



HAL
open science

Optical flow methods for the pixel-wise measurement of fields in solid mechanics

Ahmed Chabib

► **To cite this version:**

Ahmed Chabib. Optical flow methods for the pixel-wise measurement of fields in solid mechanics. Solid mechanics [physics.class-ph]. Université de Lille, 2024. English. NNT : 2024ULILN003 . tel-04625217

HAL Id: tel-04625217

<https://theses.hal.science/tel-04625217>

Submitted on 26 Jun 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Title:

**Optical flow methods for the pixel-wise measurement
of fields in solid mechanics**

Author:

Ahmed Chabib

A dissertation submitted for the degree of
Doctor of Philosophy
In Mechanical Engineering

January 8, 2024

Jury:

Mr.	Reviewer	Jean-Charles Passieux	Professor, INSA Toulouse
Mr.	Reviewer	Julien Réthoré	CNRS Professor, Ecole Centrale de Nantes
Mrs.	Examiner	Sylvia Feld-Payet	Research Engineer, ONERA, Université Paris Saclay
Mr.	Jury president	Bertrand Wattrisse	Professor, Université Montpellier
Mr.	Examiner	Jan Neggens	Associate Professor, CentraleSupélec
Mr.	Thesis Director	Pierre Gosselet	CNRS Professor, Université de Lille
Mr.	Supervisor	Jean-François Witz	Research Engineer, Ecole Centrale de Lille
Mr.	Supervisor	Vincent Magnier	Associate Professor, Université de Lille

Keywords:

Non-Linear Mechanics, Strain, Parallel computing, Digital Image correlation, Optical Flow, Metrics

Titre:

**Méthodes de flot optique pour la mesure de champ au
pixel en mécanique des solides**

Auteur:

Ahmed Chabib

Thèse présentée en vue d'obtenir le grade de
Docteur
en Mécanique

8 Janvier 2024

Jury:

M.	Rapporteur	Jean-Charles Passieux	Professeur, INSA Toulouse
M.	Rapporteur	Julien Réthoré	DR CNRS, Ecole Centrale de Nantes
Mme.	Examinatrice	Sylvia Feld-Payet	IR ONERA, Université Paris Saclay
M.	Président du jury	Bertrand Wattrisse	Professeur, Université Montpellier
M.	Examineur	Jan Neggers	Maître de conférences, Centrale-Supélec
M.	Directeur de thèse	Pierre Gosselet	DR CNRS, Université de Lille
M.	Encadrant	Jean-François Witz	IR CNRS, Ecole Centrale de Lille
M.	Encadrant	Vincent Magnier	Maître de conférences, Université de Lille

Mots-clés:

Mécanique non linéaire, Déformation, Calcul parallèle, Corrélation d'images numériques, Flot optique, Métriques

Abstract

During mechanical tests, the materials undergo strain which is often non-uniform and complex to measure. This is particularly the case when the material under study exhibits local variations in its properties, as is the case with composite materials, or in the presence of cracking. The optical non intrusive methods for the full-field measurement such as the Digital Image Correlation (DIC), widely used in mechanics, help to overcome this issue. The Optical Flow technique share the same objective as the DIC, but it is rarely used in our community. This dissertation work aims to prove the efficiency and the relevance of some optical flow configurations for estimating mechanical quantities of interest, in particular the strain.

We first present the link between these methods for motion estimation, their common characteristics and distinctions. At the core of these methods lies the quantification of the similarity between two images in order to find the transformation that allows transitioning from one to the other. To solve this ill-posed problem, the global approaches of DIC use interpolation functions to describe the movement. In the other side, the optical flow approaches describe the displacement at each pixel and employ a Tikhonov penalty to impose the regularity of the field and to ensure the existence and uniqueness of the solution.

If the least-square error (L^2 norm) is omnipresent in DIC, several metrics have been studied in the optical flow for the calculation of the displacement field, to avoid the smoothing effect on discontinuities when the L^2 norm is used in the regularization term. This dissertation inspects the effect of some of these metrics (Charbonnier and Lorentz) over the restitution of discontinuities and strain. We show that these functions can provide satisfactory results while the Charbonnier function, with the usual parameters for optical flow, may encourage the appearance of some parasitic cracks in the fields. With the help of an image mask containing different weights of regularization, we show that we can force the emergence of local phenomena while maintaining the homogeneity of the solution.

The rapid acquisition of kinematic fields is of great importance, as it enables us to gain a better understanding of the material's behavior by comparing a maximum number of image sequences. Another purpose of this dissertation is the generation of a pixel-wise code of correlation, GPU-accelerated and easy to understand and modify. Our open source code was used for different applications such as the calibration of cameras.

Finally, efforts have been made to improve the Conjugate Gradient, a Krylov

linear solver which is essential for solving large-scale symmetric positive definite systems, by proposing a preconditioner guided by Tikhonov's regularization and by taking advantage of Ritz values for filtering and recycling the digital information in order to adjust regularization.

Résumé

Pendant les essais mécaniques, les matériaux subissent des déformations souvent non uniformes dont la mesure est complexe. Cela est particulièrement le cas quand le matériau étudié présente des variations locales dans ses propriétés comme les matériaux composites, ou en présence de fissuration. Les méthodes optiques non intrusives pour la mesure de champs, telles que la Corrélation d'Images Numériques (CIN) très utilisée en mécanique, peuvent surmonter cette difficulté. La technique du Flot Optique partage le même objectif que la CIN mais elle est peu fréquemment employée dans notre communauté. Ce travail de thèse vise à prouver la pertinence et l'efficacité de certaines configurations du Flot Optique pour l'estimation des quantités d'intérêt du mécanicien, en particulier la déformation.

On présente, tout d'abord, le lien entre ces deux méthodes pour l'estimation du mouvement, leurs caractéristiques communes et leurs distinctions. Au cœur de ces méthodes, réside la quantification de la similarité entre deux images afin de trouver la transformation qui permet de passer de l'une à l'autre. Pour résoudre ce problème mal posé, les approches globales de la DIC décrivent le mouvement en se basant sur des fonctions d'interpolation. Les approches du Flot Optique décrivent le déplacement sur chaque pixel et utilisent un terme de Tikhonov afin d'imposer la régularité du champ et pour assurer l'existence et l'unicité de la solution.

Si l'erreur aux moindres carrés (norme L^2) est omniprésente dans la CIN, beaucoup de métriques ont été étudiées pour le Flot Optique appliqué au calcul de champs de déplacement, afin d'éviter l'effet de sur-lissage des discontinuités quand la norme L^2 est utilisée dans le terme de régularisation. Cette thèse analyse l'effet de certaines métriques (Charbonnier et Lorentz) sur la restitution des discontinuités et des déformations. On montre que ces fonctions peuvent fournir des résultats satisfaisants, alors que la fonction de Charbonnier, telle qu'elle est habituellement utilisée pour le flot optique, peut encourager l'apparition des discontinuités parasites dans les champs. À l'aide d'une image masque avec différents poids de régularisation, on montre que l'on peut forcer l'apparition des phénomènes locaux tout en gardant l'homogénéité de la solution.

L'acquisition rapide des champs cinématiques revêt une grande importance, puisqu'elle favorise la meilleure compréhension du comportement du matériau en comparant un maximum de séquences d'images. Un autre objectif de la thèse est la génération d'un code de corrélation à l'échelle du pixel accéléré par GPU, facile à comprendre et à modifier. Notre code open source a été

utilisé pour des applications différentes comme la calibration d'image. En fin de compte, des efforts ont été déployés pour améliorer le gradient conjugué, solveur linéaire de Krylov essentiel pour la résolution des systèmes symétriques définis positifs de très grande dimension, en utilisant un préconditionneur guidé par la régularisation de Tikhonov et en profitant des valeurs de Ritz pour filtrer et recycler l'information dans le but d'ajuster la régularisation.

Remerciements

Je souhaite tout d'abord exprimer ma profonde gratitude envers les rapporteurs de cette thèse, M. Julien Réthoré et M. Jean-Charles Passieux, pour la précieuse évaluation de ce travail ainsi que l'autorisation qu'ils m'ont accordée à soutenir. Mes remerciements s'adressent également à M. Bertrand Wattrisse qui a accepté avec bienveillance de présider le jury de ma thèse, à Mme. Sylvia Feld-Payet et M. Jan Neggens qui ont contribué à l'examen de cette recherche.

J'aimerais également exprimer ma reconnaissance envers mes encadrants, grâce à qui ce travail a pu être mené à terme. Tout d'abord à mon directeur de thèse Pierre Gosselet, pour sa confiance, sa gentillesse et sa disponibilité tout au long de ces trois années. Mes remerciements vont également à Jean-François Witz, qui enrichi ma pensée par ses nombreuses discussions scientifiques et a constamment partagé de nouvelles idées à explorer, ainsi qu'à Vincent Magnier pour ses remarques pertinentes, sa disponibilité et la configuration de la machine de calcul que j'ai utilisée. Je remercie chaleureusement toute l'équipe du Lamcube avec laquelle j'ai partagé des moments agréables, inoubliables.

Mes remerciements s'adressent à tous mes amis proches pour leurs encouragements et les moments partagés qui m'ont donné la force de persévérer. Ce travail n'aurait pas pu être accompli sans le soutien et l'encouragement de ma sœur Karima et de sa famille, qui ont cru en moi et m'ont soutenu. Je dédie cette réalisation à tous les membres de ma famille, en particulier à mes frères et sœurs et à ma mère qui a toujours veillé sur moi.

Pour finir, j'aimerais dédier ce travail à l'âme de mon père, celui qui m'a appris à écrire mes premiers mots et qui nous a malheureusement quitté pendant la rédaction de ce manuscrit.

Contents

1	Introduction	17
2	State of the art	19
2.1	Digital Image Correlation(DIC)	19
2.1.1	Introduction	19
2.1.2	Digital Image Correlation (DIC) principles	19
2.1.3	Techniques for improving the full-field measurement	25
2.1.4	Conclusion	26
2.2	Optical Flow	27
2.2.1	Introduction	27
2.2.2	Global approach	27
2.2.3	Local approach	29
2.2.4	Conclusion	32
2.3	Solvers for linear systems and Parallel computing	33
2.3.1	Solvers for linear systems	33
2.3.2	Preconditioning	35
2.3.3	Parallel computing	38
2.3.4	General-purpose computing on graphics processing units	42
2.3.5	Conclusion	43
3	The impact of metrics in mechanical imaging	47
3.1	Introduction	48
3.2	Methodology	49
3.2.1	Graduated Non-Convexity (GNC) and pyramidal approach	50
3.2.2	Methodological details	52
3.2.3	Implementation	52
3.3	Results	53
3.3.1	Crossing crack	53
3.3.2	Crack with tip	54
3.3.3	Local regularization	59
3.4	Conclusion	67
4	GCPU_OpticalFlow: a GPU accelerated Python software for strain measurement	73
4.1	Motivation and significance	74
4.1.1	Introduction	74
4.1.2	Principles of DIC and Optical Flow	74
4.2	Software description	75
4.2.1	Software functionalities	75
4.2.2	Software architecture	76
4.3	Illustration	77
4.4	Impact	77

4.5	Conclusion	80
4.6	Acknowledgements	81
4.7	Applications	82
4.7.1	Calibration of cameras	83
4.7.2	Microstructure gradient's	85
5	Interplay between preconditioning and regularization for linear ill-posed problems solved by conjugate gradient: Application to optical flow estimation	89
5.1	Introduction	89
5.2	Basic notions about the optical flow	91
5.3	Preconditioned Conjugate Gradient and Ritz elements	92
5.3.1	Role of the preconditioner	94
5.3.2	Stopping criteria	94
5.3.3	A posteriori filtering	95
5.4	Preconditioning by regularization	96
5.5	Assessments	97
5.5.1	Application to optical flow reconstruction	97
5.5.2	Quality of the preconditioner	97
5.5.3	Ritz filtering	99
5.5.4	Subspace recycling	102
5.5.5	Tuning of λ	102
5.6	Conclusion	103
6	General Conclusion	107

List of Figures

2.1	Schematic representation of the correlation problem. I_1 the reference image, I_2 the deformed image, p the coordinates of the framed subset in I_1 and w the displacement vector.	20
2.2	Illustration of motion field calculation using an iterative method	21
2.3	Illustration of displacement field modeling and image partitioning for local DIC	23
2.4	Example of a Q4 finite element mesh for global DIC displacement modeling applied to an image with a speckle pattern. . . .	23
2.5	Generalized Charbonnier loss function for different values of a . . .	30
2.6	Lorentzian loss function for different values of σ	30
2.7	Illustration of the data parallel model	39
2.8	Illustration of the task parallel model	40
2.9	The shared memory model	40
2.10	The distributed memory model	41
2.11	The hybrid memory model	42
2.12	GPU and CPU Architectures	43
3.1	The curve of the tested penalty functions: the quadratic (L^2), Charbonnier (Cha.) using $\varepsilon = 0.001$ and the Lorentzian with different values of σ	50
3.2	Representation of the pyramidal approach, OF=Optical Flow . .	51
3.3	First synthetic test case	54
3.4	Identification of the crossing crack for different metrics.	55
3.5	The deformed image for the second test case, the part framed in blue contains the generated crack	56
3.6	Horizontal displacement field estimated with different values of λ , and different loss functions for the crack with tip experiment.	57
3.7	The horizontal strain fields ε_{xx} provided with different regularization amplitude of λ for for different metrics.	59
3.8	The vertical strain fields ε_{yy} provided with different regularization amplitude of λ for different metrics.	61
3.9	NSDE for the different metrics using different regularization amplitudes. Their corresponding MNSDE is presented in the table 3.9h. The image 3.9g represents the analytical norm of the second-order gradient.	63
3.10	ε_{xx} strain of Charbonnier function using different ε values and fixed $\lambda = 4 \times 10^2$	63
3.11	ε_{yy} strain of Charbonnier function using different ε values and fixed $\lambda = 4 \times 10^2$	64
3.12	ε_{xx} strain of Lorentz function using different σ values and fixed $\lambda = 10^2$	65

3.13	ε_{yy} strain of Lorentz function using different σ values and fixed $\lambda = 10^2$	66
3.14	(a) Local regularization mask. The cut along the red line is shown in the figure (b).	68
3.15	The strain fields in terms of λ . The third column shows a zoom of the second column over the crack. The first and second rows show fields obtained with values close to the minimum and maximum regularization parameter ($\lambda = 7$ and $\lambda = 600$) used in the mask, while the third row shows the results obtained with the local parameter change. Finally, the analytical fields are presented in the last row. The dotted line corresponds to the analytical abscissa of the crack tip.	70
3.16	NSDE obtained with different regularization amplitudes and using the local regularization mask. The table indicates the number of pixel required to capture the crack as well as the AEE, the MNSDE of the global field as well as in the vicinity and outside the crack.	71
3.17	Example of the Structure-Texture decomposition	72
4.1	Representation of the pyramidal approach	77
4.2	Flow-chart of flow field estimation with <i>GCPU_OpticalFlow</i>	78
4.3	Experimental setup with (A) the camera, (B) the sample under testing and (C) the uniaxial tensile testing machine.	79
4.4	Dimensions of the used ± 45 Carbon Epoxy specimen.	79
4.5	Uniaxial strain ε_{xx} in function of λ with fixed 3×3 median filter	80
4.6	Uniaxial strain ε_{xx} in function of the median size with fixed $\lambda = 3 \times 10^3$	81
4.7	Strain field computed by YaDICs and <i>GCPU_OpticalFlow</i>	82
4.8	Energy image in terms of gradient displacement energy for both software. The dotted vertical lines indicate the limits of the zone where the two software are compared	82
4.9	The device used for camera calibration, where (A) Left camera, (B) Right camera and (C) The calibration pattern,(D) Moving table	84
4.10	(a) Profilometric topography. (b) Soloff topography. (c) Direct topography. (d) AI topography. (e)z-distribution of the 450 th column [88]	86
4.11	Strain fields in the loading direction using YaDICs for different correlation window sizes and <i>GCPU</i> for several regularization values [102]	87
5.1	Speckle of the test specimen.	99
5.2	L-curves ($\varepsilon = 10^{-5}$) compared with Ritz post-treatment, for different regularization intensity λ . \triangle Different scales on each plot.	100
5.3	Spectral analysis of the system for $\varepsilon = 10^{-5}$ and different regularization intensity λ . A 5-width median filter was used on the contribution curves.	101

5.4	Identified ε_{xx} field, for different regularization intensity λ , with $\varepsilon = 10^{-5}$	101
5.5	Costless postprocessing for different (λ_i) : top, initial computation with $\lambda \in \{1, 100, 1000, 10000\}$; bottom, solution deduced for $\lambda = 1$. ε_{xx} strain field.	103
6.1	ε_{zz} strain field, $\lambda = 10^3$	110

List of Tables

4.1	Statistical indicators (in pixels) of the root-mean-square error (RMSE) of the kinematic fields computed by YaDICs and those estimated by GCPU_OpticalFlow.	78
4.2	Code metadata	83
5.1	ε_{xx} strain field (range = mean value \pm 3 st.dev.). Comparison of the effect of preconditioning by diagonal (simple approach) vs by regularization, for different weights $\lambda \in \{1, 1000\}$ and linear solver precision $\varepsilon \in \{10^{-2}, 10^{-3}\}$	98
5.2	Performance of recycling	102

1. Introduction

It is currently challenging for researchers and industrialists to understand the dissipation mechanisms in heterogeneous materials. This complexity arises from the diverse and non-uniform nature of these materials. Consequently, reliably modeling them at a scale suitable for engineering is difficult. Recent advances in experimentation, particularly the use of imaging, provide access to extremely detailed data regarding material composition. By coupling these fine observations with image correlation techniques, it becomes possible to precisely observe deformation mechanisms. This combination of experimental and numerical tools should enable the development of models based on observations rather than idealizations of microscopic behaviors.

The digital image correlation method is one of the numerical and optical techniques used to describe the behavior of a material by comparing two images captured at different times to compute displacement and strain fields. The fundamental assumption behind these methods is the graylevel conservation, initially introduced by Horn and Schunk in 1981 [1], which represents one of the first optical flow algorithms. Optical flow methods are formulated in a similar manner to the digital image correlation methods, but they are not widely recognized, and their use is not common in mechanics. The early methods of DIC resemble the local Lucas-Kanade [2] optical flow model, and they were introduced by Sutton [18] in 1983. These initial works involved subdividing a reference image into multiple regions and searching for matches in the moving image. Sutton methods demonstrated their performance when they were used for measuring planar rigid body motions [98].

These two families of optical methods generally offer local and global approaches for estimating the transformation. Local optical flow and DIC approaches assume that for a given pixel in an image, the neighboring pixels follow a movement similar to that of that pixel. In contrast, the global approach of DIC employs interpolation functions to describe the displacement field over a reduced set of degrees of freedom. On the other hand, the global approach of optical flow introduces a regularization term and imposes a new constraint to add information and enforce uniqueness to the optimization problem to be solved. The graylevel conservation can be violated. To quantify this violation in DIC, we often choose the mean square error function, which leads to simple computations but which is sensitive to discontinuities and outliers. This norm has been shown to diffuse cracks as regularization increases. This problem can be solved by a two-pass approach where the first computation permits to detect zones where a discretization suited to discontinuities, like the extended finite element method (XFEM) [23], must be introduced before running the second computation.

The same problem of diffusion has been solved in optical flow by the use of new, so-called robust, metrics. This name comes from the fact that, unlike the quadratic norm, these functions are less sensitive to noise and able to preserve the discontinuities. However, studies in this community are limited to displacements. This approach to changing the error function is less complex than the XFEM method, and independent of the shape of the field to be calculated. The choice and

study of the impact of these functions in mechanics represents a blind spot in the literature, since they have not been sufficiently dealt with. The first objective of this thesis is therefore to study the influence of these metrics and their parameters on the computed strain fields.

Recent advances in experimentation, particularly the development of camera sensors, provide access to extremely detailed data about material composition and enable precise observation of deformation mechanisms. However, the trade-off for this wealth of observation is the enormous increase in the computational cost associated with data processing. Models derived from pixel-wise measurements involve millions of degrees of freedom. Therefore, the second objective of this thesis is to gain a strong understanding of optical flow computation techniques and to develop a fast and memory-optimized code for these purposes.

The first chapter of this thesis comprises a bibliographic study to give a clear idea of the technical aspects of the optical methods studied in this work. The other objective of this chapter is to establish the link between the digital image correlation community and the optical flow community while emphasizing their similarities and differences. Finally, the last section deals not only with the numerical solvers needed to estimate displacements, but also with parallel computing [100] and GPU computing [67, 68], which are important for a proper understanding of this work.

The quadratic norm as well as other similar functions are often used in mechanics in contrast to optical flow where other penalty functions [6] are used. The second chapter is a study of the impact of some non-classical error functions on mechanical images, with a particular focus on cracks. In this chapter, we also look at the impact of local variations of regularization over on kinematic fields. At the end of this chapter, a GPU-accelerated code is provided to allow mechanics to test their images with the studied functions.

The quadratic norm frequently used in DIC is a convex function and therefore it is simpler to obtain its optimum, which translates into a low computational cost compared with non-convex functions that require an iterative process detailed in the previous chapter. Before concluding, we present in the last chapter a time and memory optimised implementation of one of the classic approaches to optical flow computation entitled Horn & Schunk with a comparison of YaDICs [61] which is an open source software for DIC developed within our laboratory LamCube. Finally, thanks to iterative solvers based on projection and to Ritz modes [83], we were able to demonstrate that we can calculate solutions for different regularisation values with a simple calculation.

2. State of the art

2.1. Digital Image Correlation(DIC)

2.1.1. Introduction

Understanding the behavior of materials is of crucial importance in many fields of science and engineering. To ensure their safety, durability and optimal performance, it is necessary to understand how materials behave under different loading conditions. When materials are subjected to external loads such as compression, tension, bending, or torsion, they can undergo strain. However, these strain fields are not uniform throughout the material and they vary locally depending on the geometry, micro-structure, and applied stresses. Several methods are available for measuring these strain in materials. Among the commonly used methods are strain gauges [26] and extensometers. These traditional techniques offer the advantages of reliability and cost-effectiveness. However, they have certain limitations. For instance, they do not provide a complete picture of the material's behavior since they allow measuring strain only at specific points in the sample.

To overcome these limitations, full-field measurement methods have been developed to obtain a more comprehensive image of strain in materials. These methods, such as Digital Image Correlation (DIC) (or Digital Volume Correlation (DVC) in the context of three-dimensional field), enable mapping strain across the entire surface or volume of the material. They offer the advantage of providing information on a large number of points, allowing for a detailed analysis of stresses and strains throughout the material. Moreover, since these methods are non-invasive they do not alter the mechanical properties of the samples.

DIC is particularly popular as a full-field measurement method. It relies on using a camera to capture images of the sample at different time points. By analyzing the differences between these images, it is possible to calculate local deformations and reconstruct kinematic fields across the material. This technique has numerous advantages, including its ease of use, accuracy, and ability to measure motion under real loading conditions.

The objective of this chapter is to present an overview of the various digital image correlation methods found in the literature and their underlying principles. By exploring these methods, we aim to provide a comprehensive understanding of DIC and its applications in the field of material testing and strain analysis.

2.1.2. Digital Image Correlation (DIC) principles

Let's assume that we have two images of the same object at two different instants. Digital Image Correlation aims to obtain the displacement field to be applied to each pixel in the first

image in order to locate it in the second one. An illustration representing the principle of this technique is shown in Fig 2.1.

Since their existence [18], digital image correlation methods have supposed that the graylevels of pixels do not change over an image sequence. This assumption has been introduced few years before the existence of DIC techniques by Horn and Schunck [1] and it is often referred to as the graylevel conservation assumption.

Let (I_1, I_2) be an image sequence, w the displacement vector that represents the transformation between the image I_1 (the reference image) and the image I_2 (the deformed image). Let p be the coordinates of a given pixel of the image. The graylevel conservation assumption can be traduced by the equation (2.1):

$$I_1(p) = I_2(p + w(p)). \quad (2.1)$$

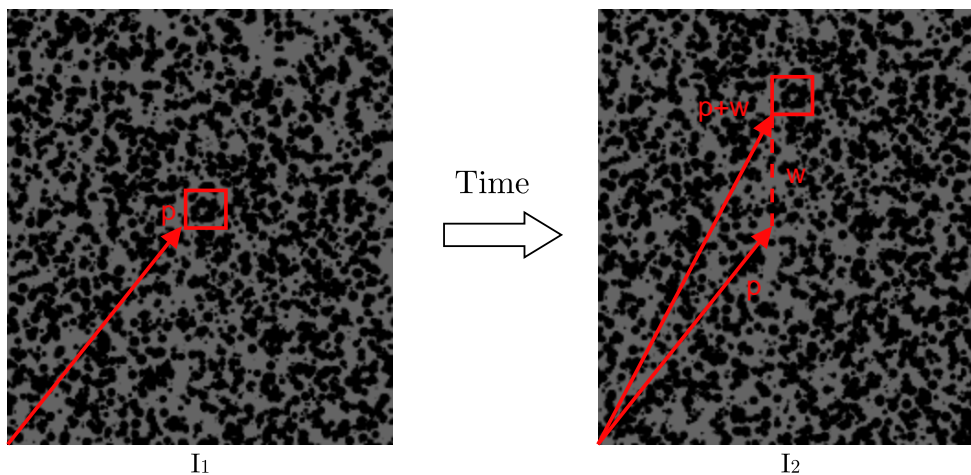


Figure 2.1: Schematic representation of the correlation problem. I_1 the reference image, I_2 the deformed image, p the coordinates of the framed subset in I_1 and w the displacement vector.

In order to quantify the difference between the images, the quadratic norm is often used in the literature, leading to the formulation of the function E , which is dependent on the displacement w .

$$E(w) = \int_{\Omega} (I_1(p) - I_2(p + w(p)))^2 dx, \quad (2.2)$$

where Ω is the domain of correlation. The main purpose of DIC is the minimization of E . In general the graylevel conservation assumption is not respected due to different factors like the noise or sometimes simply due to changes in brightness, which is why it is advisable to pay attention to the light sources used during the tests.

$$\min_w E(w) = \min_w \int_{\Omega} (I_1(p) - I_2(p + w(p)))^2 dx. \quad (2.3)$$

The optimization problem (2.3) is ill-posed since the sought field has multiple components, while the data at our disposal are scalar and correspond to the graylevels of the images. To find the displacement, additional constraints must be added.

In practice, the displacement field w is sought within a finite dimension vector subspace of approximation denoted by V_h and defined by a basis consisting of a family of interpolation functions [27] $(\Phi_i)_{i \in [1, n_d]}$. Then the displacement can be written as:

$$w(p) = \sum_i^{n_d} \lambda_i \Phi_i(p)$$

where (λ_i) are the desired degrees of freedom.

In order to efficiently solve this nonlinear optimization problem, iterative methods [21] are often used, see Fig.2.2, and the field w is updated at each iteration using a spatial increment denoted by dw . The increments (dw^j) can be interpreted as steps to be taken in order to reach the optimal direction for transforming w^k into w^{k+1} .

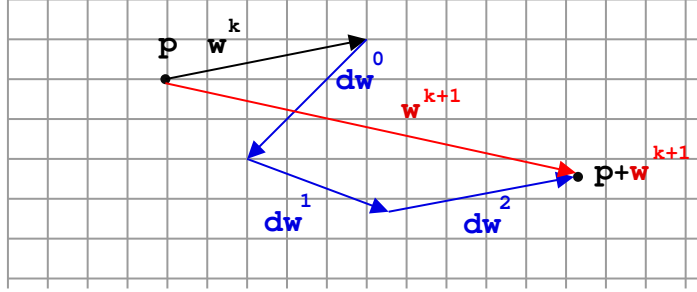


Figure 2.2: Illustration of motion field calculation using an iterative method

We assume that the displacement is small. We use a first order Taylor expansion with the respect to $dw = \sum_i^{n_d} d\lambda_i \Phi_i(p)$ on the deformed image.

$$I_2(p + w(p) + dw(p)) \approx I_2(p + w(p)) + \nabla I_2(p + w(p)) dw(p). \quad (2.4)$$

The optimality condition $\frac{\partial E(w + dw)}{\partial d\lambda_i} = 0$ leads to the resolution of a linear system:

$$Md\lambda = b \quad (2.5)$$

where:

$$M_{ij} = \int_{\Omega} (\Phi_j(p) \nabla I_2(p + w(p))) (\Phi_i(p) \nabla I_2(p + w(p))) dp,$$

$$b_i = \int_{\Omega} (I_1(p) - I_2(p + w(p))) (\Phi_i(p) \nabla I_2(p + w(p))) dp.$$

After solving the system and determining the values $(d\lambda_j)_j$ that contribute to the best spatial increment, and the deformed image will subsequently be corrected by the newly calculated displacement and this process is repeated until convergence is reached.

The calculation of the gradient of the corrected image $\nabla I_2(p + w)$ necessitates an interpolation at each step. Since w is supposed to be small, the gradient of the corrected $\nabla I_2(p + w)$ can

be approximated by the gradient of the reference image $\nabla I_1(p)$ [12, 21]. This result allows the calculation of the image derivatives once in an iterative scheme as the approximation of the gradient does not depend on the time. This result is essential as it allows creating the matrix system only once, making the calculation much simpler. It should be noted that only an update of the right hand side is required at each stage. However, the approximation of the gradient of the corrected image has a limited field of application, covering small deformations in particular [21].

In that case, the linear system (2.5) simplifies to:

$$M_{ij} = \int_{\Omega} (\Phi_j(p) \nabla I_1(p)) (\Phi_i(p) \nabla I_1(p)) dp,$$

$$b_i = \int_{\Omega} (I_1(p) - I_2(p + w(p))) (\Phi_i(p) \nabla I_1(p)) dp.$$

In the literature, the most popular digital image correlation approaches are the local and the global approaches [19] and the choice of the approach depends on the studied phenomena and the available computational resources.

2.1.2.1 Local Methods

Local methods [18] consist of searching for the displacement at a given central point by considering its neighbors. The correlation criterion defined by equation (2.3) is minimized independently over grids of pixels, called Zones Of Interest (ZOI). Typically, the zones of interest have a rectangular or square shape since images are composed of pixels, see figure 2.3. The distance between the ZOIs enables them to either be contiguous, overlapping, or entirely separated. The objective is to locate each ZOI from the reference image on the deformed image. Generally, larger ZOIs are easier to locate on the deformed image. However, this increase in size comes at the cost of longer computation time. Conversely, smaller ZOIs are more challenging to distinguish, but they require less computational effort.

$$\min_w E_{ZOI}(w) = \min_w \int_{ZOI} (I_1(p) - I_2(p + w(p)))^2 dp \quad (2.6)$$

This approach has the advantage to be easily parallelizable as problems are solved independently for each ZOI, no assembly operation is required and furthermore, the number of degrees of freedom to be determined for each zone is small. This allows for very low computational time, making this approach favored by commercial software.

Initially the local methods [18] were developed with the aim of measuring the movements of rigid bodies without rotation, which involves using a piecewise constant shape function base $(\Phi_i)_i$. These works are similar to the Lucas-Kanade [2] optical flow method, which we will explain in detail later. In order to accommodate other types of motion, additional types of basis functions [37] were used, such as linear or polynomial functions.

2.1.2.2 Global Methods

Global methods were introduced after the local methods. The displacement field is directly measured over the entire study area, commonly referred to as Region Of Interest (ROI). The

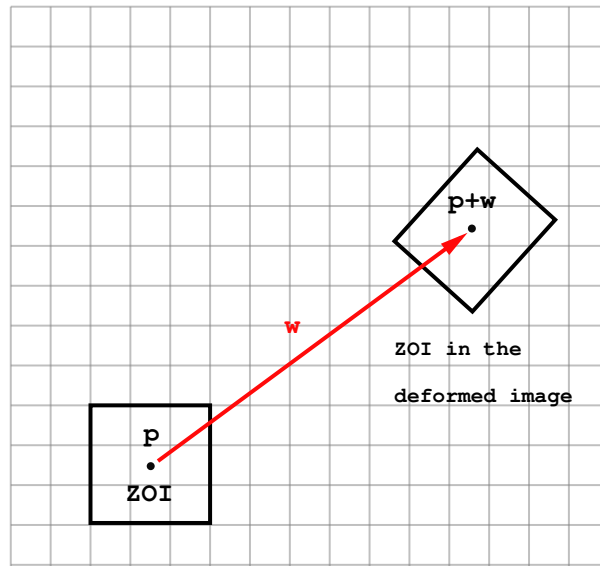


Figure 2.3: Illustration of displacement field modeling and image partitioning for local DIC

global methods allow the continuity of the fields over the whole region of interest by an assembly operation that leads to the creation of a large system to be solved, in contrast to the local approach, where the regularity of the fields is not necessarily guaranteed and the system to be solved is defined locally at each ZOI.

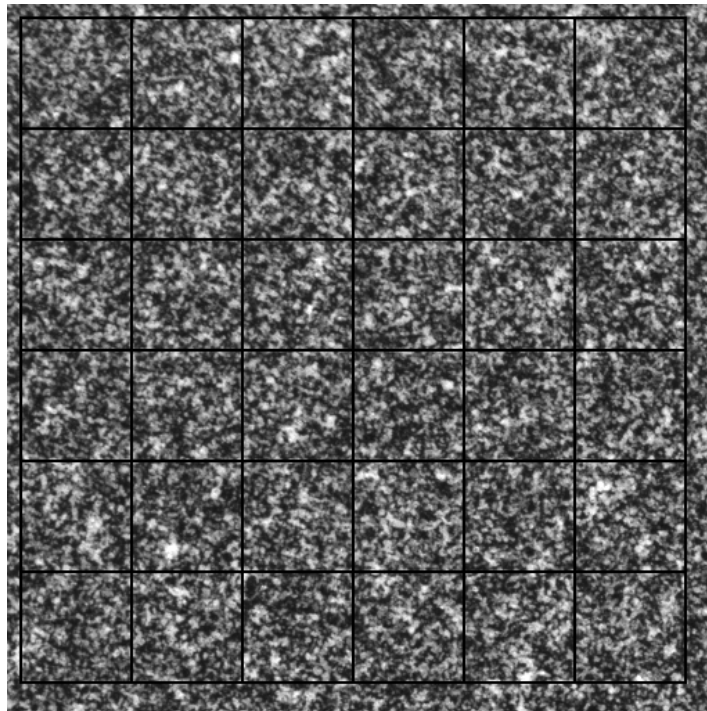


Figure 2.4: Example of a Q4 finite element mesh for global DIC displacement modeling applied to an image with a speckle pattern.

Different basis functions $(\Phi_i)_i$ can be used to interpolate the displacement field over the grids Fig. 2.4. For instance, Lagrange finite element or BSpline [20] basis functions can be used.

Global method based on finite elements are widely employed and have been developed to enable the measurement of continuous displacements using a finite element mesh. Initially, this approach was limited to linear rectangular elements (with 4 nodes) before being generalized to other types of elements [35].

This approach has the advantage of facilitating the exchange of information between simulation and experimentation or measurement by using the same mesh.

This family of DIC methods has the advantage to be easily extended to the case of Digital Volume Correlation. Réthoré [23] has proposed an approach based on eXtended Finite Element method (XFEM) to handle two-dimensional or three-dimensional images involving propagating cracks. This method allows for the propagation of a discontinuity within the material and to avoid its diffusion.

2.1.2.3 Error regimes in finite elements DIC

The measurement uncertainty depends on the size of the finite elements [31, 32], leading to the identification of two distinct error regimes in digital image correlation [97, 33]. Due to the decrease in the number of available pixels within each finite element during correlation, a type of error known as correlation error occurs. Indeed, the decrease in the chosen element size results in a reduction of the amount of information contained within each element, leading to a loss of precision in displacement measurement.

On the other hand, increasing the size of the elements leads to difficulties in accurately representing local displacements within the elements. The interpolation methods used to assign a value to the displacement field within the pixels forming the element can become less accurate as the element size increases. Consequently, interpolation error can introduce additional errors in the measurement of the displacement field. Finally, the user must take into account the choice of element size and implicitly these error regimes in order to obtain accurate measurements of kinematic fields.

2.1.2.4 Tikhonov regularization

The Tikhonov regularization [30] is a method used in various fields to solve inverse problems, particularly in image processing [22]. It aims to obtain a regularized and stable solution by adding a weighted constraint based on the smoothness of the displacement field or the image itself to the objective energy E :

$$E(w) = \int_{\Omega} \left\{ (I_1(p) - I_2(p + w(p)))^2 + \lambda \mathcal{R}(w) \right\} dp \quad (2.7)$$

where λ represents the amplitude of the regularization term \mathcal{R} .

This form of regularization has been added to many digital image correlation codes [59, 25], as it allows them to calculate correlation at the pixel-scale. This enables software to provide results even in poorly meshed areas or when the size of the elements becomes small, which can

destabilize correlation codes. Generally, the regularization term \mathcal{R} is chosen as a differential operator such as a Laplacian or a gradient applied to the displacement field. In this case, the larger the parameter λ , the smoother the fields become. However, determining the optimal regularization parameter value is not straightforward and depends on the processed data and the phenomena under study.

2.1.3. Techniques for improving the full-field measurement

2.1.3.1 Multi-scale resolution

Image correlation approaches use a Taylor expansion to linearize the equation for the displacement, but this expansion is only valid if the spatial increment is small. This leads to limitations in these methods. To avoid the impact of local minima, a multi-scale [28] (also called pyramidal) approach Fig. 3.2 is often used. Instead of considering a single scale for the analysis, this approach divides the image into multiple scales or levels of resolution. The algorithm is initially applied at a coarser resolution scale to capture global deformations of large amplitude. Then, the algorithm is iteratively applied at finer resolution scales to capture local deformations of smaller amplitude.

Assuming, for example, that we have three scales, a reduction factor of 0.5, and starting images of 512×512 pixels. We first create downsampled images, i.e., images of size 256×256 and 128×128 . We start by calculating the correlation between the two smallest-sized images. The computed vector field will be resized to serve as an initialization for the medium-sized images. This process is repeated until the last level, where the correlation step is applied to the original sequence of images.

2.1.3.2 Interpolation

The operation of interpolating graylevels is a necessary operation at each step of the iterative process for updating the right-hand side of Problem (2.5) which depends on a term involving the corrected image $I_2(p+w(p))$. Various interpolation methods can be used, including bilinear interpolation, bicubic interpolation, or even spline-based interpolation. These methods allow for the estimation of missing pixel values by considering the values of neighboring pixels and taking into account the continuity of graylevel variations. Accurate interpolation enhances the correspondence between images by enabling sub-pixel displacement measurement, which can reach up to 10^{-3} pixels [19, 38].

2.1.3.3 Speckle pattern

The speckle pattern is a random texture often added to the surface of tested specimens. It plays a crucial role in digital image correlation as it can help enhance the accuracy of pixel matching. The iterative resolution of the image correlation system (2.5) involves calculating the gradient of the image, which provides the descent direction for the iterative resolution algorithms. Maximizing this gradient throughout the image domain can lead to more accurate results. In general, the speckle pattern consists of a set of circular black and white paint marks

placed randomly with a size preferably between 3 and 5 pixels [36]. Before applying a speckle pattern on a surface, it is important to ensure that the mechanical and thermal properties of the paint being used are suitable for the specific test or application in order to avoid disrupting the quality of the obtained results. For instance, in the case of a high-temperature test, it is important to ensure that the paint used does not melt or peel off.

It is important to mention that there are several techniques for applying a speckle pattern, such as digital printing that can be used to achieve precise speckle patterns. Another popular technique is projection, which involves spraying paint particles using a spraying device such as airbrushes or spray paints.

2.1.4. Conclusion

Throughout this section, we have presented the fundamental principles of digital image correlation (DIC) and the techniques employed for motion estimation. We have detailed the mathematical formalism associated with DIC, as well as the processes of minimization and resolution. Additionally, we have made a clear distinction between global and local approaches and thoroughly discussed their respective advantages and limitations. We also seen that utilizing a multiscale approach enables the detection of large movements, and by employing a suitable speckle pattern along with appropriate interpolation operations, it is possible to enhance the quality of the estimated fields.

Local approaches, which involve subdividing the image into independent problems, yield results that are obtained through pixel windows or sub-images. On the other hand, global approaches ensure continuous fields but demand more computational resources and effort. It is worth noting that global methods based on finite elements have gained popularity and are widely adopted within the mechanics community due to their ability to facilitate communication between experimental and numerical domains.

The selection of the element size in DIC plays a critical role in accurately measuring the fields, as it can introduce different error regimes. In the case of taking into account a regularization term in the energy to be minimized, an additional parameter comes into play besides the size of the elements, which is the the amplitude of this regularizing term. This complicates the adjustment of these parameters, as they both play a similar role.

In the upcoming section, we will explore alternative methods for defining kinematic fields at each pixel of the image. These methods rely on regularization terms, such as Tikhonov regularization, which are commonly employed in image correlation and have already been presented.

2.2. Optical Flow

2.2.1. Introduction

In processing sequences of images, one of the primary challenges is to approximate the motions present in the scenes. This problem is of great importance in several scientific fields. To address this issue, a technique known as optical flow is commonly employed.

Both optical flow and digital image correlation are techniques that aim at extracting motion information. However, they differ in their approach. Optical flow methods describe the displacement vector at each pixel of the image, providing a dense motion field. In contrast, DIC techniques use small elements or subsets of the images to calculate the displacements.

This section aims to explore in depth the fundamental principles of optical flow and explore its underlying assumptions and limitations and discuss various methods for computing it.

2.2.2. Global approach

In 1981 Horn and Schunck presented the first variational method [1] to estimate the optical flow fields. It has been assumed that pixels intensities remained unchanged in the images sequence. This assumption has been utilized not only by the optical flow community but has also been employed in the development of digital image correlation method as shown in the previous section 2.1 (under the name of graylevel conservation) and the problem is formulated using a simple quadratic norm, which lead to the creation of the energy E (2.2).

Despite its convexity, the previous problem is ill-posed because of the small number of equations compared with the number of unknowns. To overcome this issue, a smoothness term has been added to the objective function (2.2).

$$\min_w E(w) = \min_w \int_{\Omega} \left\{ (I_1(p) - I_2(p + w))^2 + \lambda \|\nabla w\|^2(p) \right\} dp, \quad (2.8)$$

where $p = (x, y)$ are the coordinates of the pixel, $w(p) = (u(p), v(p))$ the sought flow field with horizontal and vertical components, ∇ represents the spatial gradient, and Ω is the image domain. λ is a scalar allowing to handle the weight of the smoothness term compared to the first graylevel conservation term. This formulation is similar to the one used in DIC, where a Tikhonov regularization term is added to the energy as in (2.7). The difference between this formulation and the one used in DIC is that the latter is optimized over the entire image domain, pixel by pixel, whereas the energy in DIC is solved over either ZOIs or ROIs with displacement interpolated by shape functions.

Practical calculation Generally, solving the problem (2.8) often involves seeking the best spatial increment as in DIC, denoted as dw , using an iterative process. This approach is referred to as "warping" [14] in the literature.

The minimum is reached where the gradient vanishes. This is translated as the following

optimality condition:

$$\begin{cases} \frac{\partial E}{\partial du} = 0 \\ \frac{\partial E}{\partial dv} = 0 \end{cases} \quad (2.9)$$

Using a first-order Taylor expansion as in (2.4), the optimality condition (2.9) can be written as:

$$\begin{cases} 2((I_x^2 + \lambda\Delta)du + I_x I_y dv + \lambda\Delta u + I_x I_t) = 0, \\ 2(I_x I_y du + (I_y^2 + \lambda\Delta)dv + \lambda\Delta v + I_y I_t) = 0, \end{cases} \quad (2.10)$$

where I_x and I_y are the diagonal matrices containing the spatial derivatives of the corrected image $\tilde{I}_2(p) = I_2(p + w)$ and $I_t = I_1 - \tilde{I}_2$, while Δ represents the Laplace operator. We rewrite these equations in matrix notation, which yields the following linear system:

$$\begin{pmatrix} I_x^2 + \lambda\Delta & I_x I_y \\ I_x I_y & I_y^2 + \lambda\Delta \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix} = - \begin{pmatrix} I_x I_t + \lambda\Delta u \\ I_y I_t + \lambda\Delta v \end{pmatrix} \quad (2.11)$$

One approach to solve the problem consists of approximating Laplace operator using finite differences method and then solving the previous linear system with an iterative algorithm.

Remark. In the first work of Horn and Schunk, the full displacement field (u, v) was directly searched for instead of increments (du, dv) . The Laplacian operator was reformulated in the energy (2.8) using the average of neighboring pixels, i.e. the 5-point stencil:

$$\Delta u \approx 4(\tilde{u} - u), \text{ where } \tilde{u}_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j-1} + u_{i,j+1}).$$

which is a direct consequence of these finite differences approximations:

$$\begin{aligned} u_{i+1,j} &\approx u_{i,j} + \frac{\partial u_{i,j}}{\partial x} + \frac{1}{2} \frac{\partial^2 u_{i,j}}{\partial^2 x} \\ u_{i-1,j} &\approx u_{i,j} - \frac{\partial u_{i,j}}{\partial x} + \frac{1}{2} \frac{\partial^2 u_{i,j}}{\partial^2 x} \\ u_{i,j+1} &\approx u_{i,j} + \frac{\partial u_{i,j}}{\partial y} + \frac{1}{2} \frac{\partial^2 u_{i,j}}{\partial^2 y} \\ u_{i,j-1} &\approx u_{i,j} - \frac{\partial u_{i,j}}{\partial y} + \frac{1}{2} \frac{\partial^2 u_{i,j}}{\partial^2 y} \end{aligned}$$

Then, directly assessing (u, v) without introducing increment, the minimization of the energy E suggested a stationary iteration (with index k):

$$\begin{pmatrix} I_x^2 + 4\lambda I & I_x I_y \\ I_x I_y & I_y^2 + 4\lambda I \end{pmatrix} \begin{pmatrix} u_k \\ v_k \end{pmatrix} = \begin{pmatrix} -I_{t,k} I_x + 4\lambda \tilde{u}_{k-1} \\ -I_{t,k} I_y + 4\lambda \tilde{v}_{k-1} \end{pmatrix} \quad (2.12)$$

The matrix on the left-hand side being made out of diagonal blocks (I is the identity matrix), the system is decoupled for each pixel and direct inversion was possible:

$$\begin{pmatrix} ((I_x^2 + 4\lambda I)(I_y^2 + 4\lambda I) - I_x^2 I_y^2) u_k \\ ((I_x^2 + 4\lambda I)(I_y^2 + 4\lambda I) - I_x^2 I_y^2) v_k \end{pmatrix} = \begin{pmatrix} I_y^2 + 4\lambda I & -I_x I_y \\ -I_x I_y & I_x^2 + 4\lambda I \end{pmatrix} \begin{pmatrix} -I_{t,k} I_x + 4\lambda \tilde{u}_{k-1} \\ -I_{t,k} I_y + 4\lambda \tilde{v}_{k-1} \end{pmatrix} \quad (2.13)$$

which could be simplified as:

$$\begin{pmatrix} u_k \\ v_k \end{pmatrix} = \begin{pmatrix} \tilde{u}_{k-1} - I_x z_{k-1} \\ \tilde{v}_{k-1} - I_y z_{k-1} \end{pmatrix} \text{ with } z_{k-1} = (I_x^2 + I_y^2 + 4\lambda I)(I_{t,k} + I_x \tilde{u}_{k-1} + I_y \tilde{v}_{k-1}) \quad (2.14)$$

New error measurements for enhanced robustness When the euclidean norm of the gradient is used for the smoothness term, it makes the algorithm sensitive to noise and tends to diffuse discontinuities which are quantities of importance in mechanics because they may represent cracks in materials. A first possibility is to weight the smoothing term with the help of an increasing function g [16]:

$$\min_w E(w) = \min_w \int_{\Omega} \left\{ (I_1(p) - I_2(p+w))^2 + g(\nabla I_1) \|\nabla w\|^2 \right\} dx \quad (2.15)$$

It's important to note that, although this regularization, often referred to as "image-driven", is easily implementable, it can be sensitive to noise in the input data, especially in regions where the information is less clear.

Another strategy, which aims to avoid the issue of discontinuity diffusion associated with the least squares norm, was proposed by Black and Anandan [6]. It consists of replacing the L^2 norm by new robust functions. Different penalty functions have been studied [4, 6, 13, 14]. Thus new robust models are proposed to compute the optical flow, where different penalties interleave:

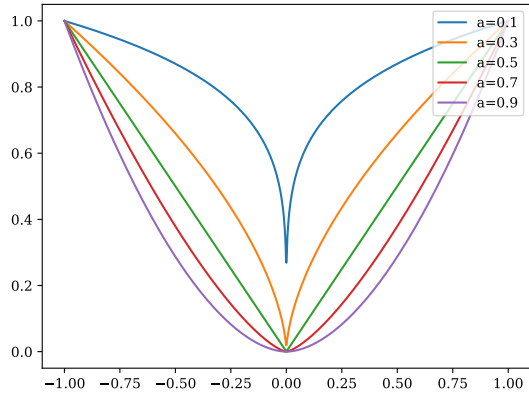
$$E(w) = \int_{\Omega} \left\{ \rho_D(I_1(p) - I_2(p+w)) + \lambda \rho_S(\nabla w) \right\} dp \quad (2.16)$$

where ρ_D is the robust function used for the graylevel conservation and ρ_S the one used for the smoothness. Often one chooses $\rho_D = \rho_S$ in order not to cumulate numerical difficulties.

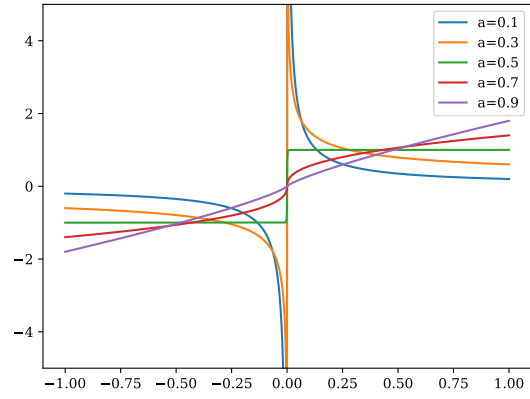
Beside the classical quadratic norm, the most commonly used penalty is the Charbonnier function [15] defined as $\rho(x) = \sqrt{x^2 + \varepsilon^2}$, where ε is a small scalar used to regularize the function, which provides a differentiable approximation of the L^1 norm. In the optical flow literature ε is fixed to be equal 10^{-3} . Charbonnier can be seen as a differentiable variant of the L^1 norm. In Sun's paper [4], the impact of the generalized Charbonnier loss function family $\rho(x) = (x^2 + \varepsilon^2)^a$ has been investigated over the displacements. The generalized Charbonnier is non-convex when $a < 0.5$ and equal to the Charbonnier when $a = 0.5$. This penalty acts like L^1 norm when a tend to 0.5 and acts as a quadratic function when a tends to 1 as shown in Fig. 2.5. An other robust penalty showed its efficiency to compute the optical flow in the presence of outliers. This function is called Lorentzian [6] $\rho(x) = \log(1 + \frac{x^2}{2\sigma^2})$ where σ is a scalar controlling the shape of function. The Lorentzian is a non-convex and therefore the energies where it interleaves require more effort to be optimized. Some technical details for estimating the optical flow using robust functions are given at the appendix 2.3.5.

2.2.3. Local approach

Lucas-Kanade [2] is a different approach for optical flow estimation, classified as "local". In addition to the graylevel conservation, it assumes that that each pixel follows the same movement as its surrounding area.

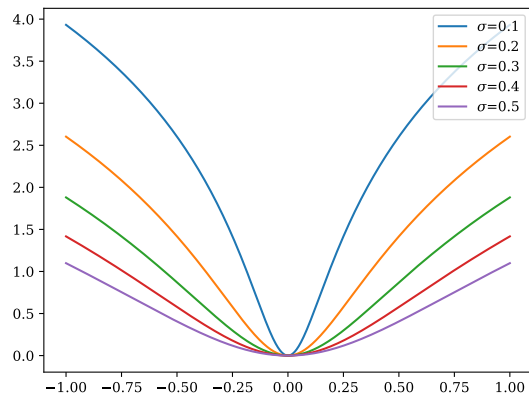


(a) Functions

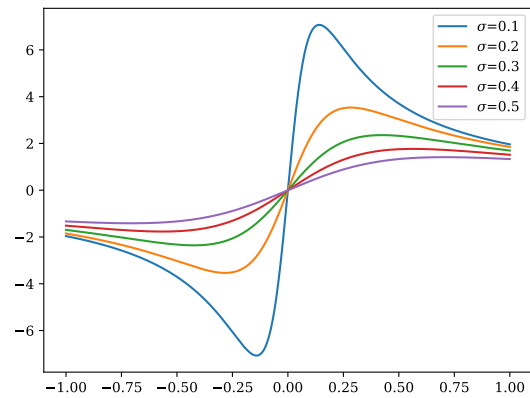


(b) Derivatives

Figure 2.5: Generalized Charbonnier loss function for different values of a .



(a) Functions



(b) Derivatives

Figure 2.6: Lorentzian loss function for different values of σ .

In order to model this assumption, let denote $p_1, p_2 \dots p_n$ the indices of the nearest neighboring pixels. We have previously seen that by using a Taylor expansion, we can rewrite the conservation of graylevels as follows:

$$I_x u + I_y v + I_t = 0$$

After applying this expression to all neighboring pixels, the problem can be expressed locally in a matrix form as:

$$\begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_n) \end{pmatrix}. \quad (2.17)$$

Unlike the first formulation where we had more unknowns, the matrix of the system is rectangular and contains more equations than unknowns, it is thus to be understood in a least-square sense.

Let us introduce the notations:

$$A = \begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{pmatrix}, \quad w = \begin{pmatrix} u \\ v \end{pmatrix}, \quad b = - \begin{pmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_n) \end{pmatrix}$$

The problem (2.17) is then transformed to:

$$\min_w \|Aw - b\|^2 \text{ or equivalently } (A^T A)w = A^T b, \quad (2.18)$$

in detail:

$$\begin{pmatrix} \sum_{i=1}^n I_x(p_i)I_x(p_i) & \sum_{i=1}^n I_x(p_i)I_y(p_i) \\ \sum_{i=1}^n I_x(p_i)I_y(p_i) & \sum_{i=1}^n I_y(p_i)I_y(p_i) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} \sum_{i=1}^n I_x(p_i)I_t(p_i) \\ \sum_{i=1}^n I_y(p_i)I_t(p_i) \end{pmatrix} \quad (2.19)$$

To solve the problem, the matrix $(A^T A)$ must be invertible and well-conditioned. Lukas-Kanade algorithm has the advantage to be easily parallelized since the the problem is defined independently on each pixel.

In (2.19) every pixel exerts the same influence onto the final displacements. In the literature, weights [17] are introduced to link the impact of the pixel onto the solution and its distance from the pixel where the optical flow is calculated. It is quite common to use a normal distribution to define the weight coefficient, noted here $\mu_i, i \in 1, 2, \dots n$.

$$\begin{pmatrix} \sum_{i=1}^n \mu_i I_x(p_i)I_x(p_i) & \sum_{i=1}^n \mu_i I_x(p_i)I_y(p_i) \\ \sum_{i=1}^n \mu_i I_x(p_i)I_y(p_i) & \sum_{i=1}^n \mu_i I_y(p_i)I_y(p_i) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} \sum_{i=1}^n \mu_i I_x(p_i)I_t(p_i) \\ \sum_{i=1}^n \mu_i I_y(p_i)I_t(p_i) \end{pmatrix} \quad (2.20)$$

In addition to its lower computational cost compared to the global Horn and Schunck method, this approach is known for its robustness to noise. However, in the line of local DIC approaches, the Lukas-Kanade method, which can be considered as a local approach for the optical flow estimation, does not result in continuous flow fields. Another disadvantage of this model is the potential loss of local motion features, especially when the spatial resolution of the phenomena is smaller than the size of the window used. As a result, certain local phenomena may escape measurement. Finally, The assumption that pixels follow the same motion is not always satisfied, for example when the considered neighborhood includes pixels on both sides of a boundary.

Remark. Methods for estimating optical flow, whether local or global, commonly employ a multi-scale approach known as “coarse to fine” which has been described in the chapter on digital image correlation (DIC). This approach aims at avoiding local minima during optical flow estimation, particularly when estimating significant motions where the Taylor expansion used to approximate the image $I_2(p + w)$ is not accurately fulfilled. For each level of the pyramid, the optical flow estimation follows an iterative scheme. Finally, it is quite common [7] to apply a median filter to the estimated fields in order to remove outliers.

2.2.4. Conclusion

The optical flow methods are optical capable of predicting motion in a sequence of images. These methods are based on the same assumption as image correlation algorithms, but they are able to provide pixel-scale displacement fields, unlike DIC methods that provide it on image patches. In this chapter, we have presented the Lukas-Kanade model, which is considered to be a local approach for optical flow estimation. It takes advantage of the pixel’s neighborhood to create a signature and search for it in the deformed image. This strategy is different from global methods that bear resemblance to the classical Horn and Schunck model, where a constraint is introduced using a Tikhonov regularization term. Global methods ensure the continuity and smoothness of the estimated field in the image, but they are computationally expensive.

Finally, we have presented the new robust models that use alternative error functions to the L^2 norm, which diffuse discontinuities. These error functions are designed to handle challenging scenarios with outliers, discontinuities and noise in the image sequences. They aim at improving the accuracy and robustness of the estimated optical flow.

In Chapter 3, we will present an article discussing the impact of optical flow parameters and the specific effects of some robust functions (e.g., Charbonnier and Lorentz) on mechanical images, with a particular focus on strain fields.

Regardless of the optical method used, whether it’s digital image correlation or optical flow, and regardless of the chosen approach, the motion estimation often linked to solving a linear system. In the next section, we focus on the presentation and understanding of these methods, with a focus on those targeting very large-scale problems as obtained with the recent development of sensors used for correlation.

2.3. Solvers for linear systems and Parallel computing

2.3.1. Solvers for linear systems

2.3.1.1 Introduction

Linear system solvers are important numerical tools that enable an efficient approximation of the solution of complex systems of linear equations which arise in different scientific disciplines of physics, mathematics or even in computer science, and in particular in optical flow and digital image correlation.

There are various numerical methods to solve such problems. In the literature, they are classified into direct and iterative methods. Direct methods provide the solution after applying a finite number of operations, while iterative methods involve a loop during which an initial vector is updated at each step in order to converge towards the solution. To select an appropriate method, several factors come into play, such as the dimensions, sparsity, symmetry and positivity of the matrix or the accessible computational resources.

The aim of this section is to present an overview of numerical solvers for linear systems, which includes both direct and iterative methods, with a specific emphasis on Krylov solvers which is a subfamily of iterative methods adapted to the resolution of very large systems.

Let A be an invertible $n \times n$ matrix, and b a given vector. We are interested in solving linear systems of the form:

$$Ax = b \tag{2.21}$$

2.3.1.2 Direct methods

Direct solution methods involve the computation of the solution of a given linear system by solving simpler triangular or diagonal systems that are obtained through the factorization of the system matrix. There are several types of matrix factorizations, the most widely recognized in the literature are LU and Cholesky decompositions [40].

LU method factorizes the matrix A into the product of two matrices: a lower triangular matrix L and an upper triangular matrix U , such that $A = LU$. After this factorization, the solution of the system (2.21) can be given by successively solving two triangular systems: $Ly = b$ and $Ux = y$, using forward and backward substitution respectively. This method is commonly used when the matrix has a moderate size, as the computational cost of the LU factorization is $O(n^3)$ [70] for dense matrix.

Cholesky decomposition is a particular case of LU decomposition used for symmetric positive definite matrices. Cholesky decomposition consists into factorizing the matrix A into the product of a lower triangular matrix L and its transpose L^T , such that $A = LL^T$. The computational cost of the Cholesky factorization is $O(\frac{n^3}{3})$ [71], which is less than that of the LU decomposition.

There are other types of matrix decomposition, like the QR -factorization. Making a good choice of the factorization method depends on many factors. Even if sparsity makes it possible to lower the costs, direct methods are generally not chosen to solve large-scale problems such as

those encountered in digital image correlation or optical flow, due to their high time complexity which is linked to the cost of the factorization step to their memory cost since they require the storage of the system matrix.

2.3.1.3 Iterative methods

For solving large-scale problems, iterative methods are often preferred. Moreover, problems in DIC are known to be poorly conditioned and, if stopped sufficiently early, the iterative process can itself be a regularization which limits the necessity of using other corrections.

2.3.1.3.1 Stationary iterative methods

The iterative methods involve repeatedly refining an initial guess to get closer to the final solution. Historically, the first iterative methods used were stationary iterative methods. These methods are defined as approaches that involve modifying a few components of the approximate solution at each iteration until convergence is reached. The most famous examples of stationary iterative methods are the Jacobi, Gauss-Seidel and Successive Over-Relaxation (SOR) methods.

The Jacobi method calculates the new approximation of the solution at a given iteration using only the components calculated in the previous iteration, whereas the Gauss-Seidel method takes into consideration the newly computed components at the current step of the process. The Jacobi method is easy to implement but it has the disadvantage of being relatively slow compared to other stationary methods mentioned. The SOR method represents an enhancement of the Gauss-Seidel method, utilizing a relaxation parameter to accelerate the convergence process.

Algorithm 1 SOR Algorithm

Require: $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $x^0 \in \mathbb{R}^n$ initial guess, ε the desired tolerance, the relaxation parameter ω

Ensure: x^k an approximation of the solution x of (2.21)

```

1:  $k \leftarrow 1$ 
2:  $Err \leftarrow \|Ax^0 - b\|$ 
3: while  $Err > \varepsilon$  do
4:   for  $i = 1, \dots, n$  do
5:     
$$x_i^k = (1 - \omega)x_i^{(k-1)} + \frac{\omega}{A_{ii}}(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^k - \sum_{j=i+1}^n A_{ij}x_j^{(k-1)})$$

6:   end for
7:    $Err \leftarrow \|Ax^k - b\|$ 
8:   if  $Err < \varepsilon$  then
9:     Break
10:  end if
11:   $k \leftarrow k + 1$ 
12: end while

```

Despite their ease of implementation, the stationary iterative methods have some drawbacks. They are very sensitive to the choice of initial values and the choice of parameters that are difficult to estimate. Moreover, they may not converge for all matrices, especially those that are poorly conditioned, and they are slow to converge, especially for large matrices that can be solved more quickly with Krylov solvers.

2.3.1.3.2 Krylov iterative methods

Krylov methods are a family of iterative methods that are widely used as they allow for the solution of large sparse linear systems. We consider in this section the same system (2.21), except that in this case the dimension n of the matrix A is very large. These solvers consist of seeking an approximation of the solution of the linear system x by projecting the initial problem onto a vector subspace of dimension m called the Krylov subspace, which we will denote by K_m . In addition to its dimension, a Krylov subspace depends on a matrix B and a vector v , it is defined as:

$$K_m(B, v) = \text{span} \{v, Bv, \dots, B^{m-1}v\} \quad (2.22)$$

Let x^0 be an initial approximation of the solution x and $r^0 = b - Ax^0$ be the initial residual. It has been shown [60] that the exact solution x belongs to the affine subspace $x^0 + K_m(A, r^0)$. Krylov methods are part of polynomial methods that seek to approximate the inverse of the matrix A^{-1} by a polynomial $p_{m-1}(A)$ and to deduce the solution in the form $x = x^0 + p_{m-1}(A)r^0$. The Krylov methods also assume that the residual r^m is orthogonal to a vector subspace L_m of the same dimension as K_m ($r^m = b - Ax^m \perp L_m$). These projection methods differ by the choice of the subspaces [41] K_m and L_m . For instance, the Full Orthogonalization Method (FOM) [46] and Conjugate Gradient (GC) [44] seeks a solution such that the residual is orthogonal to $L_m = K_m(A, r^0)$, Bi-Conjugate Gradient method (BiGC) [45] uses $L_m = K_m(A^T, r^0)$, while Generalized Minimal Residual (GMRES) [43] use $L_m = AK_m(A, r^0)$. The GMRES method can be used with arbitrary matrices, which makes it more general than other methods. Several methods are derived from GMRES, such as the example of the MINRES [39] method, which works on symmetric linear equation systems. MINRES takes advantage of the symmetry of the matrix to reduce the time and space complexity of the GMRES method. In the case of symmetric and positive-definite linear problems like those encountered in optical flow or digital image correlation, the well-known and efficient conjugate gradient method is often used.

The time complexity of the conjugate gradient method is of the order of $O(n^2)$ per iteration, making the conjugate gradient faster for large-scale problems. Finally, Krylov methods guarantee that the convergence to the exact solution of the linear system is achieved at most n iterations, even though a satisfying approximation can be achieved much faster.

2.3.2. Preconditioning

The conditioning or the condition number of a matrix A is the value, noted $\kappa(A)$, that quantifies the sensitivity of its solutions to change in the input data. The closer to 1 this value, the better

Algorithm 2 GMRES Algorithm

Require: $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $x^0 \in \mathbb{R}^n$ initial guess, $m \ll n$, ε the desired tolerance

Ensure: An approximation of the solution x of (2.21)

```
1:  $r^0 \leftarrow b - Ax^0$ 
2:  $v_1 \leftarrow \frac{r^0}{\|r^0\|}$ 
3: for  $j = 1, \dots, m$  do
4:    $w \leftarrow Av_j$ 
5:   for  $i = 1, \dots, j$  do
6:      $h_{i,j} \leftarrow \langle w, v_i \rangle$ 
7:      $w \leftarrow w - h_{i,j}v_i$ 
8:   end for
9:    $h_{j+1,j} \leftarrow \|w\|$ 
10:  if  $h_{j+1,j} = 0$  then return  $x_j$ 
11:  end if
12:   $v_{j+1} \leftarrow \frac{w}{h_{j+1,j}}$ 
13:   $y_j \leftarrow \min_{y \in \mathbb{R}^j} \|b - Ax_j - AV_j y\|_2$ 
14:  if  $\|b - Ax_j - AV_j y_j\|_2 \leq \varepsilon$  then
15:    return  $x_j + V_j y_j$ 
16:  end if
17: end for
18: return  $x^m + V_m y_m$ 
```

Algorithm 3 Conjugate Gradient Algorithm

Require: $A \in \mathbb{R}^{n \times n}$ symmetric and positive-definite, $b \in \mathbb{R}^n$, $x^0 \in \mathbb{R}^n$ initial guess, ε the desired tolerance

Ensure: An approximation of the solution x of $Ax = b$

```
1:  $r^0 \leftarrow b - Ax^0$ 
2:  $p^0 = r^0$ 
3: for  $k = 0, 1, 2, \dots$  do
4:    $\alpha_k = \frac{(r^k, r^k)}{(Ap^k, p^k)}$ 
5:    $x^{k+1} = x^k + \alpha_k p^k$ 
6:    $r^{k+1} = r^k - \alpha_k Ap^k$ 
7:   if  $r^{k+1} < \varepsilon$  then
8:     Break
9:   end if
10:   $\beta_k = \frac{(r^{k+1}, r^{k+1})}{(r^k, r^k)}$ 
11:   $p^{k+1} = r^{k+1} + \beta_k p^k$ 
12: end for
```

conditioned the matrix is, and conversely, if this value is large, the matrix is said to be ill-conditioned.

$$\kappa(A) = \|A\| \|A^{-1}\|, \text{ for a matrix norm.} \quad (2.23)$$

When a matrix is well-conditioned, small perturbations in the input data will not have a large influence on the output result, while an ill-conditioned matrix is one where small changes in the input can cause large changes in the output result. The conditioning of a matrix can affect the performance of iterative methods. These methods can converge quickly and efficiently to an approximate solution when the matrices are well-conditioned, while they may converge slowly by requiring more iterations or not converge at all if the matrices have high conditioning. The preconditioning technique involves using an invertible matrix M such that the product $M^{-1}A$ is better conditioned than A itself. Hence, the new system to be solved is:

$$M^{-1}Ax = M^{-1}b \quad (2.24)$$

The solution of this modified system is identical to that of the original system, yet it may exhibit faster convergence when solved. The perfect system to be solved is one where the condition number is equal to 1, which corresponds to the identity matrix, which is equivalent to choosing a matrix M^{-1} that coincides with the inverse of matrix A . Since the calculation of the matrix inverse is expensive, and it becomes even more costly when the matrix is very large, the idea is to seek an approximation of it. This approximation is as difficult as the initial problem. Various techniques are used to construct the preconditioner. The following is a brief overview of some of these techniques.

2.3.2.1 Matrix decomposition

The most well-known and easy-to-implement preconditioners are based on the decomposition of the matrix A . The easiest example to implement is called the Jacobi diagonal preconditioner detailed in [42], which is defined by a diagonal matrix containing the inverse of the diagonal entries of matrix A which means that $M^{-1} = \text{diag}(A)^{-1}$.

Other commonly used preconditioners are Gauss-Seidel defined by setting $M = (D - L)$ and the symmetric SOR (SSOR) [42], which is defined by setting $M = \frac{1}{\omega(2 - \omega)}(D - \omega L)D^{-1}(D - \omega U)$, where D , L , and U are respectively the diagonal, lower triangular, and upper triangular parts of the matrix $A = L - D - U$, and ω is a scalar.

2.3.2.2 Approximate Inverse Preconditioner

The goal of this approach is to seek a preconditioner for the system by solving the following minimization problem:

$$M^{-1} = \arg \min_P \|I - PA\|$$

The approximate inverse translates to the search for a matrix whose product with matrix A is very close to the identity matrix which is a well-conditioned matrix, this lead to a remarkable improvement of performance, but this kind of preconditioners is computationally expensive to calculate. Generally, the Frobenius norm is used for formulating the optimization problem, but other norms can also be used.

2.3.2.3 Incomplete factorization

Unlike direct methods such as LU decomposition or Cholesky decomposition that are memory-intensive due to fill-in during the factorization stage. The incomplete factorization provides an approximation of the matrix by limiting the fill-ins, and finally, the inverse of this decomposition is used as a preconditioner. Incomplete LU factorization is often used for sparse matrices with many zero coefficients.

The most common and simplest approach consists of allowing fill-in only where it already exists in the initial matrix which leads to the preconditioner $ILU(0)$. While constructing the preconditioner through this approach is simple and effective, it may deviate significantly from A^{-1} in certain cases. This is because incomplete LU decomposition leaves out a significant amount of information. However, another approach that allows for some fill-ins can lead to an improvement in the quality of the preconditioner, except that increasing fill-ins causes an increase in the computational cost associated with matrix-vector products and with the ILU factorization.

There are many other types of preconditioning methods, like the ones based on multigrid or domain decomposition methods. In the current section, we attempted to briefly present the most classical and simple methods. For more details, we refer to the work of Saad [42].

2.3.3. Parallel computing

The presence of parallel processing in scientific computing codes has become increasingly prominent, primarily due to technological advancements and to the growing demands of users. Parallel computing involves harnessing the architecture and computational power of machines, which have undergone significant advancements in recent times.

The measurement of the kinematic fields using DIC or optical flow techniques often involves processing large quantities of images and complex data generated by increasingly advanced sensors. In this context, parallel computing allows not only for simultaneous operations on multiple pixels, regions, or image windows but also for the application of specific tasks to multiple data in parallel. This results in a reduction in the processing time of these data and improves the ability to process real-time data, which is required for certain types of experiments. In the following, we will describe the distinction between parallel programming models and parallel execution models.

2.3.3.1 Parallel programming models

The programming models refer to the manner in which the programmer specifies parallel tasks or operations within a program. Two parallel programming models are generally distinguished: data parallelism and task parallelism.

2.3.3.1.1 Data parallelism

Data parallelism [72] is a parallel programming model that aims to apply the same operation simultaneously to different data packets. This model involves subdividing a large dataset into smaller segments and assigning each segment to a processing unit. Subsequently, these

processing units execute the same task in parallel. This programming model is particularly useful when dealing with a large amount of data, such as in the case of digital volume correlation. For instance, let's assume that we have a machine with four computing units and a matrix. The objective is to increment the elements of this matrix. Data parallelism involves executing the increment operation simultaneously on multiple processing cores, with each core processing a different set of data, rather than performing the calculation using a single core and thus each processor handles a part of the initial array to be incremented as shown in Fig 2.7. The execution time in sequential mode will be then divided by the number of parallel cores, which theoretically leads to a speedup of 4 in this example.

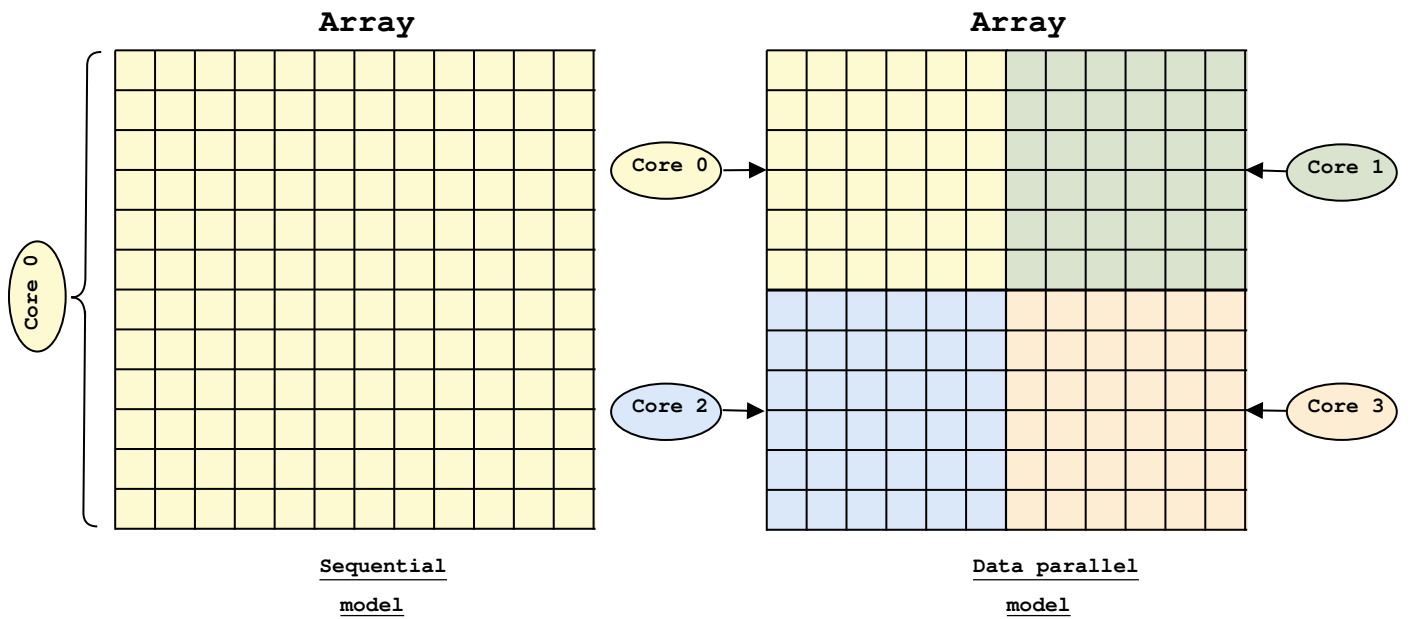


Figure 2.7: Illustration of the data parallel model

2.3.3.1.2 Task parallelism

Task parallelism [73] is a parallel programming model that enables the simultaneous execution of different tasks on separate computing units. In contrast to the previously described data parallel model, this model does not focus on the data and it is used to divide the computational code into multiple independent tasks that can be executed in parallel on different computing units. This allows for improved performance by leveraging the hardware architecture Fig 2.8.

2.3.3.2 Parallel execution models

Execution models serve as a bridge between hardware architecture and the software layers available for applications. Execution models differ from the chosen language or programming interfaces. Indeed, execution models are an abstraction of the system that allows us to express the algorithm as well as the data structures. Unlike programming languages and APIs, which provide information about the implementation of these models and how they will be executed, execution models are independent of the choice of language or APIs.

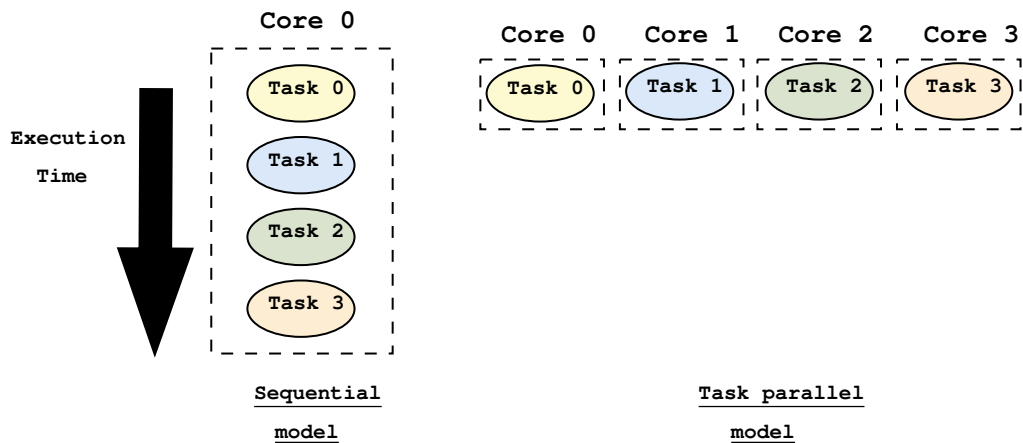


Figure 2.8: Illustration of the task parallel model

There are several parallel execution models, and the most well-known ones will be presented in the following.

2.3.3.2.1 Shared memory model

This model is characterized by the fact that all processes share a common address space in which they read and write asynchronously. The shared memory execution model is one of the simplest models to understand. One advantage of this model is the absence of the notion of data ownership since all processes share the same memory, which they can access for reading or writing. Memory accesses in this model are controlled by mechanisms such as locks or semaphores [74].

There are several libraries for targeting this type of memory architectures, but OpenMP remains predominantly used. It is a parallel programming API compatible with C, C++, and Fortran, which utilizes directives and libraries to add parallelism to sections of code.

This model is limited generally by the size of the machine's memory. To handle problems whose size exceeds the capacity of shared memory, a distributed memory model is often chosen.

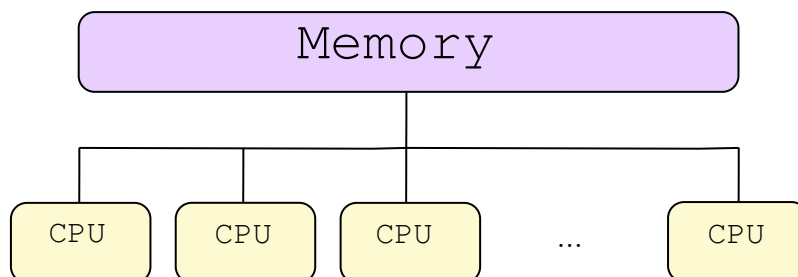


Figure 2.9: The shared memory model

2.3.3.2 Distributed memory model

The distributed memory execution model is characterized by the fact that each set of tasks, whether they are located on the same machine or different machines, use their own resources during computation. When necessary, tasks communicate with each other to exchange data using messages. The data transfer requires cooperation from each process, for example, a send operation requires a corresponding receive operation, which means that most of the communications (data transfer) are managed by the developer and can have an impact on the performance of the computation. The most widely used library for managing communication between compute nodes in distributed environments is MPI, which is also compatible with C, C++, Fortran, Python and other languages.

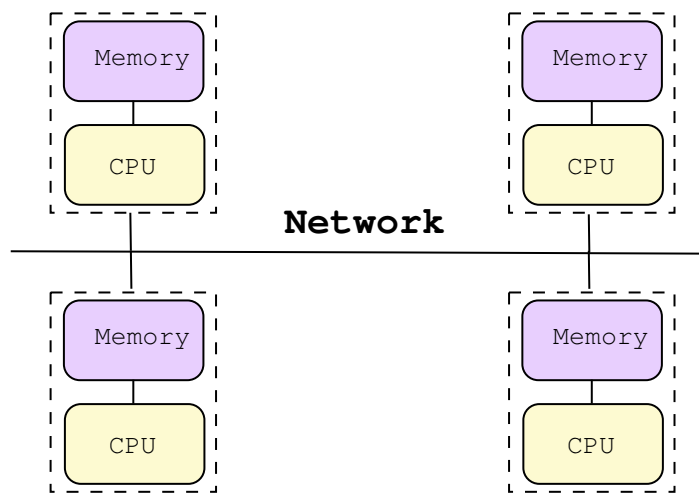


Figure 2.10: The distributed memory model

2.3.3.2.3 Hybrid memory model

Hybrid memory supercomputers play a significant role in the high-performance computing domain, and hybrid execution models are particularly well-suited for such machines because they combine multiple types of memory. A hybrid memory model is a model that combines multiple execution models, such as a combination of shared memory and distributed memory. This can involve a system comprising multiple compute nodes, where each node has its own local shared memory. However, communication between nodes is performed using a network. The goal of this type of model is to exploit the advantages of each model when the hardware architecture allows it and it gives the possibility for multiple levels of parallelism which leads to an improvement in performance.

To take advantage of intranode and internode parallelism, a combination of libraries is often used. The first one is used to accelerate calculations on parts with shared memory, such as OpenMP, and another one handles inter-processor communications, for example, MPI.

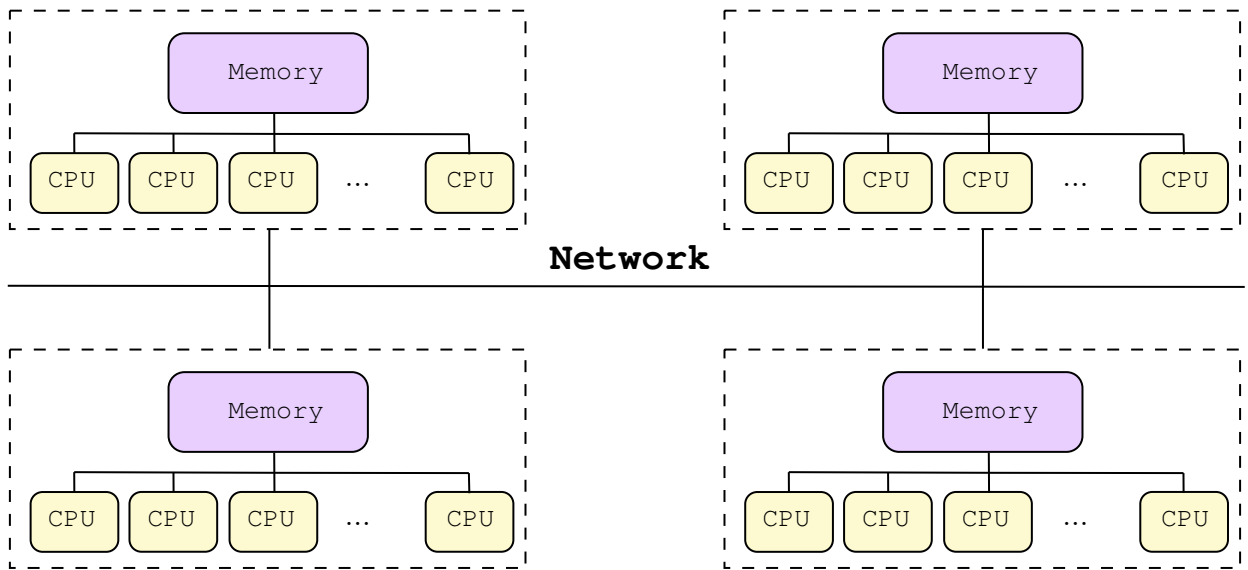


Figure 2.11: The hybrid memory model

2.3.4. General-purpose computing on graphics processing units

Graphics Processing Units (GPUs) are electronic circuits primarily used for graphics processing in gaming applications before exploiting their performance for numerical computations and massively parallel tasks. This usage is called GPGPU, which stands for General-purpose computing on graphics processing units. GPUs enable parallelism for repetitive and highly parallel tasks due to their vast number of computing cores, which can execute multiple operations simultaneously. Moreover, graphics cards are designed for data parallelism and utilize a SIMD [66] (Single Instruction, Multiple Data) architecture, allowing a single set of instructions to be applied concurrently to multiple data elements.

To facilitate code development and execution of computations on this type of accelerators, there are several software environments available. These platforms provide specific libraries that enable developers to fully harness the power of GPUs. Among the most widely used platforms are OpenCL [67] and CUDA [68]. OpenCL is a parallel programming platform developed by the Khronos Group, which can be used on a variety of GPUs from different manufacturers, making the code more portable as it can be executed on different accelerator architectures in contrast to CUDA which is the most well-known GPU programming platform, developed by Nvidia and which allows developers to utilize libraries to write code with C/C++ that can be executed on Nvidia GPUs.

PyCUDA [69] has made GPU programming accessible to Python users, which is a widely used and straightforward programming language within the mechanics community. PyCUDA serves as a Python wrapper, enabling the creation of GPU-accelerated code using Python by encapsulating CUDA functions within a Python interface. Even though this interface is easy to use and is called from Python, the CUDA kernels are written using C/C++, which makes the writing process more challenging for developers who are not familiar with C programming or programming on the GPU. To provide a higher level of abstraction than what is offered

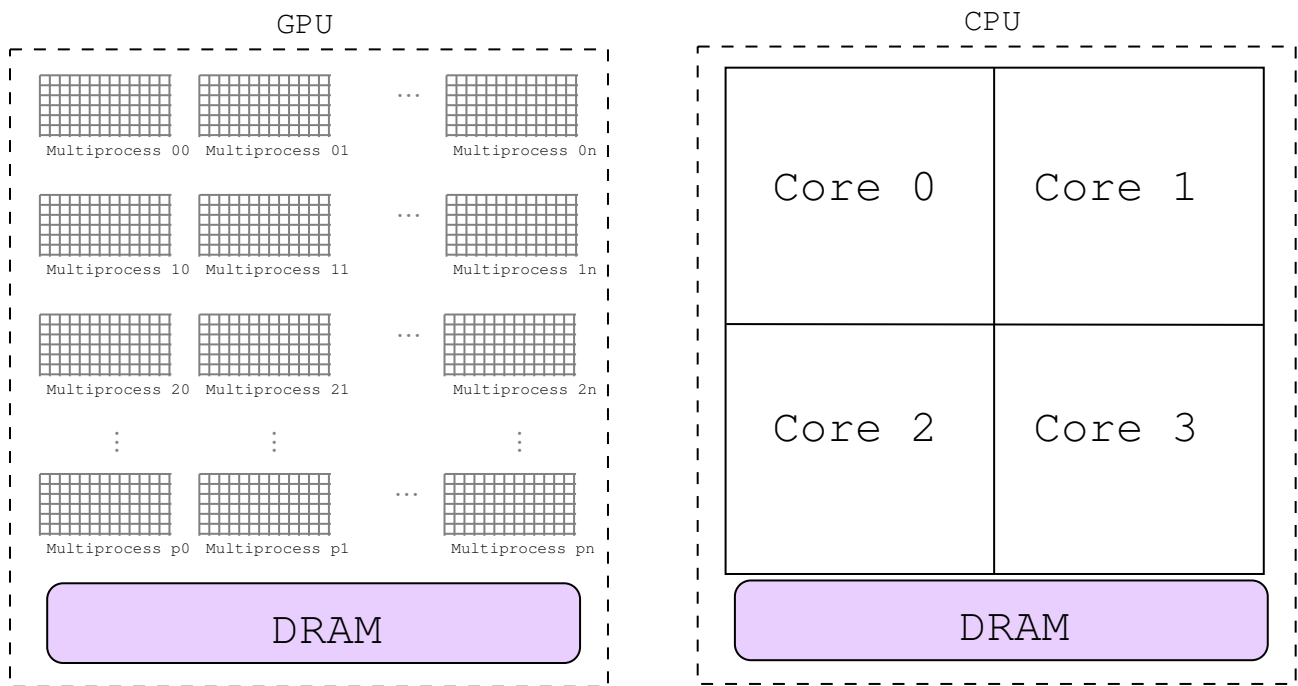


Figure 2.12: GPU and CPU Architectures

by PyCUDA, where the user is required to write the source code themselves, NVIDIA has developed a library called CuPy [47]. It incorporates several basic matrix calculation functions already developed with NumPy, thus avoiding the need for the user to write complex code, as is the case with PyCUDA. With CuPy, users can take the advantage of these pre-implemented functions, making it more accessible and user-friendly for performing GPU-accelerated matrix computations without the necessity to manually write intricate code like in PyCUDA.

2.3.5. Conclusion

Through this section, various well-known methods for solving linear systems have been introduced. We started with direct methods, which are capable of solving problems in a finite number of operations. We also introduced iterative methods, which involve using an iterative scheme to approach the solution based on a given initialization, such as those encountered in digital image correlation or optical flow. Krylov methods consist of solving the problem in a lower-dimensional space compared to the original problem. Among a variety of algorithms, two methods were presented: GMRES, which operates in the general case where no assumption other than invertibility is made about the nature of the system matrix, and the conjugate gradient method, which is suitable for symmetric positive definite matrices.

The solution of large-scale linear systems is often associated with a significant demand in terms of computational resources and memory. The resolution of large-scale linear systems is often faced with such a high demand for computational and memory resources that it becomes impossible to execute this resolution operation due to the limitations of computing machines or the significantly high computational time required. By using parallelism strategies, it is possible

to distribute the computation across different compute nodes, which accelerates significantly the resolution process. This highlights the importance of presenting various parallelism strategies to enhance the overall efficiency of solving large-scale linear systems. Task parallelism is a strategy that involves executing multiple tasks simultaneously, each representing a distinct operation. This approach efficiently utilizes computing resources by executing tasks independently and concurrently. On the other hand, data parallelism aims to execute the same operation on smaller portions of data, obtained by subdividing the initial data. This allows for parallel computation on different data partitions, which can accelerate the overall processing and allow for solving larger systems. We also discussed GPU architectures and how computations are executed within such an architecture before introducing PyCUDA and Cupy a couple of modules that enable the utilization of these hardware accelerators with Python.

After selecting a programming model based on available resources and the nature of the problem to be solved, it is necessary to choose an appropriate execution model to ensure efficient execution of the parallel program. Finally, understanding programming models and execution models can help optimize parallel programs, efficiently utilize hardware resources, and achieve good performance.

Appendix: Optical flow estimation

In this section, we provide some technical details used to iteratively optimize the optical flow problem (3.5) with an anisotropic regularization term. The energy to be minimized here differs from the one detailed in [96], where an isotropic regularization term is employed. We use the Iterative Reweighted Least Squares (IRLS) [89] approach. We can choose two different regularization metrics, ρ_D for the term associated with the image, and ρ_S for the one related to regularization. Regardless of the chosen metric, it can be expressed as a function of the square of the variable. We will use this notation, which will simplify the calculations.

The method aims to find the best increments du and dv so that the gradient of the objective over these increments is null. The problem (2.16) can be written as:

$$E(du, dv) = \int_{\Omega} \left\{ \rho_D((I_2(p + w + dw) - I_1(p))^2) + \lambda [\rho_S(\nabla_x(u + du)^2) + \rho_S(\nabla_y(u + du)^2) + \rho_S(\nabla_x(v + dv)^2) + \rho_S(\nabla_y(v + dv)^2)] \right\} dp \quad (2.25)$$

Given that: $p = (x, y, t)$ are the coordinates of the pixel, $w(p) = (u(p), v(p), 1)$ where u and v are the horizontal and vertical flow fields respectively. Finally we denote by $\nabla_x *^2 = \frac{\partial *^2}{\partial x}$ and $\nabla_y *^2 = \frac{\partial *^2}{\partial y}$, and:

$$I_t(p) = I_2(p + w) - I_1(p), \quad I_x(p) = \nabla_x I_2(p + w), \quad I_y(p) = \nabla_y I_2(p + w)$$

Let \mathbf{u} and \mathbf{v} be the vectorized flow fields u and v respectively. To simplify the notations, we keep ∇_x and ∇_y to denote the discrete derivative filters, $\nabla_x \mathbf{u} = \mathbf{u} * \frac{1}{2} \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$. We denote

by δ_p the vector column with an only non zeros value equal to one at the pixel $p = (x, y)$. After employing a first-order Taylor expansion and the descritization of the equation (2.25), we obtain:

$$E(\mathbf{du}, \mathbf{dv}) = \sum_p \rho_D (\delta_p^T (I_t + I_x \mathbf{du} + I_y \mathbf{dv}))^2 + \lambda [\rho_S ((\delta_p^T \nabla_x (\mathbf{u} + \mathbf{du}))^2) + \rho_S ((\delta_p^T \nabla_y (\mathbf{u} + \mathbf{du}))^2) + \rho_S ((\delta_p^T \nabla_x (\mathbf{v} + \mathbf{dv}))^2) + \rho_S ((\delta_p^T \nabla_y (\mathbf{v} + \mathbf{dv}))^2)] \quad (2.26)$$

and

$$\frac{\partial E}{\partial \mathbf{du}} = 0 \quad \text{and} \quad \frac{\partial E}{\partial \mathbf{dv}} = 0 \quad (2.27)$$

We denote by g_{px} , g_{py} , and f_p the following expressions:

$$\begin{aligned} g_{px} &= (\delta_p^T \nabla_x (\mathbf{u} + \mathbf{du}))^2 \\ g_{py} &= (\delta_p^T \nabla_y (\mathbf{u} + \mathbf{du}))^2 \\ f_p &= \delta_p^T (I_t + I_x \mathbf{du} + I_y \mathbf{dv})^2 \end{aligned} \quad (2.28)$$

Then:

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{du}} &= \sum_p \rho'_D (f_p) \frac{\partial f_p}{\partial \mathbf{du}} + \lambda \left\{ \rho'_S (g_{px}) \frac{\partial g_{px}}{\partial \mathbf{du}} + \rho'_S (g_{py}) \frac{\partial g_{py}}{\partial \mathbf{du}} \right\} \\ &= 2 \sum_p \rho'_D (f_p) (I_x \delta_p \delta_p^T I_x \mathbf{du} + I_x \delta_p \delta_p^T (I_t + I_y \mathbf{dv})) + \lambda \left\{ \rho'_S (g_{px}) \nabla_x^T \delta_p \delta_p^T \nabla_x \right. \\ &\quad \left. + \rho'_S (g_{py}) \nabla_y^T \delta_p \delta_p^T \nabla_y \right\} (\mathbf{du} + \mathbf{u}) \end{aligned} \quad (2.29)$$

Let us define a generalized Laplace filter related to the horizontal field L_u as:

$$L_u = \nabla_x^T \rho'_{S1} \nabla_x + \nabla_y^T \rho'_{S2} \nabla_y$$

where $\rho'_{S1} = \text{diag}(\rho'_S (g_{px}))$, $\rho'_{S2} = \text{diag}(\rho'_S (g_{py}))$, and $\rho'_D = \text{diag}(\rho'_D (f_p))$.

The expression (2.29) can be written as

$$\frac{\partial E}{\partial \mathbf{du}} = 2((\rho'_D I_x^2 + \lambda L_u) \mathbf{du} + \rho'_D I_x I_y \mathbf{dv} + \rho'_D I_x I_t + \lambda L_u \mathbf{u}) \quad (2.30)$$

Similarly, we obtain the following expression by deriving the objective E with respect to \mathbf{dv} :

$$\frac{\partial E}{\partial \mathbf{dv}} = 2(\rho'_D I_x I_y \mathbf{du} + (\rho'_D I_y^2 + \lambda L_v) \mathbf{dv} + \rho'_S I_y I_t + \lambda L_v \mathbf{v}) \quad (2.31)$$

Given that:

$$\begin{aligned} L_v &= \nabla_x^T \rho'_{S3} \nabla_x + \nabla_y^T \rho'_{S4} \nabla_y \\ \rho'_{S3} &= \text{diag}(\rho'_S (h_{px})), \quad \rho'_{S4} = \text{diag}(\rho'_S (h_{py})) \\ h_{px} &= (\delta_p^T \nabla_x (\mathbf{v} + \mathbf{dv}))^2 \\ h_{py} &= (\delta_p^T \nabla_y (\mathbf{v} + \mathbf{dv}))^2 \end{aligned}$$

The equations (2.30) and (2.31) can be written as:

$$\begin{pmatrix} \rho'_D I_x^2 + \lambda L_u & \rho'_D I_x I_y \\ \rho'_D I_x I_y & \rho'_D I_y^2 + \lambda L_v \end{pmatrix} \begin{pmatrix} \mathbf{d}\mathbf{u} \\ \mathbf{d}\mathbf{v} \end{pmatrix} = - \begin{pmatrix} \rho'_D I_x I_t + \lambda L_u \mathbf{u} \\ \rho'_D I_y I_t + \lambda L_v \mathbf{v} \end{pmatrix} \quad (2.32)$$

Since the system (2.32) contains nonlinear terms (that need to be updated with $(\mathbf{d}\mathbf{u}, \mathbf{d}\mathbf{v})$), the main problem of minimization (2.27) is solved using a fixed-point algorithm. Since the problem is high-dimensional, Krylov iterative methods [60] can be used to solve the inner linear system as shown in the previous chapter.

Algorithm 4 IRLS algorithm

- 1: Initialization $\mathbf{d}\mathbf{u}^0, \mathbf{d}\mathbf{v}^0$
 - 2: At each iteration k :
 - a: Compute $\rho'_{S1}, \rho'_{S2}, \rho'_{S3}, \rho'_{S4}$ and ρ'_D using the known fields $\mathbf{d}\mathbf{u}^{k-1}, \mathbf{d}\mathbf{v}^{k-1}$
 - b: Compute L_u and L_v
 - c: Solve the linear system (2.32) and deduct $\mathbf{d}\mathbf{u}^k, \mathbf{d}\mathbf{v}^k$
 - d: Test convergence over $(\mathbf{d}\mathbf{u}^{k-1}, \mathbf{d}\mathbf{v}^{k-1})$ and $(\mathbf{d}\mathbf{u}^k, \mathbf{d}\mathbf{v}^k)$.
 - e: If convergence Stop else $k \leftarrow k + 1$.
-

3. The impact of metrics in mechanical imaging

In the previous section, we reported that several error functions which encourage the appearance of local phenomena are utilized instead of the quadratic function which is known to smooth the discontinuities. The objective of this chapter is to gain a better understanding of these metrics in the optical flow methods and to give details about the often underexplored topic of metric selection in the literature. Additionally, this chapter aims to explore the impact of regularization on the nature of the estimated results. Our approach is based on one of the key articles in optical flow titled "Secrets of optical flow estimation and their principles" [4] written by D. Sun in 2010, which introduced enhanced optical flow methods with inspiration from modern techniques, as well as more sophisticated non-classical models. Several robust functions resilient to outliers were proposed in the previous work of optical flow estimation. We focus on those studied in the cited paper [4]. This includes the Lorentz function and the Charbonnier function, which has a formulation that resembles the L^1 norm, with a non-continuous derivative. We will subsequently explore the influence of the regularization amplitude as well as the localized impact of changing this parameter. Furthermore, concerted efforts have been made to provide a fast and efficient code so that the mechanics community can test these metrics on their images. This chapter is presented in the format of a scientific article [85], which is being submitted to the Strain ISSN:1475-1305 journal.

Authors Ahmed Chabib, Jean-François Witz, Pierre Gosselet, Vincent Magnier.

Affiliation Univ. Lille, CNRS, Centrale Lille, UMR 9013 - LaMcube - Laboratoire de Mécanique, Multiphysique, Multi-échelle, F-59000 Lille, France.

Abstract Optical Flow (OF) is an alternative technique to the more classical digital image correlation methods (DIC), able to measure the motion between two images. It is a pixel-wise method, in the sense that the displacement is defined at each pixel and its computation does not require overpixel grids. This paper examines the impact on the identified strain field, the main quantity of interest for solid mechanics, of the metrics used to quantify the conservation of the optical flow and to impose the regularity of the displacement. The Charbonnier and Lorentzian loss functions are inspected in this work. A new regularization approach is used to locally vary the Tikhonov parameter using a mask in order to preserve the discontinuities and encourage the appearance of local phenomena while smoothing the computed fields elsewhere. Finally, an open-source GPU-accelerated python code has been implemented.

Keywords DIC, Optical Flow, GPU, Strain, Measurement

3.1. Introduction

Digital Image Correlation (DIC) and Optical Flow (OF) refer to techniques that analyze a sequence of images in order to estimate the displacement. In recent years, DIC has gained popularity as an efficient technique to analyze digital images of materials under testing, allowing for the high-resolution full-field measurement of strain. The basic principle of DIC is to track the displacement of a surface or a volume by comparing digital images taken before and after deformation. DIC methods involve partitioning images into smaller regions and then analyzing their intensity patterns in order to identify corresponding subsets between two images. The intensity patterns are encoded using a speckle or random dot pattern applied to the surface (or by the variation of the texture in the volume), and the resulting data can be presented as displacement or strain maps. These maps provide useful information about the mechanical behavior of the material or the structure being analyzed. Besides its full-field advantage compared to point sensors, DIC is a non-destructive technique that does not require direct contact with the surface, making it a suitable option to analyze fragile or sensitive materials.

Optical Flow is a popular method used in the field of computer vision and image processing to measure displacement and motion fields. This technique also provides full-field measurements and does not require a physical contact with the object being analyzed as DIC. After more than 40 years of research, optical flow models have steadily improved in accuracy over time. Unlike DIC, these models are capable of describing the movement at every pixel of the image. Both Optical flow and DIC techniques are based on graylevels conservation assumption (3.1) introduced by Horn and Schunk [1], assuming that the graylevels of the pixels do not change during the motion:

$$I_1(x) = I_2(x + u(x)), \forall x \in \Omega, \quad (3.1)$$

where I_1 is the reference image, I_2 is the deformed image, x are the coordinates of a given pixel, Ω represents the domain of the image and u is the displacement field.

Moreover, most of DIC software [58, 59] uses least square as a metric to quantify the difference between the sequence of the images and they subsequently try to minimize this gap. This optimization problem is ill-posed due to the number of degrees of freedom exceeding the number of equations. To overcome the issue, DIC approaches may only focus on subsets (Ω_e) of the image and use a kinematic vector space V_h based on shape functions associated with a grid spaced by several pixels [27] (like in finite element methods):

$$\min_{u_h \in V_h} \sum_{\Omega_e \subset \Omega} \int_{\Omega_e} \left(I_1(x) - I_2(x + u_h(x)) \right)^2 dx. \quad (3.2)$$

It is very common when decreasing the spacing of the grid to add a Tikhonov regularization [22] to the previous functional:

$$E_{DIC}(u_h) = \sum_{\Omega_e} \int_{\Omega_e} \left\{ \left(I_1(x) - I_2(x + u_h(x)) \right)^2 + \lambda (\nabla u_h)^2 \right\} dx, \quad \text{with } \lambda > 0. \quad (3.3)$$

Most of the current optical flow methods closely resemble the first Horn and Schunck formulation (3.4). As done with DIC, they typically combine a data term that assumes graylevel

conservation, along with a weighted smoothness term that models the variation of the flow across the image. The only difference is that the problem here is formulated at each pixel of the whole image with a kinematic defined at the pixel scale (without shape functions over grids):

$$E(u) = \int_{\Omega} \left\{ \left(I_1(x) - I_2(x + u(x)) \right)^2 + \lambda ((\nabla u)^2) \right\} dx, \quad \text{with } \lambda > 0. \quad (3.4)$$

It has been noticed that this classical formulation utilizing the quadratic function is not robust to outliers. Black and Anandan [6] solved this issue by introducing a new formulation (3.5) where the quadratic error function is substituted by an adequate error function robust to outliers, hence the term "robust functions". The paper of Sun [4] drew inspiration from these robust models to define new classical methods capable of enhancing the quality of the computed fields. It analyzed the impact of the penalty functions and the effect of other parameters on the displacement field from Middlebury optical flow benchmark [5] which contains real-life images. For those datasets, the texture that refers to the visual features that allow different regions of an image to be distinguished is not imposed as is the case of mechanical sequences where the speckle pattern is imposed by the user. It has been chosen in different works [7, 103] to utilize an anisotropic diffusion approach as a replacement for the homogeneous regularization in the Horn-Schunck model. This decision stems from the frequent occurrence of discontinuities in optical flow, which often coincide with high image gradients. Introducing a metric ρ , the functional to be minimized can be written as:

$$E(u) = \int_{\Omega} \left\{ \rho((I_1(x) - I_2(x + u(x)))) + \lambda (\rho((\nabla_x u)) + \rho((\nabla_y u))) \right\} dx. \quad (3.5)$$

Our research contributes to a better understanding of the methods proposed in [4] and their implementation, in the context of the analysis of mechanical strain and the detection of mechanical cracks. In particular, we compare the effect of replacing the quadratic norm by the Charbonnier's or the Lorentzian for the image and smoothness terms. Note that we provide a simple matrix-free python code accelerated by GPU so that researchers from all the scientific disciplines who are not familiar with GPU programming can test their images with these techniques easily in a reasonable time.

The rest of the paper is organized as follows. In Section 3.2, we provide the details for the robust and efficient minimization of the functional (3.5). In Section 3.3, we assess the method and compare it different settings on a numerically generated cracked samples before presenting a local regularization approach and moving on to Section 3.3.3 to present our conclusions and perspectives.

3.2. Methodology

The methods used in this work are similar to the classical brightness methods [4] introduced by Sun with some differences that will be mentioned later. The penalty functions used in [4] are taken into account: first, the classical quadratic norm $\rho(x) = x^2$ utilized by Horn and Schunck(HS) and widely used in DIC, then the Charbonnier [14] loss function defined as $\rho(x) = \sqrt{x^2 + \varepsilon^2}$, which is a differentiable and convex robust function and a variant of the L^1 norm.

The parameter ε allows the Charbonnier to converge towards the L^1 norm and it is often set in the literature to 10^{-3} . In the beginning, we will test the Charbonnier function as it is commonly used in most optical flow papers, which means setting its parameter to the mentioned value. We finally present the non-convex Lorentzian [6] function defined as $\rho(x) = \log(1 + \frac{x^2}{2\sigma^2})$, $\sigma > 0$. These functions are illustrated by their curves on Figure 3.1.

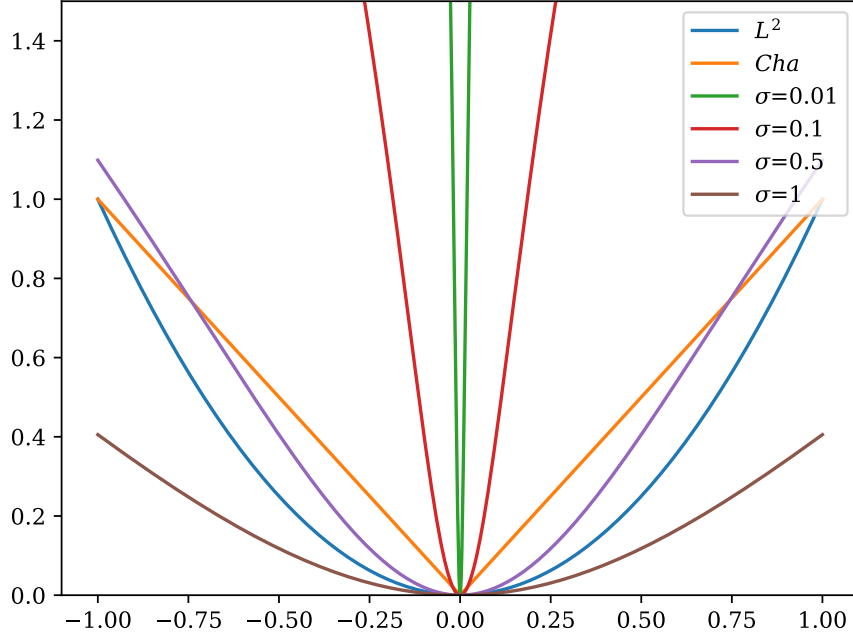


Figure 3.1: The curve of the tested penalty functions: the quadratic (L^2), Charbonnier (Cha.) using $\varepsilon = 0.001$ and the Lorentzian with different values of σ

We denote by u and v the horizontal and the vertical displacement fields respectively. The equation (3.5) can be discretized in the form of:

$$\begin{aligned}
 E(u, v) = \sum_{i,j} \rho((I_2(i + u_{i,j}, j + v_{i,j}) - I_1(i, j))) + \lambda \left(\rho((\nabla_x u)_{i,j}) + \rho((\nabla_y u)_{i,j}) + \rho((\nabla_x v)_{i,j}) \right. \\
 \left. + \rho((\nabla_y v)_{i,j}) \right)
 \end{aligned} \tag{3.6}$$

where (i, j) are the coordinates of pixels, ∇_x and ∇_y are respectively the horizontal and vertical derivation operators.

3.2.1. Graduated Non-Convexity (GNC) and pyramidal approach

The minimization of a non-convex function is not always straightforward due to the existence of local optima that may appear during the minimization process. The Graduated Non-Convexity

(GNC) scheme is used to tackle the complex optimization problems. GNC algorithm works by minimizing a sequence of energy functions $(E_\omega)_\omega$ (3.7) given as a linear combination of a fully convex quadratic energy function E_Q and the fully robust energy function E (3.5). E_Q is obtained by changing the robust penalties in (3.5) by a quadratic loss function. The parameter ω decreases gradually from 1 to 0 and varies the convexity of the compound objective, allowing the transition from the quadratic form to the proposed robust form.

$$E_\omega(u, v) = \omega E_Q(u, v) + (1 - \omega)E(u, v) \quad (3.7)$$

Throughout the process, the solution obtained from the previous stage is utilized as an initial guess for the current stage. It has been demonstrated [9] that using three stages of *convexification* typically yields satisfactory results. During each step, a straightforward local minimization of the energy function is executed.

For the purpose of predicting large motion fields, a coarse-to-fine [14] approach is often adopted as in DIC. The coarse-to-fine approach consists in creating a pyramid of images by repeatedly reducing the size of the original image. The pyramidal structure is created by stacking the smaller resized images on top of each other, ending with the highest resolution image at the bottom. Afterwards, at every level of the pyramid we perform a processing step, the output is then resized and used as a starting point for the next level. The process is repeated until reaching the original image. An illustration of this approach is shown in Fig. 3.2. We smooth the images before downsampling them using a Gaussian filter, with the standard deviation fixed at $\sqrt{\frac{1}{2d}}$, where d represents the downsampling factor. During the minimization of the quadratic formulation which appears during the first iteration of the GNC process, we dynamically set the number of pyramid levels to ensure that the highest level has a width or height of approximately 20 to 30 pixels with a downsampling factor of 0.5 since it has been explored [4] that using a convex penalty, a factor of 0.5 is sufficient. For the remaining stages of the GNC, we will utilize two levels with a factor of 0.8 to benefit from the solution obtained in the prior *convexification* stage. The GNC is also used for the Charbonnier function despite its convexity, since it leads

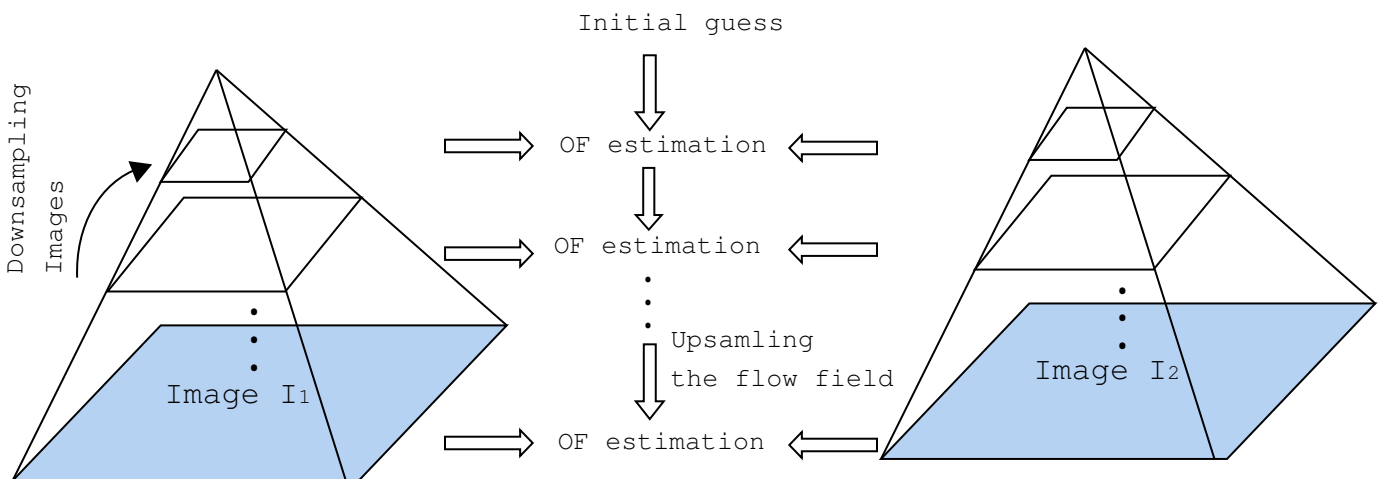


Figure 3.2: Representation of the pyramidal approach, OF=Optical Flow

to better minimization [4] due to the nonlinearity of its derivative. In this method ten flow

increments are computed per direction at each pyramid level and a 5×5 median filter is applied in the end of each warping step to the current fields to remove the outliers as described in [4].

3.2.2. Methodological details

The estimation of the optical flow fields requires computing a finite difference approximation of the spatial derivatives of images as well as the temporal derivative defined as the difference between the reference image and the corrected image $I_2(i+u_{i,j}^k, j+v_{i,j}^k)$ where u^k and v^k represent the displacement fields computed at the iteration k obtained by a bi-cubic interpolation. Instead of the classical finite difference kernels, we approximate the spatial derivatives using a five-point kernel [8] $h = \frac{1}{12}[-1, 8, 0, -8, 1]$. We neglect pixels whose movement exceeds the border by setting their corresponding derivatives to zero. We interpolate both the second image and its derivatives using the current flow field. In agreement with [9], we use the average of the gradient of the reference image and the interpolated derivatives of the second image to define the tangent system to be solved.

Unlike the methods used in Sun’s paper [4], where the regularization parameter λ and the parameter σ of the Lorentzian function are fixed to provide a displacement close to the real fields obtained from the image sequences in the Middlebury dataset, our approach consists of varying the value of these parameters to see their impact on the strain field and to study the effect of this change on discontinuities.

The choice of the parameter in the Lorentzian function is far from evident, and there is no universal parameter value that suits all situations. The selection of this parameter requires a thorough understanding of the data and objectives. For instance, if the expected motion fields are smooth and continuous, a higher value of this parameter can be a good choice to encourage regularization and reduce discontinuities. Conversely, if the field contains discontinuities, a lower value of σ allows increasing their effect. It may be necessary to conduct empirical tests by adjusting the parameter of the Lorentzian function and evaluating the results on real optical flow data.

It should be noted that the Rudin-Osher-Fatemi [8](ROF) texture-structure decomposition appendix 3.4 frequently used to eliminate changes in brightness is not applied here. Indeed, this decomposition consists in filtering input images by breaking each of them down into two parts, a texture part which preserves image details, and a part containing shadows called structure. In contrast with real life scenes, in mechanical engineering, experimental setups are designed to limit as much as possible the change in brightness during the test. All the more so as, in the example presented in this paper, the images are synthetic and do not pose any brightness-related problems.

3.2.3. Implementation

We provide a python source code accelerated by GPU using *CuPy* [47] which proposes a NumPy-compatible API for GPU computing with NVIDIA CUDA. As the matrix to be solved at each iteration is a positive definite symmetric matrix, the conjugate gradient method [44] is used to solve the problem. The choice of this type of solvers is not only motivated by its ability to

solve large-scale systems, but also by its capacity to be implemented in a matrix-free version, which allow fewer resources consumption and consequently enables the processing of much larger datasets. The code [95] as well as the examples are uploaded to GitHub.

3.3. Results

We tested our algorithm on real images and demonstrated that it yields satisfactory results. The choice to use synthetic images is also influenced by the absence of a reference solution in the context of physical experiments. The use of synthetic images is of great importance in computer vision, as they allow for the generation of precise data for assessing the effectiveness of image processing algorithms. By using these images, it is possible not only to avoid the difficulties associated with collecting real experimental data, which can be costly in terms of time and resources, but also they allow for the creation of various test scenarios.

A natural measure of quality in this context is the Average Endpoint Error (AEE) which corresponds to the average (over N pixels) of the Euclidean distance between the calculated (u, v) and the ground-truth (\hat{u}, \hat{v}) motion fields:

$$AEE^2 = \frac{\|u - \hat{u}\|^2 + \|v - \hat{v}\|^2}{N^2}$$

In order to study the influence of the previous cited functions over the discontinuities, a synthetic 512×512 speckle patterns image is generated with grain sizes averaging between three and five pixels using the algorithm developed in [33].

3.3.1. Crossing crack

We create an artificial transverse horizontal crack of one pixel by imposing a one-pixel downward displacement on the lower part of the created discontinuity. The images are presented in the Fig. 3.3.

The horizontal displacement in this test case is zero across the entire image whereas the vertical displacement field is invariant for each part of the image, since it is zero for the entire upper part of the crack and imposed to be one pixel for the lower part as shown in the analytic motion field of Fig. 3.4c. The average of vertical displacements along the horizontal direction of the image must then be constant and the curve of these averages over the entire image should be represented by a Heaviside step function, discontinuous at the crack row. In this case, the maximum standard deviation reached on each section is an indicator of the noise of the calculated solution. The results of Fig. 3.4 demonstrate that the quadratic norm behaves as predicted by diffusing discontinuities and by requiring more pixels to capture the crack than the robust functions (Charbonnier and Lorentzian) that are able to converge towards a better solution in terms of AEE and standard deviation. Indeed, the L^2 norm captures the discontinuities with seven pixels, while other metrics that only require three pixels. This highlights the importance of studying the impact of these metrics over the estimation of strain on mechanical images.

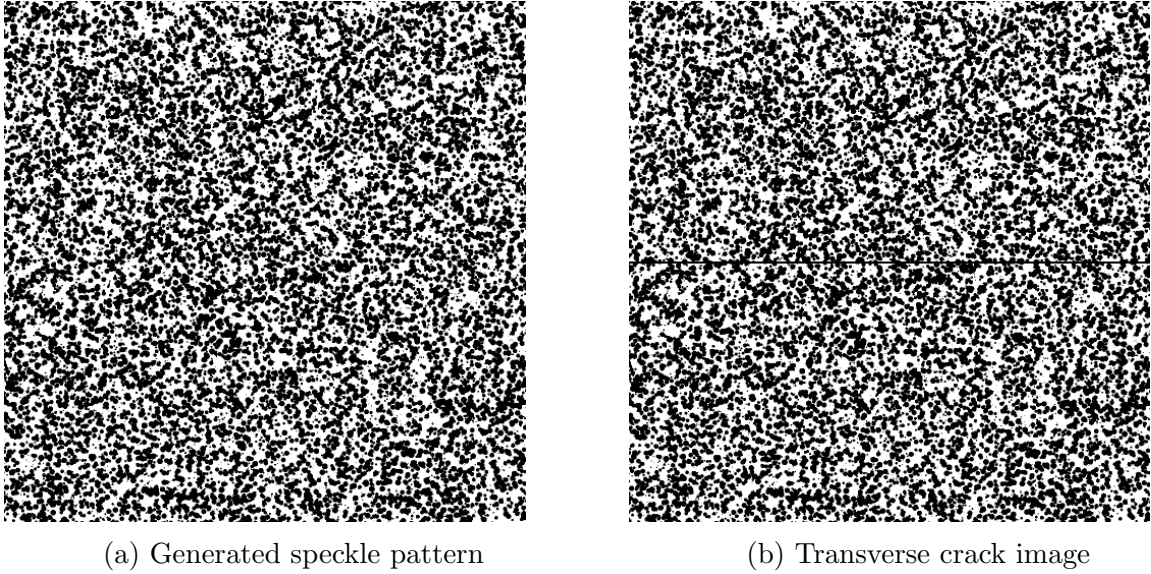


Figure 3.3: First synthetic test case

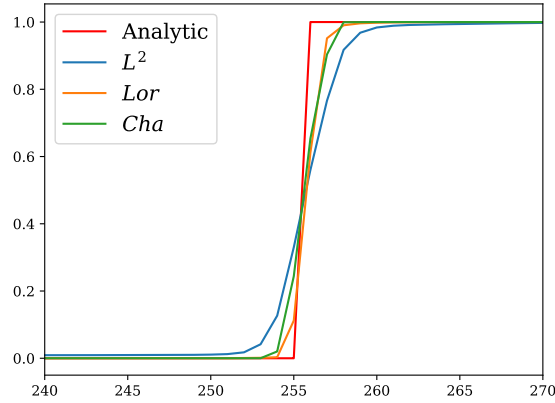
3.3.2. Crack with tip

The William’s series [65] allows for the description of displacement and strain fields near a crack or notch in a solid material. This method assumes that the motion near a crack can be expressed as a series of functions dependent on the distance from the crack tip. From the pre-defined speckle pattern Fig. 3.3a, a deformed image is generated using Williams’ fields resulting from a planar loading mode presented in Fig. 3.6j. The deformed image Fig. 3.5 of the generated sequence contains a 1×250 pixels discontinuity.

The regularization parameter in optical flow methods is used to balance the smoothness of the estimated motion field and the accuracy of the motion vectors. This parameter allows controlling the trade-off between fitting the data and maintaining the smoothness of the estimated flow field. The behavior of this parameter is similar to that of the grid size in DIC techniques [65]. A higher value of the regularization parameter leads to a smoother but less accurate motion estimation, while a lower value results in a more detailed but potentially noisy flow field, see Fig. 3.6.

The initial results show that the Charbonnier and the Lorentzian functions give more promising results than the quadratic norm by preserving discontinuities and providing better quality of motion fields.

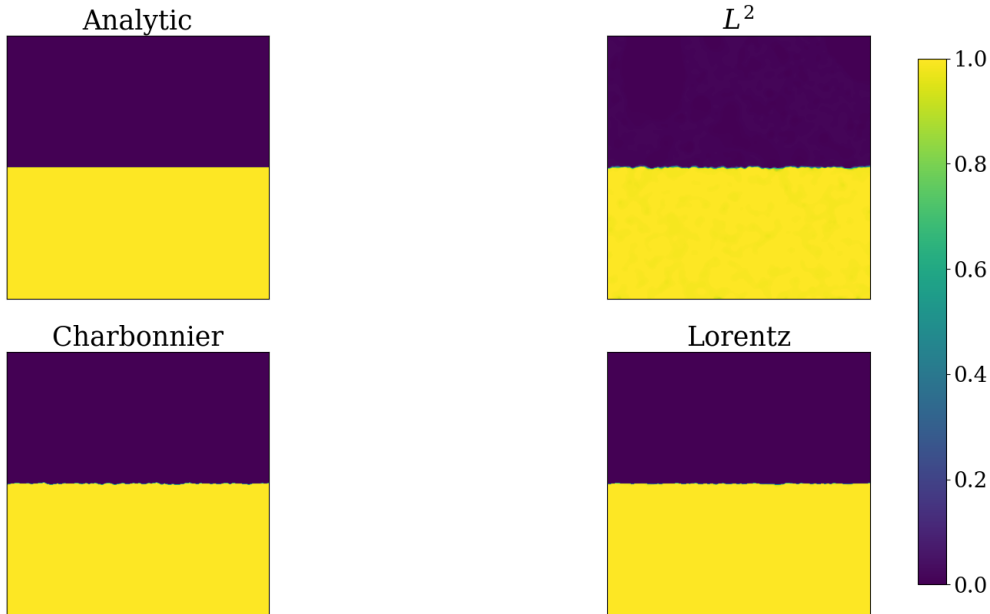
To highlight the difference between the used metrics, we present the horizontal and vertical strain fields obtained with different regularization in Fig 3.7 and Fig 3.8. We observe that the fields obtained using the L^2 norm are noisier compared to those generated by the robust functions. It is also observed that Charbonnier produces spurious discontinuities (see Fig. 3.7c Fig 3.8c) that are not observed either in the analytical fields Fig. 3.7g and Fig. 3.8g or in the fields generated by the Lorentzian penalty function presented in the first rows of Fig 3.7 and Fig 3.8. Moreover, these unreal discontinuities become structured and difficult to distinguish from real cracks. This outcome can be explained by the fact that Charbonnier is an approx-



(a) Average of vertical displacements along the horizontal direction in the vicinity of the crack.

Metric	AEE $\times 10^{-5}$	Std. dev. max	Crack width (pixels)
L^2 $\lambda = 10^3$	7.86	1.64×10^{-2}	7
Lorentz $\lambda = 10$ $\sigma = 0.05$	6.05	2.97×10^{-3}	3
Charbonnier $\lambda = 10$	7.04	3.20×10^{-5}	3

(b) AEE, maximum standard deviation reached outside the crack, and number of pixels required to capture the discontinuity for each penalty function.



(c) Calculated and analytic motion fields.

Figure 3.4: Identification of the crossing crack for different metrics.

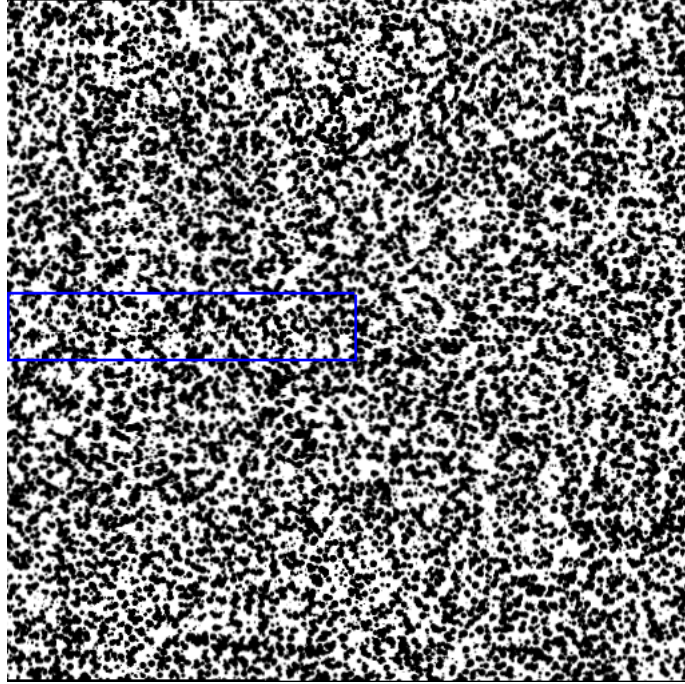


Figure 3.5: The deformed image for the second test case, the part framed in blue contains the generated crack

imation of the non-smooth L^1 norm, which results in abrupt gradients in regions where the displacement values are slightly different.

Discontinuities like cracks can be characterized as places where the strain field varies rapidly. This can be illustrated by Fig. 3.9g which shows the norm of the gradient of the strain field for the ground-truth solution, and where the crack clearly stands out. Thus, the norm of the difference between the value of the analytical second derivatives of the displacement and their calculated counterpart is an indicator that quantifies the ability of the methods to capture cracks. We denote this field by NSDE which stand for Frobenius Norm of the Strain Derivative Error, its relative mean value is denoted by MNSDE, see Eq. (3.8). We observe from the NSDE of the Fig. 3.9, that the fields obtained with the Lorentzian function shown in Fig. 3.9a and Fig. 3.9b are more homogeneous outside the crack, closer to reality, and contain less discontinuities than those estimated by the Charbonnier function presented in Fig. 3.9c and Fig. 3.9d or the noisy fields given by the L^2 . Furthermore, Fig. 3.9h shows that the lowest values of MNSDE are provided by the Lorentzian compared to the other metrics. Even though the fields generated by the quadratic function are noisier, as reflected by higher MNSDE, these fields are not structured as those generated by the Charbonnier function. This behaviour makes the Charbonnier's function as used by the optical flow community ($\varepsilon = 10^{-3}$) challenging to use in mechanics.

$$NSDE(\varepsilon) = \sum_{i,j} \left\| \frac{\partial(\varepsilon_{x_i x_j} - \hat{\varepsilon}_{x_i x_j})}{\partial x_j} \right\|_F^2, \quad MNSDE(\varepsilon) = \frac{\overline{NSDE(\varepsilon)}}{\sum_{i,j} \left\| \frac{\partial \hat{\varepsilon}_{x_i x_j}}{\partial x_j} \right\|_F^2} \quad (3.8)$$

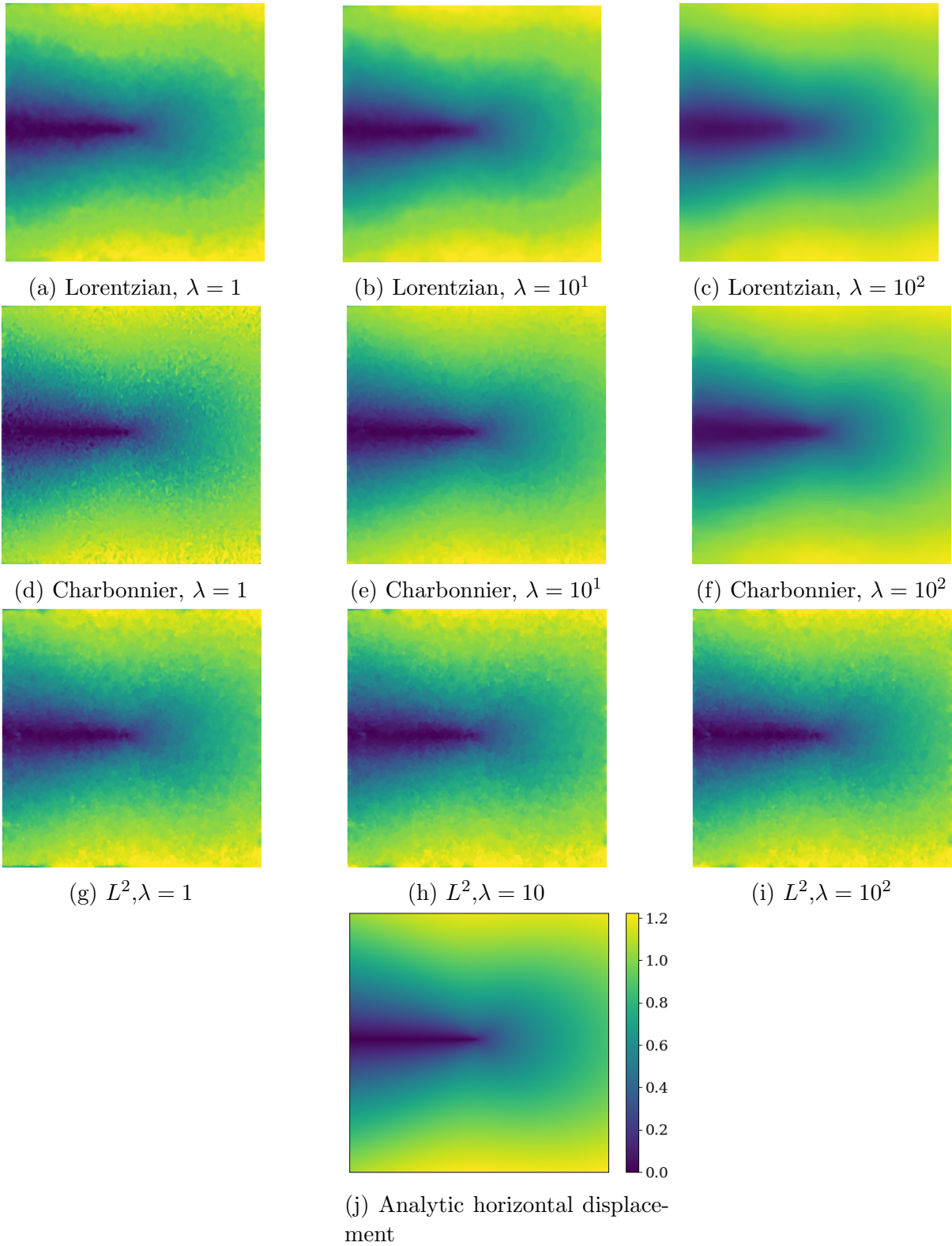
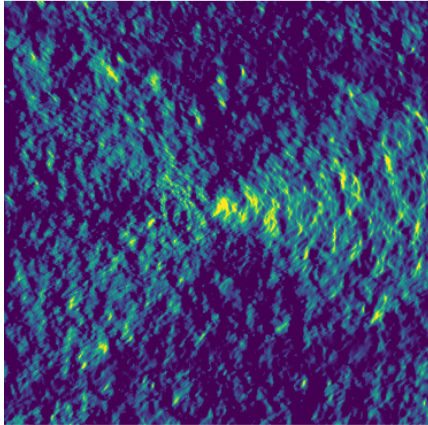
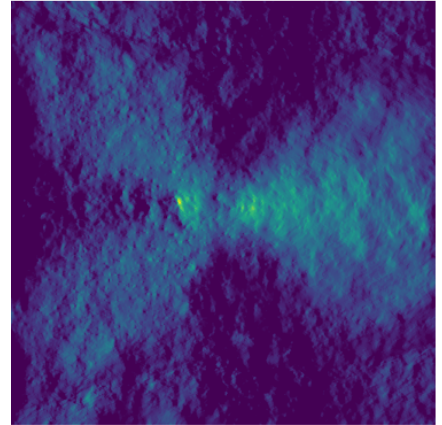


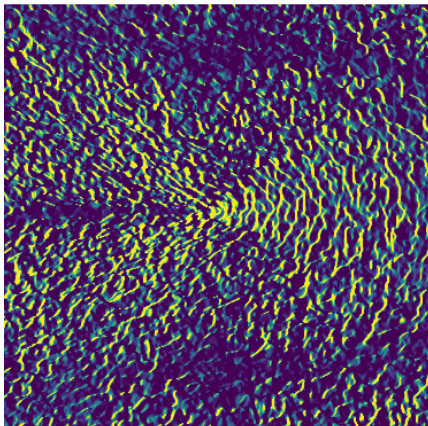
Figure 3.6: Horizontal displacement field estimated with different values of λ , and different loss functions for the crack with tip experiment.



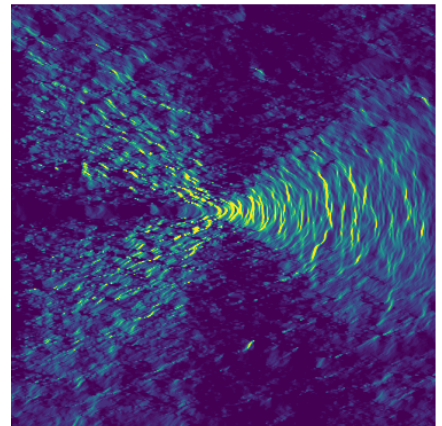
(a) Lorentzian, $\lambda = 10^1$



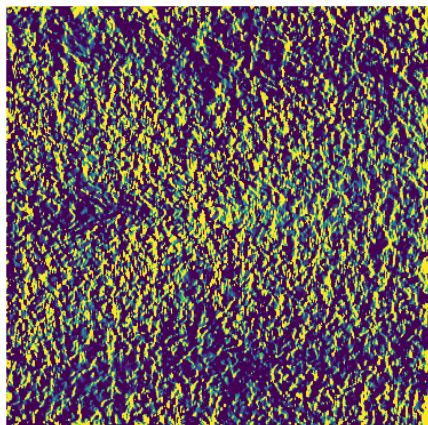
(b) Lorentzian, $\lambda = 10^2$



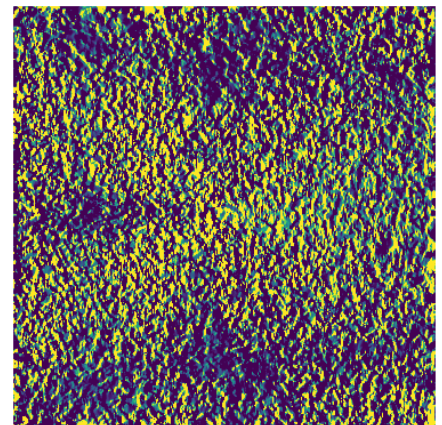
(c) Charbonnier, $\lambda = 10^1$



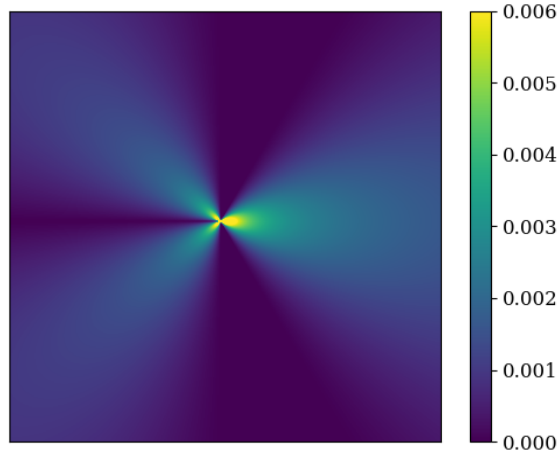
(d) Charbonnier, $\lambda = 10^2$



(e) L^2 , $\lambda = 10^1$



(f) L^2 , $\lambda = 10^2$



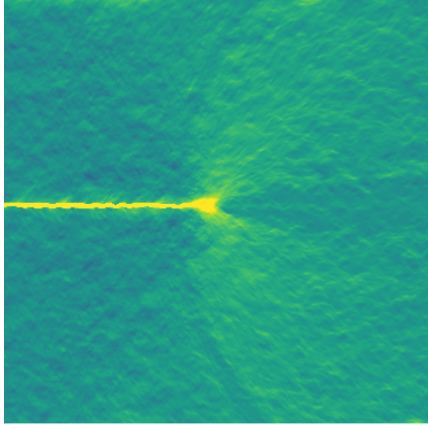
(g) Analytic

Figure 3.7: The horizontal strain fields ε_{xx} provided with different regularization amplitude of λ for for different metrics.

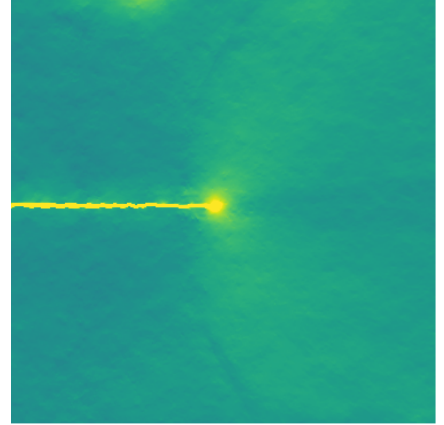
Choice of the parameter for the robust metrics. Let's recall that the shown results given by the different robust metrics are obtained by fixing the parameter ε for Charbonnier as done by the optical flow community, and using a parameter $\sigma = 0.05$ for the Lorentzian, which we fixed through empirical testing. In this paragraph, we assess the impact of varying these parameters over the strain field. The results are shown in Figures 3.10, 3.11, 3.12 and 3.13. It is visible that the discontinuities observed for Charbonnier with $\varepsilon = 10^{-3}$ disappear when the value of this parameter is increased. On the other hand, we observe that we begin to have increasingly noisy fields and we start to diffuse the discontinuities when it is increased. We remark a similar behavior for the parameter σ of the function of Lorentz, with the appearance of spurious discontinuities when its value decreases and the presence of noisy fields resembling those of the L^2 norm when its value increases. This means that the parameters of the robust metrics allow us to give more weight either to the behavior of the L^1 norm, avoiding aberrant values and preserving cracks, or to that of the L^2 norm.

3.3.3. Local regularization

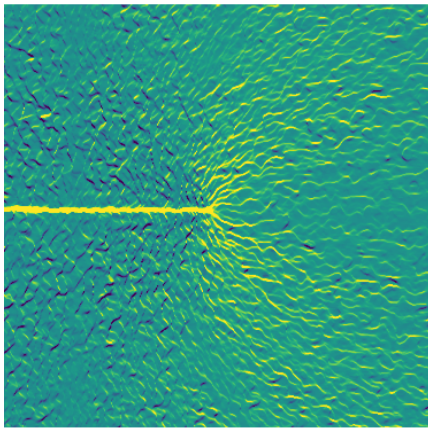
Global regularization of the fields may result in excessive smoothing of the solution, leading to loss of details and structures in the image and decreasing the accuracy of the registration computation. Moreover, this strategy may not be suitable for situations involving regions with discontinuities. Hence, the interest in considering local regularization weights is to preserve the discontinuities as much as possible on the calculated fields. We have adopted the strategy to adjust the amplitude of the regularization in such a way that it decreases around the crack to ensure that there is no diffusion of the crack whereas the strain fields becomes homogeneous



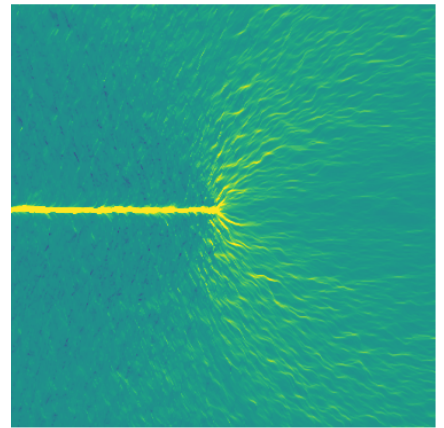
(a) Lorentzian, $\lambda = 10^1$



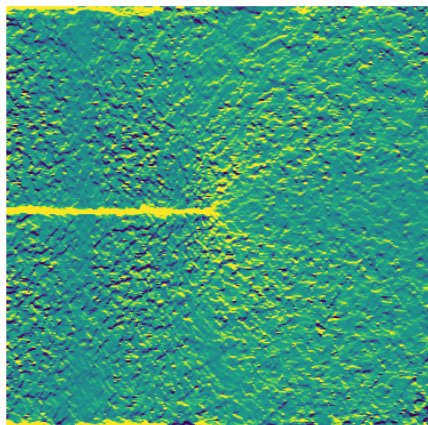
(b) Lorentzian, $\lambda = 10^2$



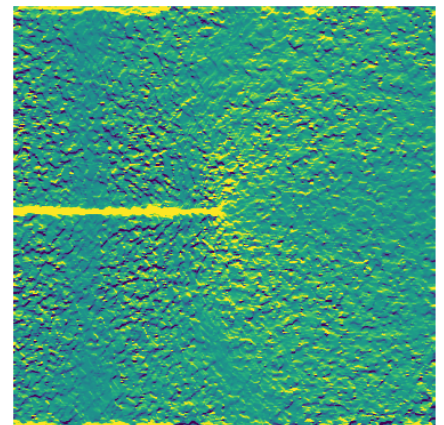
(c) Charbonnier, $\lambda = 10^1$



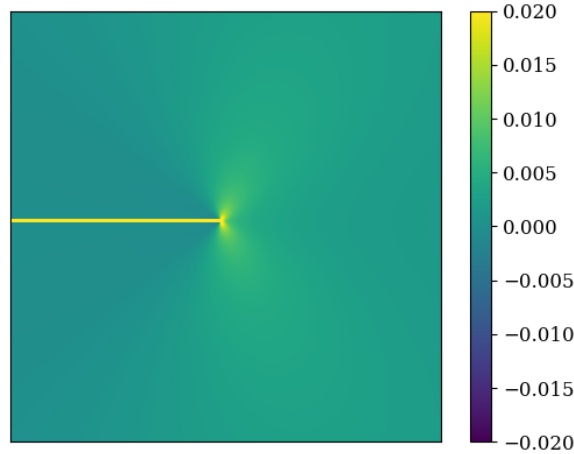
(d) Charbonnier, $\lambda = 10^2$



(e) L^2 , $\lambda = 10^1$



(f) L^2 , $\lambda = 10^2$



(g) Analytic

Figure 3.8: The vertical strain fields ε_{yy} provided with different regularization amplitude of λ for different metrics.

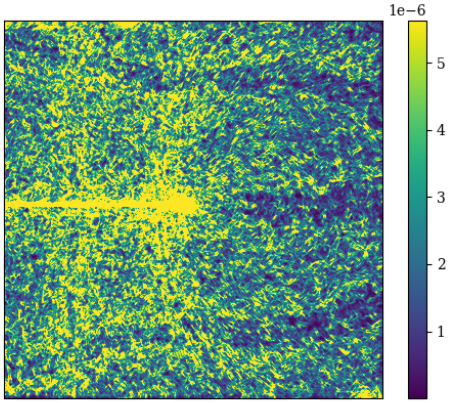
and smooth outside the area containing the discontinuity. This leads to the minimization of a novel energy whose regularization weight ($\lambda_{i,j}$) is defined locally and changes depending on the pixel coordinates:

$$\begin{aligned}
 E(u, v) = \sum_{i,j} & \rho((I_2(i + u_{i,j}, j + v_{i,j}) - I_1(i, j))) + \lambda_{i,j} \left(\rho((\nabla_x(u_{i,j}))) + \rho((\nabla_y(u_{i,j}))) \right. \\
 & \left. + \rho((\nabla_x(v_{i,j}))) + \rho((\nabla_y(v_{i,j}))) \right)
 \end{aligned} \tag{3.9}$$

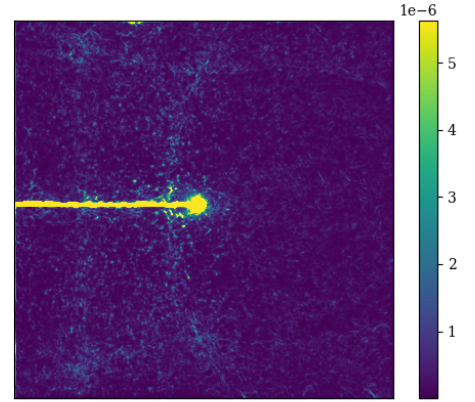
To test our approach, we create a local smoothing mask Fig. 3.14 with values varying between 7 (near the crack) and 600 (away), using the function of Lorentz with $\sigma = 0.05$. In order to avoid sudden changes in λ , a Gaussian filter is applied as shown in Fig. 3.14b. The analytical strain field ε_{yy} shown in Fig. 3.15k and 3.15l allows to visualize the whole length of the discontinuity. The choice of a global large smoothing weight results in excessive smoothing in the area near the crack tip Fig. 3.15f which is not observed for the small values of λ Fig. 3.15c. This can be explained by the fact that when the regularization amplitude is high, there is a strong influence of the neighboring pixels on the smoothing process, which can lead to a loss of important details. But the global small λ results in structured noise away from the crack.

The local change of regularization shown in Fig. 3.15g, 3.15h, 3.15i allows us to take advantage of the high regularization outside the crack by homogenizing the strain fields and to preserve the totality of discontinuity on the ε_{yy} field as in the case of small regularization, as the impact of the neighboring pixels is forced to become weak at the crack's neighborhood.

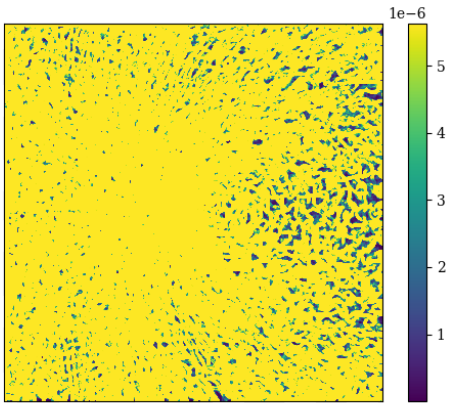
One can observe in Fig. 3.16 that the local regularization permits to correctly capture the strain gradients. Also, the MNSDE computed with the local regularization matches the best approaches in their region of validity (near the crack for the low regularization, away from the crack for the large regularization). Observing the AEE, it is important to note that a low regularization results in a low value for this index. Therefore, we focus on displacements, the use of a regularization value of 7 might be more favorable.



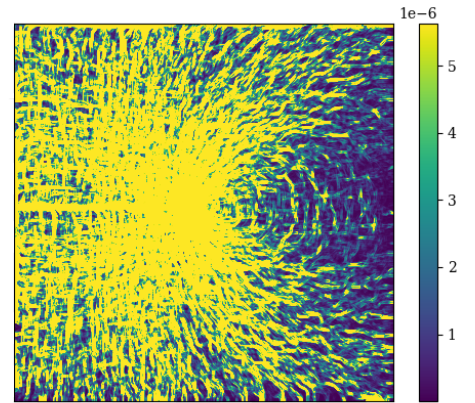
(a) Lorentzian, $\lambda = 10^1$



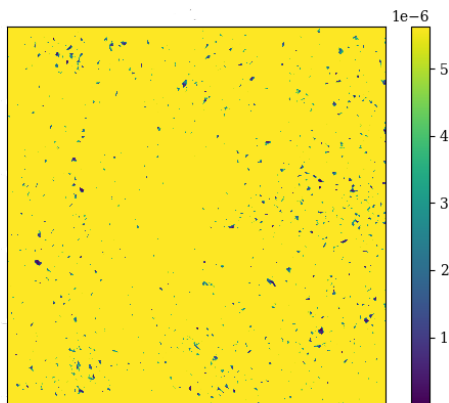
(b) Lorentzian, $\lambda = 10^2$



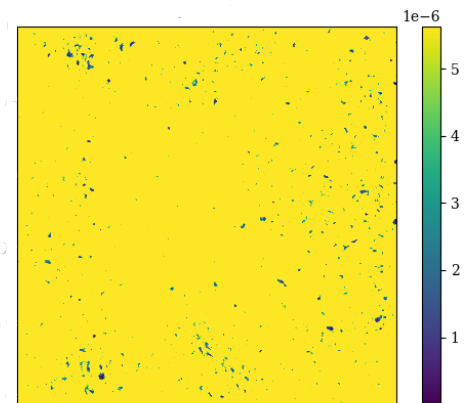
(c) Charbonnier, $\lambda = 10^1$



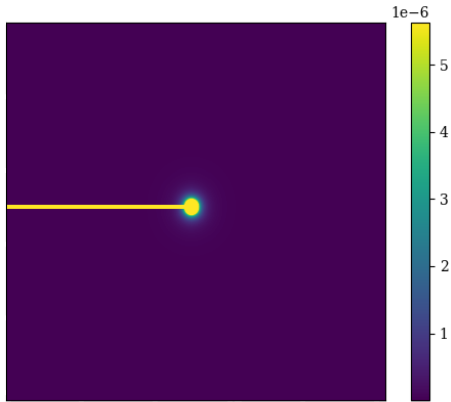
(d) Charbonnier, $\lambda = 10^2$



(e) L^2 , $\lambda = 10^1$



(f) L^2 , $\lambda = 10^2$



(g) Analytic

Function	λ	MNSDE
<i>Lorentz</i>	10	3.037×10^{-1}
<i>Lorentz</i>	10^2	8.226×10^{-2}
<i>Charbonnier</i>	10	2.029×10^0
<i>Charbonnier</i>	10^2	5.681×10^{-1}
L^2	10	3.661×10^0
L^2	10^2	3.550×10^0

(h) The MNSDE in terms of λ

Figure 3.9: NSDE for the different metrics using different regularization amplitudes. Their corresponding MNSDE is presented in the table 3.9h. The image 3.9g represents the analytical norm of the second-order gradient.

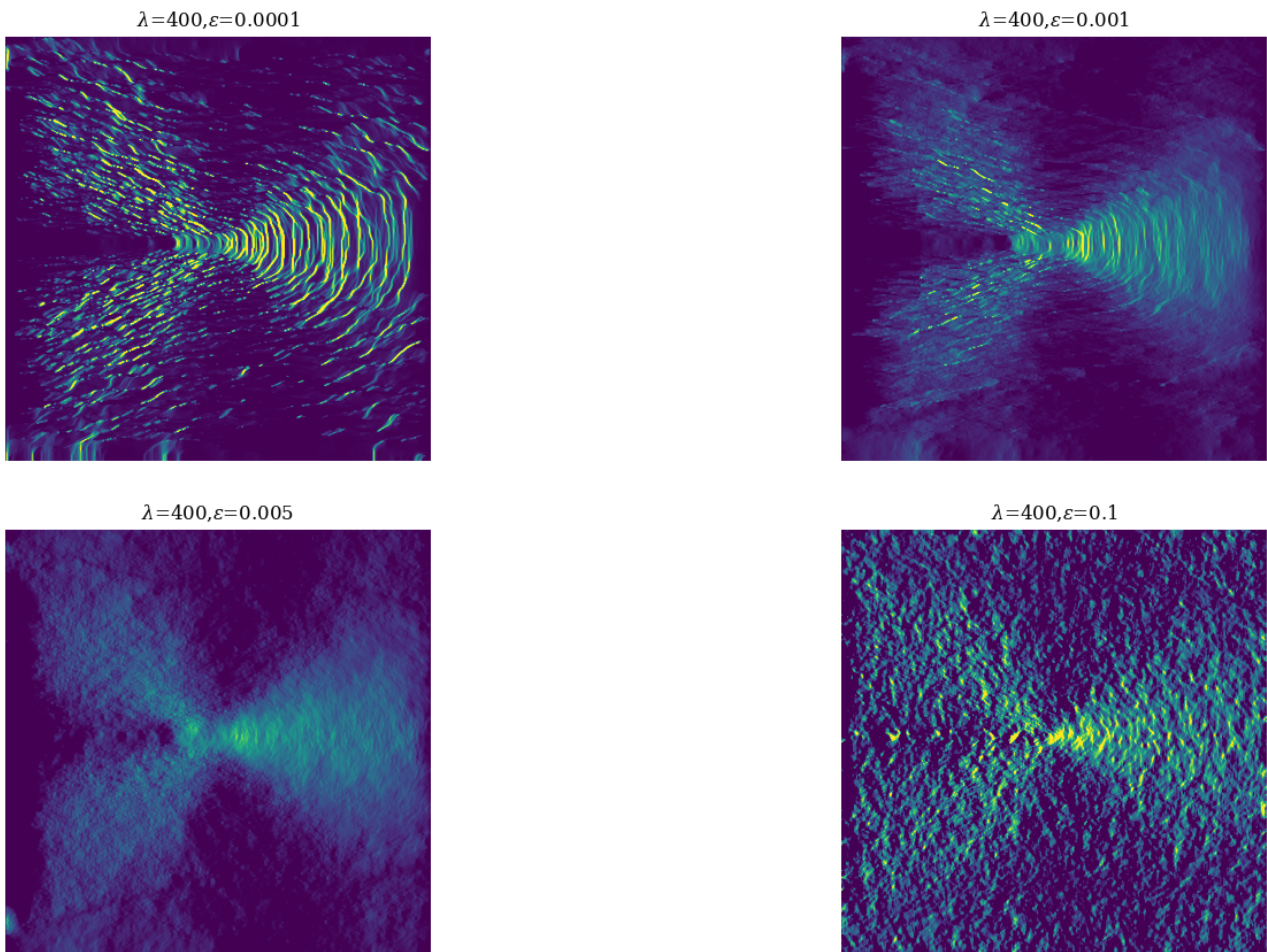


Figure 3.10: ε_{xx} strain of Charbonnier function using different ε values and fixed $\lambda = 4 \times 10^2$

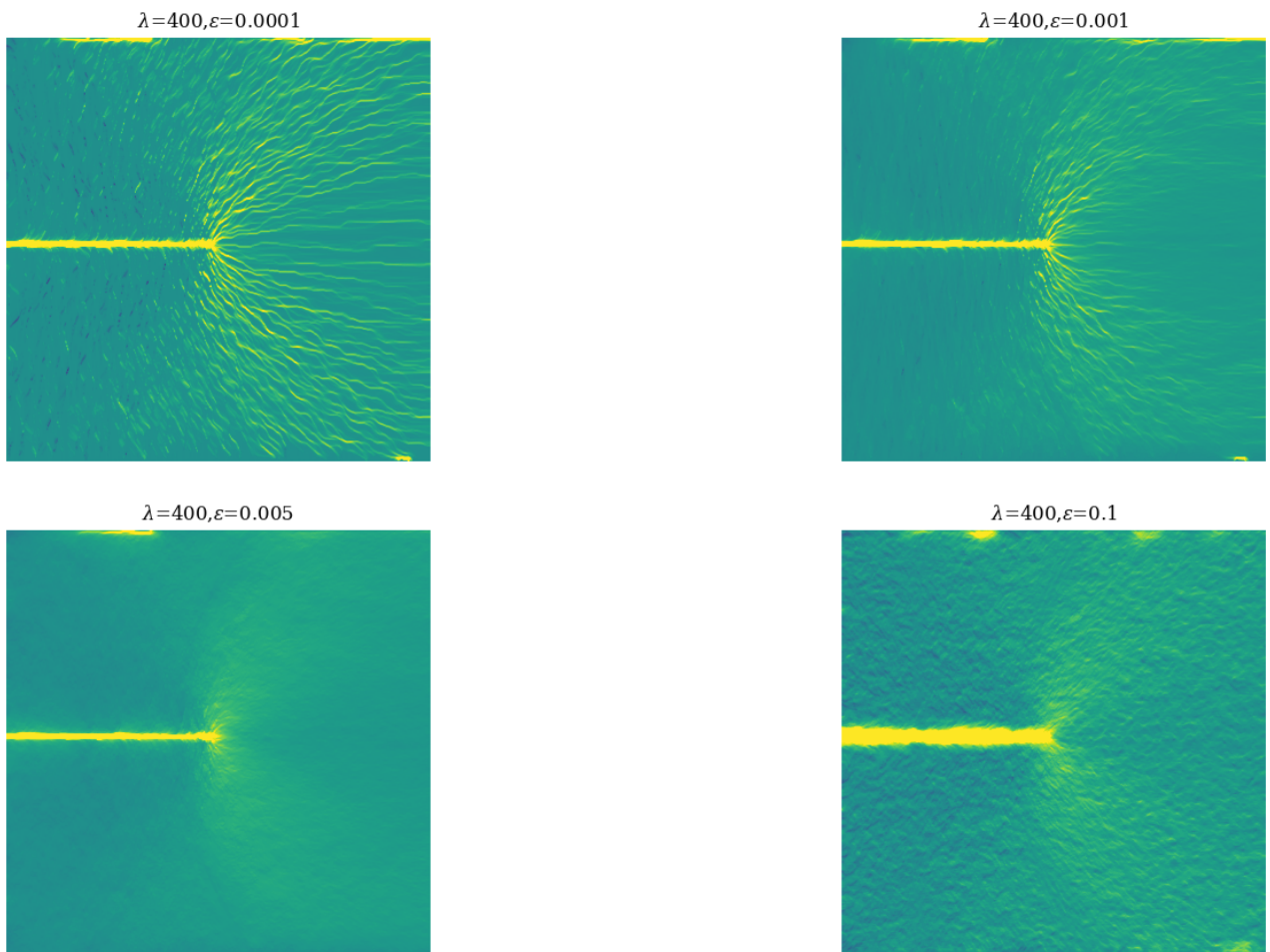


Figure 3.11: ε_{yy} strain of Charbonnier function using different ε values and fixed $\lambda = 4 \times 10^2$

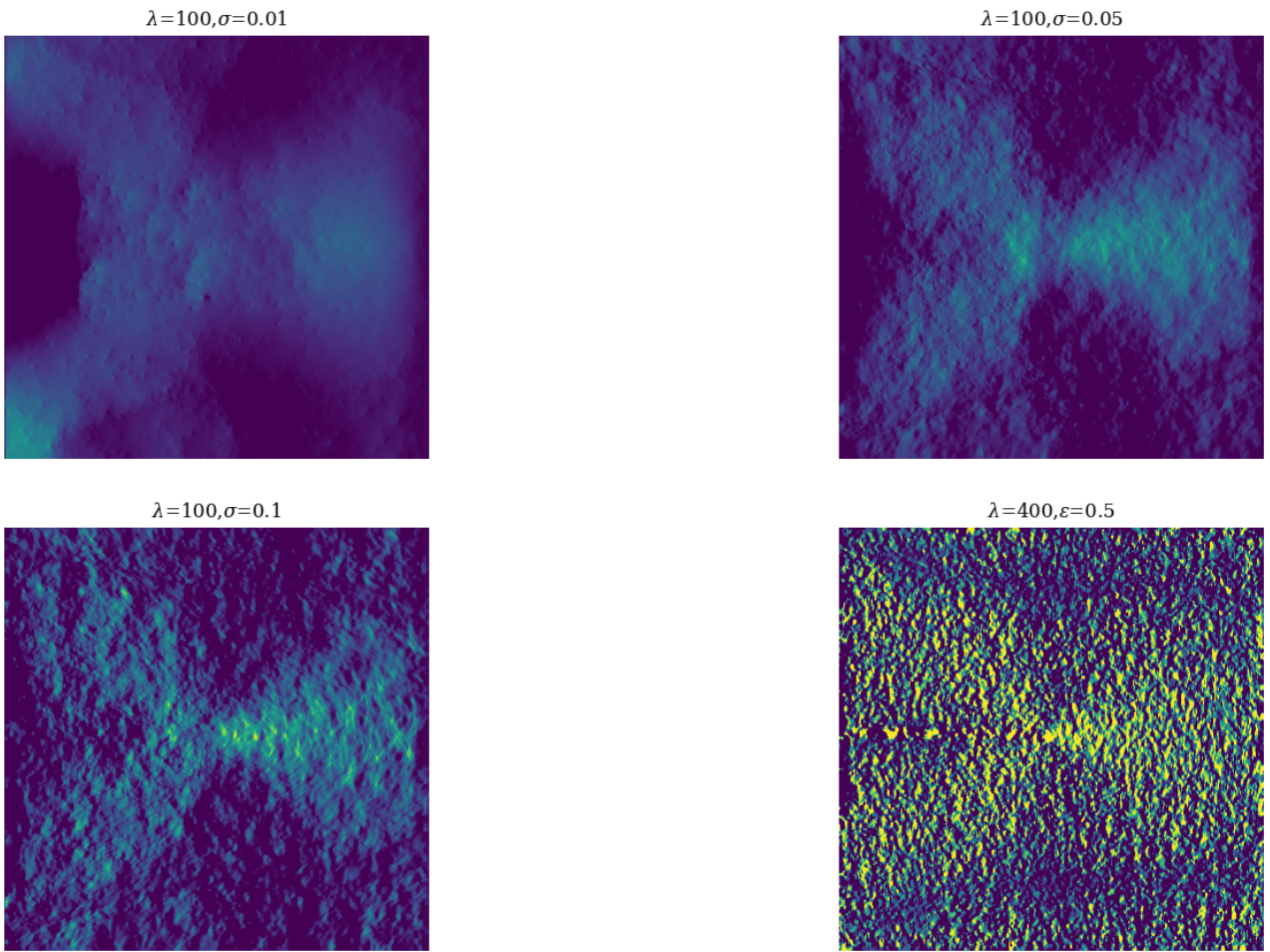


Figure 3.12: ε_{xx} strain of Lorentz function using different σ values and fixed $\lambda = 10^2$

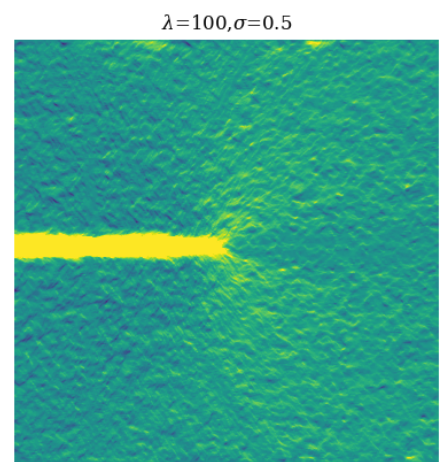
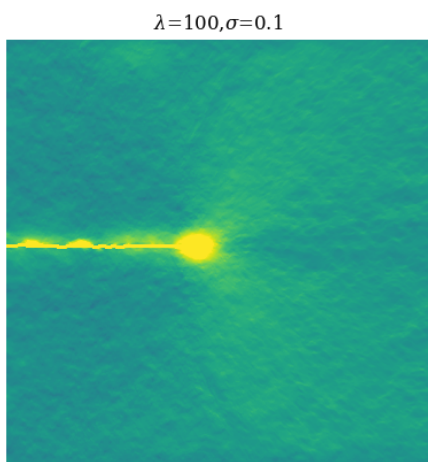
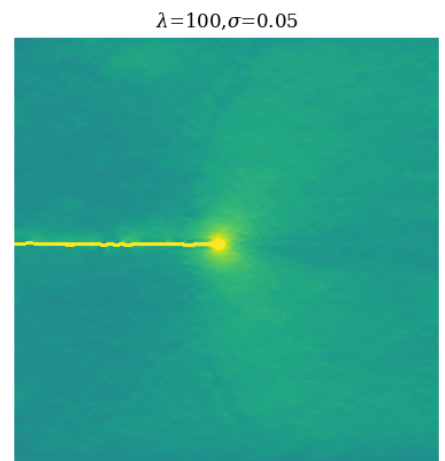
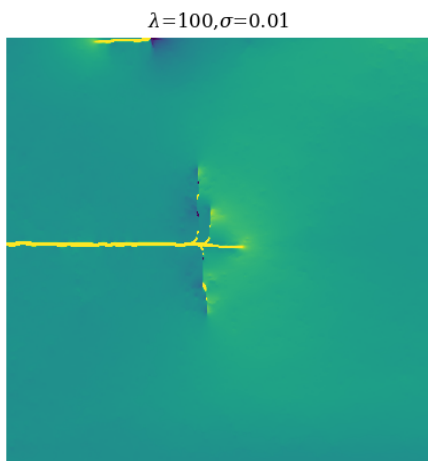


Figure 3.13: ε_{yy} strain of Lorentz function using different σ values and fixed $\lambda = 10^2$

3.4. Conclusion

Along this paper, we have tested the impact of different metrics on the strain fields estimated using optical flow methods. Robust penalty functions like Charbonnier and Lorentzian often give very good results on displacements and preserve cracks unlike the quadratic L^2 norm often used in DIC that tends to diffuse them. Despite its advantage of preserving discontinuities, the Charbonnier function as it used in optical flow creates some spurious cracks which could be hardly distinguished from the physical ones. The use of the Lorentzian and Charbonnier functions with suitable values of their parameters that allow us to achieve a transition between the behaviors of the two norms L^1 and the L^2 , leads to satisfactory results.

The Tikhonov parameter that balances the influence of the fidelity term and the smoothness term plays a major role in these methods. Increasing this term is similar to enlarging the size of the sub-images in DIC algorithms. The increase of this parameter can cause oversmoothing on some regions of the image, which could be unpleasant in the presence of local phenomena, while its decrease can encourage the presence of noise in the estimated strain fields. A strategy is adopted to change the value of this parameter locally using a mask. This approach enabled us to combine the advantage of preserving the discontinuities in the areas where this parameter is low and of smoothing the fields where its weight is greater.

This work has enabled us to test a few examples of robust metrics. In future work, we'd like to explore other metrics and their added value in mechanics. The creation of the regularization mask used in the illustrative example of this article is manual and with advance knowledge of the kinematic field which is not often the case. One of the perspectives is to automate the creation of masks. One initial idea to reach this objective could be to take the advantage of the pyramidal approach and to leverage the GNC structure used during the minimization process. This automation could also be achieved through the introduction of an artificial intelligence model able to distinguish between image areas requiring stronger smoothing and those requiring weaker smoothing.

Finally, in order to use these penalties on other image sequences, we implemented an open source, GPU-accelerated, and matrix-free [code](#) [95] written in python that can be easily modified and allows the use of a regularization mask.

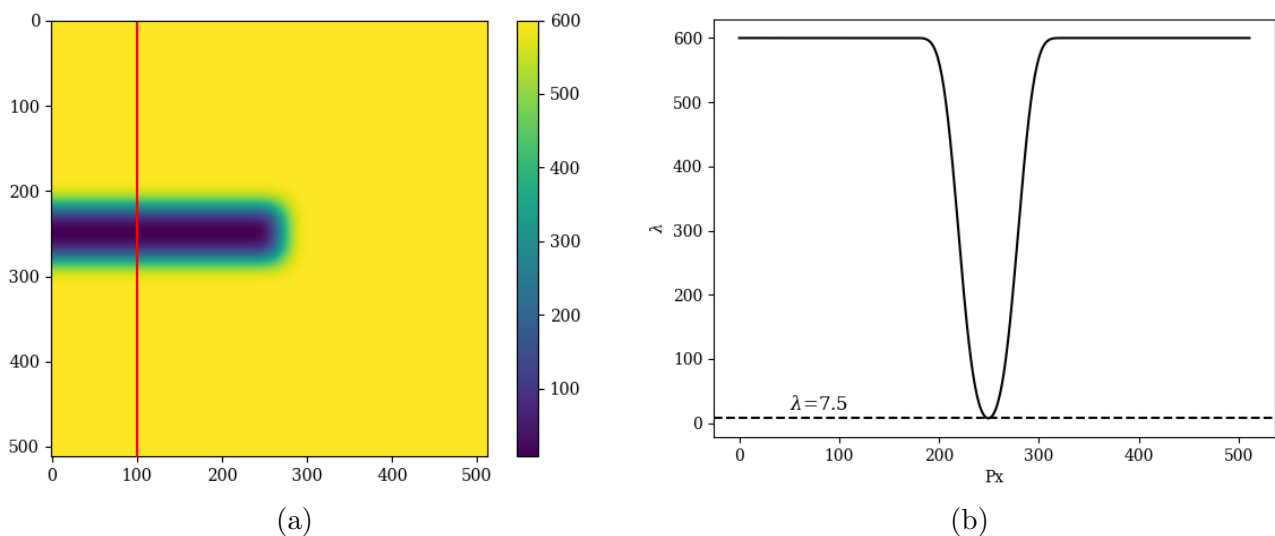


Figure 3.14: (a) Local regularization mask. The cut along the red line is shown in the figure (b).

Appendix: Structure-texture decomposition

The assumption of the graylevel conservation has been taken into account since the creation of optical flow methods. In other words, it has been supposed that the values of the pixels of the image sequence do not change during the motion. For many sequences, this constraint is not respected. For instance, this can be caused by camera noise, shadows, reflections, brightness changes...

To solve this issue, one solution was proposed in [7], called structure-texture decomposition. This idea of this approach is to split each image into two parts. The first part contains the main objects of the images and is called the structural part while the second one is the textural part and corresponds to the fine scale-details of the image, this is illustrated on Figure 3.17. The decomposition is achieved using Rudin, Osher and Fatemi (ROF) image denoising [8] model. The structural part I_S of the image I can be obtained by solving the following problem:

$$\min_{I_S} \int_{\Omega} \left\{ |\nabla I_S| + \frac{1}{2\theta} (I_S - I)^2 dx \right\} \quad (3.10)$$

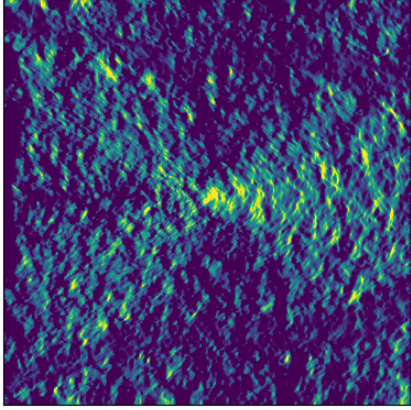
Finally, the textural image I_T is given as the difference between the main image and its structural part $I_T = I - I_S$.

For the purpose of solving (3.10), the iterative scheme introduced in [8] is adopted. $I_S = I + \theta \operatorname{div} p$ where p is defined iteratively by $p^0 = 0$ and:

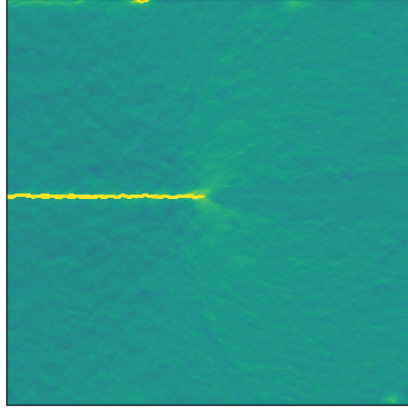
$$\tilde{p}^{n+1} = p^n + \frac{\tau}{\theta} (\nabla(I + \theta \operatorname{div} p^n))$$

and

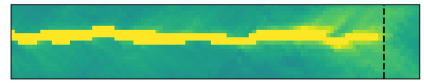
$$p^{n+1} = \frac{\tilde{p}^{n+1}}{\max(1, |\tilde{p}^{n+1}|)}$$



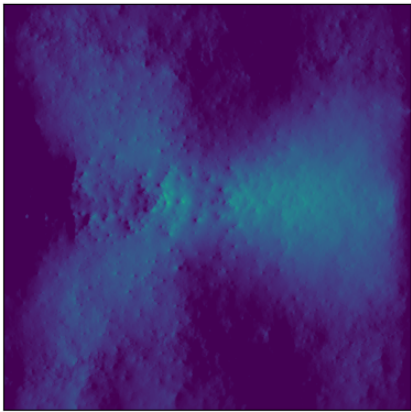
(a) $\epsilon_{xx}, \lambda = 7$



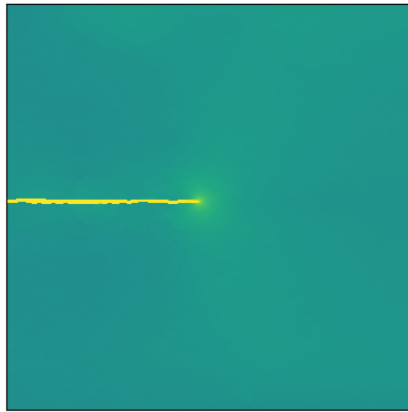
(b) $\epsilon_{yy}, \lambda = 7$



(c) $\epsilon_{yy}, \lambda = 7$



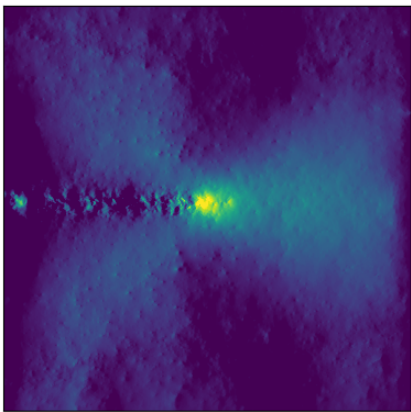
(d) $\epsilon_{xx}, \lambda = 6 \times 10^2$



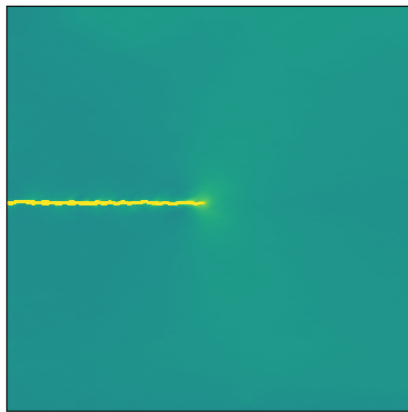
(e) $\epsilon_{yy}, \lambda = 6 \times 10^2$



(f) $\epsilon_{yy}, \lambda = 6 \times 10^2$



(g) $\epsilon_{xx}, \lambda \text{ local}$



(h) $\epsilon_{yy}, \lambda \text{ local}$



(i) $\epsilon_{yy}, \lambda \text{ local}$

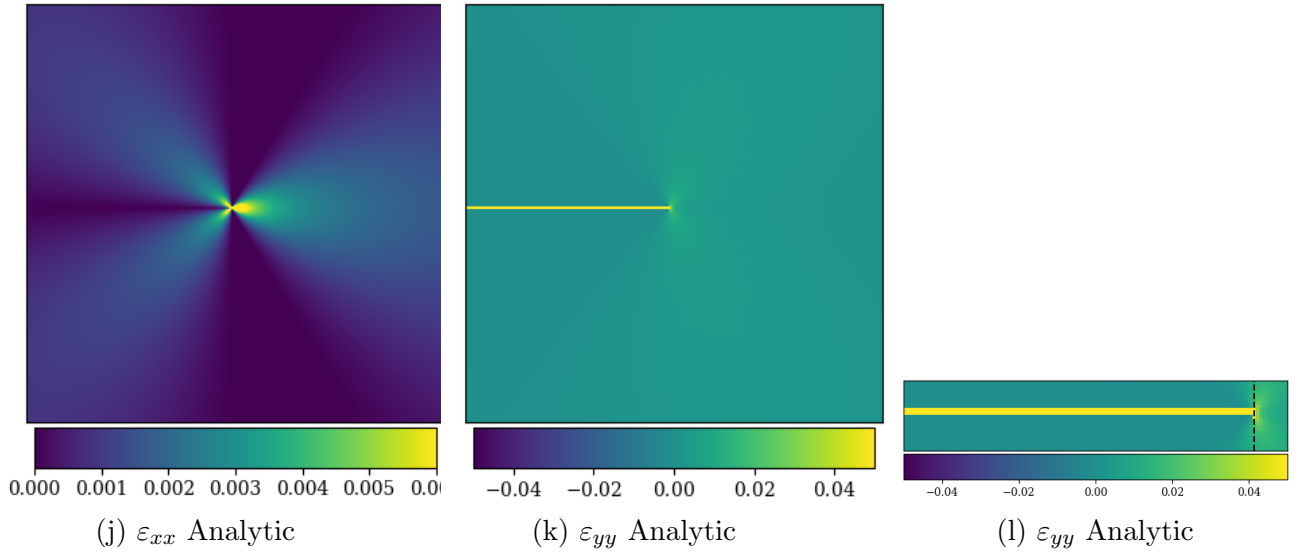
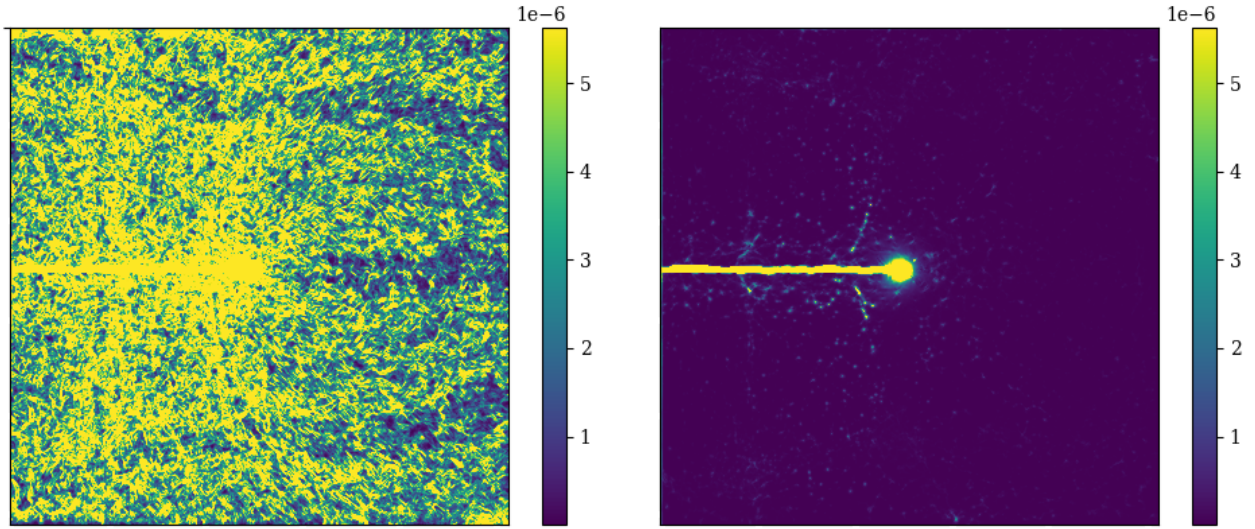


Figure 3.15: The strain fields in terms of λ . The third column shows a zoom of the second column over the crack. The first and second rows show fields obtained with values close to the minimum and maximum regularization parameter ($\lambda = 7$ and $\lambda = 600$) used in the mask, while the third row shows the results obtained with the local parameter change. Finally, the analytical fields are presented in the last row. The dotted line corresponds to the analytical abscissa of the crack tip.

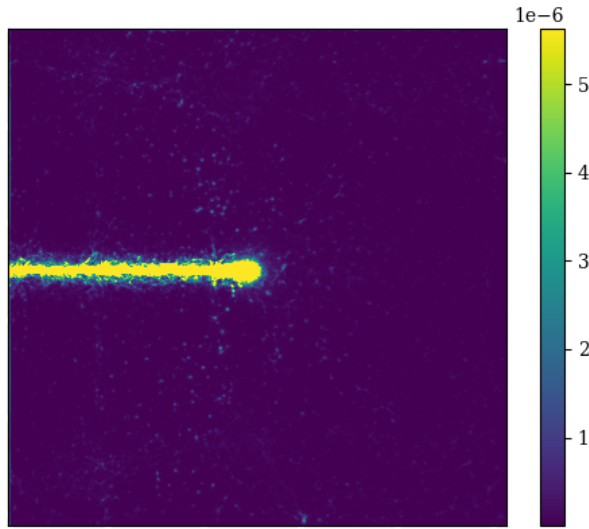
The time step τ must be less or equal to $\frac{1}{4}$ and θ is a small constant.

This type of decomposition could be a promising avenue to explore in the case of tomographic images, as the image texture is determined by the nature of the studied material, and this can, in some cases, pose a challenge for the optical methods. One can find a suggestion for setting the parameter values in the paper of Wedel [7].



(a) $\lambda = 7$

(b) $\lambda = 6 \times 10^2$

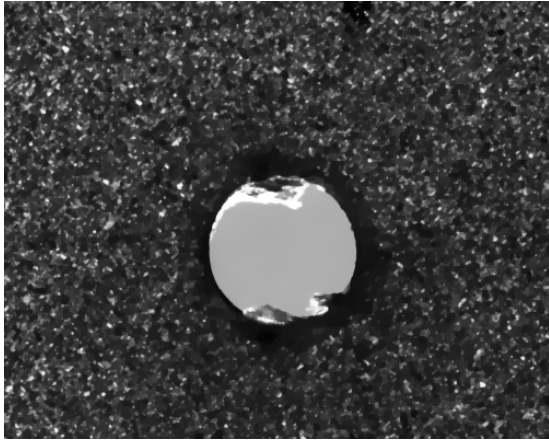


(c) λ local

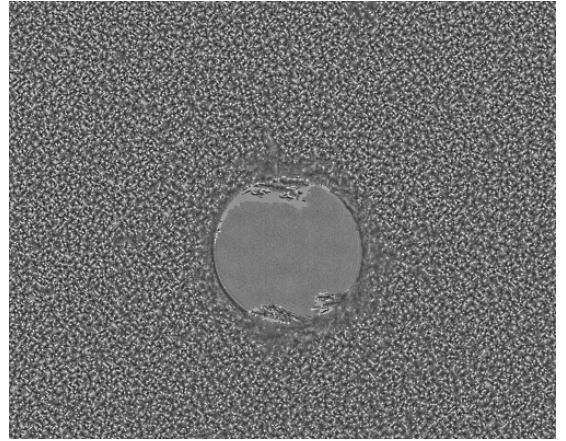
λ	Crack Pixels	AEE $\times 10^{-4}$	MNSDE		
			Global	Near Crack	Out Crack
7	3	1.46	0.39	17.44×10^{-1}	45.62×10^0
600	6	1.68	0.19	24.35×10^{-1}	3.25×10^0
<i>Local</i>	3	1.53	0.15	19.18×10^{-1}	3.12×10^0

(d)

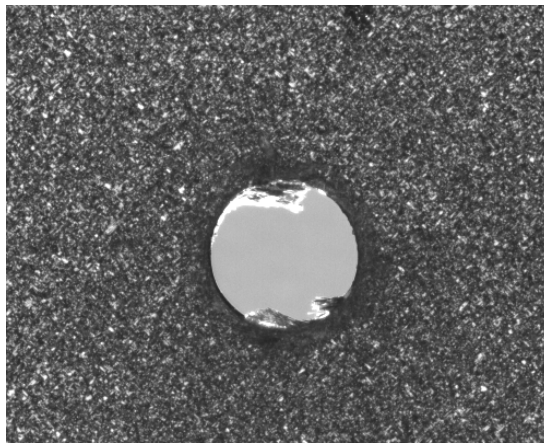
Figure 3.16: NSDE obtained with different regularization amplitudes and using the local regularization mask. The table indicates the number of pixel required to capture the crack as well as the AEE, the MNSDE of the global field as well as in the vicinity and outside the crack.



(a) Structure part



(b) Texture part



(c) Original image

Figure 3.17: Example of the Structure-Texture decomposition

4. GCPU_OpticalFlow: a GPU accelerated Python software for strain measurement

In the previous chapter, we examined a few examples of metrics not often utilized in DIC. These include the Lorentzian and the Charbonnier metrics. These metrics depend on an adjustable parameter that allows for a transition between the behavior of the L^1 and L^2 norms. Besides, the linear system to be solved is created in each iteration of the calculation since the derivatives of these metrics are not linear as shown in 2.3.5. On the other hand, the L^2 norm yields results less satisfying than those generated with the other penalty functions, but its derivative is linear, making the minimization straightforward. This result eliminates the need for complex algorithms to find the optimum of the energy using the L^2 norm.

It is also pertinent when obtaining results within a reasonable time frame is crucial, as seen in real-time mechanical testing scenarios where rapid data acquisition is necessary to identify any abnormal behavior or defect occurring during the test. To address this, we explore in this chapter the techniques employed in developing a parallel and a memory-optimized software for full-field measurement. This work has been employed within our laboratory in various experiments with different objectives, such as fatigue tests or the calibration of cameras [88] which we will discuss in the following sections in the form of an application of GCPU_OpticalFlow which is the name of the pixel-wise DIC software [99] we have developed.

Finally, the remainder of this chapter is a work that has been submitted to the SoftwareX journal with the ISSN 2352-7110.

Authors Ahmed Chabib, Jean-François Witz, Pierre Gosselet, Vincent Magnier.

Affiliation Univ. Lille, CNRS, Centrale Lille, UMR 9013 - LaMcube - Laboratoire de Mécanique, Multiphysique, Multi-échelle, F-59000 Lille, France.

Abstract This paper introduces an open-source pixel-wise Digital Image Correlation tool written in Python and targeting graphics processing units (GPUs) with the help of Cupy and Rapids-cuCim libraries. It is capable of computing the kinematic fields that transform an image into another in an efficient and quick way and it allows to treat large images in the GPU. Even if GCPU_OpticalFlow can be easily used by communities concerned by the estimation of displacement, it is particularly tuned to estimate consistent strain (gradient) field. The detection of a crack in a material is presented in this work as a demonstration.

Keywords DIC, Optical Flow, Python, GPU, Mechanics, Strain, Measurement, CUDA

4.1. Motivation and significance

4.1.1. Introduction

Digital Image Correlation (**DIC**) is one of the most popular optical methods that provide full-field displacement between consecutive images. After the notable advancement of this technique, DIC has become more and more used in several scientific disciplines, notably in materials science for strain measurement in order to deduce the properties of materials.

Most of DIC software are based on a local [50, 53, 54, 55, 56] or a global approach [52, 57, 25, 59]. Although local DIC can provide pixel-wise displacement when the step size is set to 1 pixel, the results in this case are still derived from a template and not from a single pixel as is the case with GCPU_OpticalFlow which is an open source code based on D. Sun [4] methods.

The calculation of the kinematic fields from a sequence of images captured with modern high-resolution cameras become increasingly expensive in terms of memory and computation, due to the copious volumes of data (tomography etc.). Hence the need to develop an optimized fast code that takes up less memory. We propose to use a matrix-free implementation of Krylov iterative solvers [60] suited to Graphics Processing Units(GPU) so that no voluminous matrix needs to be stored.

Some of the existing DIC programs are written with a low level programming language (C++ for the example) like DICe [52], Ncorr [53] which make their understanding, execution and modification complicated. Our code is implemented in the popular python language and relies on classical libraries to optimize its performance and offload computations on the available hardware accelerators. Presently, the DIC programs are adapted to different architectures, like CPU [52, 53, 54, 55, 56], Raspberry Pi [51] or even in GPU [58]. In contrast to this software where no assumption on the nature of the displacements is made, the previous work of our research team GPUcorrel [58] is based on a global approach and uses an integrated correlation step. In other terms, the program expects a list of displacement fields to extract from the image sequence and the calculated result is a linear combination of the given basis. Moreover, GPUcorrel uses PyCUDA kernels which have the form of CUDA-C code to target the GPU. Our software uses Cupy [47] and Cucim [64] libraries that contain a GPU-accelerated version of Numpy, Scipy and Skimage functions and called easily using the same syntax. The engine is able to use the alternatives of the GPU libraries on CPU when the GPU or one of its related libraries is missing. For instance, if CuPy is not found, Numpy will be used instead, all in the same single straightforward Python code comprehended by non-experimented developers.

This article and software represents a bridge between the optical flow community that does not deal with mechanical imaging and the DIC community, and enables the users, particularly from mechanical engineering, to measure the pixel-wise displacements and efficiently deduce strain.

4.1.2. Principles of DIC and Optical Flow

Let I_1 and I_2 be a sequence of two images, where I_1 is the reference image and I_2 is the deformed one. The objective of DIC is to find a transformation of the image that keeps the graylevel

invariant between two images.

$$I_1(x) = I_2(x + u(x)), \quad u \text{ is the displacement field.} \quad (4.1)$$

It is impossible to estimate the displacement that transforms I_1 to I_2 at each pixel directly from the conservation law (4.1) as the number of unknowns exceeds the number of equations. To solve this problem, DIC algorithms transform the ill-posed optical flow problem into a well-posed one by using a grid of pixels, aiming at reducing the number of the degrees of freedom [2]. This grid can be local, FEM (Finite Element Method) [27] or even X-FEM (Extended Finite Element Method) [3]. Therefore, the displacement, written u_h , is interpolated on a basis of continuous shape functions with local support. In order to quantify the difference between the images, the integral of squared differences is chosen as a metric. Then the purpose of DIC is the minimization of the objective function ((4.2)) with respect to u_h .

$$E(u_h) = \int_{\Omega} \left(I_1(x) - I_2(x + u_h(x)) \right)^2 dx \quad (4.2)$$

The use of a reduced dimension search space acts in itself as a regularization. Nevertheless, when willing to decrease the size of the grid, a Tikhonov-type regularization is commonly added, leading to a new objective function:

$$E(u_h) = \int_{\Omega} \left(I_1(x) - I_2(x + u_h(x)) \right)^2 + \lambda (\nabla u_h)^2 dx, \quad \text{for } \lambda > 0. \quad (4.3)$$

Optical flow methods share the purpose of DIC but they make the hypothesis that a well mastered regularization makes it possible to get rid of the interpolation grid, leading to computations at the pixel scale, see [1, 6]:

$$E(u) = \int_{\Omega} \left(I_1(x) - I_2(x + u(x)) \right)^2 + \lambda (\nabla u)^2 dx \quad (4.4)$$

Notice that in [4], other energies were proposed where Li and Osher's median filter [11] was interspersed. This resulted in the removal of the outliers at the cost of a slight energy increase.

4.2. Software description

4.2.1. Software functionalities

GCpu_OpticalFlow is coded in Python language and requires some external modules. It is implemented as simply as possible in order to be understandable and editable by all the researchers regardless of their programming skills. Therefore, the software uses Cupy and cuCim to run on the GPU. Cupy is a high-level GPU-accelerated computing library which contains a GPU version of NumPy and SciPy libraries able to run on NVIDIA CUDA or on AMD ROCm platforms, whereas cuCim is a library that provides a CUDA-accelerated implementation of a wide range of image processing operations existing in scikit-image. This software only works with NVIDIA GPUs with at least the 3.0 version of Compute Capability.

At every warping step, the algorithm requires the resolution of the following linear system:

$$\begin{pmatrix} I_x^2 + \lambda\Delta & I_x I_y \\ I_x I_y & I_y^2 + \lambda\Delta \end{pmatrix} \begin{pmatrix} dU^{k+1} \\ dV^{k+1} \end{pmatrix} = - \begin{pmatrix} I_x I_t + \lambda\Delta U^k \\ I_y I_t + \lambda\Delta V^k \end{pmatrix}, \quad (4.5)$$

where k is the current iteration, dU^{k+1} and dV^{k+1} are respectively the horizontal and the vertical flow increment, U^k and V^k are respectively the horizontal and the vertical current estimated flow field, λ represents the regularization parameter, I_x and I_y are the spatial derivatives of the image, I_t is the temporal derivative and Δ is the discrete Laplace operator. When handling high resolution images, the matrix can not fit in the memory and therefore the resolution is computationally costly.

The preconditioned minimum residual method MINRES [39] is used in this software. The choice of this solver is motivated not only by its ability to take advantage of the symmetry of the system matrix but also by the possibility of a matrix-free implementation. While evaluating the matrix-vector product in the resolution step, the Laplace operator is computed using the *laplace()* function of *scipy* or *cupyx.scipy* for GPU, to avoid any additional matrix storage.

In numerical analysis, the preconditioner is a practical tool to increase the rate of convergence, in our case it has been observed that using the following preconditioner P can decrease the number of iterations that MINRES takes to converge.

$$P = \begin{pmatrix} (I_x^2 + 8\lambda)^{-1} & 0 \\ 0 & (I_y^2 + 8\lambda)^{-1} \end{pmatrix} \quad (4.6)$$

As usually done in DIC methods, the gradient of the second warped image $\nabla I_2(x+u)$ is replaced by the gradient of the reference image of the sequence $\nabla I_1(x)$ so that costly computation can be avoided. This can be explained by the fact that at the convergence the images of the sequence converge to the same solution [12].

4.2.2. Software architecture

To estimate flow fields with large motion, the method uses an incremental multi-resolution technique. A pyramid of images is built by down-sampling the sequence. The flow field estimated in a level is thereafter up-sampled and used to initialize the next level as illustrated in Fig. 4.1. The main function of the software is *compute_flow_base()*. It returns u_l and v_l , the horizontal and the vertical displacements for a specified level l after solving the system (4.5) 10 times and getting the best flow increments, and it takes as one of its arguments the up-sampled displacements computed in the previous level as an initial state. It should be noted that in the top level, the initialization is given by *optical_flow_tv1()* function of *skimage* or *cucim.skimage*. *compute_flow_base()* is called at each level by the *compute_flow()* function which allows the management of the pyramid and returns the final estimated displacements. The flowchart in Fig. 4.2 shows the main functions of the software modules.

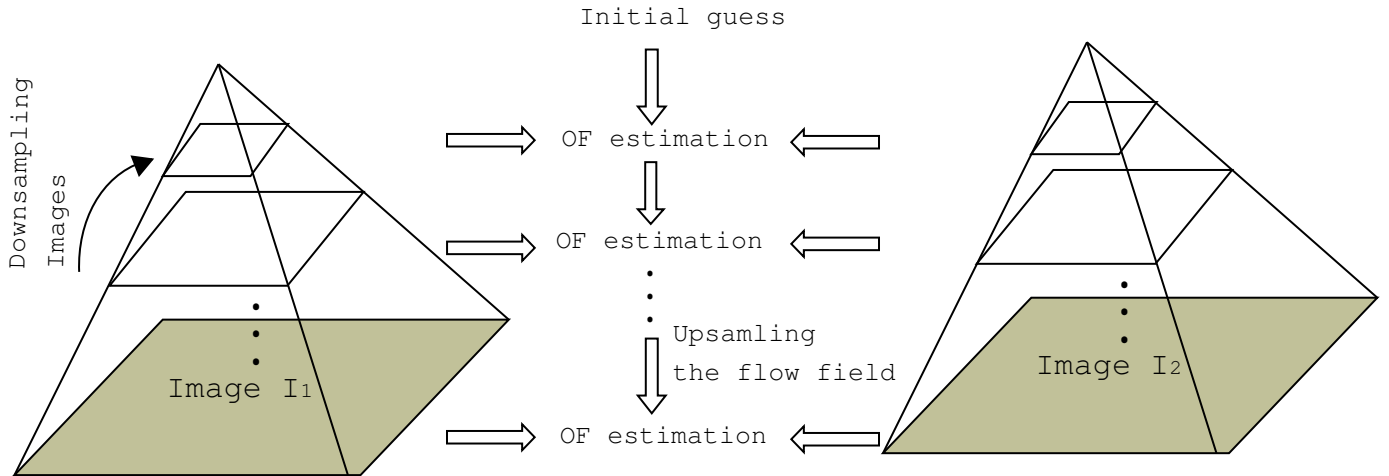


Figure 4.1: Representation of the pyramidal approach

4.3. Illustration

We created, in the folder *Test*, a script named *mainscript.py* which allows the user to quickly test the program. This script is able to estimate the motion from the sequence of images in the *Images* folder. It generates two *.npy* files, *u_cucim* and *v_cucim* corresponding to the horizontal and vertical displacements.

To demonstrate the usefulness of the method, we used two images of a holed ± 45 carbon/epoxy composite material specimen Fig. 4.4 on which a uniaxial tensile force is applied. The setup is schematized in Fig. 4.3, it generates a sequence of 1120×7830 images where each pixel is equivalent to 0.022 mm. This type of material is chosen because of its particularity of creating cracks, which provides an excellent case to test and to analyze the method on discontinuities.

The results presented in Fig. 4.5 and Fig. 4.6 show that the method can detect the only crack in the sample. We observe that the choice of the amplitude of the regularization parameter as well as the size of the window of the median filter could play an important role. Increasing the value of λ leads to decreasing the noise on the image but it can lead to strain diffusion as the result of using a quadratic norm. We also remark that increasing the size of the median filter can provide less noisy results.

4.4. Impact

In order to compare the performance of our method with a global DIC software, we use YaDICS [61], a program which demonstrated its competence to effectively generate the kinematic fields [62] in various disciplines of experimental mechanics. The window size in the DIC approach is decreased to the maximum in order to compare the performance, since GCPU_OpticalFlow generates the fields at every pixel. The result is shown in Fig. 4.7 and Table 4.1. We observe, that the crack is clearly detected. Note that the interpolation error given by the norm of the difference between the warped image and the reference one, is

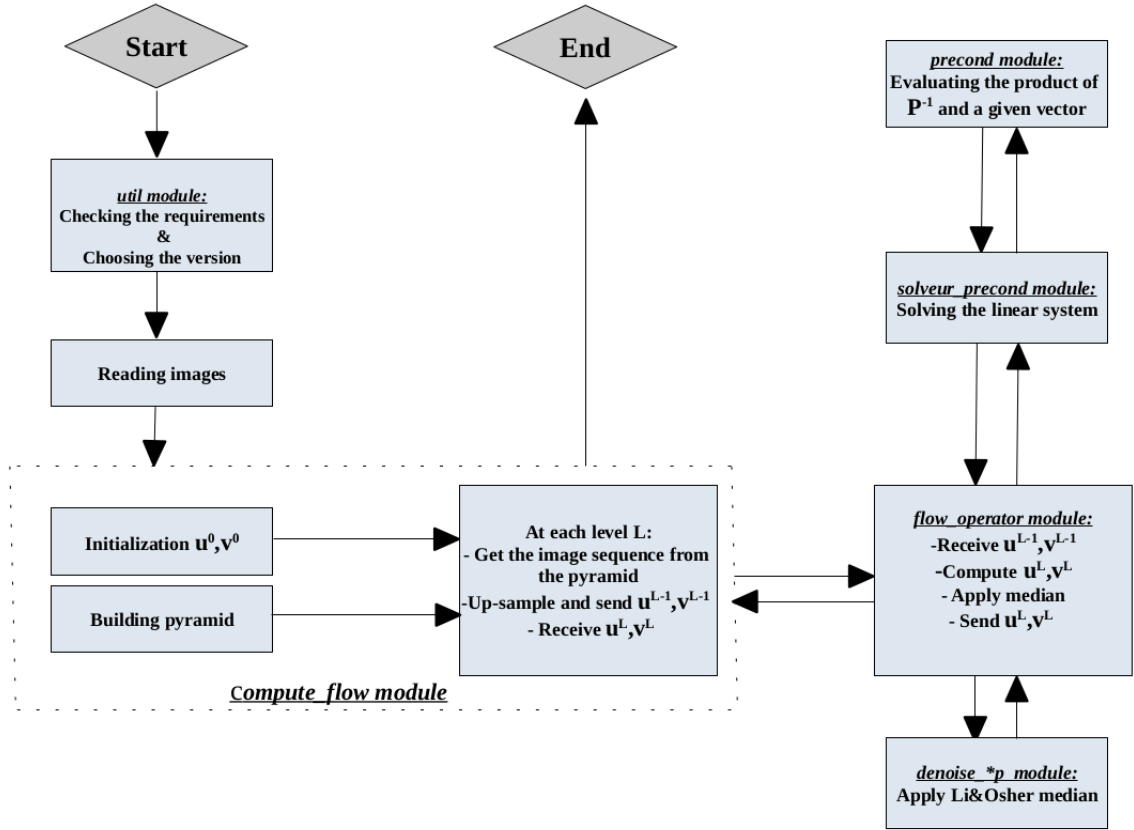


Figure 4.2: Flow-chart of flow field estimation with *GCPU_OpticalFlow*

5.42×10^{-2} and 5.64×10^{-2} for YaDICs and *GCPU_OpticalFlow* respectively, meaning that the quality of the solution is unchanged. The main difference is that the DIC method takes more than 11 minutes for YaDICs using an AMD Ryzen Threadripper 1950X 16-Core Processor. On the other side, due to the performance improvement efforts, the *GCPU* takes only 15 seconds using a the same processor with an NVIDIA GeForce RTX 3070 GPU, meaning that our software in this case is 44 times faster.

Field	Mean	Std	Median	NMAD
u	$2.119e-1$	$2.820e+0$	$2.719e-2$	$1.374e-2$
v	$2.144e-1$	$7.220e-1$	$2.691e-2$	$1.305e-2$
ε_{xx}	$3.046e-2$	$1.343e+0$	$4.746e-3$	$3.031e-3$
ε_{yy}	$2.075e-2$	$2.433e-1$	$3.942e-3$	$2.526e-3$
ε_{xy}	$1.486e-1$	$2.863e+0$	$2.475e-3$	$1.726e-3$

Table 4.1: Statistical indicators (in pixels) of the root-mean-square error (RMSE) of the kinematic fields computed by YaDICs and those estimated by *GCPU_OpticalFlow*.

For the purpose of getting a clear idea about the difference between the residuals of both software, a graph is presented in Fig. 4.8. We mean by the image energy, the energy calculated by first term of the function E (4.4) related to the graylevel conservation, while the gradient

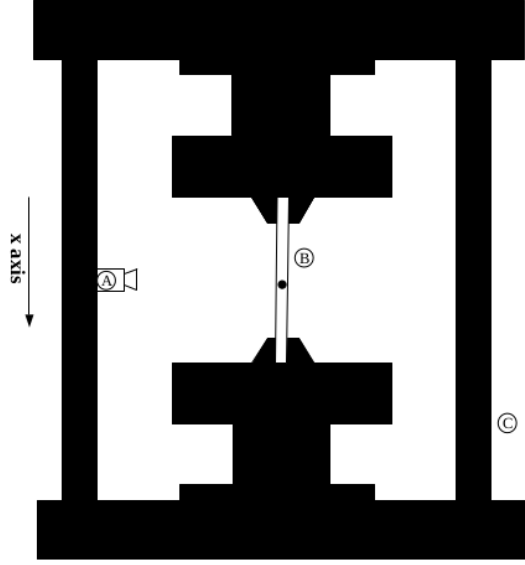


Figure 4.3: Experimental setup with (A) the camera, (B) the sample under testing and (C) the uniaxial tensile testing machine.

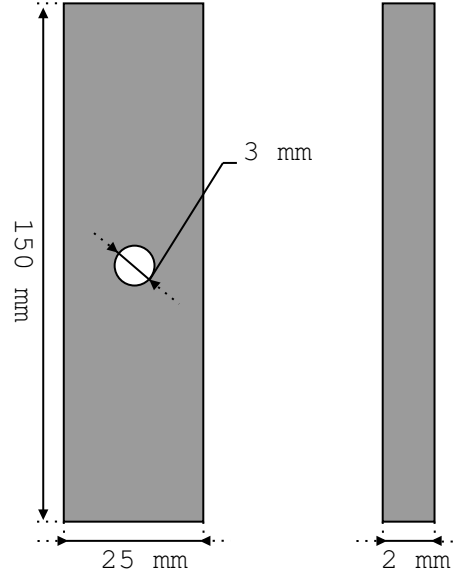


Figure 4.4: Dimensions of the used ± 45 Carbon Epoxy specimen.

displacement energy is the value given by the gradient of the transformation that figures in the second term of the same function E . Different values of λ varying from 3×10^3 to 10^6 are used for GCPU, and various element size between 2 and 50 are used in YaDICs case, since these two parameters are meant to smooth the calculated fields by each program. It can be clearly observed that for an equivalent smoothness level, GCPU provides better quality results as its image energy is lower than YaDICs. To quantify the difference, the areas under the curves are measured and we can notice that the value given by YaDICs is 6.42 greater than the one calculated by GCPU_OpticalFlow.

The methods used by the program were coded in [48] using Matlab. Unlike Python which became a vastly accepted open-source programming language, Matlab isn't free nor open-source. The Matlab implementation was expensive in terms of memory and calculations. To estimate the motion for an $(N \times N)$ pixels sequence, the storage of $O(N^4)$ coefficients of Laplace matrix and of the main matrix of the linear problem (4.5) was needed. Thanks to GCPU_OpticalFlow, we are able to solve the same problem using only $O(N^2)$ coefficients. Using this approach, we can treat larger volume of data on the GPU.

In the future this work will be integrated to Crappy [49], which is a locally developed Python module capable of commanding advanced experimental mechanical tests and able to acquire data in real time.

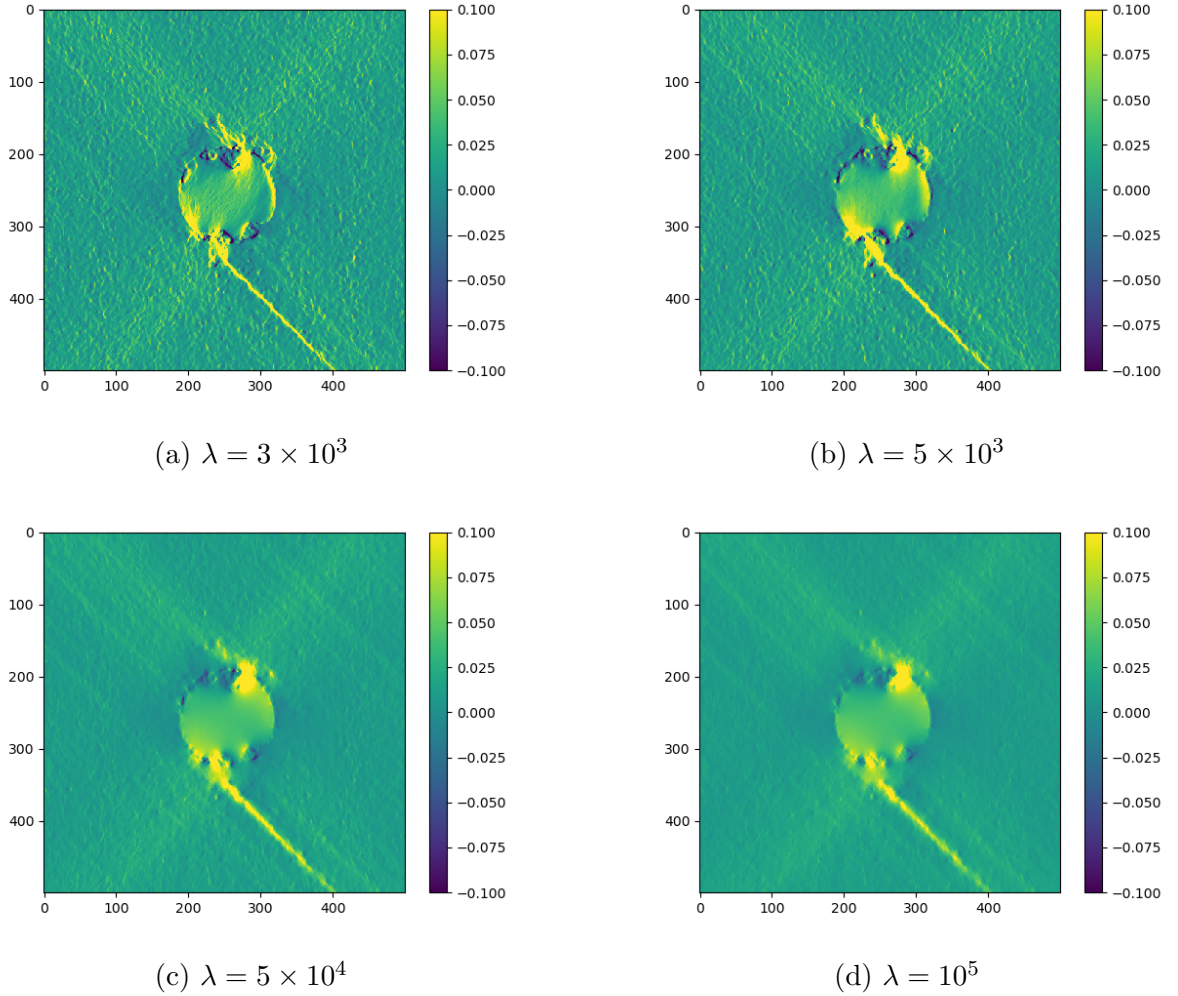
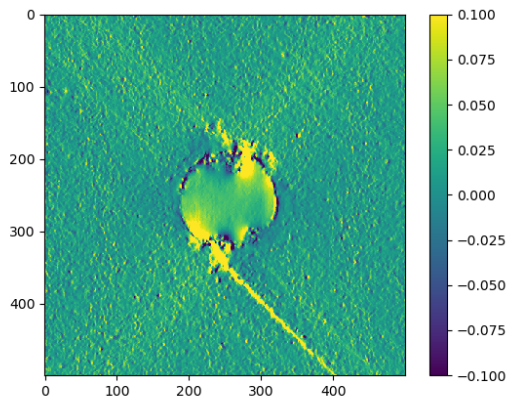


Figure 4.5: Uniaxial strain ε_{xx} in function of λ with fixed 3×3 median filter

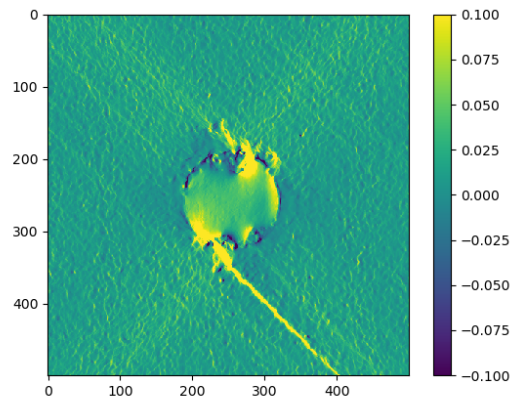
4.5. Conclusion

The presented software is an open-source Python module for full field displacement and strain computing, able to run in both GPU and CPU and based on D. Sun models. In our implementation, we used a matrix-free Krylov solver to reduce the computational cost and the memory storage. It has been shown by the example that the code computes efficiently the strain at each pixel of the image. We also presented the benefits and limitations of the size of the median filter and the parameter of regularization on the discontinuities.

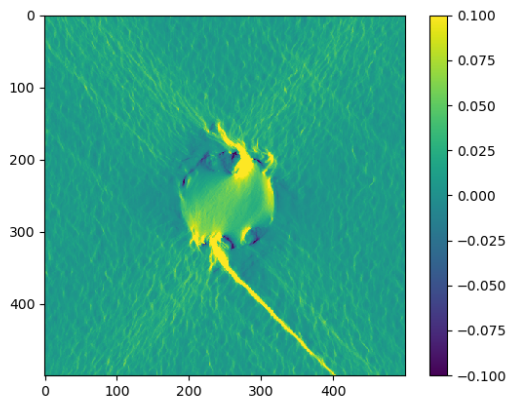
Finally, in the long term this method will be extended to work on three-dimensional tomographic images.



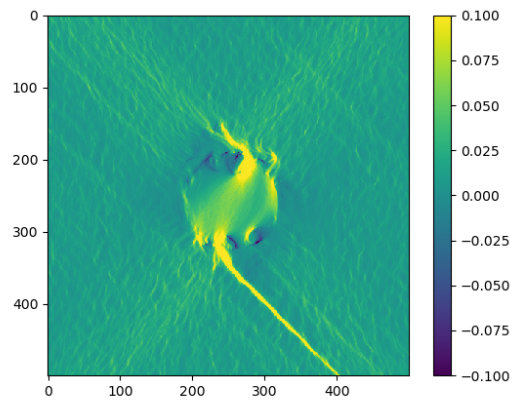
(a) Median 3×3



(b) Median 5×5



(c) Median 9×9



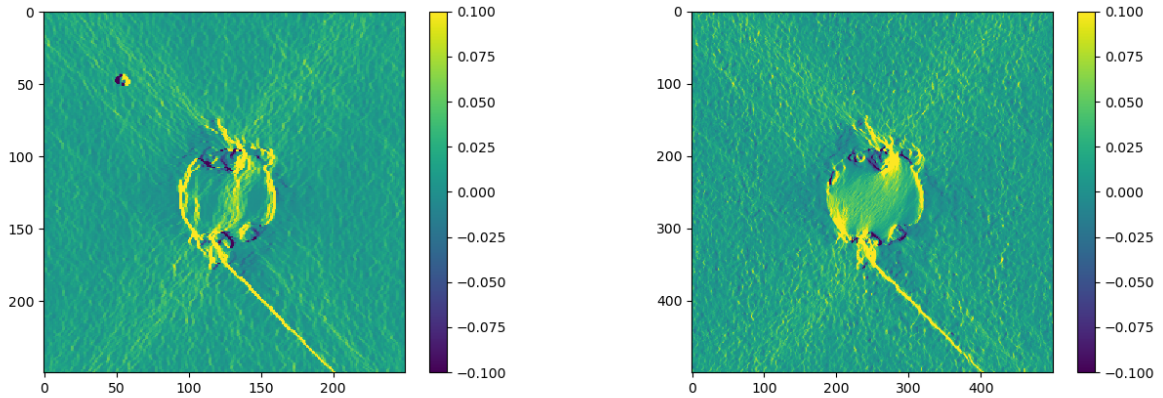
(d) Median 11×11

Figure 4.6: Uniaxial strain ε_{xx} in function of the median size with fixed $\lambda = 3 \times 10^3$

4.6. Acknowledgements

The authors wish to express their very special appreciation to Mr. Victor Couty and Mr. Adrien Berger for providing them with test images.

Required Metadata



(a) YaDICs strain with 2×2 window, 3×3 median (b) GCPU strain $\lambda = 3 \times 10^3$, 5×5 median

Figure 4.7: Strain field computed by YaDICs and GCPU_OpticalFlow

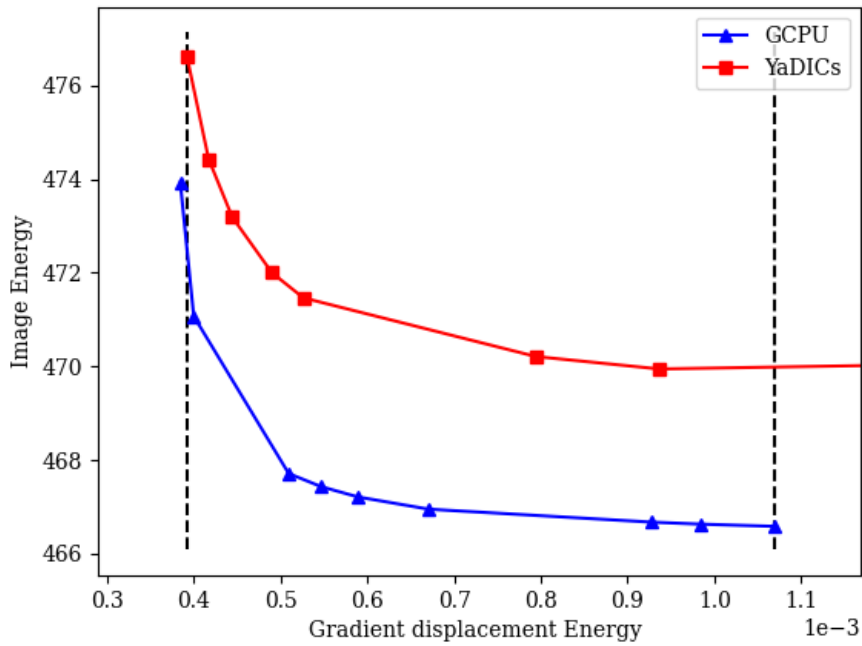


Figure 4.8: Energy image in terms of gradient displacement energy for both software. The dotted vertical lines indicate the limits of the zone where the two software are compared

4.7. Applications

This short section presents two contexts where the software developed in this thesis was applied by other PhD students for their own studies. We focus on technical and performance details

Nr.	Code metadata description	
C1	Current code version	v1.0
C2	Permanent link to code, repository used for this code version	https://github.com/chabibchabib/GCpu_OpticalFlow
C3	Permanent link to Reproducible Capsule	
C4	Legal Code License	GPLv2+
C5	Code versioning system used	git
C6	Software code languages, tools, services used	Python, Cupy, OpenCv, Rapids Cucim and Numba
C7	Compilation requirements, operating environments & dependencies	Numpy 1.20.3, Scikit-image 0.16.2, Scipy 1.6.3, OpenCV 4.2.0 or newer versions. Numba, NVIDIA CUDA GPU with the Compute Capability ≥ 3.0 , CUDA Toolkit ≥ 10.2 , Cupy and Cucim
C8	Link to developer documentation, manual	https://gcpu-opticalflow.readthedocs.io/
C9	Support email for questions	ahmed.chabib@univ-lille.fr

Table 4.2: Code metadata

and leave the scientific conclusions to the thesis' authors.

4.7.1. Calibration of cameras

Calibration of cameras is a widely used technique in the field of computer vision aimed at establishing the relationship between the position of a point in the image and its spatial coordinates, thus modeling a camera system by estimating its properties. The parameters of a vision system are divided into two types: intrinsic parameters, which represent camera parameters such as focal length and the position of the camera's principal point, and extrinsic parameters, which describe the camera's orientation and position relative to a global coordinate system.

Understanding these parameters allows us to generate a three-dimensional representation from a two-dimensional image. This property is highly advantageous for reconstructing surface topography by measuring height variations. The GCPU_OpticalFlow contributed to the development of a Python model for image calibration using the Soloff methods [93], developed within our laboratory [88], known as PyCASO. This work has been published in the SoftwareX journal. PyCASO requires two sets of images of the same object taken by two cameras (Left and Right) at different heights z . To accomplish this, a special setup is typically used, consisting of a light source, a calibration target containing distinguishable pixel patterns that will be used in the image correlation step, and an instrument capable of translating along the z -axis. A schematic representation of this setup is presented in Fig. 4.9

The objective is to establish the relationship between spatial coordinates $X = (x, y, z)$ and the coordinates of the two cameras, which we will denote as $P_r = (C_r, R_r)$ and $P_l = (C_l, R_l)$ for

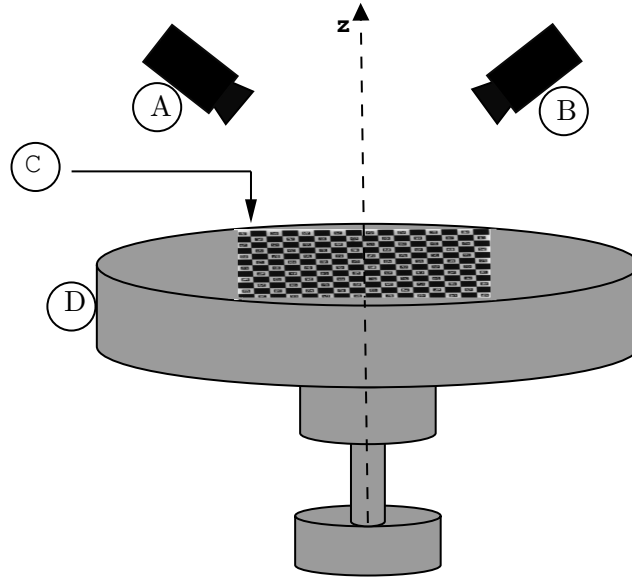


Figure 4.9: The device used for camera calibration, where (A) Left camera, (B) Right camera and (C) The calibration pattern,(D) Moving table

the right and left camera respectively. First, the module proposes to reconstruct the object's surface using direct methods [94] that assume that each component of the spatial coordinates is the image of the pixel coordinates in the left and right images through a polynomial. For example, for the component x , we assume that there exists a polynomial D_x such that:

$$x = D_x(P_l, P_r) = d_0 + d_1 C_l + d_2 R_l + d_3 C_r + \dots + d_{n-2} C_r R_r^{p-1} + d_{n-1} R_r^p$$

where p is the polynomial degree and n is the number of calibration constants to determine for each component. The same assumption is made for each direction, leading to the following linear system:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} D_x \\ D_y \\ D_z \end{pmatrix} = \begin{pmatrix} d_{0x} & d_{1x} & \dots & d_{(n-1)x} \\ d_{0y} & d_{1y} & \dots & d_{(n-1)y} \\ d_{0z} & d_{1z} & \dots & d_{(n-1)z} \end{pmatrix} \begin{pmatrix} 1 \\ C_l \\ \vdots \\ R_r^p \end{pmatrix} = DM \quad (4.7)$$

This assumption is applied to a set of points with a cardinal N less than the number of points in the image in order to generate more equations, allowing the system to be solved using a pseudo-inverse, such as the Moore-Penrose pseudo-inverse [92], for instance. PyCASO also offers, in addition to the direct methods, another approach called Soloff, which consists of assuming this time that for each component of the coordinates P_r and P_l of each camera is the image of the spatial coordinates X by a polynomial. For example, for the component C_r , we assume that there exists a polynomial S_{C_l} such that $C_l = S_{C_l}(X)$. Given that:

$$S_{C_l}(X) = s_0 + s_1 x + s_2 y + \dots + s_{n-2} y z^{p-1} + s_{n-1} z^p$$

By grouping these equations into a matrix, we obtain the following problem:

$$\begin{pmatrix} C_l \\ R_l \\ C_r \\ R_r \end{pmatrix} = \begin{pmatrix} S_{C_l} \\ S_{R_l} \\ S_{C_r} \\ S_{R_r} \end{pmatrix} = \begin{pmatrix} S_{0C_l} & \dots & S_{(n-1)C_l} \\ S_{0R_l} & \dots & S_{(n-1)R_l} \\ S_{0C_r} & \dots & S_{(n-1)C_r} \\ S_{0R_r} & \dots & S_{(n-1)R_r} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \\ \vdots \\ z^p \end{pmatrix} = SM \quad (4.8)$$

The method for determining the coefficients of the matrix S is identical to that of the direct methods. Regardless of the calibration method used, it is necessary to identify the pixels in both the left and right images, and this is where image correlation comes into play especially with the use of our software that can provide the deformation linking the two images. In other words, `GCPU_OpticalFlow` determines the displacement fields $w = (u, v)$ such that $P_l(x) = P_r(x + w(x))$ for all points x of the image. Once the matrix D is determined along with the displacement field, the surface topography can be generated using the direct approaches by calculating the image of each point in the image using the polynomial whose coefficients are stored in D . However, for the Soloff methods, an optimization step is required, relying on the Levenberg-Marquardt method [90] or regression methods [91].

The different methods were tested for reconstructing the surface of a coin with a speckle pattern and compared with the results provided by a profilometer. It was observed Fig. 4.10 that the direct methods deviate more from the reference curve compared to the Soloff methods, which provide more accurate results. However, the direct methods are faster than the Soloff methods, which can be explained by the computational complexity of the optimization algorithms used for point identification in the image.

Finally, smoothing appears in regions with a significant gradient of height, which is explained by the nature of the least squares function used for calculating the displacement fields. It could be interesting to investigate the impact of changing this norm to more robust metrics that ensure the preservation of discontinuities in the field, as described in the chapter 3.

4.7.2. Microstructure gradient's

This work was applied as part of a doctoral thesis [102] aimed at studying the microstructure gradient's effect on the fatigue of a metallic material. More specifically, it was used in a mechanical test under constant load amplitude for crack detection in a test specimen.

It has been observed that post-processing a thousand image sequences acquired during the test requires eight days of computation using 2x2 elements for YaDICs. However, using the same machine equipped with a Nvidia GeForce RTX 3070 graphics card, this only requires four hours. A figure showing the crack length in a strain field as a function of window size and Tikhonov parameter is presented in Fig. 4.11. Notably, in addition to the regularization parameter effect, the element size exhibits a similar behavior. Increasing the element size results in excessive smoothing of the crack.

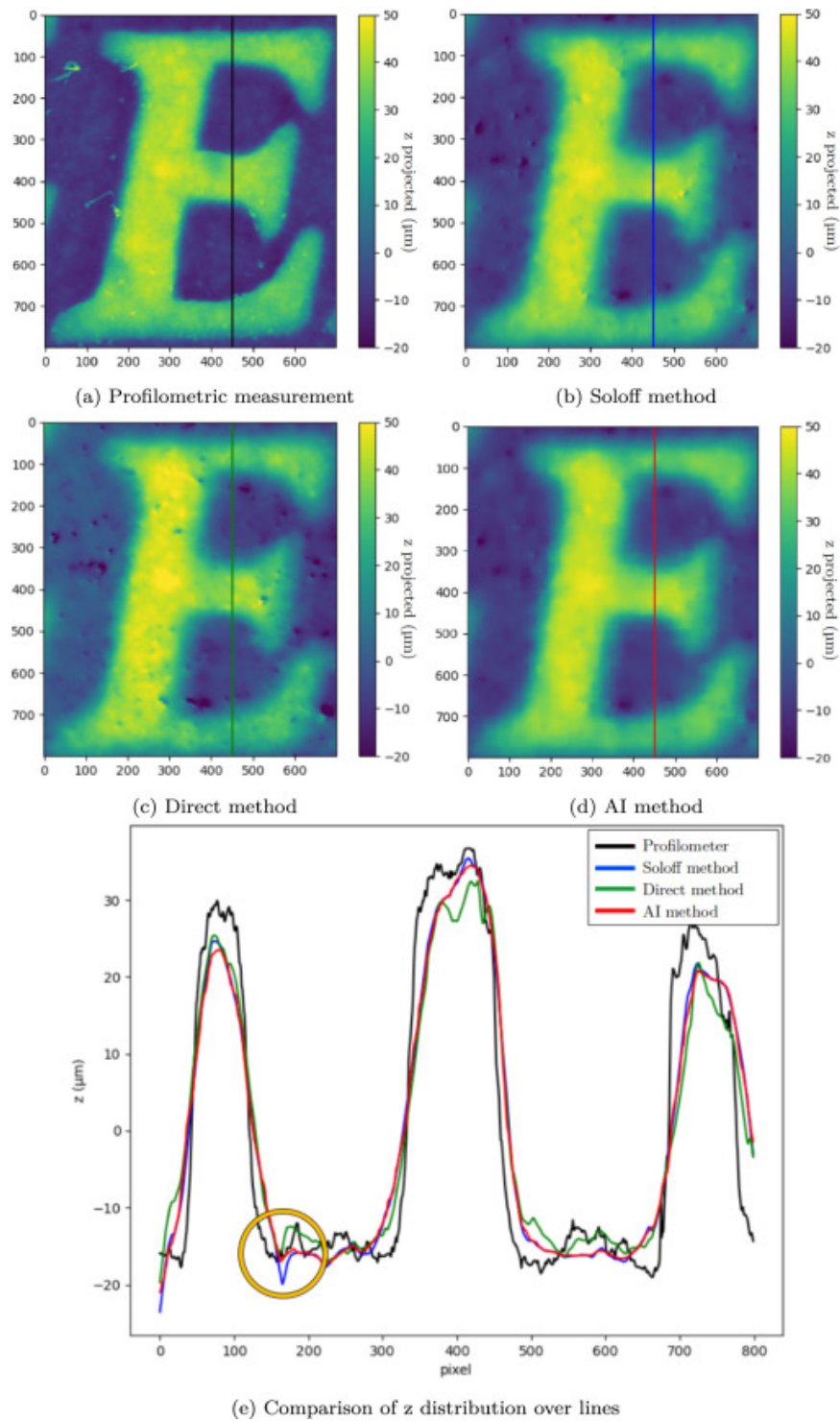


Figure 4.10: (a) Profilometric topography. (b) Soloff topography. (c) Direct topography. (d) AI topography. (e) z -distribution of the 450th column [88]

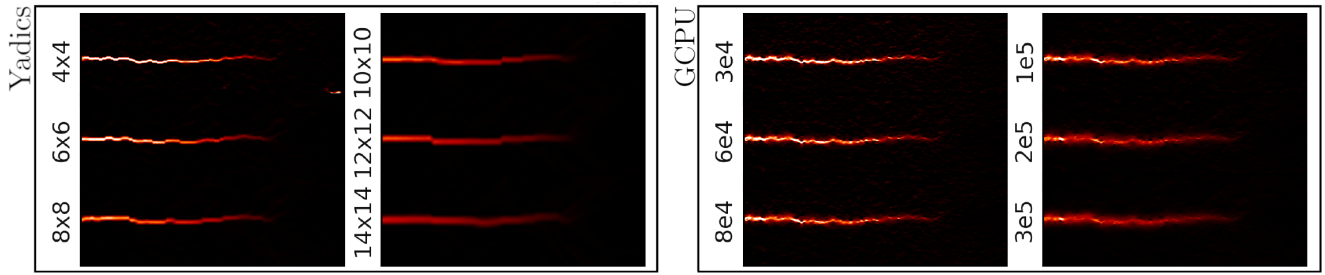


Figure 4.11: Strain fields in the loading direction using YaDICs for different correlation window sizes and GCPU for several regularization values [102]

5. Interplay between preconditioning and regularization for linear ill-posed problems solved by conjugate gradient: Application to optical flow estimation

In the previous chapter we aimed at creating a simple code, hence the use of already implemented and accelerated solvers which seem to be adapted to our needs. The conjugate gradient algorithm seems to be better suited to this problem thanks to its positive definite aspect. This chapter contributes to a better understanding of the system to be solved generated by Horn & Schunk, and benefit from our knowledge of Krylov solvers in the context of ill-posed problems, in particular those on the Augmented Preconditioned Conjugate Gradient and filtering by Ritz values, in order to adjust the Tikhonov parameter and carry out a Picard-type spectral analysis. This chapter is presented under the form of a draft article, to be soon submitted in International Journal for Numerical Methods in Engineering ISSN: 1097-0207.

Authors Ahmed Chabib, Jean-François Witz, Pierre Gosselet, Vincent Magnier.

Affiliation Univ. Lille, CNRS, Centrale Lille, UMR 9013 - LaMcube - Laboratoire de Mécanique, Multiphysique, Multi-échelle, F-59000 Lille, France.

Abstract This paper investigates the possibilities offered by combining regularization and preconditioning by the same symmetric positive semi-definite operator when solving ill-posed problems. We study the question of the stopping criterion, and the possibility offered by Ritz eigen elements of a posteriori filtering of the solution and tuning of Tikhonov's weight. The method is applied as the linear solver of an optical flow estimator and it is coupled with a subspace recycling strategy.

Keywords DIC, Optical Flow, Augmented Conjugate Gradient, Strain

5.1. Introduction

Ill-posed systems of equations are ominous in mechanics. They are particularly present in identification problems, like the boundary completion in elasticity [79, 75]. They also appear in methods which involve some compact operation, like the Herglotz' transform to build solutions

to the Helmholtz problems [80]. Beside existence and uniqueness issues, ill-posed problems are defined by the lack of stability between the cause and the effect, in other words small perturbations in the input potentially lead to large modifications of the output.

In this paper, we focus on discrete $n \times n$ linear symmetric positive semi-definite systems of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, so that all properties can be analyzed in terms of the spectrum of \mathbf{A} . Existence and uniqueness are linked to the null-space of \mathbf{A} (strictly zeros eigenvalues) whereas stability is associated with the accumulation eigenvalues near zero. Indeed, a small contribution of \mathbf{b} in an eigendirection associated with a small eigenvalue of \mathbf{A} has a significant impact on \mathbf{x} . Ill-posed problems thus result in poorly-conditioned operators.

Solving such systems amounts to finding a satisfying treatment to these small eigenvalues: truncation, shift, filtering. Truncation consists in not considering the problematic directions, for example using an eigenvalue decomposition (or more generally a singular value decomposition) and only keeping the part of the matrix associated with eigenvalues larger than a given criterion ε [76]:

$$\mathbf{A} \simeq \sum_{i=1}^m \sigma_i \mathbf{u}_i \mathbf{u}_i^T, \quad \sigma_i > \varepsilon > 0, \quad m < n. \quad (5.1)$$

Note that this idea is approximately implemented by iterative solvers since they tend to favor the upper part of the spectrum in the first iterations, so that one only needs to stop the solver early enough.

Shift is generally achieved thanks to a Thikonov regularization [30], that can be written as, in its simplest form:

$$\mathbf{A} \simeq \mathbf{A}_\lambda = \mathbf{A} + \lambda \mathbf{I}. \quad (5.2)$$

In that case $\lambda > 0$ becomes the lower bound of the spectrum of \mathbf{A}_λ . Often, a matrix with more physical sense, acting more locally on the small eigenvalues, is available instead of the identity. Filtering tries to improve a solution after it was computed using another technique by trying to enforce some physical properties. For instance, smoothing can be used to recover regularity in a oscillating solution.

All these techniques are often controlled by a parameter (ε for the truncation, λ for the regularization, the stopping criterion of an iterative solver. . .) which needs to be tuned in order to find a balance between the information inside the original system and the information brought (or removed) by the treatment.

When the accuracy of the data is known, Morozov's principle [81] provides an objective criterion to choose the parameter: the correction brought by the added information should not exceed the noise in the measurement.

When no such data is available, a compromise must be found. Picard's principle [77] compares the eigenvalues (sorted in decreasing order) and the decomposition of the right-hand side (rhs) in the eigendirections. While eigenvalues decrease faster than the rhs, their contribution to the solution remains controlled. The L-curve [78] is a visual aid to find a balance. The solutions for various level of regularization are positioned in a plane ("norm of the residual", "norm of the solution"), in general large regularization leads to low norm of the solution but high error, whereas small regularization leads to lower level of error but large solutions (highly perturbed). Hopefully, some corner exists which realizes a compromise between residual and oscillating solution.

In this paper, we try to combine cleverly these ideas inside a sophisticated solver which offers many useful features: several stopping criteria, filtering of the solution, easy tuning of the regularization. It extends previous work in [75], by exploring the interplay between Tikhonov regularization and preconditioning. This work was initiated in the context of digital image correlation by the optical flow technique, even though it can be applied to very general studies. The paper is organized as follows: in Section 5.2, we briefly present the system that needs to be solved for the computation of the optical flow between two images. In Section 5.3, we recall the augmented preconditioned conjugate gradient algorithm and the computation of Ritz eigenlements, we provide a first discussion on the effects of the preconditioner. In Section 5.4, we consider the case of regularized systems preconditioned by the regularization matrix. Section 5.5 provides assessments and Section 5.6 concludes the paper.

5.2. Basic notions about the optical flow

The optical flow is a digital image correlation technique which aims at estimating the displacement field between two images at the scale of the pixel. Contrarily to very popular approaches in solid mechanics inspired by the Finite Element Method [27], it does not rely on a mesh and on shape functions to approximate the displacement field. Given a sequence of two images (I_1, I_2) , viewed as $N \times M$ arrays of graylevel pixels (in the discrete segment $0 : G_{max}$), it directly aims at finding the transformation $\phi = (\phi_x, \phi_y)$ such that $I_1 - I_2 \circ \phi = 0$. Note that we use interpolation between pixels so that the images can be defined on the rectangle $[0, N - 1] \times [0, M - 1] \subset \mathbb{R}^2$ with values in the continuous segment $[0, G_{max}] \subset \mathbb{R}$ and the displacement $u := (\phi - I_d)$ can take non-integer values (I_d is the identity operator). It is even common to obtain precision below one tenth of a pixel. In order to gain flexibility, and adapt to unavoidable noisy measurements which make the zero unachievable, the problem is better rephrased in terms of the minimization of the “image energy” E_I :

$$E_I^2 = \frac{1}{2} \|I_1 - I_2 \circ \phi\|^2, \quad \text{where } \|I\|^2 = \sum_{\substack{0 \leq i < N \\ 0 \leq j < M}} I(i, j)^2. \quad (5.3)$$

Even under that form the problem is not well-posed, would it only be because there are two times more unknowns than equations.

A solution to recover a well-posed problem is to enforce regularity to the displacement field. A penalty term related to the gradient is then introduced:

$$E^2 = E_I^2 + \frac{\lambda}{2} \|\nabla u\|^2, \quad (5.4)$$

where we kept the Euclidean norm notation for $\|\nabla u\|^2 := \|\partial_x u_x\|^2 + \|\partial_y u_y\|^2 + \|\partial_x u_y\|^2 + \|\partial_y u_x\|^2$. λ is a weight that needs to be tuned in order to balance the contributions of the image energy and of the regularization.

In general a modified Newton approach is used to minimize the energy. Starting from a guess u , the update $u + du$ is computed by solving the system:

$$(\mathbf{A} + \lambda \mathbf{M})\mathbf{x} = \mathbf{b}_\mathbf{A} + \lambda \mathbf{b}_\mathbf{M}, \quad (5.5)$$

with

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} \mathbf{J}_x & \\ & \mathbf{J}_y \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{J}_x & \\ & \mathbf{J}_y \end{pmatrix}, & \mathbf{M} &= \begin{pmatrix} \Delta & \\ & \Delta \end{pmatrix} \\ \mathbf{x} &= \begin{pmatrix} \text{vec}(du_x) \\ \text{vec}(du_y) \end{pmatrix}, & \mathbf{b}_\mathbf{A} &= \begin{pmatrix} \text{vec}((I_1 - I_2 \circ \phi)J_x) \\ \text{vec}((I_1 - I_2 \circ \phi)J_y) \end{pmatrix}, & \mathbf{b}_\mathbf{M} &= \begin{pmatrix} \text{vec}(\Delta u_x) \\ \text{vec}(\Delta u_y) \end{pmatrix}. \end{aligned} \quad (5.6)$$

The vec operator converts images to vectors ($N \times M$ array to NM vector). For $z \in \{x, y\}$, J_z is the z component of the gradient of I_1 , Δu_z is the (scalar) Laplace operator applied to u_z . \mathbf{J}_z is the NM diagonal operator containing the values of the gradient J_z , and Δ is the NM matrix version of gradient operator (with Neumann boundary conditions). All the operators are in fact obtained by discrete difference on the image. As classically done, the gradient of I_1 , which can be computed once for all, is used to approximate that of $I_2 \circ \phi$. As commonly done in image treatment, a median filter is applied to all the computed increments in order to remove outliers caused by the imperfect speckle.

It is extremely simple to work with \mathbf{A} and \mathbf{M} without assembling them, one only needs to compute and store the two $N \times M$ images $(\mathbf{J}_x, \mathbf{J}_y)$ and use Hadamard product and Laplace function when computing matrix-vector multiplication.

The system is of dimension $2MN$. As said earlier, \mathbf{A} is strongly deficient since its rank is at most MN , a first part of its kernel has the following basis:

$$\text{span} \begin{pmatrix} \mathbf{J}_y \\ -\mathbf{J}_x \end{pmatrix} \subset \ker(\mathbf{A}). \quad (5.7)$$

With a simple computation, it can be shown that the complement part of the matrix can be diagonalized as $(\mathbf{J}_x^2 + \mathbf{J}_y^2)$, and pixels where the gradient is zero (bad speckles) are also associated with zero eigenvalues.

\mathbf{M} is also rank deficient, the dimension of its kernel is 2, a basis of its null space is well known:

$$\ker(\mathbf{M}) = \text{span} \begin{pmatrix} \mathbf{1} & 0 \\ 0 & \mathbf{1} \end{pmatrix}, \quad (5.8)$$

where $\mathbf{1}$ is the vector filled with 1: the kernel of the scalar Laplace operator consists of constant functions.

5.3. Preconditioned Conjugate Gradient and Ritz elements

We use the classical notation: normal font for scalars, boldface lowercase the vectors, boldface uppercase for matrices. For a collection of vectors, we note $\mathbf{X}_m = (\mathbf{x}_0, \dots, \mathbf{x}_{m-1})$ (the index m thus corresponds to the number of columns of the matrix).

Let \mathbf{A} be a $n \times n$ symmetric definite positive matrix and \mathbf{b} be a n vector. We search the solution to the system $\mathbf{A}\mathbf{x} = \mathbf{b}$. We use a conjugate gradient, preconditioned by the symmetric positive semi-definite matrix \mathbf{M} , and augmented by the $n \times n_C$ full-rank matrix \mathbf{C} such that $\ker(\mathbf{M}) \subset \text{Range}(\mathbf{C})$.

For an approximation \mathbf{x}_i , we note $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$ the residual. We introduce the Krylov subspace $\mathcal{K}_i(\mathbf{M}^{-1}\mathbf{A}, \mathbf{C}, \mathbf{M}^{-1}\mathbf{r}_0)$:

$$\mathcal{K}_i(\mathbf{M}^{-1}\mathbf{A}, \mathbf{C}, \mathbf{M}^{-1}\mathbf{r}_0) = \text{span}(\mathbf{M}^{-1}\mathbf{r}_0, \dots, (\mathbf{M}^{-1}\mathbf{A})^{(i-1)}\mathbf{M}^{-1}\mathbf{r}_0) \oplus \text{Range}(\mathbf{C}) \quad (5.9)$$

If no confusion is possible, we will simply write \mathcal{K}_i .

Given an arbitrary initialization \mathbf{x}_{00} , the i_{th} iteration can be defined as:

$$\begin{cases} \text{find} & \mathbf{x}_i \in \mathbf{x}_{00} + \mathcal{K}_i(\mathbf{M}^{-1}\mathbf{A}, \mathbf{C}, \mathbf{M}^{-1}\mathbf{r}_0) \\ \text{such that} & \mathbf{r}_i \perp \mathcal{K}_i(\mathbf{M}^{-1}\mathbf{A}, \mathbf{C}, \mathbf{M}^{-1}\mathbf{r}_0) \end{cases} \quad (5.10)$$

This iteration is achieved by Algorithm 5 where the augmentation is managed by the correction of the initialization (in order to obtain \mathbf{x}_0) and the projector \mathbf{P} on $\ker(\mathbf{C}^T\mathbf{A})$, which together ensure that the residual is always orthogonal to $\text{Range}(\mathbf{C})$.

Algorithm 5 Augmented Conjugate Gradient

\mathbf{x}_{00} and \mathbf{C} given
 $\mathbf{P} = \mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{A}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A}$
 $\mathbf{x}_0 = \mathbf{P}\mathbf{x}_{00} + \mathbf{C}(\mathbf{C}^T\mathbf{A}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{b}$
 $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \mathbf{P}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_{00})$
 $\mathbf{z}_0 = \mathbf{P}\mathbf{M}^{-1}\mathbf{r}_0, \mathbf{w}_0 = \mathbf{z}_0$
 $\gamma_0 = (\mathbf{z}_0^T\mathbf{r}_0)$
for $i = 0, 1, \dots, m$ (convergence) **do**
 $\mathbf{q}_i = \mathbf{A}\mathbf{w}_i$
 $\delta_i = (\mathbf{w}_i^T\mathbf{q}_i), \alpha_i = \delta_i^{-1}\gamma_i$
 $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{w}_i\alpha_i$
 $\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{q}_i\alpha_i$
 $\mathbf{z}_{i+1} = \mathbf{P}\mathbf{M}^{-1}\mathbf{r}_{i+1}$
 $\gamma_{i+1} = (\mathbf{z}_{i+1}^T\mathbf{r}_{i+1})$
 $\beta_i = \gamma_i^{-1}\gamma_{i+1}$
 $\mathbf{w}_{i+1} = \mathbf{z}_{i+1} + \mathbf{w}_i\beta_i$
end for

The algorithm builds two special basis of \mathcal{K}_i : \mathbf{Z}_i is a \mathbf{M} -orthogonal whereas \mathbf{W}_i is \mathbf{A} -orthogonal:

$$\begin{aligned} \mathbf{Z}_i^T\mathbf{M}\mathbf{Z}_i &= \mathbf{Z}_i^T\mathbf{R}_i = \text{diag}(\gamma_j)_{0 \leq j < i} \\ \mathbf{W}_i^T\mathbf{A}\mathbf{W}_i &= \mathbf{W}_i^T\mathbf{Q}_i = \text{diag}(\delta_j)_{0 \leq j < i} \end{aligned} \quad (5.11)$$

It is convenient to introduce the \mathbf{M} -normalized version of the \mathbf{Z}_i basis:

$$\hat{\mathbf{z}}_i = \frac{(-1)^i\mathbf{z}_i}{\sqrt{\gamma_i}} \quad \text{so that } \hat{\mathbf{Z}}_i^T\mathbf{M}\hat{\mathbf{Z}}_i = \mathbf{I} \quad (5.12)$$

This basis is in fact the basis that would have been obtained by an Arnoldi procedure [82], and we have:

$$\begin{aligned} \hat{\mathbf{Z}}_i^T\mathbf{A}\hat{\mathbf{Z}}_i &= \mathbf{T}_i = \text{Tridiag}(\eta_{j-1}, \mu_j, \eta_j) \\ \text{with } \mu_0 &= \frac{1}{\alpha_0}, \quad \mu_j = \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}}, \quad \eta_j = \frac{\sqrt{\beta_j}}{\alpha_j} \end{aligned} \quad (5.13)$$

We can diagonalize $\mathbf{T}_i = \mathbf{\Xi}_i \mathbf{\Theta}_i \mathbf{\Xi}_i^T$ where $\mathbf{\Theta}_i$ is the diagonal matrix of eigenvalues sorted in decreasing order and $\mathbf{\Xi}_i$ the orthonormal matrix of eigenvectors.

The Ritz vectors are $\mathbf{V}_i = \hat{\mathbf{Z}}_i \mathbf{\Xi}_i$, while $\mathbf{\Theta}_i$ are the Ritz values of the system. They satisfy:

$$\mathbf{V}_i^T \mathbf{M} \mathbf{V}_i = \mathbf{I} \quad \text{and} \quad \mathbf{V}_i^T \mathbf{A} \mathbf{V}_i = \mathbf{\Theta}_i. \quad (5.14)$$

More, as the number of iterations i increases, they tend to approximate the generalized eigenvalues of the couple (\mathbf{A}, \mathbf{M}) [83].

5.3.1. Role of the preconditioner

It is often said that the preconditioner should be a good approximation of the inverse, in the sense that the spectrum of $\mathbf{M}^{-1} \mathbf{A}$ should be as concentrated as possible around a non-zero value (1 after scaling if needed). This can be roughly estimated by the condition number of $\mathbf{M}^{-1} \mathbf{A}$, but more sophisticated studies are available [84].

It is important to note the proximity between the conjugate gradient algorithm to solve linear systems and the Lanczos procedure to compute eigenvalues [60], as was made explicit by the Ritz analysis in this section's introduction. It is also useful to see that the higher part of the spectrum is explored in priority due to the repeated power in the construction of Krylov subspace.

In the case of poorly-conditioned systems, the preconditioner can play a regularization role, as was explored in [75]. Schematically, for a direction \mathbf{d} , what matters is the ratio $(\mathbf{d}^T \mathbf{A} \mathbf{d} / \mathbf{d}^T \mathbf{M} \mathbf{d})$. If the preconditioner measures the irregularity of a field, it penalizes the highly oscillating directions and delays their exploration.

Also, the preconditioner can be viewed as providing a physic-based alternative to the simple Euclidean orthogonality. It thus defines "natural" norms which are useful in the analysis of the iterations as discussed in next subsection.

5.3.2. Stopping criteria

Conjugate gradient give valuable pieces of information at no cost, but in specific norms, in the course of the iterations. First, we have error estimators [86]:

$$\begin{aligned} \|\mathbf{r}_i\|_{\mathbf{M}^{-1}}^2 &= \gamma_i \\ \|\mathbf{x}_{i+1} - \mathbf{x}\|_{\mathbf{A}}^2 &= \|\mathbf{x}_i - \mathbf{x}\|_{\mathbf{A}}^2 - \gamma_i^2 \delta_i^{-1} \end{aligned} \quad (5.15)$$

of course the difficulty is that $\|\mathbf{x}_0 - \mathbf{x}\|_{\mathbf{A}}^2$ is unknown. Then we have measurement of the norm of the correction brought by iterations:

$$\begin{aligned} \|\mathbf{x}_{i+1} - \mathbf{x}_0\|_{\mathbf{M}}^2 &= \|\mathbf{x}_i - \mathbf{x}_0\|_{\mathbf{M}}^2 + \alpha_i^2 \|\mathbf{w}_i\|_{\mathbf{M}}^2 + 2\alpha_i (\mathbf{w}_i^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_0)) \\ \text{with } \begin{cases} \|\mathbf{w}_{i+1}\|_{\mathbf{M}}^2 &= \gamma_i + \beta_i^2 \|\mathbf{w}_i\|_{\mathbf{M}}^2, & \|\mathbf{w}_0\|_{\mathbf{M}}^2 &= \gamma_0 \\ (\mathbf{w}_{i+1}^T \mathbf{M} (\mathbf{x}_{i+1} - \mathbf{x}_0)) &= -\beta_i ((\mathbf{w}_i^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_0)) + \alpha_i \|\mathbf{w}_i\|_{\mathbf{M}}^2) \end{cases} \end{aligned} \quad (5.16)$$

Finally, we have an estimator on the preconditioned operator:

$$\begin{aligned} \|\mathbf{T}_0\|_F^2 &= \mu_0^2 \\ \|\mathbf{T}_{i+1}\|_F^2 &= \|\mathbf{T}_i\|_F^2 + \mu_i^2 + \eta_i^2 + \eta_{i-1}^2 \rightarrow \|\mathbf{M}^{-1} \mathbf{A}\|_F^2 \end{aligned} \quad (5.17)$$

where index F stands for the Frobenius norm, $\|\mathbf{M}^{-1}\mathbf{A}\|_F^2$ is the sum of the squares of the generalized eigenvalues of (\mathbf{A}, \mathbf{M}) .

We can then devise costless stopping criteria:

$$\begin{aligned} \|\mathbf{r}_i\|_{\mathbf{M}^{-1}} &< \varepsilon \|\mathbf{r}_0\|_{\mathbf{M}^{-1}} \\ \|\mathbf{r}_i\|_{\mathbf{M}^{-1}} &< \varepsilon \|\mathbf{T}_i\|_F \|\mathbf{x}_i - \mathbf{x}_0\|_{\mathbf{M}} \end{aligned} \quad (5.18)$$

The first one is very classical, but it is risky in the sense that it may be too strict if the initialization or the augmentation were well-chosen ($\|\mathbf{r}_0\|_{\mathbf{M}^{-1}}$ is already small). The second one is inspired from the Scipy implementation of MinRes (with a more adapted choice of norms), we are sorry not to know who to attribute it to; it is interesting in the sense that it balances the reduction of error and the increase of the norm of the solution, which is always a dilemma when solving ill-posed problems. It is often interesting to combine the criteria, add stagnation detection, and to also use safeguards in absolute value in case of too good initialization and augmentation.

5.3.3. A posteriori filtering

In the case of a poorly conditioned system, the reduction of the error can be obtained at the price of an explosion of the norm of the solution. This is well explained by Picard analysis: the phenomenon occurs when the eigenvalues of the operator decrease faster than the contribution of the right-hand side in the associated direction. It can also be visualized on a L-curve, in the positive quarter of a frame of the form $(\|\mathbf{r}_i\|, \|\mathbf{x}_{i+1}\|)$: the curve starts in the bottom right corner (large error, small norm) with a fast decay of the error, and finishes in the top left corner (reduced error, large norm).

As shown earlier, conjugate gradient provides natural norms to evaluate the error and the norm of the solution: $\|\mathbf{x}_i - \mathbf{x}\|_{\mathbf{A}}$ and $\|\mathbf{x}_i - \mathbf{x}_0\|_{\mathbf{M}}$. With this choice of norms, the curve is always oriented toward the upper left corner (at each iteration, the norm of the error decreases and the norm of the solution increases).

Ritz elements offer a convenient way to filter the solution. Assuming m iterations were conducted, we can process the basis \mathbf{V}_m and the values Θ_m . We can post process a reduced solution $\tilde{\mathbf{x}}_i$ in projection on \mathbf{V}_m .

$$\tilde{\mathbf{x}}_i = \mathbf{x}_0 + \sum_{j=1}^i \mathbf{v}_j \frac{(\mathbf{v}_j^T \mathbf{r}_0)}{\theta_j} \quad (5.19)$$

We have:

$$\begin{aligned} \|\tilde{\mathbf{x}}_i - \mathbf{x}\|_{\mathbf{A}}^2 &= \|\tilde{\mathbf{x}}_i - \mathbf{x}_0\|_{\mathbf{A}}^2 - \sum_{j=1}^i \frac{(\mathbf{v}_j^T \mathbf{r}_0)^2}{\theta_j} \\ \|\tilde{\mathbf{x}}_i - \mathbf{x}_0\|_{\mathbf{M}}^2 &= \sum_{j=1}^i \frac{(\mathbf{v}_j^T \mathbf{r}_0)^2}{\theta_j^2} \end{aligned} \quad (5.20)$$

and of course:

$$\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i-1}\|_{\mathbf{A}}^2 = \frac{(\mathbf{v}_i^T \mathbf{r}_0)^2}{\theta_i} \quad \text{and} \quad \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i-1}\|_{\mathbf{M}}^2 = \frac{(\mathbf{v}_i^T \mathbf{r}_0)^2}{\theta_i^2}. \quad (5.21)$$

Since the (θ_j) are sorted in decreasing order, we see that the error of $(\tilde{\mathbf{x}}_i)$ tends to decrease less than its norm tends to increase. The L-curve for $(\tilde{\mathbf{x}}_i)_i$ is then convex and the corner may be easier to define. The slope of the L-curve between the point $i - 1$ and i is $-\theta_i^{-1}$. A possibility is to define the corner as the point which maximizes the variation of slope: $i = \arg \max(\theta_{i+1}^{-1} - \theta_i^{-1})$. Ritz' elements also make it possible to use Picard's theory and stop the construction of $\tilde{\mathbf{x}}_i$ when the contribution $(\mathbf{v}_i^T \mathbf{r}_0)$ starts to decrease less fast than (θ_i) . This criterion has the advantage to take into account the properties of the right-hand side.

5.4. Preconditioning by regularization

In general, preconditioning is important because it can speed up the convergence of iterative solvers. As evoked in Section 5.3, it becomes crucial for ill-posed problem for the physical information it provides and also because the resolution will be stopped "early". Since a too strict convergence criterion can not be attained in a reasonable amount of time, two distinct preconditioners lead to different history of resolution and to qualitatively very different solutions.

As described in Section 5.2, we are interested in Tikhonov-regularized systems of the form:

$$\underbrace{(\mathbf{A} + \lambda \mathbf{M})}_{\mathbf{A}_\lambda} \mathbf{x}_\lambda = \underbrace{\mathbf{b}_\mathbf{A} + \lambda \mathbf{b}_\mathbf{M}}_{\mathbf{b}_\lambda} \quad (5.22)$$

As suggested by the notation, we investigate the effects of using the same operator for the regularization and the preconditioning, in particular when there exists a cheap technique to apply the preconditioner (i.e. \mathbf{M}^{-1}). Conceptually, this idea makes sense as the same physical motivation underlies the choice of the regularization and that of the preconditioner. Moreover, many opportunities are opened by this choice.

If we assume that the system (5.22) was solved for a given λ in m iterations, then we can process the Ritz basis \mathbf{V}_m . The strong point is that the properties of \mathbf{V}_m are independent of λ :

$$\begin{aligned} \mathbf{V}_m^T \mathbf{M} \mathbf{V}_m &= \mathbf{I} \\ \mathbf{V}_m^T \mathbf{A}_\lambda \mathbf{V}_m &= \Theta_{\lambda,m} = \Theta_m + \lambda \mathbf{I}. \end{aligned} \quad (5.23)$$

Remark. λ can be viewed as a shift in the generalized eigenvalues of (\mathbf{A}, \mathbf{M}) . Since λ alters the initial residual and only a limited number of iterations is made, the content of \mathbf{V}_m is influenced by λ , without impairing the orthogonality properties. The arguments of the algorithm P and x_0 of the algorithm 5 are independent of λ .

We can define the Ritz' approximation:

$$\tilde{\mathbf{x}}_{\lambda,i} = \mathbf{x}_0 + \sum_{j=1}^i \mathbf{v}_j \frac{(\mathbf{v}_j^T \mathbf{r}_{\lambda,0})}{\theta_j + \lambda} = \mathbf{x}_0 + \sum_{j=1}^i \mathbf{v}_j \frac{\mathbf{v}_j^T \mathbf{b}_\mathbf{A} + \lambda \mathbf{v}_j^T \mathbf{b}_\mathbf{M}}{\theta_j + \lambda} \quad (5.24)$$

This approximation can be computed at zero cost for any λ , and one could search for the optimal choice of (λ, i) for a compromise between error and norm of the solution.

5.5. Assessments

5.5.1. Application to optical flow reconstruction

The recovery of the optical flow is in fact a nonlinear minimization problem. A pyramidal approach is developed [85], as often in image correlation, in order to provide a meaningful initialization. For simplicity, we focus on the last system to be solved, associated with the full image. Anyhow, the initialization of this system was impacted by the choice of the regularization.

We consider the solution to system ((5.5),(5.6)) with augmented preconditioned conjugate gradient, Algorithm 5. Due to the rectangular shape of the images, there exists an extremely cheap way to solve the Laplacian using Fast Fourier transform or more precisely discrete cosine transform see Appendix 5.6.

In order to apply the inverse, it is necessary to ensure that the orthogonality with respect to constant fields (null-space of \mathbf{M}). For the augmentation, we may use:

$$\mathbf{C} = \begin{pmatrix} \mathbf{1} & 0 \\ 0 & \mathbf{1} \end{pmatrix} \quad \text{or} \quad \mathbf{C} = \begin{pmatrix} \frac{\mathbf{1}}{\sqrt{s_{xx}}} & -\mathbf{1} \frac{s_{xy}s_b}{s_{xx}} \\ 0 & s_b \mathbf{1} \end{pmatrix} \quad \text{with} \quad \begin{cases} s_{xx} = \mathbf{1}^T \mathbf{J}_x^2 \mathbf{1} \\ s_{xy} = \mathbf{1}^T \mathbf{J}_x \mathbf{J}_y \mathbf{1} \\ s_{yy} = \mathbf{1}^T \mathbf{J}_y^2 \mathbf{1} \\ s_b = 1/\sqrt{s_{yy} - s_{xy}^2/s_{xx}} \end{cases} \quad (5.25)$$

the second expression has the advantage to make the matrix ($\mathbf{C}^T \mathbf{A} \mathbf{C}$) identity.

The proposed test case is a holed composite plate in traction, with a 45° crack to be identified at the bottom of the hole. The speckle in the initial configuration is shown in Figure 5.1. To quantify the bad conditioning, the pixel-wise gradient norm is between 10^{-6} and 10^2 .

5.5.2. Quality of the preconditioner

We first wish to verify that preconditioning by regularization actually leads to better enforcement of the regularity. In Table 5.1, we can qualitatively compare the classical approach of preconditioning by the diagonal of the operator and the proposed preconditioning by regularization. The increased regularity is particularly visible for low weight λ and low precision ε of the linear solver.

Preconditioning by the regularization operator thus makes it possible to make meaningful computations with low weight in the regularization and to solve with less precision, hence with less iterations. Nevertheless, one has to mention that our preconditioner is computationally more expensive per iteration than the diagonal one.

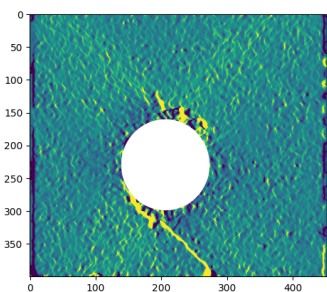
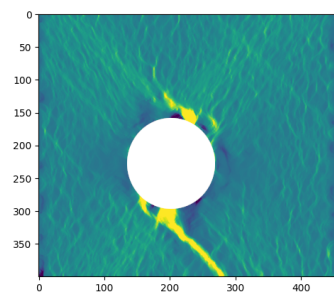
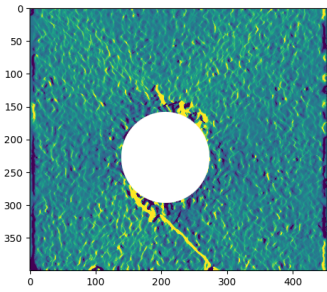
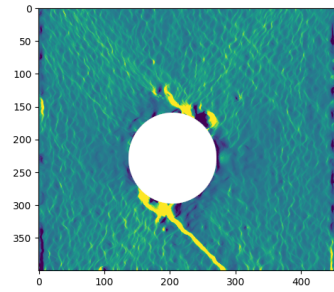
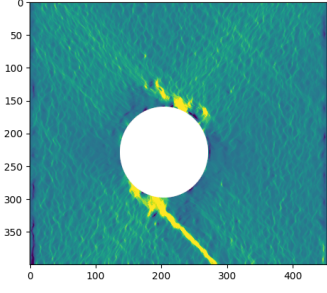
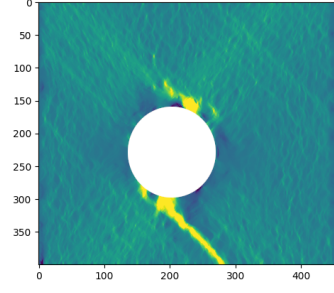
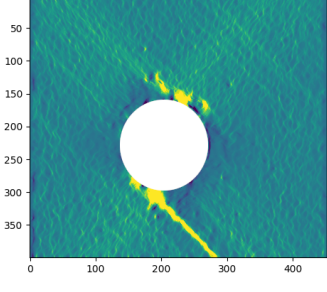
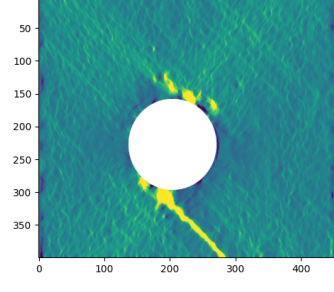
λ	ε	Diagonal Prec.	Regularization Prec.
Low	Low		
Low	High		
High	Low		
High	High		

Table 5.1: ε_{xx} strain field (range = mean value \pm 3 st.dev.). Comparison of the effect of preconditioning by diagonal (simple approach) vs by regularization, for different weights $\lambda \in \{1, 1000\}$ and linear solver precision $\varepsilon \in \{10^{-2}, 10^{-3}\}$.

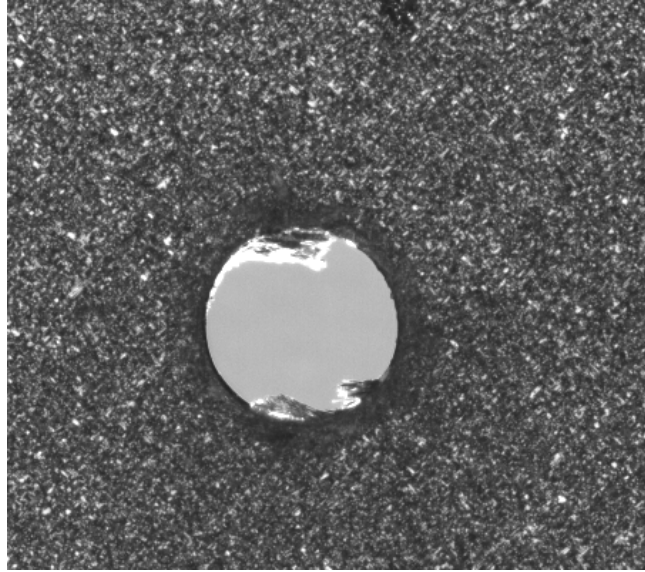


Figure 5.1: Speckle of the test specimen.

5.5.3. Ritz filtering

We analyze the solving process for high ($\lambda = 1000$) and low ($\lambda = 10$) levels of regularization. We use the second stopping criterion of Equation (5.18) with $\varepsilon = 10^{-5}$, which corresponds to a rather high degree of convergence. The identified strain field are given in Figure 5.4.

We analyze the convergence in terms of compromise between the decrease of the error and the increase of the norm of the gradient of the solution which stems from the oscillations in the identified fields. Figure 5.2 presents two L-curves associated with high ($\lambda = 1000$) and low ($\lambda = 10$) regularization. We use the natural CG-norms, please note that the position of the 0-abscissa is conventional because $\|x_0 - x\|_{\mathbf{A}}$ is unknown. The L-curves of the CG iterations (dotted lines) have similar shapes, like pieces of hyperbola, but due to the difference of magnitude, different scales had to be used: the error decreases four times less when the high regularization is used, and the norm of the solution remains fifty times smaller.

In order to better understand the convergence, we conduct a Ritz analysis and the a posteriori filtering of the solution. We present the L-curves of (5.24) in terms of modes included in the reconstruction. We show the curves in terms of full error and only taking into account the image error ($\mathbf{v}_i^T \mathbf{r}_{\mathbf{A}}$) — they are almost overlaid on each other, a slight discrepancy only appears for high regularization. The shape of the Ritz L-curves corresponds to most the modes (the highest) only slightly decreasing the error and almost not changing the norm, only the last 5 modes are associated with significant decrease of the error (but of course at the cost of much increased solution norm). More or less, if a corner was to be selected it would correspond to just suppressing the contribution of the last mode.

In order to better understand this behavior, we first analyzed the convergence of the Ritz values by comparing the spectrum obtained at the last iteration with the one obtained just one iteration before (Ritz(N-1) full error in Fig. 5.2), like was done in [87] in the case of a well posed problem. It appears that the largest Ritz values were quite well approximated and only

the lowest part of the spectrum evolves (remember the Ritz values correspond to the inverse of the slope of the segments in the L-curve). In other words, even though the last iterations seem not to modify the solution much (accumulation of the dots in the upper left part on the CG L-curves), they play an important role in terms of estimation of the lower part of the spectrum, without adding lots of small eigenvalues.

To support this analysis, we conducted a Picard's study on Figure 5.3 which shows the distribution of the Ritz values (θ_i) as well as the decomposition of the right-hand side on the eigen-space $\mathbf{v}_i^T \mathbf{b}_A$ and $\lambda \mathbf{v}_i^T \mathbf{b}_M$. It is worth recalling that low and high regularized systems have the same spectrum, except that it is more sampled for the low regularization which requires two times more iterations to converge. The Ritz values are slowly decreasing and only the last 10% really decay, the low regularization is not associated with an overpopulation of the lowest part of the spectrum. What stands out is the fact that the right-hand side contributes almost equally on all modes (at least it does not decrease for larger Ritz values). Picard's theory thus suggests that we should stop the reconstruction when the Ritz values start to decay. Unfortunately, it appears that the crack is mostly represented in this part of the spectrum which makes it impossible to eliminate.

By the way, Figure 5.3 permits to compare the smallest Ritz value θ_{\min} with the regularization parameter λ . The case that we called "low regularization" corresponds to λ being negligible with respect to θ_{\min} , and thus only marginally modifying the active Ritz spectrum. On the contrary, the high regularization corresponds to a $\lambda > \theta_{\min}$ which means that the lower part of the spectrum is flattened.

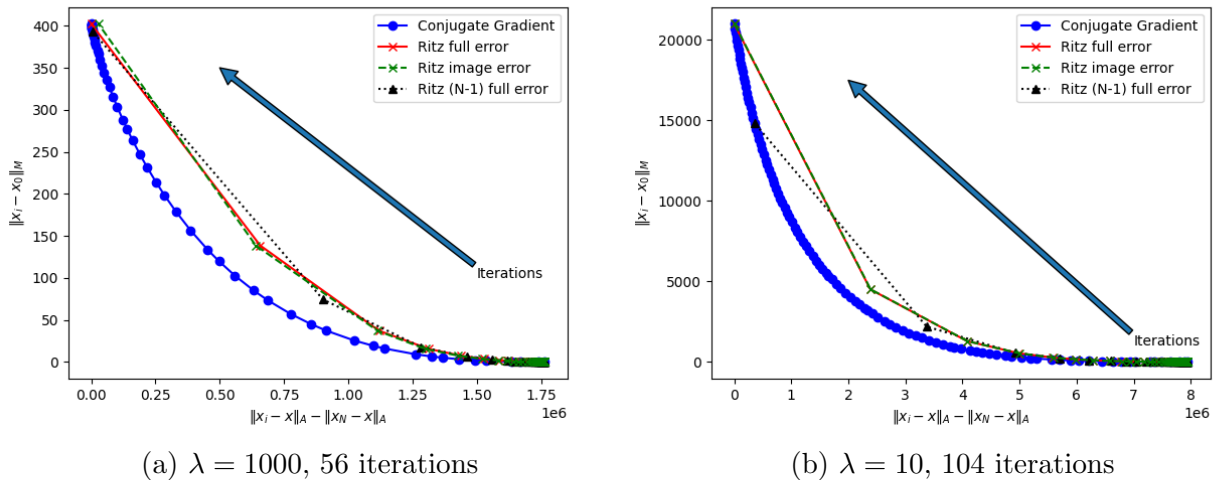
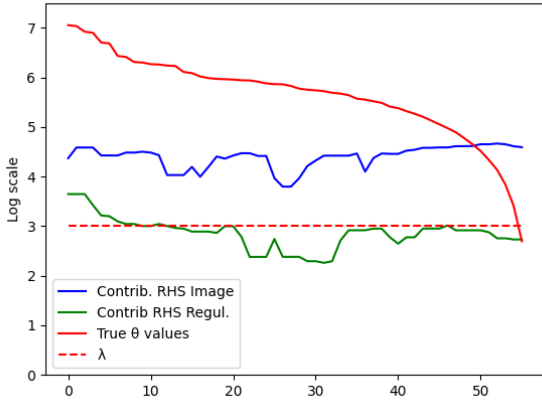
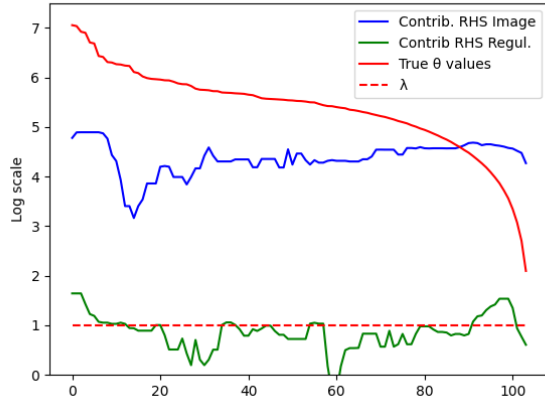


Figure 5.2: L-curves ($\varepsilon = 10^{-5}$) compared with Ritz post-treatment, for different regularization intensity λ . \triangleleft Different scales on each plot.

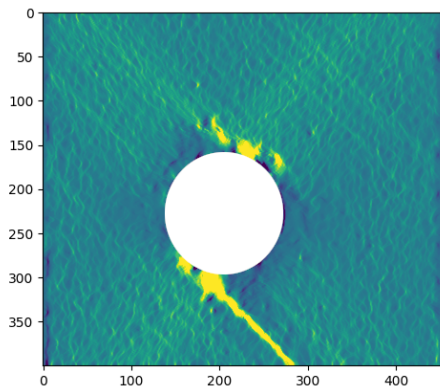


(a) $\lambda = 1000$

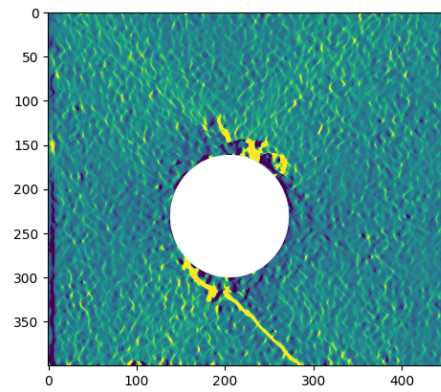


(b) $\lambda = 10$

Figure 5.3: Spectral analysis of the system for $\varepsilon = 10^{-5}$ and different regularization intensity λ . A 5-width median filter was used on the contribution curves.



(a) $\lambda = 1000$



(b) $\lambda = 10$

Figure 5.4: Identified ε_{xx} field, for different regularization intensity λ , with $\varepsilon = 10^{-5}$.

5.5.4. Subspace recycling

Even though it appears that Ritz filtering is difficult to apply to the studied system, we can still benefit from Ritz vectors to accelerate the solution. As a sequence of linear systems with identical matrix has to be solved, it is natural to augment the system with the previously generated Ritz vectors by concatenating $\mathbf{C} \leftarrow (\mathbf{C} \ \mathbf{V})$. Indeed, augmentation comes with optimized block operations that make augmenting by one vector much cheaper than one iteration.

Moreover, Ritz vectors possess two advantages. First the product $\mathbf{A}\mathbf{V}_i$ which is required during augmentation can be obtained at low computational cost using the formula: $\mathbf{A}\hat{\mathbf{z}}_{i+1} = (-1)^{i+1}(\mathbf{q}_{i+1} - \beta_i \mathbf{q}_i) / \sqrt{\gamma_{i+1}}$ and $\mathbf{A}\mathbf{V}_i = \mathbf{A}\hat{\mathbf{Z}}_i \mathbf{\Xi}_i$. Second using normalization: $\mathbf{V}_i \leftarrow \mathbf{V}_i \mathbf{\Theta}_i^{-1/2}$, we have $\mathbf{V}_i^T \mathbf{A}\mathbf{V}_i = \mathbf{I}$.

Aug.	0	10	20	30	40	50	60	70	max (77)
Iter.	77	64	57	49	44	41	40	40	38
Time (s)	11.7	10.1	8.3	7.1	6.6	6.3	6.2	6.3	6.6

Table 5.2: Performance of recycling

Table 5.2 illustrates the performance of recycling for the nonlinear system to be solved on the full image at the end of identification. The first linear system, only augmented by the kernel of the preconditioner is first solved in 77 iterations. Then a certain portion of the Ritz vectors is used to augment the next 8 linear systems (same matrix, different right-hand sides). As the augmentation results in an excellent initialization, we use a criterion in terms of absolute value of $\|\mathbf{r}_i\|_{\mathbf{M}}$ to halt the iterations because other comparison as given in Equation (5.18) might use an unfair reference. We measure the performance in terms of gain in iterations, and mainly gain in computational time (measures are conducted on an upper mid-range laptop with Nvidia RTX A2000 graphic card). The gain in terms of iterations is moderate, with best obtained for small augmentation space (at most 1.3 iterations per vector, for 10 vectors). In terms of time, the optimal is obtained for augmentation space of 80%-90% of available vectors, with a global CPU time divided by almost 2 (this time includes all the extra cost associated with computing and using Ritz vectors). This size of subspace agrees with what we observed on the stability of the largest Ritz values in the L-curves plots.

5.5.5. Tuning of λ

It is often hard to automatize the selection of the regularization intensity λ . Picard's plots like in Figure 5.3 permit to put λ in relation with the spectrum of the preconditioned operator and thus to understand the effect of the regularization in terms of flattened spectrum. Still, the final judge is often the expert's impression of a strain map, and it is convenient to compute maps associated with several (λ_i) at low cost.

Formula (5.24) makes it possible, after the solution to one linear system for a given λ_0 , to post-process the solution for any λ_i at the simple cost of computing the associated right-hand side $\mathbf{b}_{\mathbf{A},i}$ (which depends on the history of the nonlinear solution for λ_i), and basic linear algebra operations.

Figure 5.5 presents the solution deduced for $\lambda = 1$ from initial computations with $\lambda \in \{1, 10, 100, 1000\}$ and $\varepsilon = 10^{-4}$. Again, a median filter was applied after the Ritz reconstruction.

It seems that the Ritz vectors make it possible to postprocess a solution with a λ divided by up to 1000. The reconstructed strain field appears to be much less smooth than the original computation (with high λ) while less noisy than the direct low-regularization computation with $\lambda = 1$. Amazingly, deconvoluting by a greater factor (10000) leads to results quite similar to what could be obtained with a quasi- L^1 norm, see the results with the Charbonnier's norm of [85].

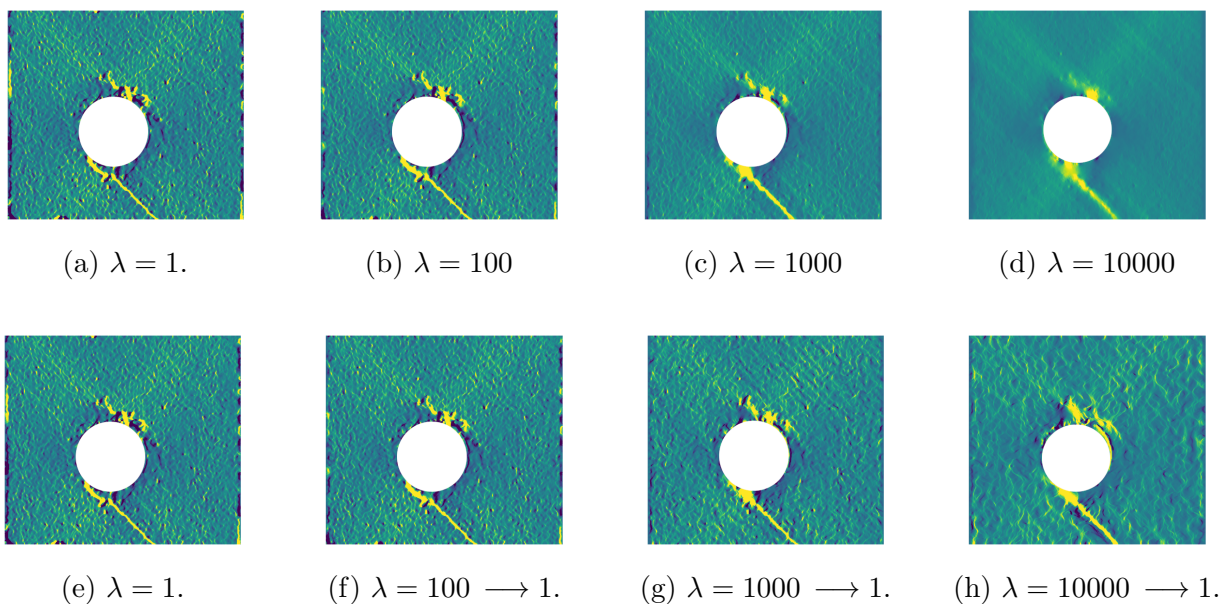


Figure 5.5: Costless postprocessing for different (λ_i): top, initial computation with $\lambda \in \{1, 100, 1000, 10000\}$; bottom, solution deduced for $\lambda = 1$. ε_{xx} strain field.

5.6. Conclusion

In this paper, we have studied how preconditioning and Tikhonov regularization could be efficiently combined in an augmented preconditioned conjugate gradient. We have shown that this association makes sense from a physical point of view and it made it possible to combine filtering, recycling of subspaces, and postprocessing of all regularized solutions at zero cost. This gives a favorable framework to apply criteria like the L-curve or Picard's analysis. We showed how all this can be implemented at very cost while providing meaning information to monitor the solution.

The solver was applied to a problem of optical flow reconstruction which introduced the extra difficulty of nonlinearity. Satisfying results were obtained on actual measurements from digital image correlation of a mechanical test.

Appendix: Inverse of Laplacian on a rectangle with Neumann boundary condition

It is well known that plane waves $x \mapsto e^{i\omega \cdot x}$, with $\omega \in \mathbb{R}^2$, form a set of eigenfunctions for the Laplace operator in \mathbb{R}^2 with eigenvalues $-\|\omega\|^2$ (using the Euclidean norm). This can be equivalently formulated by saying that the Fourier transform diagonalize the Laplacian. Hence, the powerful solution technique (in that case ω is the variable in the Fourier domain):

$$\begin{aligned} \Delta u + f &= 0 && \text{in } \mathbb{R}^2 \\ u &= \mathcal{F}^{-1} \left(\frac{\mathcal{F}(f)}{\|\omega\|^2} \right) \end{aligned} \quad (5.26)$$

What is remarkable is that the eigenvectors are preserved by discretization. For instance, if we consider the classical 5-point stencil on a unit grid:

$$(\Delta_h u)(k, l) = u(k+1, l) + u(k-1, l) + u(k, l+1) + u(k, l-1) - 4u(k, l) \quad (5.27)$$

and one can check that

$$(\Delta_h e^{i\omega \cdot x})(k, l) = e^{i(k\omega_1 + l\omega_2)} (e^{i\omega_1} + e^{-i\omega_1} + e^{i\omega_2} + e^{-i\omega_2} - 4) \quad (5.28)$$

Now, considering a rectangular domain, the boundedness of the domain and the boundary conditions lead to selecting only certain eigenvalues and eigenvectors are made out of a good combination of plane waves. Consider the unit square $[0, 1]^2$, the eigenvalues $\lambda_{n,m}$ and eigenvectors $v_{n,m}$ of the Laplacian with (homogeneous) Neumann boundary conditions are given by:

$$\begin{aligned} v_{n,m}(k, l) &= \cos\left(\frac{ml\pi}{M}\right) \cos\left(\frac{nk\pi}{N}\right) \\ \lambda_{n,m} &= 2 \left(1 - \cos\left(\frac{n\pi}{N}\right)\right) + 2 \left(1 - \cos\left(\frac{m\pi}{M}\right)\right) \end{aligned} \quad (5.29)$$

As eigenvectors are cosine functions, the specialization of the Fourier transform to this case takes the name of discrete cosine transform (DCT).

One just needs to take some care of the $\lambda_{0,0} = 0$ -eigenvalue, associated with the constant eigenvector. The classical solution is to work on functions with 0 mean value and nullify the constant term in the transformed function.

We give the python code for the inverse of the discrete Laplacian on a rectangle with Neumann boundary conditions, which can be directly invoked by the function

`from scipy.ndimage import laplace` with default arguments (that is to say `border='reflect'`).

```
import numpy as np
from scipy.fftpack import dctn, idctn

# Prepare transform of Laplacian for (N,M) images
mw_x = 2*(np.cos(np.pi*np.arange(0,N)/N)-1)
mw_y = 2*(np.cos(np.pi*np.arange(0,M)/M)-1)
```

```
[MWX, MWY] = np.meshgrid(mwx,mwy,indexing='ij')
MW = MWX+MWY
MW[0,0]=1.
iMW = 1./MW
iMW[0,0]=0

def SolveLaplaceNeumann(U,iMW): #U must have zero mean value
    dctU = dctn(U,norm='ortho')
    uhat = dctU*iMW
    u = idctn(uhat,norm='ortho')
    return(u)
```


6. General Conclusion

Through this work, we have presented and discussed several commonly used optical methods for calculating kinematic fields, specifically displacements and strain, in solid mechanics. These methods include Digital Image or Volume Correlation (DIC or DVC) and optical flow. The first method involves describing motion, whether through its local or global approach, by using independent sub-images referred to as zones of interest (ZOI) in the case of local approaches, and interdependent sub-images referred to as Regions of Interest (ROI) in the case of the global approach. The second technique, optical flow, is based on the same assumption of graylevel conservation, that the graylevel of each pixel in the integration domain remains invariant in an image sequence.

In contrast to image correlation algorithms, optical flow algorithms are capable of describing kinematic fields at each pixel, but they require an extra term that contains a regularization constraint. Note that such a regularization is also often added to the DIC problem formulation as well.

The global approaches of both methods have the advantage of providing continuous displacement fields over large areas, which is their main benefit. However, they require solving a very large linear system, the dimensions of which depend on the size of the images and the number of degrees of freedom sought. Regardless of the chosen method, users always have an interest in obtaining an accurate motion description within a reasonable time, especially when dealing with real-time applications. Parallelism of codes can help reduce their execution time.

The problems to be solved often involve very high dimensions, and the most suitable solvers for such problems are referred to as Krylov solvers, which are a specific type of iterative methods which search for the solution in a lower-dimensional space compared to the dimensions of the matrix to be solved and then return to the initial space via a transformation matrix (examples: GMRES for general matrices and the Conjugate Gradient method for symmetric positive-definite matrices). In most cases, Krylov-type solvers offer the advantage of being implemented in a matrix-free version, meaning they don't require storing the system matrix. They only need a well-defined matrix-vector product operation, making them memory-optimized. The use of this tool is essential, whatever the methods or the metrics employed.

In order to quantify the difference between two images, the quadratic norm is often used. This work and the literature have shown that this norm has the disadvantage of diffusing the discontinuities that are often the subject of mechanical studies. On the other hand, in the optical flow community, other types of error functions have been proposed, known as robust functions. This designation comes from the fact that they are robust to outliers and discontinuities. Two examples have been studied in this work: The Charbonnier and Lorentzian functions. Both give satisfactory results when their parameter, which allows the transition between the behaviors of the two norms L^1 and L^2 is well adjusted, but it has been found that the Charbonnier function, which is seen as a variant of the L^1 norm as used by the optical flow community, can

create spurious discontinuities in the strain field.

The drawback of these metrics is that they require additional effort to minimize them, which is due either to their non-convexity or to the non-linearity of their derivative. It was found that the Tikhonov regularisation parameter plays a very important role, since it control the weight given to the regularity part of the solution. Its behaviour was found to be similar to that of window size in DIC. We proposed a local regularization that contains different values, larger outside the crack and smaller in its vicinity. We were able to detect cracks without diffusion, while achieving very homogeneous strain similar to those obtained with very large λ .

It has been mentioned that the GNC minimization algorithm is crucial not only for the Lorentzian due to its non-convex nature but also for the Charbonnier function because of its non-linearity. GNC is not useful in the case of the quadratic norm since it is convex and its derivative is linear. We then set out to provide an algorithm for calculating fields based on the Horn and Schunk method, while introducing more sophisticated techniques for improving estimation. For this, we used the basic method described in the Sun's [4] paper. To simplify the understanding and modification of the "GCPU_Opticalflow" code by the users who are not familiar with low-level programming, we chose to use Python as programming language. The provided software is GPU-accelerated, and its simplicity lies in the fact that it uses CuPy to target the GPU instead of other libraries like PyCUDA, whose kernel understanding requires a mastery of C/C++. GCPU_Opticalflow is also matrix-free, which means it doesn't consume a lot of memory resources to operate. This is possible due to the nature of the Krylov solvers used, which allow an implementation that avoids unnecessary storage and relies on functions to compute matrix-vector products on demand. In our case, this was achieved through operators based on convolution. Finally, it's also multiplatform, since it only works on CPU if hardware or software resources are not met. Indeed, in the case where one or more requirements are not met, the alternatives of GPU-accelerated libraries will be used (e.g. NumPy instead of CuPy) and the solution will be calculated at a slower pace.

We saw in the last chapter that it is possible to use the regularization term to precondition the system to be solved using conjugate gradient methods. We have seen that the use of Ritz modes in the post-processing stages to filter and recycle the digital information not only serves to analyse the spectral properties of the problem in order to perform a Picard's type analysis, but can also be used to generate the kinematic fields resulting from different regularization weights parameters at zero cost.

In terms of perspectives, it is important to note that first of all the local regularization mask used was generated manually, which is not always possible to do. Our first idea is to take advantage of the optimization algorithm's iterative scheme and generate a mask automatically from the results of the first GNC iteration, where the quadratic formulation is minimized. Another approach to achieve this automation is to use artificial intelligence algorithms, such as deep learning, to identify the areas where parameters need to be changed in order to detect local mechanical phenomena. In the medium term, we will be interested in discovering other robust metrics, as well as a thorough comprehension of their parameters. We would also aim to explore other less complex optimisation approaches other than the GNC used in this work in order to improve the calculation time so that this type of algorithm can be easily extended to the case of tomographic images, which is our long-term objective. Extending this type of

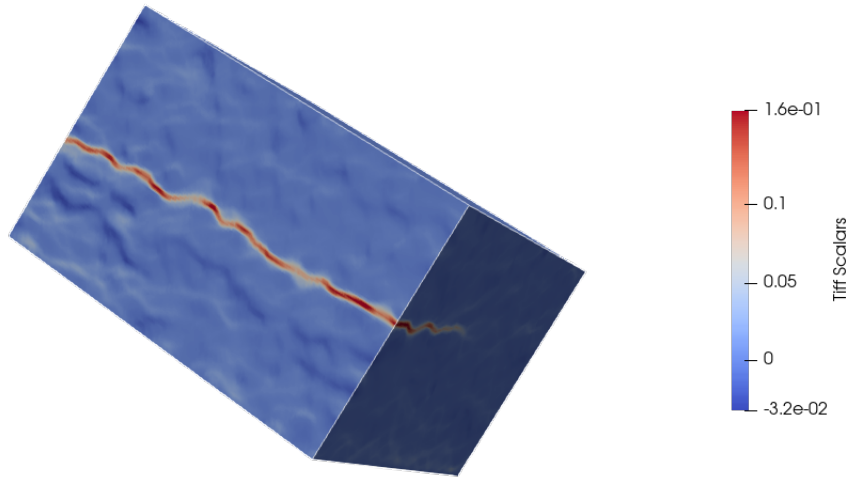
algorithm to three-dimensional images generated using modern sensors with a resolution that can reach up to 4000 voxels per direction, will require massive parallelism of the calculation code on several levels, as well as the proposal of new solvers and interpolation operations to optimize the communications during the calculation.

An initial work aimed at achieving this extension and estimating three-dimensional pixel-wise, kinematic fields was carried out using a Horn and Schunk algorithm. The code implementation consistently adhered to the same logic while utilizing CuPy to target the GPU and using a matrix-free version of Krylov solvers, in order to maintain the code's simplicity. This time, however, the system to be solved (6.1) at a given iteration k , for a sequence of images of size $(N \times M \times P)$ has a dimension of $3NMP$.

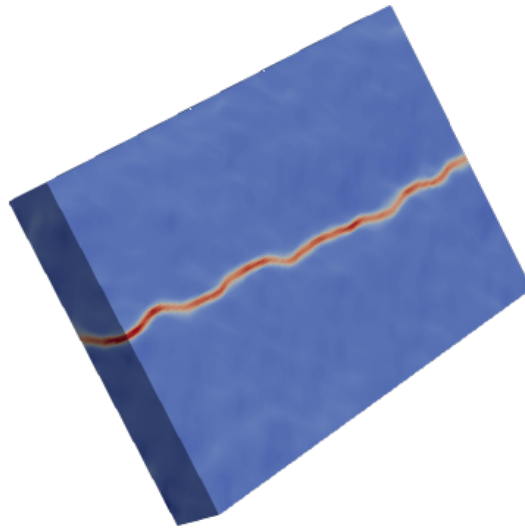
$$\begin{pmatrix} I_x^2 + \lambda\Delta & I_x I_y & I_x I_z \\ I_x I_y & I_y^2 + \lambda\Delta & I_y I_z \\ I_x I_z & I_y I_z & I_z^2 + \lambda\Delta \end{pmatrix} \begin{pmatrix} du^{k+1} \\ dv^{k+1} \\ dw^{k+1} \end{pmatrix} = - \begin{pmatrix} I_x I_t + \lambda\Delta u^k \\ I_y I_t + \lambda\Delta v^k \\ I_z I_t + \lambda\Delta w^k \end{pmatrix}, \quad (6.1)$$

I_z , dw^{k+1} , and w^k represent, respectively, the gradient, the increment, and the displacement at iteration k for the third space component.

This model was tested on images of size $150 \times 150 \times 100$ voxels of a sample of nodular graphite cast iron [101] with a crack. The obtained results are promising, since we were able to detect the crack in the specimen, as shown in Fig. 6.1. We can observe a kind of noise at the image edge, and we believe this issue can be explained by the boundary conditions used for the interpolation operation or for the Laplacian operator. Hence, the importance of inspecting and addressing this question before proceeding to parallelism.



(a) Front view



(b) Slice view

Figure 6.1: ε_{zz} strain field, $\lambda = 10^3$

Bibliography

- [1] B.K.P. Horn and G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981, V. 17, pages 185–203.
- [2] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, 1981, pages 121–130.
- [3] J. Réthoré, F. Hild, S. Roux, Shear-band capturing using a multiscale extended digital image correlation technique, *Computer Methods in Applied Mechanics and Engineering*, V. 196, Issues 49–52, 2007.
- [4] D. Sun, S. Roth and M. J. Black. Secrets of optical flow estimation and their principles. *Computer society conference on computer vision and pattern recognition. IEEE*, 2010. p. 2432-2439.
- [5] S. Baker, D. Scharstein, J. Lewis, S. Roth, MJ. Black and R. Szeliski. A database and evaluation methodology for optical flow. In *ICCV*, 2007
- [6] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *CVIU*, 63:75–104, 1996.
- [7] A. Wedel, T. Pock, C. Zach, D. Cremers, H. Bischof. An improved algorithm for TV-L1 optical flow. *Dagstuhl Motion Workshop*, 2008.
- [8] LI. Rudin, S. Osher, E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992
- [9] D. Sun, S. Roth, J. Lewis, and M. J. Black. Learning optical flow. *ECCV*, vol. 3, p. 83–97 (2008)
- [10] A. Blake and A. Zisserman. *Visual Reconstruction*. The MIT Press, Cambridge, Massachusetts, 1987.
- [11] Y. Li and S. Osher. A new median formula with applications to PDE based denoising. *Commun. Math. Sci.*, 7(3):741–753, 2009
- [12] J. Neggers, B. Blaysat, JPM. Hoefnagels, MGD. Geers. On image gradients in digital image correlation, 105 (4) 243–260.
- [13] J. Sanchez, E. Meinhardt-Llopis, G. Facciolo. TV-L1 Optical Flow Estimation. In *Image Processing On Line*, 2013.

- [14] T. Brox, A. Bruhn, N. Papenberg, Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In European Conference on Computer Vision (ECCV), V. 3024 of Lecture Notes in Computer Science, p. 25–36, May 2004. http://dx.doi.org/10.1007/978-3-540-24673-2_3.
- [15] P. Charbonnier, L. Blanc-Feraud, G. Aubert and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging, Proc. IEEE Int. Conf. Image Process., p. 168-172, 1994.
- [16] L. Alvarez, J. Monreal, J. Sánchez. A PDE model for computing the Optical Flow. XVI Congreso de Ecuaciones Diferenciales y Aplicaciones 1999.
- [17] JY. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm. Intel Corporation 2000.
- [18] M.A. Sutton, W.J. Wolters, W.H. Peters, W.F. Ranson, S.R. McNeill. Determination of displacements using an improved digital correlation method. Image Vision Comput. 1983, 1, 133–139.
- [19] F. Hild, S. Roux. Comparison of local and global approaches to digital image correlation, Experimental Mechanics, V. 52,2012.
- [20] P. Cheng, M. Sutton, H. Schreier, S. McNeill. Full-field Speckle Pattern Image Correlation with B-Spline Deformation Function. Experimental mechanics, V. 42, 2002
- [21] JC. Passieux, R. Bouclier. Classic and Inverse Compositional Gauss-Newton in Global DIC. International Journal for Numerical Methods in Engineering, In press, 119 (6), pp.453-468. [ff10.1002/nme.6057](https://doi.org/10.1002/nme.6057). [ffhal-02059472f](https://doi.org/10.1002/nme.6057)
- [22] JC. Passieux, JN. Périé. High resolution digital image correlation using proper generalized decomposition: PGD-DIC. International Journal for Numerical Methods in Engineering, 2012, 92 (6), pp.531-550. [ff10.1002/nme.4349](https://doi.org/10.1002/nme.4349). [ffhal-00708541](https://doi.org/10.1002/nme.4349)
- [23] J. Réthoré, F. Hild, S. Roux. Extended digital image correlation with crack shape optimization. International Journal for Numerical Methods in Engineering, 2008, 73 (2), pp.248-272.
- [24] F. Hild, A. Bouterf, P. Forquin, S. Roux. On the Use of Digital Image Correlation for the Analysis of the Dynamic Behavior of Materials. The Micro-World Observed by Ultra High-Speed Cameras, pp.185-206, 2018, 978-3-319-61490-8. [ffhal-01674588f](https://doi.org/10.1007/978-3-319-61490-8)
- [25] JC. Passieux. [jcpassieux/pyxel](https://github.com/jcpassieux/pyxel), original-date: 2018-10-06T17:55:07Z. <https://github.com/jcpassieux/pyxel>.
- [26] S. Keil. Technology and Practical Use of Strain Gages With Particular Consideration of Stress Analysis Using Strain Gages: With Particular Consideration of Stress Analysis Using Strain Gages. John Wiley & Sons, Ltd. ISBN 978-3-433-60666-7.

- [27] G. Besnard, F. Hild, S. Roux. "Finite-element" displacement fields analysis from digital images : application to Portevin-Le Châtelier bands. *Experimental Mechanics*, 2006, 46, pp.789-804. [ff10.1007/s11340-006-9824-8](https://doi.org/10.1007/s11340-006-9824-8). [ffhal-00124513f](https://doi.org/10.1007/s11340-006-9824-8)
- [28] F. Hild, B. Raka, M. Baudequin, S. Roux, Florence Cantelaube. Multi-Scale Displacement Field Measurements of Compressed Mineral Wool Samples by Digital Image Correlation. *Applied optics*, 2002, IP 41, pp.6815-6828. [ff10.1364/AO.41.006815](https://doi.org/10.1364/AO.41.006815). [ffhal-00002901f](https://doi.org/10.1364/AO.41.006815)
- [29] B. Bay, T. Smith, D. Fyhrie, M. Saad. Digital volume correlation: Three-dimensional strain mapping using X-ray tomography. *Experimental Mechanics*. 1999, V 39, pp-217:226.
- [30] AN. Tikhonov, VY. Arsenin, *Solutions of ill-posed problems*, (1977), Wiley.
- [31] F. Hild, S. Roux. Digital Image Correlation: from Displacement Measurement to Identification of Elastic Properties – a Review. *Strain*, 2006, V. 42, pp. 69-80.
- [32] J. Réthoré, T. Elguedj, P. Simon, M. Coret. On the Use of NURBS Functions for Displacement Derivatives Measurement by Digital Image Correlation. *Experimental Mechanics*, 2009, V. 5, pp.1099-1116.
- [33] M. Bacconnais. Méthode intégrée de corrélation d'images et de corrélation d'images virtuelles. *Traitement des images [eess.IV]*. École centrale de Nantes, 2019. Français. (NNT : 2019ECDN0069). (tel-02950836). PhD thesis(in french).
- [34] G. Golantonio. Stéréo-corrélation d'images numériques éléments finis : de l'étalonnage à l'identification. University of Toulouse, 2020. PhD thesis(in french).
- [35] JE. Pierré, JC. Passieux, JN. Périé, F. Bugarin, L. Robert. Unstructured finite element-based digital image correlation with enhanced management of quadrature and lens distortions. *Optics and Lasers in Engineering* 77 2016, pp. 44–53.
- [36] International Digital Image Correlation Society, Jones, E.M.C. and Iadicola, M.A. (Eds.) (2018). *A Good Practices Guide for Digital Image Correlation*. DOI: 10.32720/idics/gpg.ed1
- [37] HW. Schreier, MA. Sutton. Systematic Errors in Digital Image Correlation Due to Under-matched Subset Shape Functions. *Experimental Mechanics* 2002. v. 42. pp 303–309.
- [38] G. Hallo, C. Lacombe, R. Parreault, N. Roquin, T. Donval, L. Lamaignère, J. Neauport and F. Hild. Sub-pixel detection of laser-induced damage and its growth on fused silica optics using registration residuals. *Optics Express* 2021. v. 29.
- [39] Paige, C. C., M. A. Saunders. Solution of Sparse Indefinite Systems of Linear Equations. *SIAM J. Numer. Anal.*, Vol.12, 1975, pp. 617-629.
- [40] G.H. Golub, C.F. van Loan, *Matrix computations*, North Oxford Acad, 1983.
- [41] A. Greenbaum: *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, PA, USA, 1997. 17.

- [42] Y. Saad. Practical use of polynomial preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.*, 6(4),865–881, 1985.
- [43] Y. Saad, M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving non symmetric linear systems. *Siam Journal on Scientific and Statistical Computing* 1986, v. 7.
- [44] M.R. Hestenes, E. Stiefel. *Methods of Conjugate Gradients for Solving Linear Systems.* *Journal of Research of the National Bureau of Standards* 1952, v. 49, pp. 409–436.
- [45] R. Fletcher. *Conjugate gradient methods for indefinite systems. Numerical analysis.* Springer 1976, pp. 73–89.
- [46] W. Kahan, B. Parlett. The Fully-Orthogonal Arnoldi Method for Generalized Eigenproblems. *SIAM Journal on Scientific and Statistical Computing* 1980, v. 1, pp. 155–173.
- [47] Akiba and Takuya and Fukuda and Kota. CuPy: A NumPy-compatible Library for NVIDIA CUDA. *IPSN Transactions on Programming* 2016, V 9, p 4–7.
- [48] Link to Matlab implementation by D. Sun. <http://www.cs.brown.edu/people/dqsun/>
- [49] V. Couty, J-F. Witz, C. Martel, F. Bari, A. Weisrock. CRAPPY: Command and Real-Time Acquisition in Parallelized Python, a Python module for experimental setups. *SoftwareX*, Vol. 16, 2021, pp.100848.
- [50] D. Turner , P. Crozier, P. Reu. Digital image correlation engine (DICE). 2015, Sandia National Laboratory: Albuquerque, NM, USA.
- [51] P. P. Das, M. R. P. Elenchezian, V. Vadlamudi, K. Reifsnider, R. Raihan. RealPi2dDIC: A Low-cost and open-source approach to in situ 2D Digital Image Correlation (DIC) applications. *SoftwareX*, 2021, Vol. 13, pp.100645.
- [52] D. Turner , P. Crozier, P. Reu. Digital image correlation engine (DICE). 2015, Sandia National Laboratory: Albuquerque, NM, USA.
- [53] J. Blaber , B. Adair, A. Antoniou. Ncorr: open-source 2D digital image correlation matlab software. *Exp Mech* 2015;55(6):1105-22.
- [54] V. Belloni, R. Ravanelli, A. Nascetti, M. Di Rita, D. Mattei, M. Crespi. Py2DIC: A new free and open source software for displacement and strain measurements in the field of experimental mechanics. *Sensors* 2019;19(18). <http://dx.doi.org/10.3390/rs12182906>.
- [55] V. Belloni V, R. Ravanelli, A. Nascetti, M. Di Rita, D. Mattei, M. Crespi. Digital image correlation from commercial to fos software: A mature technique for full-field displacement measurements. *Int Arch Photogr Remote Sens Spatial Inf Sci* 2018;42(2).
- [56] R. Ravanelli, A. Nascetti, M. Di Rita, V. Belloni, D. Mattei, N. Nistico, M. Crespi, 2017. A new digital image correlation software for displacements field measurement in structural applications. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-4/W2*, pp. 139-145.

- [57] SN. Olufsen, ME. Andersen, E. Fagerholt μ DIC: An open-source toolkit for digital image correlation. *SoftwareX* 2020;11:100391.
- [58] V. Couty, J-F. Witz, P. Lecomte-Grosbras, J. Berthe, E. Deletombe, M. Brieu. GPUCorrel: A GPU accelerated Digital Image Correlation software written in Python. *SoftwareX*, 2021, Vol. 16, pp.100815.
- [59] Réthoré J. UFreckles, language: eng. <http://dx.doi.org/10.5281/zenodo.1433776>. <https://zenodo.org/record/1433776>.
- [60] Y. Saad. Numerical methods for large eigenvalue problems, SIAM, (2011), Classics in Applied Mathematics.
- [61] R. Seghir, J-F. Witz, S. Coudert, 2014. Yadics-digital image correlation 2/3d software. <http://yadics.univ-lille1.fr/wordpress/>
- [62] N. Dahdah, N. Limodin, A. Bartali, JF. Witz, R. Seghir, E. Charkaluk, JY. Buffiere. Damage Investigation in A319 Aluminium Alloy by X- ray Tomography and Digital Volume Correlation during In Situ High-Temperature Fatigue Tests. In *Strain*, Wiley-Blackwell, 2016, 52 (4), pp.324-335.
- [63] A. Plyer, G. Le Besnerais, F. Champagnat. Massively parallel lucas kanade optical flow for real-time video processing applications 11 1–18. <http://dx.doi.org/10.1007/s11554-014-0423-0>.
- [64] *Nvidia cuCim documentation*. <https://docs.rapids.ai/api/cucim/stable>
- [65] C. Roux-Langlois, A. Gravouil, M.-C. Baietto, J. Réthoré, F. Mathieu, F. Hild and S. Roux. DIC identification and X-FEM simulation of fatigue crack growth based on the Williams’ series. *International Journal of Solids and Structures* 2018. V. 53, pp 38–47.
- [66] M.J. Flynn. Very high-speed computing systems, *Proceedings of the IEEE*, 1966, V. 54, pp 1901 - 1909.
- [67] Khronos Group. The OpenCL Specification (Version 3.0). Khronos Group 2011. <https://registry.khronos.org/OpenCL/>
- [68] NVIDIA Corporation. (2023). CUDA Toolkit Documentation (Version 12.2). NVIDIA Corporation. <https://docs.nvidia.com/cuda/>
- [69] R. Badea. PyCUDA: Even Simpler GPU Programming with Python. Parallel for All. NVIDIA Corporation 2012.
- [70] J. Dongarra, V. Eijkhout, P. Luszczek. Recursive Approach in Sparse Matrix LU Factorization. *Scientific Programming*, 2001.
- [71] BS. Andersen and J. Wasniewski. A Recursive Formulation of Cholesky Factorization of a Matrix in Packed Storage. *ACM Transactions on Mathematical Software*, 2001, V. 27, pp. 214–244.

- [72] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, G. E. Dahl. Measuring the Effects of Data Parallelism on Neural Network Training. *Journal of Machine Learning Research*, 2019, V. 20, pp 1–49.
- [73] P. Thoman, K. Dichev, T. Heller, R. Iakymchuk, X. Aguilar, K. Hasanov, P. Gschwandtner, P. Lemarinier, S. Markidis, H. Jordan, T. Fahringer, K. Katrinis, E. Laure, D. S. Nikolopoulos. A taxonomy of task-based parallel programming technologies for high-performance computing. *The Journal of Supercomputing*, 2018, V. 74, pp. 1422–1434.
- [74] D. Scholefield. Proving properties of real-time semaphores. *Science of Computer Programming*, 1995, V. 24, pp. 159–181.
- [75] R. Ferrier, M. L. Kadri, P. Gosselet. The Steklov-Poincaré technique for data completion: Preconditioning and filtering. *International Journal for Numerical Methods in Engineering*, 2018, V. 116(4), pp. 270–286.
- [76] P. C. Hansen. The truncated SVD as a method for regularization. *BIT Numerical Mathematics*, 1987, V. 27, pp. 534–553.
- [77] P. C. Hansen. The discrete Picard condition for discrete ill-posed problems. *BIT Numerical Mathematics*, 1990, V. 30, pp. 658–672.
- [78] P. C. Hansen. Analysis of discrete ill-posed problems by means of the L-curve. *SIAM review*, 1992, V. 34, pp. 561–580.
- [79] M. L. Kadri, J. Ben Abdallah, T. Nouri Baranger. Identification of internal cracks in a three-dimensional solid body via Steklov–Poincaré approaches. *Comptes Rendus Mécanique*, 2011, V. 339, pp. 674–681.
- [80] L. Kovalevsky and P. Gosselet. A quasi-optimal coarse problem and an augmented Krylov solver for the Variational Theory of Complex Rays. *International Journal for Numerical Methods in Engineering*, 2016, V. 107, pp. 903–922.
- [81] V. A. Morozov. The error principle in the solution of operational equations by the regularization method. *USSR Computational Mathematics and Mathematical Physics*, 1968, V. 8, pp. 63–87, 1968.
- [82] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003
- [83] Z. Jia, G.W. Stewart. On the convergence of the Ritz values, Ritz vectors and refined Ritz vectors. Institute of Advanced Computer Studies, Department of Computer Science, University of Maryland at College Park. 1999.
- [84] O. Axelsson, G. Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 1986, V. 48, pp. 499–523.
- [85] Ahmed Chabib, Jean-Francois Witz, Pierre Gosselet, Vincent Magnier. The impact of metrics in mechanical imaging. 2023. [\(hal-04251608\)](#)[Submitted].

- [86] O. Axelsson, I. Kaporin, Error norm estimation and stopping criteria in preconditioned conjugate gradient iterations *Numerical Linear Algebra with Applications*, 2001, V. 8, pp. 265–286.
- [87] P. Gosselet, C. Rey, J. Pebrel. Total and selective reuse of Krylov subspaces for the resolution of sequences of nonlinear structural problems. *International Journal for Numerical Methods in Engineering*, 2013, V. 94, pp. 60–83.
- [88] E. Caron, JF. Witz, C. Cuvier, A. Beaurain, V. Magnier, A. El Bartali. PYCASO: Python module for calibration of cameras by Soloff’s method, 2023, *SoftwareX*, V. 23.
- [89] P. Meer. Robust techniques for computer vision. *Emerging Topics in Computer Vision*, p 107-190, 2004
- [90] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 1944, V. 2, pp. 164–168.
- [91] L. Breiman, J. Friedman, CJ Stone, RA. Olshen. *Classification and Regression Trees*, Taylor & Francis, 1984.
- [92] R. Penrose, A Generalized Inverse of Matrices. *Proceedings of the Cambridge Philosophical Society*, 1954, V. 51, pp. 406–413.
- [93] SM. Soloff, RJ. Adrian, ZC. Liu. Distortion compensation for generalized stereoscopic particle image velocimetry, *Measurement Science and Technology*, 1997, V. 8, pp. 1441–1454.
- [94] I. Léandry, C. Brèque, V. Valle. Calibration of a structured-light projection system: Development to large dimension objects. *Optics and Lasers in Engineering*, 2012, V. 50, pp. 373-379.
- [95] A. Chabib, JF. Witz, P. Gosselet and V. Magnier, Link to the code repository, 2023, https://github.com/chabibchabib/robust_metrics
- [96] Ce Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis* (Thesis). MIT, 2009.
- [97] M. Bornert, F. Brémand, P. Doumalin, JC. Dupré, M. Fazzini, M. Grédiac, F. Hild, S. Mistou, J. Molimard, JJ. Orteu, L. Robert, Y. Surré, P. Vacher, and B. Wattrisse. Assessment of Digital Image Correlation Measurement Errors: Methodology and Results. *Experimental Mechanics*, 2009, V. 49, pp. 353–370.
- [98] WH. Peters, WF. Ranson, MA. Sutton, TC Chu, J. Anderson, Application Of Digital Correlation Methods To Rigid Body Mechanics. *Optical Engineering*, 1983, V.22(6).
- [99] Ahmed Chabib, Jean-Francois Witz, Pierre Gosselet, Vincent Magnier. GPCU_OpticalFlow: a GPU accelerated Python software for strain measurement. 2022. (hal-04025948) [Submitted]

- [100] Zbigniew J. Czech. Introduction to Parallel Computing. Cambridge University Press, 2017, ISBN 9781316795835.
- [101] N. Limodin, J. Réthoré, JY. Buffière, F. Hild, W. Ludwig, J. Rannou, S. Roux. 3d x-ray microtomography volume correlation to study fatigue crack growth, *Advanced engineering materials*, 2011, V. 13, pp. 186-193.
- [102] Mohamed Amine Faham. Influence of a microstructure gradient on the fatigue of a metallic material, 2024. Ecole Centrale de Lille [Thesis].
- [103] J. Weickert, T. Brox. Diffusion and regularization of vector- and matrix-valued images. *Contemporary Mathematics*, 2002, V. 268, pp. 313–251