two different types of memory access: random memory access and sequential memory access. Then it analyzes the data flow to define, in the second part of the text, a scalable architecture fitting the real-time needs and taking into account the limited energy autonomy of embedded platforms and the silicium surface on commercially available FPGAs.

The performance of the proposed architecture has been studied on two benchmarks.

The first one, computation of a weighted distance transform with simultaneous propagation of region labels, is to verify the uniformity of the data flow and the distribution of the computational burden over all the processing units. This benchmark compares the real execution time against the theoretical execution time (obtained as the time needed by one processing unit divided by the number of processing units

operating in parallel). The results show a linear increase of performance and a balanced activity at least up to four independently operating processing units.

The second benchmark implements an active-contour-based object-tracking algorithm. The purpose of this test is to evaluate the capability of this platform to run in real-time applications with intensive random memory accesses. Section 4.2.3 lists the details concerning the computational complexity of the application in terms of number of elementary operations. The simulation results show that the above-presented contour tracking application can be run on this architecture in real time, provided that the processors are clocked at 120 MHz, and one instruction executes in one clock cycle. Hence, the architecture specifications made in the first part of the text are confirmed.

The scalability of this architecture consists in replicating the processing units. Physically, their number is limited by the silicium available on the chip; and logically, by the data-flow balance on all the blocks of the architecture. A time-costly computation will allow a linear increase of the performance up to a higher number of processing units, before the busses and the memory blocks saturate. From Figure 3, it follows that the highest data flow concentrates on the READ data memory. Although it has not been used in this paper, two possible improvements will make the data flow on the individual memory blocks more uniform: (i) the entire four-neighborhood can be retrieved in one clock cycle by using another memory organization, as proposed by Noguet in [67], or (ii) the READ data memory flow can be divided by two by using a dual-port memory for the data memory pages. However, both options will lead to some increase of complexity of the switch.

REFERENCES

[1] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1988.

[2] S. Osher and R. P. Fedkiw, "Level set methods: an overview and some recent results," *Journal of Computational Physics*, vol. 169, no. 2, pp. 463–502, 2001.

[3] J. A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Science*, Cambridge University Press, Cambridge, UK, 1996.

[4] G. Sapiro, *Geometric Partial Differential Equations and Image Analysis*, Cambridge University Press, New York, NY, USA, 2001.

[5] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," *Journal of Computational Physics*, vol. 118, no. 2, pp. 269–277, 1995.

[6] R. Malladi, J. A. Sethian, and B. C. Vemuri, "A fast level set based algorithm for topology-independent shape modeling," *Journal of Mathematical Imaging and Vision*, vol. 6, no. 2-3, pp. 269–289, 1996.

[7] F. Precioso and M. Barlaud, "B-spline active contour with handling of topology changes for fast video segmentation," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 6, pp. 555–560, 2002, Special Issue on Image Analysis for Multimedia Interactive.

[8] G. Cserey, C. Rekeczky, and P. Földesy, "PDE-based histogram modification with embedded morphological processing of the level-sets," *Journal of Circuits, Systems and Computers*, vol. 12, no. 4, pp. 519–538, 2003.

[9] J. Weickert, B. M. T. H. Romeny, and M. A. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering," *IEEE Trans. Image Processing*, vol. 7, no. 3, pp. 398–410, 1998.

[10] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll, "Image selective smoothing and edge detection by nonlinear diffusion," *SIAM Journal on Numerical Analysis*, vol. 29, no. 1, pp. 182–193, 1992.

[11] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky, "Fast geodesic active contours," *IEEE Trans. Image Processing*, vol. 10, no. 10, pp. 1467–1475, 2001.

[12] P. Smereka, "Semi-implicit level set methods for curvature and surface diffusion motion," *Journal of Scientific Computing*, vol. 19, no. 1-3, pp. 439–456, 2003.

[13] S. Holmgren and D. Wallin, *Performance of High-Accuracy PDE Solvers on a Self-Optimizing NUMA Architecture*, vol. 2150 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2001.

[14] J. A. Sethian, "Parallel level set methods for propagating interfaces on the connection machine," Department of Mathematics, University of California at Berkeley, Berkeley, Calif, USA, 1989.

[15] M. Rumpf and R. Strzodka, "Nonlinear diffusion in graphics hardware," in *Proc. EG/IEEE TCVG Symposium on Visualization (VisSym '01)*, pp. 75–84, Ascona, Switzerland, May 2001.

[16] M. Rumpf and R. Strzodka, "Level set segmentation in graphics hardware," in *Proc. International Conference on Image Processing (ICIP '01)*, vol. 3, pp. 1103–1106, Thessaloniki, Greece, October 2001.

[17] J. E. Cates, A. E. Lefohn, and R. T. Whitaker, "GIST: an interactive, GPU-based level set segmentation tool for 3D medical images," *Medical Image Analysis*, vol. 8, no. 3, pp. 217–231, 2004.

[18] C. Sigg, R. Peikert, and M. Gross, "Signed distance transform using graphics hardware," in *Proc. 14th IEEE Visualization Conference (VIS '03)*, pp. 83–90, Seattle, Wash, USA, October 2003.

[19] K. Hwang, P. S. Tseng, and D. Kim, "An orthogonal multiprocessor for parallel scientific computations," *IEEE Trans. Comput.*, vol. 38, no. 1, pp. 47–61, 1989.

[20] T. Gijbels, P. Six, L. Van Gool, F. Catthoor, H. De Man, and A. Oosterlinck, "A VLSI-architecture for parallel non-linear diffusion with applications in vision," in *Proc. IEEE Workshop on VLSI Signal Processing VII*, pp. 398–407, La Jolla, Calif, USA, October 1994.

[21] E. Dejnožková and P. Dokládal, "A parallel architecture for curve-evolution PDEs," *Image Analysis and Stereology*, vol. 22, pp. 121–132, 2003.

[22] R. Wittig and P. Chow, "OneChip: an FPGA processor with reconfigurable logic," in *Proc. IEEE Symposium on FPGAs for Custom Computing Machines (FCCM '96)*, K. L. Pocek and J. Arnold, Eds., pp. 126–135, IEEE Computer Society, Napa Valley, Calif, USA, April 1996.

[23] Z. A. Ye, A. Moshvos, S. Hauck, and P. Banerjee, "CHIMAERA: a high-performance architecture with a tightly-coupled reconfigurable functional unit," in *Proc. 27th International Symposium on Computer Architecture*, pp. 225–235, British Columbia, Canada, 2000.

[24] Y. Li, T. Callahan, E. Dernell, R. Harr, U. Kurkure, and J. Stockwood, "Hardware-software co-design of embedded reconfigurable architectures," in *Proc. 37th Design Automation Conference (DAC '00)*, pp. 507–512, Los Angeles, Calif, USA, June 2000.

[25] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 7, pp. 629–639, 1990.
[26] L. Alvarez, P.-L. Lions, and J.-M. Morel, "Image selective smoothing and edge detection by nonlinear diffusion. II," *SIAM Journal on Numerical Analysis*, vol. 29, no. 3, pp. 845–866, 1992.
[27] S. Schüpp, *Prétraitement et segmentation d'images par mise en oeuvre de techniques basées sur les équations aux dérivées partielles : application en imagerie microscopique biomedicale*, Ph.D. thesis, Université de Caen Basse-Normandie, Caen Cedex, France, December 2000.
[28] B. Kimia and K. Siddiqi, "Geometric heat equation and nonlinear diffusion of shapes and images," *Computer Vision and Image Understanding*, vol. 64, no. 3, pp. 305–322, 1996.
[29] T. Lindeberg, *Scale-Space Theory in Computer Vision*, Kluwer Academic, Dordrecht, The Netherlands, 1994.
[30] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel, "Axioms and fundamental equations of image processing," *Archive for Rational Mechanics and Analysis*, vol. 123, pp. 199–257, 1993.
[31] R. Brockett and P. Maragos, "Evolution equations for continuous-scale morphology," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '92)*, vol. 3, pp. 125–128, San Francisco, Calif, USA, March 1992.
[32] R. van den Boomgaard, *Mathematical morphology: extensions towards computer vision*, Ph.D. thesis, University of Amsterdam, Amsterdam, The Netherlands, March 1992.
[33] F. Meyer and P. Maragos, "Nonlinear scale-space representation with morphological levelings," *Journal of Visual Communication and Image Representation*, vol. 11, no. 2, pp. 245–265, 2000.
[34] L. Najman and M. Schmitt, "Watershed of a continuous function," *Signal Processing*, vol. 38, no. 1, pp. 99–112, 1994.
[35] A. Montanvert and J. M. Chassery, *Géométrie discrète en analyse d'images*, Hermès, Paris, France, 1991.
[36] R. Kimmel, *Curve evolution on surfaces*, Ph.D. thesis, Technion - Israel Institute of Technology, Haifa, Israel, May 1995.
[37] J. A. Sethian, "A marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.
[38] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.
[39] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: recovering 3D shape and nonrigid motions," *Artificial Intelligence*, vol. 36, no. 1, pp. 91–123, 1988.
[40] L. Cohen, "On active contour models and balloons," *Computer Vision, Graphics, and Image Processing*, vol. 53, no. 2, pp. 211–218, 1991.
[41] V. Caselles, F. Catté, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numerische Mathematik*, vol. 66, no. 1, pp. 1–31, 1993.
[42] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Topology independent shape modeling scheme," in *Geometric Methods in Computer Vision II*, vol. 2031 of *Proceedings of SPIE*, pp. 246–258, San Diego, Calif, USA, July 1993.
[43] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Evolutionary fronts for topology-independent shape modeling and recovery," in *Proc. 3rd European Conference on Computer Vision (ECCV '94)*, vol. 800 of *Lecture Notes in Computer Science*, pp. 3–13, Stockholm, Sweden, May 1994.
[44] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: a level set approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, no. 2, pp. 158–175, 1995.
[45] R. Malladi, R. Kimmel, D. Adalsteinsson, G. Sapiro, V. Caselles, and J. A. Sethian, "A geometric approach to segmentation and analysis of 3d medical images," in *Proc. Mathematical Methods in Biomedical Image Analysis Workshop (MMBIA '96)*, San Francisco, Calif, USA, June 1996.
[46] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," in *Proc. 5th IEEE International Conference on Computer Vision (ICCV '95)*, pp. 694–699, Cambridge, Mass, USA, June 1995.
[47] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Gradient flows and geometric active contours models," in *Proc. 5th International Conference on Computer Vision (ICCV '95)*, pp. 810–815, Cambridge, Mass, USA, June 1995.
[48] R. Malladi and J. A. Sethian, "Image processing: flows under min/max curvature and mean curvature," *Graphical Models and Image Processing*, vol. 58, no. 2, pp. 127–141, 1996.
[49] S. Jehan-Besson, M. Gastaud, M. Barlaud, and G. Aubert, "Region-based active contours using geometrical and statistical features for image segmentation," in *Proc. IEEE International Conference in Image Processing (ICIP '03)*, vol. 2, pp. 643–646, Barcelona, Spain, September 2003.
[50] L. Alvarez, J. Weickert, and J. Sánchez, "A scale-space approach to nonlocal optical flow calculations," in *Proc. 2nd International Conference on Scale-Space Theories in Computer Vision (Scale-Space '99)*, pp. 235–246, Corfu, Greece, September 1999.
[51] J. B. T. M. Roerdink and A. Meijster, "The watershed transform: definitions, algorithms and parallelization strategies," *Fundamenta Informaticae*, vol. 41, no. 1-2, pp. 187–228, 2000.
[52] E. Dejnožková, *Architecture dédiée au traitement d'image basé sur les équations aux dérives partielles*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Paris, Paris, France, March 2004.
[53] J. A. Sethian, "Fast marching methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.
[54] J. A. Sethian, "Level set methods and fast marching methods," Tech. Rep., Department of Mathematics, University of California, Berkeley, Calif, USA, 1999. http://math.berkeley.edu/~sethian/level_set.html.
[55] S. Kim, "An $\mathcal{O}(N)$ level set method for eikonal equations," *SIAM Journal on Scientific Computing*, vol. 22, no. 6, pp. 2178–2193, 2001.
[56] M. J. Flynn, "Very high-speed computing systems," *Proc. IEEE*, vol. 54, no. 12, pp. 1901–1909, 1966.
[57] R. Cypher and J. L. C. Sanz, "SIMD architecture and algorithms for image processing and computer vision," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 12, pp. 2158–2174, 1989.
[58] A. Gibbons and W. Rytter, *Efficient Parallel Algorithms*, Cambridge University Press, Cambridge, UK, 1988.
[59] A. Broggi, G. Conte, F. Gregoretti, C. Sansoè, and L. M. Reyneri, "The Paprica massively parallel processor," in *Proc. 1st IEEE International Conference on Massively Parallel Computing Systems (MPCS '94)*, pp. 16–30, Ischia, Italy, May 1994.
[60] A. Manzanera, "Morphological segmentation on the programmable retina: towards mixed synchronous/asynchronous algorithms," in *Proc. 6th International Symposium on Mathematical Morphology (ISMM '02)*, H. Talbot and R. Beare, Eds., pp. 389–399, Sydney, Australia, April 2002.
[61] T. Le, W. Snelgrove, and S. Panchanathan, "SIMD processor arrays for image and video processing: a review," in *Multimedia Hardware Architectures*, vol. 3311 of *Proceedings of SPIE*, pp. 30–41, San Jose, Calif, USA, January 1998.
[62] M. Maruyama, H. Nakahira, T. Araki, et al., "A 200 MIPS image signal multiprocessor on a single chip," in *Proc. 37th IEEE International Solid-State Circuits Conference (ISSCC '90)*, pp. 122–123, San Francisco, Calif, USA, February 1990.

[63] T. Minami, R. Kasai, H. Yamauchi, Y. Tashiro, J. Takahashi, and S. Date, "A 300-MOPS video signal processor with a parallel architecture," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1868–1875, 1991.
[64] R. Duncan, "A survey of parallel computer architectures," *IEEE Computer*, vol. 23, no. 2, pp. 5–16, 1990.
[65] E. De Greef, F. Catthoor, and H. De Man, "Mapping real-time motion estimation type algorithms to memory efficient, programmable multi-processor architectures," *Microprocessing and Microprogramming*, vol. 41, no. 5-6, pp. 409–423, 1995.
[66] A. Moga, T. Viero, B. Dobrin, and M. Gabbouj, "Implementation of a distributed watershed algorithm," in *Mathematical Morphology and Its Applications to Image and Signal Processing*, pp. 281–288, Kluwer Academic, Dordrecht, The Netherlands, 1994.
[67] D. Noguet, *Architectures parallèles pour la morphologie mathématique géodésique*, Ph.D. thesis, Institut National Polytechnique De Grenoble, Techniques de l'Informatique et de la Microélectronique pour l'Architecture des ordinateurs, Grenoble, France, Janvier 2002.
[68] A. Bieniek, *Divide-and-Conquer Parallelisation Methods for Digital Image Processing Algorithms*, vol. 10 of *VDI Fortschritt-Berichte*, VDI Verlag, Düsseldorf, Germany, 2000.
[69] M. Dubois, C. Scheurich, and F. Briggs, "Memory access buffering in multiprocessors," in *Proc. 13th Annual International Symposium on Computer Architecture (ISCA '86)*, pp. 434–442, Tokyo, Japan, June 1986.
[70] K. Gharachorloo, D. Lenoski, J. Laudon, P. Gibbons, A. Gupta, and J. Hennessy, "Memory consistency and event ordering in scalable shared-memory multiprocessors," in *Proc. 17th Annual International Symposium on Computer Architecture (ISCA '90)*, pp. 15–26, Seattle, Wash, USA, May 1990.
[71] F. Meyer and P. Maragos, "Multiscale morphological segmentations based on watershed, flooding, and eikonal PDE," in *Proc. 2nd International Conference on Scale-Space Theories in Computer Vision (Scale-Space '99)*, M. Nielsen, P. Johansen, O. F. Olsen, and J. Weickert, Eds., vol. 1682 of *Lecture Notes in Computer Science*, pp. 351–362, Springer, Corfu, Greece, September 1999.
[72] P. Dokládal, R. Enficiaud, and E. Dejnožková, "Contour-based object tracking with gradient-based contour attraction field," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '04)*, vol. 3, pp. 17–20, Montreal, Quebec, Canada, May 2004.
[73] F. Meyer and C. Vachier, "Image segmentation based on viscous flooding simulation," in *Proc. 6th International Symposium on Mathematical Morphology (ISMM '02)*, vol. 2, pp. 69–77, Sydney, Australia, April 2002.
[74] R. Haralick, "Digital step edges from zero crossing of second directional derivatives," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, no. 1, pp. 58–68, 1984.
[75] R. Kimmel and A. Bruckstein, "On edge detection integration and geometric active contours," in *Proc. 6th International Symposium on Mathematical Morphology (ISMM '02)*, vol. 2, Sydney, Australia, April 2002.
[76] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Trans. Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.
[77] K. Sobottka and I. Pitas, "Extraction of facial regions and features using color and shape information," in *Proc. 13th IEEE International Conference on Pattern Recognition (ICPR '96)*, vol. 3, pp. 421–425, Vienna, Austria, August 1996.
[78] A. Nayak and S. Chaudhuri, "Self-induced color correction for skin tracking under varying illumination," in *Proc. IEEE International Confererence on Image Processing (ICIP '03)*, vol. 3, pp. 1009–1012, Barcelona, Spain, September 2003.
[79] V. Veselý, "Fast Algorithms of Fourier and Hartley Transform and their Implementation in MATLAB," http://citeseer.ist.psu.edu/vesely98fast.html.
[80] H. Karner, M. Auer, and C. Ueberhuber, "Optimum complexity FFT algorithms for RISC processors," Tech. Rep. AURORA TR1998-03, Institute for Applied and Numerical Mathematics, Technical University of Vienna, Vienna, Austria, 1998.
[81] S. Osher and N. Paragios, Eds., *Geometric Level Set Methods in Imaging, Vision and Graphics*, chapter 2, Springer, New York, NY, USA, 2003.
[82] http://www.eece.unm.edu/xup/microblazeppc.htm.
[83] http://www.arm.com/miscPDFs/4491.pdf.
[84] http://support.intel.com/design/pentium4/datashts/24919805.pdf.

Eva Dejnožková is a research engineer with the Commissariat à l'Energie Atomique à Saclay, France, which she joined in 2004. She graduated with the highest degrees from the West Bohemia University, Pilsen, Czech Republic, in 1999, as an engineer specialized in industrial electronics. Afterwards, she specialized in hardware architecture for image processing, and obtained her Ph.D. degree from the School of Mines of Paris, France. She obtained a special distinction of the rector of the West Bohemia University. Her research interests include image processing and compression, and embedded and mobile computers architecture for image processing and compression.

Petr Dokládal is a research engineer at the Centre of Mathematical Morphology, the School of Mines in Paris, France. He graduated from the Technical University in Brno, Czech Republic, in 1994, as a telecommunication engineer and received his Ph.D. degree from the University of Marne la Vallée, France, in general computer sciences, specialized in image processing. His research interests include medical imaging, image segmentation, object tracking, and pattern recognition.