Finding contours in images using geodesic lines and field lines

Fang Yang

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'Université Paris-Dauphine

Détéction de contours dans les images à l'aide de lignes géodésiques et lignes de champs

École Doctorale de Dauphine — ED 543

Spécialité **Sciences**

COMPOSITION DU JURY :

Laurent D.COHEN
Université Paris Dauphine
Directeur de thèse

Carole LE GUYADER
INSA de Rouen
Présidente du jury

Laurent NAJMAN
Université Paris Est
Rapporteur

Ke CHEN
University of Liverpool
Rapporteur

Alfred M.BRUCKSTEIN
Technion - Israel Insitute of Technology
Membre du jury

Abderrahim ELMOATAZ
University of Caen Basse Normandie
Membre du jury

Jean-Marie MIREBEAU
Université Paris Sud
Membre du jury

Soutenue le 14.09.2017
par Fang YANG

Dirigée par **Laurent D.COHEN**

DAUPHINE UNIVERSITÉ PARIS | PSL RESEARCH UNIVERSITY PARIS

Finding Contours in Images Using Geodesic Lines and Field Lines

Fang Yang

Supervisor: Prof. Laurent D.Cohen

CEREMADE,CNRS, UMR 7534
Université Paris Dauphine,PSL Research University

This dissertation is submitted for the degree of
Doctor of Philosophy

December 2017

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor, Prof. Laurent D.Cohen, without whom this thesis would not be completed. Thank you for your instructive advice and useful suggestions on this thesis and also for providing me a research assistantship over four years. I am deeply grateful of your contributions in this thesis work.

I am also deeply indebted to Prof. Alfred Bruckstein. Thank you for providing me the idea of PointFlow that I can work on and make extensions. I gratefully acknowledge your insightful suggestions, fruitful discussions and warm encouragement.

My sincere gratitude also goes to Dr. Jean-Marie Mirebeau. Thank you very much for your fruitful discussions and suggestions on the scheme of numerical solutions to isotropic and anisotropic heat equations and this scheme really facilitates my work.

Grateful acknowledgement is made to all the jury members of my thesis committee. I would like to warmly thank Prof. Laurent Najman and Prof. Ke Chen who spent their precious time on reading the manuscript and writing reports. I am really honored to have Prof. Elmoataz and Prof. Carole LE GUYADER to present in my Ph.D defense as examiners.

I would also like to extend my gratitude to Prof. Gabriel Peyré et Dr. Dario Prandi for the fruitful discussions on the applicability of Varadhan's formula under different metrics.

I would like to thank the China Scholarship Council to support me and provide me financial aid during these four years.

I would like to thank the entire team of CEREMADE for their help during years. My Ph.D time was made enjoyable in large part due to my dear friends Da, Qichong, Qilong, Ke, Sebastian, Changye, Shuoqing, Camille, Raphaël, Quentin, Clara, Isabelle, Qun, Arnaud, Guillaume, Maxime, Michaël, Laurent, Aude, Marco, Nadia, Ling, Kai, Beibei, Nan *etc*. Thank you!

Special thanks go to my beloved family for their loving considerations and great confidence in me all through these years. Thank my parents for their comprehension and support. Thank my partner Dr. Xin Su who always encouraged and helped me during the whole stage of this Ph.D.

Abstract

This thesis aims at providing methods for finding contours in images and is composed by two parts: A first one dealing with geodesic distance and curves computations using heat diffusions and a second one presenting a boundary-tracking method named PointFlow.

In the first part of this thesis, we study different methods to obtain geodesic distance and geodesic lines on Riemannian manifolds. On a Riemannian manifold, we can define a local metric (a scalar field or a symmetric positive tensor field) that involves the information about the problem we want to solve, i.e. to track the centerline of a tubular structure or to extract the edges in images or on surfaces. The geodesic distance and geodesic lines on images and surfaces play significant roles in computer vision and graphics. They can be applied to vessel segmentation, road extraction, surface remeshing, and so on. Generally, the geodesic distance could be acquired via Dijkstra's method or solving the Eikonal partial differential equation. In this thesis, different kinds of heat equations are applied to approximate the geodesic distance with different metrics. The designed geodesic metrics basically take advantages of the image gray-level, the geodesic anisotropy and so on. Additionally, we come up with two automatic algorithms for extracting the geodesic lines. At last, an approach for the segmentation of tubular structure is proposed. The experimental results testify the robustness and effectiveness of our heat-based method.

For the second part of this thesis, we are interested in one of the most fundamental problems in computer vision: edge detection. Edge detection, a low-level preprocessing step, always remains a hot issue since the 1970's. It is regarded as the basis in tasks such as image segmentation and object detection/recognition. Over the past decades, numerous approaches are proposed to detect edges. At the very beginning, researchers are dedicated to detect the sharp changes of image intensity. As the natural images become more and more complex, only by using the image gradient does not guarantee the quality of edges and contours. Then features such as textures, brightness, colors and similarities are also taken into consideration. By using such features, the performance of edge detectors are improved a lot. More recently, edge detection via machine learning especially deep learning has been highlighted because of its excellent performance. In this thesis, we propose a model called PointFlow which can simulate the process of tracking object boundaries automatically. On the way of finding

contours, we do not only consider the gray-level information, but also combine different features to detect the edges and contours. Moreover, this model can be also used to infer the illusory contours. We test our edge detection method on a well known dataset and the result is comparable to the other classical edge detectors.

Keywords: Minimal Paths, Geodesic, Eikonal Partial Differential Equation, Image Segmentation, Heat Diffusion, Isotropic, Anisotropic, Tubular Structure, Ordinary Differential Equation, Edge Detection, Point Flow Method, Inference of Illusory Contours, Machine Learning, Deep Learning

Table of contents

Notations

d	dimension, $d = 2$ or $d = 3$
Ω	image domain, an open subset of $\mathbb{R}^d$, $\Omega \subset \mathbb{R}^d$
I	a gray-level image, an integrable function $I : \Omega \to \mathbb{R}$
p	a pixel in the image domain, $p \in \Omega$
u	heat distribution on an image domain, $u : \Omega \to \mathbb{R}$
$\mathcal{P}$	a scalar potential metric
T	a tensor metric
D	diffusion tensor
∇	gradient operator
$\nabla\cdot$ or div	divergence operator
Δ	Laplacican operator
∂	partial differential operator
G_σ	a Gaussian filter with variance σ
$\mathbf{V}$	a vector field
γ	a curve
$\mathcal{A}$	a set of curves
ϕ	a distance map, $\phi : \Omega \to \mathbb{R}$
Δt	time step
Δx or Δy	space step on x coordinate or y coordinate

Introduction Abrégée

Contexte de cette these

La segmentation des images joue un rôle important dans les traitements des images. C'est vraiment utile pour beaucoup d'applications dans le domaine de computer vision, i.e, détéction d'objets, reconnaissance, système de contrôle de circulation, l'imagerie médicale. Pendant decennies, chercheurs ont proposé beaucoup de méthodes sur la segmentation des images, i.e, méthodes de seuillage, méthodes variationnelles, méthodes basée sur l'équation différentiel partiel (EDP).

Dans cette thèse, on est interresé par les méthodes basée sur EDPs. Le modèle qu'on utilise dans cette thèse est l'équation de chaleur. En gros, l'idée est d'appliquer le noyau de chaleur à la formule de Varadhan pour obtenir la distance géodésique de chaque point dans le domaine de l'image aux points source. Les chemins minimaux ou les courbes géodésiques pourraient être reculés en resolvant une équation differentielle ordinaire (ODE).

Une géodésique est une courbe qui minimise la distance entre deux noeuds terminaux sur manifold. Une façon générale d'obtenir une courbe géodésique consiste à calculer la distance géodésique en premier. La distance géodésique peut être obtenue par l'alogrithm Dijsktra ou résoudre une équation Eikonal. Elle peut également être approximée par le noyau de chaleur. Les avantages d'utiliser le noyau de chaleur pour prendre l'approximation la distance géodésique sont: il est rapide et facile à mettre en oeuvre, de plus, le noyau de chaleur est moins sensible au bruit que les autres méthodes. C'est pourquoi on est interessé pour dévélopper les méthodes géodésique realisé par chaleur.

A l'exception de la segmentation par les méthodes géodésiques, nous nous interessons également à la détéction des contours et bords. Contours et bords sont les caracristiques le plus fondamentales d'images. La détéction de contour est l'un des problème le plus classique dans le traitement d'image. C'est une étape fondamentale et importante pour l'analysis et la comprehension d'image. Elle connect le traitement de bas niveau tel que le debruitage d'image, l'amélioration d'image avec le traitement de haut niveau tel que la reconnaissance d'objet ou la navigation. Au cours des dernières decennies, de nombreux détécteurs de bord ont été etudiés et proposés. Une façon naturelle de détécter les bords est de trouver ou se

trouvent les gradients les plus élevés. Beaucoup de détécteur basée sur cette idée ont été proposé, e.g. le détécteur Robert, le opérateur Sobel, le détécteur Canny. Pour détécter les contours dans l'images compliquées, c'est pas vraiment efficace de choisir les endroits ou les gradient des caractéristique au niveau bas sont les plus élevés. Donc, les caractéristiques au niveau plus haut ont été proposé pour détécter contours.

Buts et contributions de cette thèse

Dans la partie I, notre but est d'exploiter l'équation de chaleur pour extraire les courbes géodésique et segmenter les structures tubulaire dans l'images. On se concentres sur la disponibilité de la diffusion de chaleur pour prendre l'approximation la distance géodésique sous différent métriques. De plus, deux méthodes automatiques (une s'appelle 'voting' et l'autre s'appelle 'keypoint') ont été proposé pour facilier les éxtractions de courbes géodésiques par la méthode de chaleur. Au final, on propose une méthode pour la segmentation de structure tubulaire.

Les contributions sont:

1. on a introduit des diffusions différentes pour calculer la distance géodésique et trouver les courbes géodésiques. Cette méthode d'utiliser le noyau de chaleur est facile à implementer. Sur le plan de la précision et rappel, les résultats de la méthode de chaleur est comparables a Fast Marching Method. En plus, grâce à la nature de diffusion qui cause le lissage des instantanés, la méthode de chaleur est moins influenceé par les bruits. De plus, selon [95, 96], le temps pour résoudre le système sparse d'EDP s'approche le temps linéaire. C'est-a-dire quand on utilise un schema dernier et decompose l'opérateur Laplacian dans un matrice sparse, la méthode de chaleur peut être extrêmement. Grâce à l'évolution et au development de la résolution de matrice sparse, comme SuiteSparse[24], ca devient plus facile et rapide de résoudre les problèmes d'inversion de matrice sparse et grande.

2. Deux méthodes pour trouver les courbes géodésique automatiquement sont proposées. Ils peuvent réduire beaucoup de main d'oeuvre car ils ont pas besoin de positions d'endpoints. En même temps, la qualité est aussi garantie. Une troisieme dimension qui décrit le largeur des structures tubulaires a été ajoutée a l'équation de chaleur de 2D.

3. À notre connaissance, c'est la premiere fois que la diffusion de chaleur est appliquée au segmentation de la structure tubulaire. La méthode proposée est efficace et rapide, en outre, il n'est pas sensible au bruit.

Dans la partie II, on est inspiré par le processus de diffusion, notre but est désigner un modèle de flux qui pousse les points dans l'image a les bords d'objets. On est aussi interessé

que pour un dététeur entrainé par deep learning, combien d'examples sont nécéssaire à entrainer qu'il peut générer des résultats comparables a notre modele.

Les contributions de cette partie sont:

1. On propose un modèle qui s'appelle PointFlow. Il dépasse l'autres détécteurs traditionnels. Ce modèle peut détécteur les bords, il peut aussi détuire le contours illusoire dans certains cas.

2. En plus, pour évaluer la performance de détécteur précisément, on construit un nouveau dataset composé par images synthétiques.

Organization de cette thèse

La partie I est surtour sur la méthode de chaleur et composée de quatre chapitres: chapitre 1 est un contexte de méthodes géodésiques, chapitre 2 qui décrit l'applicabilité de la méthode de chaleur, chapitre 3 et 4 sont deux extensions d'application de la méthode de chaleur. Cette partie est organisée comme suit:

- **Chapitre 2** introduit les méthodes de contours actives d'état de l'art et les méthodes géodésiques. La distance géodésique peut être obtenue par façons différentes, la façon plus populaire est de résoudre une equation différentielle Eikonal. Les courbes géodésiques peut être obtenues par appliquer une équation différentielle ordinaire sur le plan de distance géodésique. Dans cette chapitre, on a introduit la distance et les courbes géodésiques, le modèle de chemin minimal et son extension, le fast marching méthode, et la méthode de chaleur.

- **Chapitre 3** introduit une méthode pour extraire la distance et les courbes géodésiques basée sur la diffusion de chaleur. Dans cette chapitre, on utilise le formule de Varadhan pour prendre une approximation de distance géodésique numériquement basée sur différents flux de chaleur. On peut considérer l'image ou surface comme un médium pour diffusion. Et puis on choisi au moins d'un point dans ce domaine comme la source de chaleur. Toutes les deux diffusions isotropique et anisotropique sont considérés pour obtenir la distance géodésique selon métriques différents. Nos algorithms sont testés sur les images synthétiques et réeles. Les résultas sont encourageants et démontrent le robustesse de ces algorithms.

- **Chapitre 4** présente deux façons pour segmenter automatiquement par la diffusion de chaleur. Comme nous le savons, trop d'interactions d'humaines influencent l'efficacité d'un algorithm et perdent beaucoup d'energie d'humain. Donc c'est important de

réduire l'intervention efficacement. Dans cette chapitre, on a proposé deux méthodes automatiques, qui s'appellent la méthode de voter et le keypoint. Ceux que les utilisateurs doivent offir sont les points de source, les positions d'endpoints sont plus nécéssaires. Pour la méthode de voter, on a désigné trois façon pour accumuler les points nouveaux et importants. Ces trois façon peut être valable pour cas différents. En termes de la méthode keypoint, the but est pour trouver les points sur les courbes géodésiques. Pour les méthodes automatiques, quand s'arreter est de haute importance. On a désigné des stopping critéres appropriées pour les deux méthodes.

- Dans **Chapitre 5**, on a introduit une méthode interactive pour segmenter les structures tubulaires dans 2D images. Dans les chapitres précédents, les méthodes peuvent détécter que les axes des structures tubulaires, mais il est aussi important de savoir les positions des frontiéres. Dans ce chapitre, on prend l'avantage d'une troisieme dimension, qui est utilisé pour décrire le largeur des structures tubulaires. Donc on peut extraire les frontiéres et les axes en même temps. La méthode est basée sur le chemin minimal obtenu de la distance géodésique de 3D equation de chaleur. Comme ce qu'on a fait dans les chapitres précédents, pour répondre à des besoins différents, on applique des métrique a l'équation de chaleur. On peut obtenir la distance à la demande par résoudre les equations de chaleur correspondants. On a testé notre méthode sur les images synthétiques et réeles. Comparé à la méthode d'état de l'art, notre méthode est robust et efficace.

Partie II est sur la détéction des bords dans l'images. Cette partie est composé par trois chapitres. Premièrement, on a introduit les détécteurs existants qui sont les représentants d'époques différentes. Et puis, on a proposé un modèle qui s'appelle PointFlow pour la détéction de contours et bords. Ce modèle peut aussi détuire les contours illusoires. Enfin, on compare notre modèle avec une méthode entrainé par deep learning, et on fait un dataset pour tester les résultats. Cette partie est organisé comme suit:

- Dans **Chapitre 6**, on a introduit la contexte scientifique de détéction de contours. Premièrement, il donne une description de détécteur typical et represent d'époques différentes. Par exemple, les détécteur le plus tradionnel, i.e., le Robert cross détécteur, le Marr-Hildreth détécteur; les détécteurs moderns, i.e., le pb (probability of boundary) et le pb globale, le sketch tokens; et les détécteurs entrainés par techniques de deep learning, i.e. DeepEdges et DeepContours.

- **Chapitre 7** a introduit un modèle qui peut tracer les contours d'objets. Ce modèle compose de deux étages, au premier, une équation différentielle ordinaire qui décrit la motion d'un point sous l'effet d'un champ vectoriel dans une période de temps. C'est la raison que cette modèle s'appelle "PointFlow". L'étape seconde est d'intégrer les

trajectoires. Dans ce chapitre, on a d'abord décrit comment le PointFlow fonctionne. Puis basé sur le modèle PointFlow, on a présenté une méthode pour détuire les contours illusoires. Afin de regrouper les parties qui devraient être mais pas connectées, une concept "inertie" est proposée. Dans ce chapitre, on a d'abord appliqué la méthode PointFlow pour détécter les points de coins dans les images. Quand un point s'approche tout prés un point de coin, la force "d'inertie" commence fonctionner. Cette force fait le point bouger comme une circle, dont le rayon peut être obtenu par l'étape précédente. Si le trajectoire qui a été généré par 'l'inertie' suffit certaines conditions, le contour illusoire peut être détuit paifaitment.

- Dans la **Chapitre 8**, on a comparé notre modèle PointFlow à un détécteur entrainé par deep learning. Le but est de prouver que, par comparaison avec la méthode de deep learning, les méthodes basés sur modèle sont toujours utiles et efficaces. Même que le deep learning vient avec une tendance irrésistible et il est vraiment utile pour des applications différentes, l'existence de méthodes basés sur modele est toujour nécéssaire.

Dans **Chapitre 9**, nous terminons cette thèse de doctorat et présentons une perspective pour les travaux futurs.

Chapter 1

Introduction

1.1 Context of this thesis

Image segmentation plays an important role in image processing. It is useful for a lot of applications in computer vision field, such as object detection, recognition tasks, traffic control systems, medical imaging and so on. During decades, a lot of methods on image segmentation have been proposed, such as the thresholding methods, clustering methods, region-growing methods, variational methods, Partial Differential Equation (PDE) based methods and so on.

In this thesis, we are interested in the PDE based methods for image segmentation. The PDE model used in this thesis is the heat equation. Basically, the idea is to apply the heat kernel to the Varadhan's formula to obtain the geodesic distance from each point in the image domain to the source points. The minimal paths or geodesic curves could be backtracked by solving an ordinary differential equation (ODE).

A geodesic is a curve that minimizes the distance of two terminal nodes on a manifold. A general way to obtain a geodesic curve is to compute the corresponding geodesic distance first. The geodesic distance can be computed by using the Dijsktra algorithm or solving an Eikonal equation via fast marching method or fast sweeping method. It can also be approximated by using the heat kernel. The advantage of using the heat kernel to approximate the geodesic distance lies in that it is fast and easy to implement, moreover, the heat kernel is less sensitive to noise than the other methods. This is the reason why we are interested in developing the geodesic methods in heat.

Besides segmentation using geodesic methods, we are also interested in edge detection in computer vision. Edges are the most fundamental features of an image. Edge detection is one of the most classical problem in image processing and computer vision. It is a basic and important image processing step for image analysis and understanding. It bridges the low-

level processing such as image denoising, image enhancement with the high-level processing such as object recognition or navigation.

In the past decades, numerous edge detectors have been studied and proposed. Natural way to detect edges is to find out where the highest gradients are. A lot of detectors based on this idea were proposed, e.g. the Robert cross detector, the Sobel operator, the Canny detector. Then researchers found that only by detecting the highest gradient of the low level feature such as the graylevel, color, or brightness could not guarantee the quality of detection in natural images. Hence, people propose higher level features to detect edges in more complicated scenes. In these researches, the features are no longer obtained directly from the images, but are learned by some machine learning techniques such as K-means, or decision trees. Besides, on the way to detect edges, the deep learning technique should never be underestimated. It helps the edge detection work to reach a new height.

1.2 Purposes and contributions of this thesis

In part I, our main objective is to exploit the heat diffusion to extract geodesic curves or segment tubular structures in images. In this work, we focus on the availability of the heat diffusion on approximating the geodesic distance under different metrics. In addition, two automatic methods (a voting method and a keypoint method) have been proposed to facilitate the extraction of geodesic curves by using the heat method. Last but not least, we propose a tubular structure segmentation method in order to obtain the centerlines and the contours of the tubular structures at the same time.

The contributions lie in that:

1. we introduce different heat flows to compute the geodesic distance and find geodesic lines. The advantage of using heat kernel to approximate the geodesic distance is that this algorithm is easy to implement. In terms of the precision and recall, the heat method yields comparable results to the state-of-the-art method (the Fast Marching Method). In addition, because the nature of heat diffusion causes instant smoothing, the heat method is less influenced by noise compared with the other methods. Moreover, it is proved that the sparse systems arising from the elliptic PDEs can be solved in very close to linear time [95, 96]. It means that when we are using a backward scheme and discretizing the Laplacian operator into a sparse matrix, our heat method can be extremely fast. And thanks to the evolution and development of fast sparse matrix solver, such as SuiteSparse[24], to solve the inversion problems of large sparse matrix becomes easier and faster than ever.

2. two automatic methods that are designed for the heat method were proposed, which save a lot of human labor for selecting endpoints and the quality can be also guaranteed.

3. a third dimension which describes the width of the tubular structure has been added to the 2D heat equation. To the best of our knowledge, it is the first time that the heat diffusion is applied to segment the tubular structure. The proposed method is efficient and fast, moreover, it is insensitive to noise.

In part II, we are inspired by the process of diffusion and we aim at designing a flow model that drives the points on the image plane to flow on the object boundaries. And we are also interested in figuring out that how many examples of images are necessary for a deep learning edge detector that it can give comparable results as our model can.

The contributions of this part are:

1. we propose a PointFlow model which outperforms the other traditional edge detectors. This model can not only detect edges, but also has an ability to infer illusory contours for specific cases.

2. In addition, in order to evaluate the performance of edge detectors precisely, we build a new dataset that is made up of synthetic images.

1.3 Organization of this thesis

Part I is mainly concerned about the heat method and composed by four chapters: a background of the geodesic methods, a theoretical chapter which describes the applicability of the heat method and two extended applications of this method. This part is organized as follows:

- **Chapter 2** gives a general introduction to the state-of-the-art active contours methods and the geodesic methods. The geodesic distance can be obtained from different ways, the most popular way is to solve the corresponding eikonal differential equation. The geodesic curves can be achieved by applying an ordinary differential equation on the geodesic distance map. In this chapter, we introduce geodesic distance and curves, the minimal path model and its extensions, the fast marching methods and the geodesic in heat method.

- **Chapter 3** introduces a method to extract geodesic distance and geodesic curves based heat diffusion. In this chapter, we apply Varadhan's formula to obtain a numerical approximation of geodesic distance according to metrics based on different heat flows. The heat equation can be utilized by regarding an image or a surface as a medium for

heat diffusion and letting the user set at least one source point in the domain. Both isotropic and anisotropic diffusions are considered here to obtain geodesics according to their respective metrics. Our algorithms are tested on synthetic and real images as well as on a mesh. The experimental results are very promising and demonstrate the robustness of the algorithms.

- **Chapter 4** presents two ways for automatic segmentation based on heat diffusion. As we know, too much human interaction will affect the effectiveness of an algorithm and waste a lot of human energy. So it is significant to reduce human intervention effectively. In this chapter, we propose two automatic methods, which are called the voting method and the keypoint method. What the users have to supply are the source points, and the end points are no longer necessary. For the voting method, we design three ways for accumulating the newly obtained important points. These three ways can be fit for different situations. In terms of the keypoint method, as the name implies, the aim is to search for the points on the curves. For the automatic segmentation methods, when to stop is of high importance. We design proper stopping criteria for both methods.

- **Chapter 5** introduces an interactive method for tubular structure segmentation in *2D* images. In the previous chapters, the methods can only detect the centerline of tubular structures, while it is important to get the boundaries or contours of these structures. In this chapter, we take the advantage of an additional third dimension, which is used for describing the width of the tubular structure. In this way, the boundaries and centerlines of the tubular structure can be extracted simultaneously. The method is based on the minimal paths obtained from the geodesic distance solved by *3D* heat equation. Like what we have done in the previous chapters, to meet different needs, we apply different metrics to the heat equation. By solving the corresponding heat equations, we obtain the distances desired. We test our method on both synthetic and real images, and compare it with the state-of-the-art method, the promising results demonstrate the robustness and effectiveness of the algorithm.

Part II is mainly about edge detection. It is composed by three chapters. First, we briefly introduce the existing edge detectors which are representatives of different eras and the evaluation methods. Then we propose a model called PointFlow for edge detection. This model can also be applied to infer the illusory contours. Finally, we make comparisons between our model and a deep learning method, testing on a dataset built on our own. This part is organized as follows:

- **Chapter 6** introduces the scientific background of the edge detection. Firstly, it gives a brief description of typical and representative edge detectors of different times, including the very traditional edge detectors such as the Robert cross detector, the Marr-Hildreth

detector; the modern detectors such as the pb (probability of boundary) and global pb detector or the sketch tokens; and the edge detectors generated by deep learning techniques including DeepEdges and DeepContours *etc*.

- **Chapter 7** introduces a model which can be used to trace the object contours in images. This model is an ordinary differential equation which simulates the motion of a moving point under the effect of a vector field within a period of time, so we call it the "PointFlow" method. In this chapter, we first describe how the PointFlow method works. Then based on the PointFlow model, we present a method for the inference of illusory contours. In order to group those parts which should be but not connected, an "inertia-like" concept is proposed. In this chapter, we first apply the point flow method to detect the corner points in images. When a point approaches a nearby corner point, the "inertia force" begins to work. This force drives the point to move as a circle, of which the radius can be obtained from the previous step. If the trajectory generated by the inertia is in accordance with the standard that we set, the illusory contours could be perfectly inferred.

- **Chapter 8** compares our model with a deep learning method. The aim is to prove that, compared with the deep learning method, by using the model-based edge detection method is still useful and effective in small dataset. Though that deep learning comes with an overwhelming tendency and does a very good job in different kind of applications in computer vision, the existence of model-based is still necessary.

In the end of this thesis Chapter 9, we conclude this PhD thesis and presents an outlook for the future work.

1.4 Publications Related to this Thesis

The works in this manuscript have led to the following publications:

- The results in Chapter 3 and Chapter 4 have been published in the Journal of Mathematical Imaging Vision (JMIV) [123]:

  (1) **Fang Yang**, Laurent D.Cohen, Geodesic Distance and Curves Through Isotropic and Anisotropic Heat Equations on Images and Surfaces. *Journal of Mathematical Imaging and Vision*, 55(2), 210-228, 2016

- The result in Chapter 5 has been presented on the Conference on Scale Space and Variational Methods in Computer Vision (SSVM2017) and published on the proceeding [124]:

  (2) **Fang Yang**, Laurent D.Cohen, Tubular Structure Segmentation Based on Heat Diffusion. In International Conference on Scale Space and Variational Methods in Computer Vision (pp. 54-65). Springer, Cham, 2017

- Part of the work of Chapter 7 is soon to be presented on the International Conference on Image Processing (ICIP2017) [122]:

  (3) **Fang Yang**, Alfred M.Bruckstein, Laurent D.Cohen, A Model for Automatically Tracing Object Boundaries, International Conference on Image Processing (ICIP2017), Beijing, China.

- The work of Chapter 7 has been presented on the conference of Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR2017):

  (4) **Fang Yang**, Alfred M.Bruckstein, Laurent D.Cohen, PointFlow: A Model for Automatically Tracing Object Boundaries and Inferring Illusory Contours (EMMCVPR), Venice, Italy, 2017

- The work of Chapter 8 in under preparation for an image analysis journal:

  (5) **Fang Yang**, Alfred M.Bruckstein, Laurent D.Cohen, A model is worth hundreds of examples, under preparation

Part I

Geodesic in Heat

Chapter 2

Active Contours and Geodesic Methods

Abstract

The first part of this thesis is focused on the extraction and segmentation of tubular structure by using geodesic methods. A geodesic is a curve minimizing the distance between two points. It is also considered as the shortest path. In this thesis, we are dedicating to computing the geodesic distance on Riemannian metrics and the notion of geodesics also refers to minimal paths. In this chapter, we introduce the background of the geodesic methods, as well as the well known active contour models, the gradient vector flow, the minimal paths models based on PDEs, the Fast Marching Method and the Heat Method.

2.1 Geodesic Distance and Geodesic Curves

Geodesic distance and geodesic curves play an important role in image processing and computer vision. They can be applied to tubular structure segmentation in images, such as vessel segmentation in medical images and road extraction in satellite images. These applications are of high importance for human development. For example, the automatic vessel segmentation can facilitate the process of segmentation by doctors themselves, and the road extraction helps the urban or rural road network planning.

2.1.1 Geodesic Distance and Geodesic Distance Map

In this section, the image domain is defined as Ω, where $\Omega \subset \mathbb{R}^d$ and d is the dimension. The geodesic distance $d(p_s, p_e)$ on a domain Ω between two points p_s, p_e is defined as follows:

$$d(p_s, p_e) = \inf_{\gamma \in \mathcal{A}_{p_s, p_e}} L(\gamma) = L(\gamma^\star) \tag{2.1}$$

where $\mathcal{A}_{p_s,p_e}$ is the set of curves between these two points, $\gamma^\star$ is a geodesic curve and $L(\gamma)$ stands for the length of a Lipschitz curve according to a certain metric.

In the isotropic case, the length of a Lipschitz continuous curve $\gamma : [0,1] \to \Omega$ is defined as a weighted length:

$$L(\gamma) = \int_0^1 (P(\gamma(t)))\,\|\gamma'(t)\|\mathrm{d}t \tag{2.2}$$

where P is a potential function defined from the image, $\gamma'(t)$ is the derivative of $\gamma(t)$, $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^d$.

In the anisotropic case, the length of a Lipschitz continuous curve $\gamma : [0,1] \to \Omega$ is computed as:

$$L(\gamma) = \int_0^1 \|\gamma'(t)\|_{T^{-1}(\gamma(t))}\mathrm{d}t \tag{2.3}$$

where $T(\gamma(t))$ is a metric tensor. Let us denote S_d^+ the set of positive symmetric definitive matrices, d is the dimension of the domain, $T : \Omega \to S_d^+$, $T(\gamma(t)) \in S_d^+$. In addition, given a vector $u \in \mathbb{R}^d$ and a matrix $M \in S_d^+$, the norm $\|u\|_M = \sqrt{\langle u, Mu\rangle}$, so the norm $\|\gamma'(t)\|_{T(\gamma(t))} = \sqrt{\langle \gamma'(t), T(\gamma(t))\gamma'(t)\rangle}$ in Eq. (2.3), and the length can be rewritten as:

$$L(\gamma) = \int_0^1 \sqrt{\langle \gamma'(t), T(\gamma(t))\gamma'(t)\rangle}\mathrm{d}t \tag{2.4}$$

In the isotropic case, the weighted metric P depends only on the location of the curve, while in the anisotropic case, the metric depends both on the location and the orientation of the curve.

The geodesic distance defined in Eq. (2.1) can be generalized to the distance from any p to a set of points $S \subset \Omega$. This defines the distance map:

$$\phi_S(p) = \min_{s \in S} d(p,s) \tag{2.5}$$

2.1.2 Eikonal Equation

The geodesic distance map ϕ satisfies the following non-linear Eikonal partial differential equation.

In 2D anisotropic case, $\Omega \subset \mathbb{R}^2$, the anisotropic Eikonal equation is as follows:

$$\begin{cases} \forall p \in \Omega \setminus S, & \|\nabla \phi_S(p)\|_{T(x)^{-1}} = 1 \\ \forall p \in S, & \phi_S(p) = 0 \end{cases} \tag{2.6}$$

$T(x)$ can be decomposed as:

$$T(x) = \lambda_1(x)e_1(x)e_1(x)^T + \lambda_2(x)e_2(x)e_2(x)^T \tag{2.7}$$

where $\lambda_1(x)$ and $\lambda_2(x)$ are the eigenvalues of $T(x)$, and $e_1(x)$ and $e_2(x)$ are their corresponding eigenvectors.

In the isotropic case, $T(x) = P^2\mathrm{I}d$, where P is a scalar function that stands for the potential and $\mathrm{I}d$ is the identity matrix. The classical isotropic Eikonal equation is recovered:

$$\begin{cases} \forall p \in \Omega \setminus S, & \|\nabla \phi_S\| = P \\ \forall p \in S, & \|\nabla \phi_S\| = 0 \end{cases} \tag{2.8}$$

For the Euclidean case, $P = 1$ and we have:

$$\|\nabla \phi_S\| = 1 \tag{2.9}$$

and the solution becomes $\phi_{p_s}(p) = \|p - p_s\|$.

The Eikonal equation presents the solution to the geodesic distance map Eq. (2.5), its numerical solution will be presented in Section 2.3.

2.1.3 Geodesic Curves

After obtaining the geodesic distance ϕ, a geodesic curve $\gamma^\star$ can be acquired by tracing back to the source point p_s from an endpoint p_e. The process of tracing is described by an ordinary differential equation (ODE) as follows:

$$\begin{cases} \forall t > 0, \frac{d\gamma^\star(t)}{dt} = -\eta_t v(\gamma^\star(t)) \\ \gamma^\star(0) = p_e \end{cases} \tag{2.10}$$

where v is the gradient of the distance, twisted by $T(x)^{-1}$

$$v(x) = T(x)^{-1}\nabla \phi_S(x) \tag{2.11}$$

and $\eta_t > 0$ is a scalar function that controls the speed of the geodesic parameterization.

To obtain a unit speed parameterization, one should normalize v, so η_t should be:

$$\eta_t = \|v(\gamma_\star(t))\|^{-1} \tag{2.12}$$

When we are using an isotropic metric, $T(x) = P^2 \mathrm{I}d$, the ODE for tracing the geodesic curve becomes:

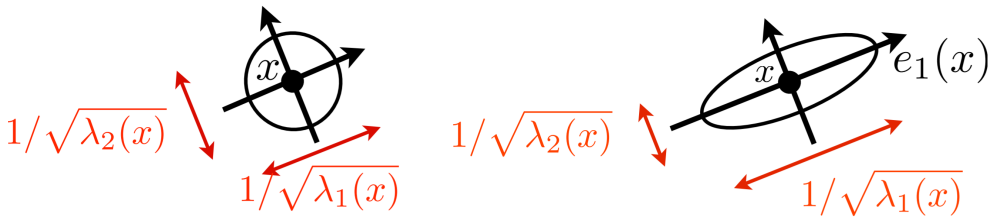
$$\forall t > 0, \frac{\mathrm{d}\gamma^\star}{\mathrm{d}t} = -\eta_t \nabla \phi_S(\gamma^\star(t)) \tag{2.13}$$

Figure.2.1 shows the isotropic and anisotropic metrics on a 2D domain respectively, where $\lambda_{1,2}$ and $\mathbf{e}_{1,2}$ are the eigenvalues and eigenvectors of $T(x)$.

The anisotropy μ is defined as:

$$\mu = \frac{\lambda_2(x) - \lambda_1(x)}{\lambda_2(x) + \lambda_1(x)} \in [0, 1] \tag{2.14}$$

In the isotropic case, $\lambda_1 = \lambda_2$, and the anisotropy $\mu = 0$.



Isotropic metric Anisotropic metric

Fig. 2.1 Different metrics [82]

2.2 Active Contours and Minimal Paths

2.2.1 Active Contour Model

Classical Snake Model

The original and pioneering work of active contours is the snake model [50], which was proposed by Kass *et al.* in 1988. The snake model and its extensions are widely used for detecting and integrating boundaries in images. The idea of snake model is to drive a given contour to converge to the interesting edges. The original snake model is to minimize the following energy functional:

$$E_{snake}(C) = \int_0^1 (\omega_1 \|C'(t)\|^2 + \omega_2 \|C''\|^2 + P(C(t)))dt \tag{2.15}$$

where $C(t) = (x(t), y(t))$ is the mapping of the active contour. t is the parameter, ranging from 0 to 1. C' and C" are the first and second order derivatives of the curve C. ω_1 and ω_2 are two positive constants, weighting the internal forces of the snake model and controlling the elasticity and rigidity of the contour C. In addition, $P(C)$ is the potential related to the external force, generally, P has a formula:

$$P = g(\|\nabla I\|) \tag{2.16}$$

where I here represents the image, and $g(\cdot)$ is a non-negative decreasing function:

$$g(a) = \eta_0 + \frac{1}{\eta_1 + a}, \qquad \text{or} \qquad g(a) = \eta_0 + \exp(-\eta_1 a) \tag{2.17}$$

where η_0 and η_1 are two positive constants which prevent g(a) from being zero or invalid.

To find a curve C which minimizes the energy of the snake model Eq. (2.15), the Euler-Lagrange equation should be satisfied:

$$-\omega_1 C''(t) + \omega_2 C''''(t) + \nabla P(C(t)) = 0, \qquad \forall t \in [0, 1] \tag{2.18}$$

The disadvantages of snake model are that: firstly, the users have to provide an initial curve which is close to the desired feature; secondly, the curves are the local minima, which are sensitive to noise and sharp corners.

Balloon Snakes

In [17], Cohen introduced a new model for active contours by proposing an addtional external force to the original snake model, which improves closed contour detection significantly. The

new active contour model was defined by adding an inflation force which makes the curve behave well in these cases. This inflation force is also called the "balloon force", because the curve (or surface) behaves like a balloon which is inflated. It is expressed as minimizing the following energy:

$$E_{balloon}(C) = -k_1 \int_C \mathrm{d}A \tag{2.19}$$

When it passes by edges, it is stopped if the edge is strong, or passes through if the edge is too weak with respect to the inflation force. This avoids the curve being trapped by spurious isolated edge points, and makes the result less sensitive to the initial conditions. In the 2D case, the energy modeling the balloon force is proportional to the inner area of the curve, and it derives a balloon inflation force:

$$F_{balloon} = k_1 \mathbf{n} \tag{2.20}$$

where $\mathbf{n}$ is the outward normal to the contour (or surface) model. Nevertheless, the balloon model should be used carefully since the initial model has to be completely inside or outside (depending on the sign of $k1$) the desired contour.

The new external force was presented:

$$F_{ext} = F_{balloon} - k_2 \frac{\nabla P}{||\nabla P||} \tag{2.21}$$

where k_2 is a constant which is a little larger than k_1. The evolution of the curve will stop under the control of Eq. (2.21). The balloon force could avoid the spurious edges and is not sensitive to the initialization of curves. Figure.2.2 gives an example of the balloon method on an MRI image.

Distance Potentials

Cohen and Cohen [18] incorporated the use of edge points extracted by a local edge detector to reduce the problem of sensitivity to the noise of the potential in the original snake model. This allows to combine the qualities of a good local edge detector, for example a Canny Deriche edge detector [14, 26], with a global active model. The introduced energy, for a contour C is:

$$E_{edges}(C)) = \int f(D(C(s)))\mathrm{d}s \tag{2.22}$$

where D is the Euclidean distance map and for each point $p \in \Omega$, $D(x)$ denotes the Euclidean distance from p to the nearest edge points.

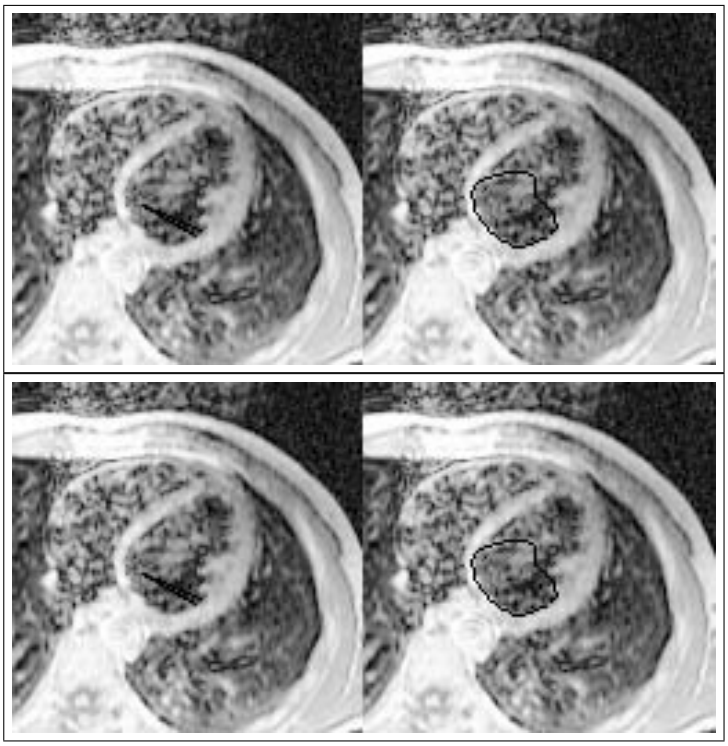


Fig. 2.2 Evolution of the balloon curve to detect the left ventricle on a Magnetic Resonance Image (MRI) [17].

The external force of the distance vector field is expressed as:

$$F_{edges} = -f'(D)\nabla D \tag{2.23}$$

∇D is the gradient vector of D and it points to the edges. It helps the curve move to its nearest edge points. The attraction forces derived from the edge detector may be used either as the only image forces, or may be combined with an intensity-gradient image to enhance the detected edges.

Gradient Vector Flow

Xu and Prince [121] proposed a new idea for active contours evolution based on the diffused gradient field. It is called the gradient vector flow (GVF). The GVF snake is an extension of the snake model and compared to the classical snake model [50], the GVF snake model is less sensitive to the initialization.

The original snake is a 2D dynamic contour that minimizes the energy function Eq. (2.15). The GVF uses a gradient vector field as a constraint energy on Eq. (2.15). The basic idea of the gradient vector field construction is to diffuse the image gradient information to the whole image domain Ω. Compared to the classical snake model, the GVF is insensitive to the initial curve.

The gradient vector flow field $\mathbf{V}(x,y) = [u(x,y), v(x,y)]$ is defined as a vector field that minimize the energy functional:

$$E_{GVF} = \int\int_{\Omega} \mu(\|\nabla u\|^2 + \|\nabla v\|^2)dxdy + \int\int_{\Omega} \|\nabla f\|^2 \|\mathbf{V} - \nabla f\|^2 dxdy \tag{2.24}$$

where μ is a positive constant which is used to balance the two terms and f is an edge map which can be obtained from the image:

$$f(x,y) = |\nabla I(x,y)|^2 \qquad \text{or} \qquad f(x,y) = |\nabla[G_\sigma(x,y) * I(x,y)]|^2 \tag{2.25}$$

where $G_\sigma(x,y)$ is a 2D Gaussian function.

The GVF field $\mathbf{V}$ in Eq. 2.24 can be found by solving the Euler-Lagrange equation:

$$\frac{\partial \mathrm{V}}{\partial t} = \mu\Delta\mathbf{V} - (\mathbf{V} - \nabla f)\|\nabla f\|^2 \tag{2.26}$$

$\mathbf{V}$ obtained by minimizing E_{GVF} is used an external force of the classical snake model, leading to the follow evolution equation:

$$\frac{\partial C}{\partial t} = \omega_1 \frac{\partial^2 C}{\partial s^2} - \omega_2 \frac{\partial^4 C}{\partial s^4} + \mathbf{V} \tag{2.27}$$

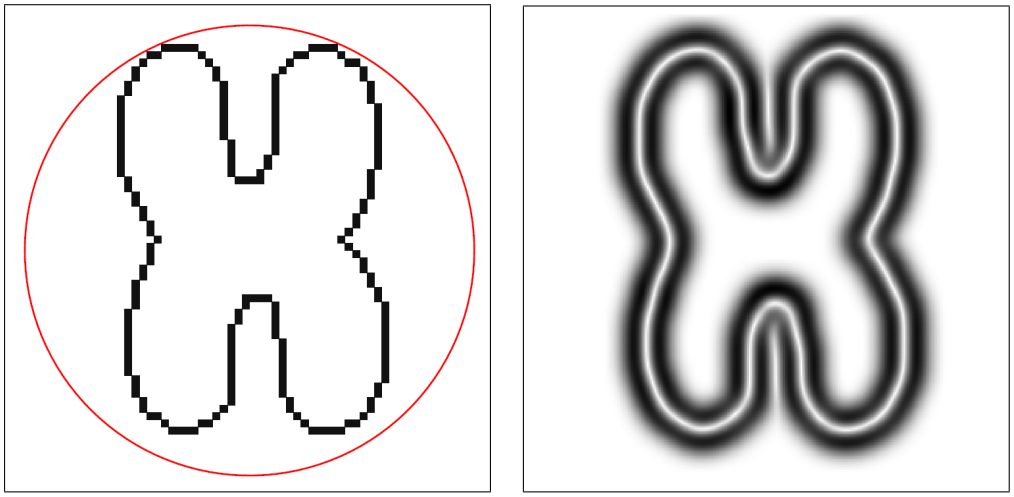
Figure.2.3 is an example of the GVF model.

Note that for a homogenous region where the f is a constant, the second term of the r.h.s in Eq. (2.26) equals to zero. In such cases, the vector field is determined by the heat equation which enhances the smoothness of the vector field. For the places which are close to the boundary, $\mathbf{V} = \nabla f$, in this case, if we use the normalized vector field of $\mathbf{V}$, the initial curve can be far away from the object boundaries.

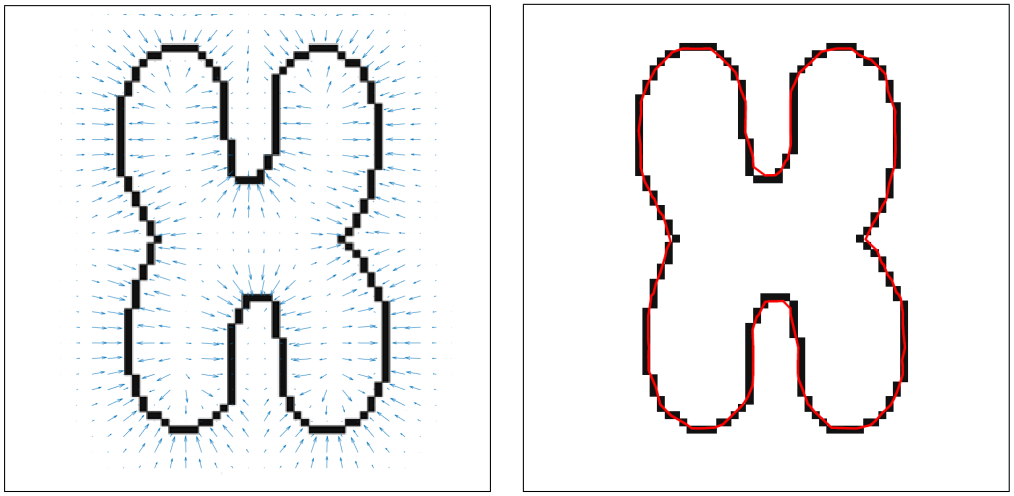
2.2.2 Minimal Path Model

The curve optimization process of the snake energy is usually blocked at unexpected local minima of the functional Eq. (2.15). So the result is highly dependent on the initialization of the curve, and is sensitive to noise. Figure.2.4 shows an example that snake model is not able to detect the spurious edges in images.

To improve the performance of the snake model, in 1997, Cohen *et al.* proposed a global minimal path model [19]. Given an image $I : \Omega \to \mathbb{R}$ and two points p_{s_0} and p_x, the geodesic γ is a curve connecting these two points that globally minimizes the following energy functional

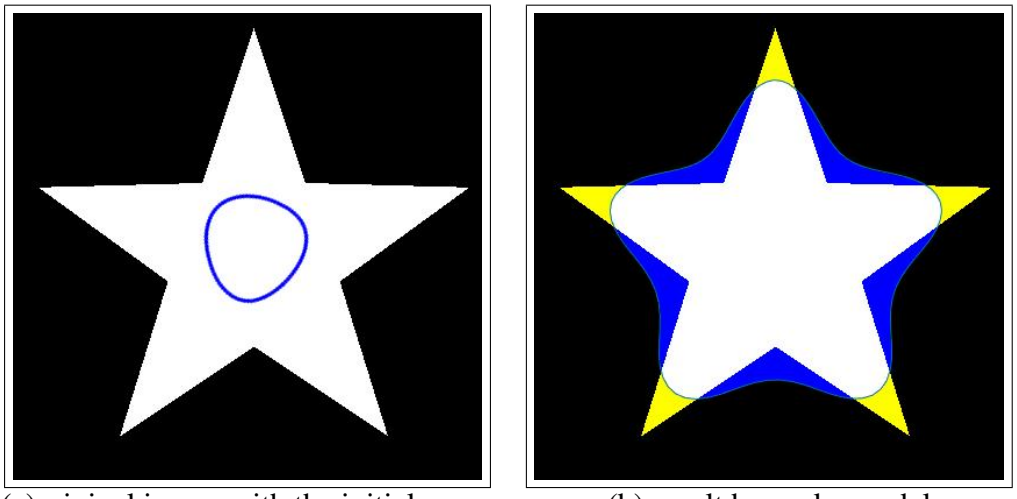


(a)original image (b) potential



(c)gradient vector (d)result

Fig. 2.3 The GVF method



(a)original image with the initial curve (b) result by snake model

Fig. 2.4 The snake stops at spurious edges.

$E : \mathcal{A}_{p_{s_0},p_x} \to \mathbb{R}^+$:

$$E(\gamma) = \int_0^L (P(\gamma) + w)ds, \quad \gamma \in \mathcal{A}_{p_0,p_x} \tag{2.28}$$

where P is a potential cost function computed from I, w is a positive constant that imposes regularity on the curve. $\mathcal{A}_{p_{s_0},p_x}(s)$ is the set of all the curves linking p_{s_0} and p_x, s is the arclength. Let us denote $\mathcal{P} = P + w$ so that $E(\gamma) = \int \mathcal{P}(\gamma(s))ds$.

The minimal action map (geodesic distance map) ϕ is defined in Eq.(2.1). In fact, the geodesic distance map ϕ defines a level set function which describes the arrival time from the source point p_{s_0}. The level set line Γ is:

$$\Gamma := \{p \in \Omega; \phi(p) = t\} \tag{2.29}$$

where t means the arrival time.

The level set evolution equation is:

$$\frac{\partial \Gamma}{\partial t} = \frac{1}{\mathcal{P}}\mathbf{n} \tag{2.30}$$

where $\mathbf{n}$ is the unit normal vector of Γ.

According to [12], Eq.(2.30) is in fact a front propagation equation with the speed $\frac{1}{\mathcal{P}}$. It propagates geodesic distance to the whole image domain.

To associate the minimal action map with the Eikonal equation Eq.(5.5), let us consider that, according to Eq.(2.29), we can obtain that:

$$\frac{\partial \phi}{\partial t} = \left\langle \nabla\phi, \frac{\partial \Gamma}{\partial t} \right\rangle = 1 \tag{2.31}$$

By associating Eq.(2.31) with Eq.(2.30), we have:

$$\left\langle \nabla\phi, \frac{\partial \Gamma}{\partial t} \right\rangle = \frac{1}{\mathcal{P}} \left\langle \nabla\phi, \frac{\nabla\phi}{||\nabla\phi||} \right\rangle = 1$$

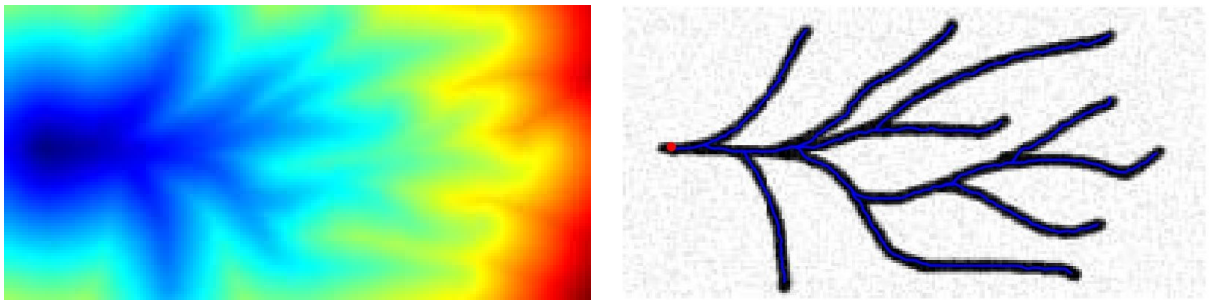
which yields

$$||\nabla\phi(x)|| = \mathcal{P}(x) \tag{2.32}$$

Thus the geodesic distance map is associated with the Eikonal equation Eq.(2.6), where the metric is isotropic.

To obtain the geodesic distance ϕ, we are indicated to solve the non-linear Eikonal equation Eq.(2.32). It is suggested in [19] that the fast marching method can be a very efficient solver.

For an image, once the source point p_{s_0} and the geodesic distance map ϕ is fixed, the users can obtain the multiple geodesic curves which connect a set of end points $p \in \Omega$ and p_{s_0} by using the ODE in Eq.(2.10), shown in Figure.2.5.



geodesic distance geodesic curves

Fig. 2.5 an example of geodesic distance and geodesic curves

2.3 Eikonal Equation and Fast Marching Method

To solve this minimization problem, Cohen and Kimmel [19] proposed a Hamiltonian approach: Find the minimal action map $\phi : \Omega \to \mathbb{R}$ that solves the isotropic Eikonal equation:

$$||\nabla \phi|| = \mathcal{P} + w \tag{2.33}$$

with the boundary condition $\phi(p_{s_0}) = 0$. Minimal paths techniques are then widely used for centerlines extraction of tubular structures.

In [46], the authors summarize popular numerical solutions to the Eikonal equation on Cartesian grids. The most common algorithms such as the Fast Marching [97, 19] and Fast Sweeping [108, 109, 125] are quite often used.

By comparing the errors, speed, accuracy and robustness of different algorithms, the authors concluded in [46] that the Fast Marching Method outperforms the other methods.

The Fast Marching Method was proposed independently by Sethian [97, 98] and Tsitsiklis [110] for an isotropic metric on a regular grid. It is an extremely fast scheme to approximate the solution to nonlinear Eikonal differential equation Eq.(5.5), and the time complexity is $O(N \log(N))$, where N is the total number of the grid points.

The Fast Marching Method is based on an optimal ordering of the grid points. Each grid point is visited only once during the process of propagation, and the exact solution is obtained by the visit.

During the process of marching, each grid point could be tagged as three states: *Accepted*, *Candidate* and *Far*. The set of *Candidate* points forms an interface between the *Accepted* points and the *Far* points. The *Accepted* points are the points of which the geodesic distance is fixed and could not be changed. The *Candidate* points are the points for which the geodesic distance has been computed at least once but not fixed. The distance value can be changed. The *Far* points are the points for which the geodesic distance has not been computed yet.

Figure.2.6 illustrates the fast marching front: the blue points are the candidate points and form the front which is expanding from the *Accepted* points to the *Far* points.

At the beginning, the source point p_{s_0} is tagged as *Candidate*, and the rest grid points are labeled as *Far* points. At each update, we choose one point with the smallest ϕ value from the Candidates, denote this point by $x_{\min}$ and label it as *Accepted*. The distance value ϕ is then updated for each neighbors $N_M(x_{\min})$ (could be *Candidate* or *Far*) of $x_{\min}$. $N_M(x_{\min})$ is the neighborhood points of $x_{\min}$, and for a 2D domain, $M = 4$ or $M = 8$. Figure.2.7 shows the 4-connectivity and 8-connectivity stencils on 2D Cartesian grid. In 2D, the local update for the isotropic Fast Marching is usually done using the 4-connectivity neighbors while for the anisotropic Fast Marching the 8-connectivity.

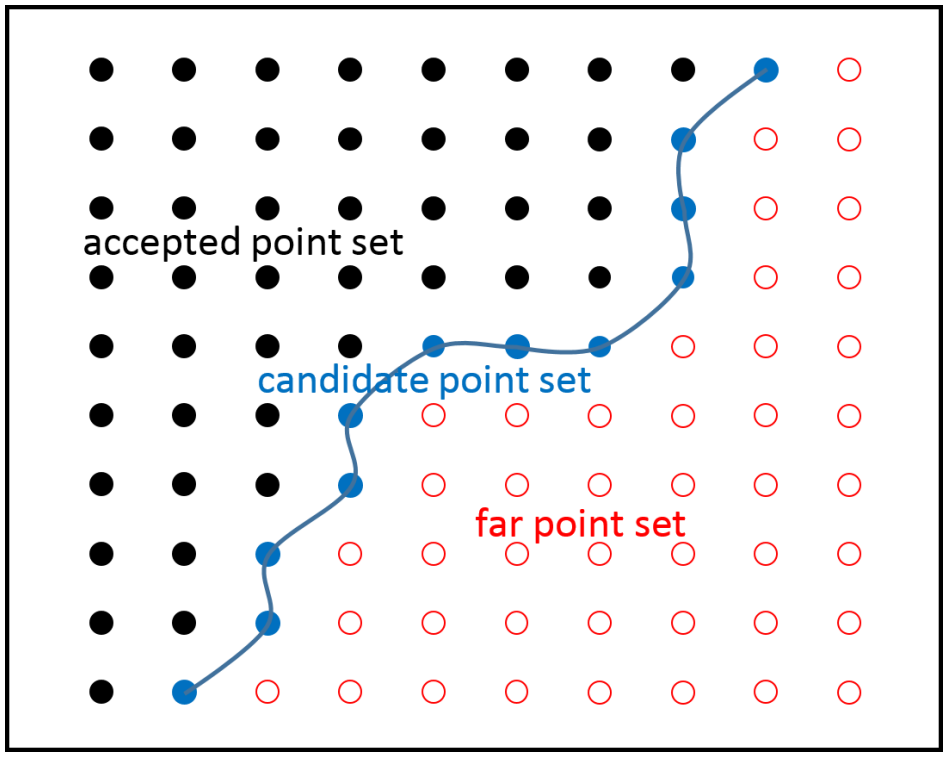
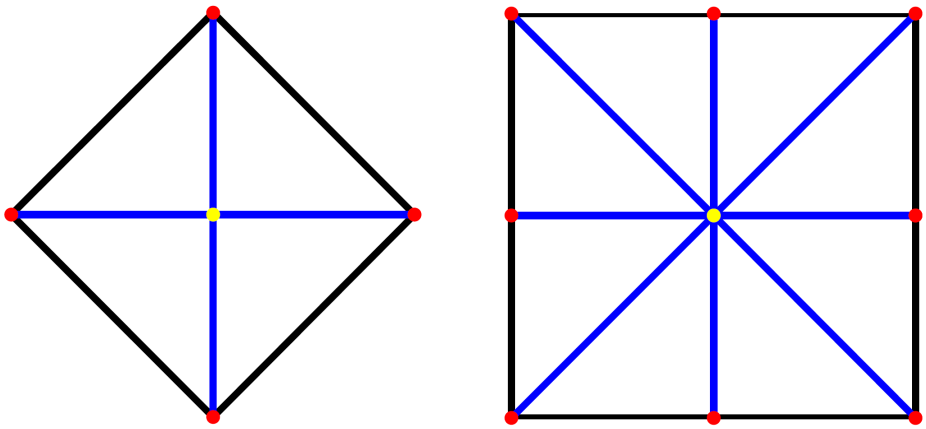


Fig. 2.6 FMM-diagram, the solid black spots are the *Accepted* points, the blue spots are the *Candidates* and the hollow spots are the *Far* points. The expanding front is formed by the blue points.



(a) 4-connectivity (b) 8-connectivity

Fig. 2.7 The center yellow points are the Accepted points, and the red points around are the neighborhood points.

Algorithm.1 describes the general fast marching algorithm. For simplicity, we illustrate the isotropic and anisotropic Fast Marching Method on 2D manifold, but it can be applied to arbitrary dimension.

Algorithm 1 Fast Marching Method

Notation:
$N_M(p)$: the set of M neighbors of a grid point p, $M = 4$ or $M = 8$ in 2D.
L: the label of the points, L can be *accepted*, *candidate*, or *far*
Input:
metric: $\mathcal{P}$
the set of source point: $\mathcal{S}$
Output:
geodesic distance map: ϕ
Initialization:
$\forall p_i \in \mathcal{S}, \phi_i \leftarrow 0, L_i \leftarrow candidate$
$\forall p_i \notin \mathcal{S}, \phi_i \leftarrow \infty, L_i \leftarrow far$

Marching Loop:
While the candidate point set is non-empty, do
Find p_{min}, the candidate point which minimizes ϕ
$L(p_{min}) \leftarrow accepted$
for $p_k \in \mathrm{N}_M(p_{min})$ and $L(\mathrm{N}_M(p_{min})) \neq accepted$, **do**
Compute $\phi_{new}(p_k)$ by local geodesic distance update scheme (see text).
if $L(p_k) = far$
$L(p_k) \leftarrow candidate$
end if
if $L(p_k) = candidate$ and $\phi_{new}(p_k) < \phi(p_k)$
$\phi(p_k) = \phi_{new}(p_k)$
end if
end for

2.3.1 Isotropic Fast Marching Method

In their original minimal path model [19], Cohen and Kimmel used Sethian's isotropic Fast Marching Method to solve the Eikonal equation Eq.(5.5).

$$\begin{cases} \|\nabla\phi\| = \tilde{\mathcal{P}} \\ \phi(x_0, y_0) = 0 \end{cases} \tag{2.34}$$

where $\tilde{\mathcal{P}} = \mathcal{P} + w$ in Eq.(5.5).

To solve the Eikonal equation in Eq.(2.34) numerically, we need to discretize Eq.(2.34) first, and this is done by using the upwind scheme:

$$\left(\frac{\max\{(\phi_{i,j}-\phi_{i-1,j}),(\phi_{i,j}-\phi_{i+1,j}),0\}^2}{h_x}\right)+\left(\frac{\max\{(\phi_{i,j}-\phi_{i,j-1}),(\phi_{i,j}-\phi_{i,j+1}),0\}^2}{h_y}\right)=\tilde{\mathcal{P}}^2 \quad (2.35)$$

where $\phi_{i,j}$ is the distance of the *Candidate* (i,j) to be computed.

Let us consider the triangle formed by these three points: $\{(i-1,j),(i,j),(i,j-1)\}$. Four kinds of situations may happen:

1. both $(i-1,j)$ and $(i,j-1)$ are *Accepted*, $\phi_{i,j}$ is the largest real solution of

$$\left(\frac{\phi_{i,j}-\phi_{i-1,j}}{h_x}\right)^2+\left(\frac{\phi_{i,j}-\phi_{i,j-1}}{h_y}\right)=\tilde{\mathcal{P}_{i,j}}^2$$

2. neither $(i-1,j)$ nor $(i,j-1)$ are *Accepted*, $\phi_{i,j}=+\infty$

3. $(i-1,j)$ is *Accepted*, but not $(i-1,j)$,

$$\phi_{i,j}=\phi_{i-1,j}+h_x\tilde{\mathcal{P}_{i,j}}$$

4. (i,j) is *Accepted*, but not $(i,j-1)$,

$$\phi_{i,j}=\phi_{i,j-1}+h_y\tilde{\mathcal{P}_{i,j}}$$

For the point (i,j), there are four triangles, and from each triangle, a value $\phi_{i,j}$ will be obtained. We retain the smallest $\phi_{i,j}$ as the geodesic distance of the point (i,j). The geodesic distance map ϕ can be obtained with a computation complexity of $O(N(\log N))$.

An example of the isotropic progressive Fast Marching propagation on a synthetic image is shown in Figure.2.8.

2.3.2 Anisotropic Fast Marching Method

The fast marching methods proposed by Sethian [97] and Tsitsiklis [110] making use of the square formed stencil Figure.2.7 (a) could not be applied to compute the anisotropic metrics based minimal action maps Eq.(2.37).

$$\begin{cases}\|\nabla\phi\|_{T^{-1}}=1\\ \phi_{p_{s_0}}=0\end{cases} \quad (2.36)$$

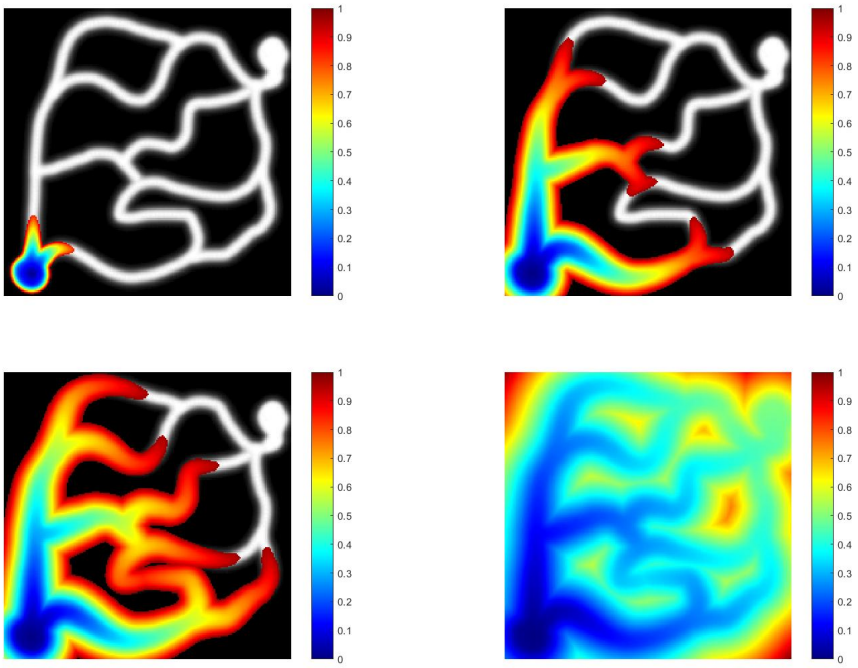


Fig. 2.8 Isotropic progressive front propagation

where T is the anisotropic metric and it is symmetric definite positive. It can be decomposed as follows:

$$T(\cdot) = \sum_{i=1}^{n} \lambda_i \cdot \mathbf{e}_i(\cdot)\mathbf{e}_i^T(\cdot) \tag{2.37}$$

where n is the dimension of the domain, $0 < \lambda_1 < \cdots < \lambda_n$ are the eigenvalues and $\mathbf{e}_i$ are the eigenvectors. And the velocity of the front along $\mathbf{e}_i$ is equal to $1/\sqrt{\lambda_i}$. The maximal anisotropy ration is defined as:

$$k = \frac{\max \sqrt{\lambda_n(\cdot)}}{\min \sqrt{\lambda_1(\cdot)}} \tag{2.38}$$

To handle with anisotropy, the authors in [100] proposed a single pass method which is based on an optimal trajectory problem rooted in control theory. The idea is still to update the grid points in a monotonic sequence by following the minimal path direction. The stencils shown in Figure.2.7 play an important role in the computation of the minimal action map ϕ by fast marching method. For the isotropic case, only *Accepted* neighboring points are considered for the update procedure. 4-connectivity stencil can give an accurate distance map. While the minimal path direction and the gradient direction of the minimal action map ϕ do not coincide in the anisotropic situation, a larger neighborhood is needed in the updating procedure to include the minimal path directions, shown as the figure in the r.h.s of Figure.2.7.

In order to solve the anisotropic Eikonal equation that is associated with an arbitrary continuous Riemannian metric and improve the stability and accuracy of the computation of the minimal action map with respect to a high anisotropic geodesic metric, Mirebeau [72] introduced an algorithm with a complexity of $O(N \ln N + N \ln k(M))$, where $k(M)$ is the anisotropy of the Riemannian metric M.

2.4 Heat Equation

In [23], Crane *et al*. proposed a different method for computing the geodesic distance on a Riemannian manifold. It is called the heat method and the detailed introduction will be presented in the next section. In this section, we would like first to discuss about the heat equation. In the following content, we will take the 1D heat equation as an example if there is no special instructions.

The heat equation is a partial differential equation (PDE) that describes the evolution of the distribution of heat on a domain Ω within time T. It has a general form:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{2.39}$$

where u stands for the heat, $\frac{\partial u}{\partial t}$ means the derivative of u, and α, a positive constant, represents the thermal conductivity, $\frac{\partial^2 u}{\partial x^2}$ denotes the second partial derivative of u in x.

The heat equation in n dimension is defined as follows:

$$\frac{\partial u}{\partial t} = \alpha \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2} \qquad \text{or} \qquad \frac{\partial u}{\partial t} = \alpha \Delta u \tag{2.40}$$

where Δ is the Laplace operator (or the Laplacian). The Laplace operator is a differential operator defined as the divergence of the gradient in Euclidean space, it is usually denoted by the symbols $\nabla \cdot \nabla$, ∇^2 or Δ. In differential geometry, the Laplace operator can be generalized to operate on Riemannian manifolds, and the more general operator is called Laplace-Beltrami operator.

2.4.1 Fundamental Solution

With respect to the initial condition $u_{p_0}(0) = \delta_{p_0}$, the solution to the heat equation is called the fundamental solution, or heat kernel. When the heat diffusion is used to approximate the geodesic distance, the desired heat distribution is in fact the fundamental solution of the corresponding heat equation, shown in Eq.(2.41).

$$\begin{cases} \frac{\partial u}{\partial t} = \alpha \Delta u, & \text{in} \qquad (0, \infty) \times \mathbb{R}^N \\ u(x, y, 0) = \delta(x, y) \end{cases} \tag{2.41}$$

Let us denote the fundamental solution Φ. In one dimension $N = 1$, the Green's function is a solution of the initial value problem:

$$\Phi(x,t) = \frac{1}{\sqrt{4\pi kt}} \exp(-\frac{x^2}{4kt}) \tag{2.42}$$

In several spatial variables, the $n-$variable fundamental solution is the product of the fundamental solutions in each variable:

$$\Phi(\mathbf{x},t) = \Phi(x_1,t)\Phi(x_2,t)...\Phi(x_n,t) = \frac{1}{\sqrt{(4\pi kt)^n}} \exp(-\frac{\mathbf{x}\cdot\mathbf{x}}{4kt}) \tag{2.43}$$

2.4.2 Numerical Schemes

Generally, the numerical solution to the heat equation is obtained using finite difference methods (FDMs), including a forward scheme or a backward scheme [87]. The FDMs solve heat equations by approximating them with difference equations, and finite differences approximate the derivatives.

Discretization of the heat equation

The continuous system of coordinates (t,x) can be replaced by a discrete grid $(n,m) = (n\Delta t, m\Delta x)$ (note that Δ here is not the Laplacian, but a small step or gap of time or location). The continuous function $u(t,x)$ under the discrete version can be rewritten as:

$$u_m^n = u(n\Delta t, m\Delta x) \tag{2.44}$$

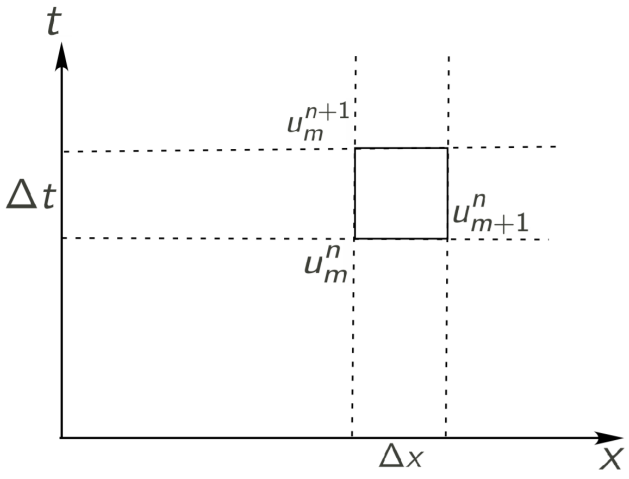


Fig. 2.9 The discretization of 1D heat equation

For a function $f(x)$, of which the derivatives to be approximated is properly-behaved, it can be expanded according to a Taylor Series expansion:

$$f(x_0+h) = f(x_0) + \frac{f'(x_0)}{1!}h + \frac{f^{(2)}(x_0)}{2!}h^2 + ... + \frac{f^{(n)}(x_0)}{n!}h^n + R_n(x) \tag{2.45}$$

By truncating the Taylor polynomial, the first derivative of $f(x_0)$ can be approximated:

$$f(x_0+h) = f(x_0) + f'(x_0)h + R_1(x) \tag{2.46}$$

Assuming that $R_1(x)$ is small enough, the approximation of $f'(x_0)$ is:

$$f'(x_0) \approx \frac{f(x_0+h) - f(x_0)}{h} \tag{2.47}$$

The first-order time derivative $\partial u/\partial t$ can be replaced by a forward finite difference in time:

$$D_t^+ u_m^n = \frac{u_m^{n+1} - u_m^n}{\Delta t} \approx u_t(t) \tag{2.48}$$

The second-order space derivative $u_{xx}(t,x)$ can be replaced by a central difference in space:

$$D_{xx}^0 u_m^n = \frac{u_{m+1} - 2u_m^n + u_{m-1}^n}{(\Delta x)^2} \approx u_{xx}(x) \tag{2.49}$$

Explicit Finite Differences

The explicit finite differences or forward scheme solves the differential equation in an iterative way. The forward difference is shown as follows:

$$D_t^+ u_m^n = \frac{u_m^{n+1} - u_m^n}{\Delta t} \approx u_t(t) + O(\Delta t) \tag{2.50}$$

For simplicity, we assume that the heat conductivity $\alpha = 1$, then the continuous heat equation $\partial u/\partial t = \alpha \Delta u$ can be discretized into:

$$\frac{u_m^{n+1} - u_m^n}{\Delta t} = \frac{u_{m+1}^n + u_{m-1}^n - 2u_m^n}{(\Delta x)^2} \tag{2.51}$$

or

$$u_m^{n+1} = u_m^n + \frac{\Delta t}{(\Delta x)^2}(u_{m+1}^n + u_{m-1}^n - 2u_m^n)$$

Let $r = \frac{\Delta t}{\Delta x^2}$, the above equation can be rearranged slightly as:

$$u_m^{n+1} = ru_{m+1}^n + (1-2r)u_m^n + ru_{m-1}^n$$

In the process of implementation, the time step Δt and the grid step Δx are set by users. The forward scheme is easy to implement because the values of u^{t+1} can be updated independently of each other. The solution could be obtained by two loops. The first one is an outer loop over time steps, and the other one is the inner loop over the space grid steps.

According to the Courant Friedrichs Lewy (CLF) condition [22], it is known that the explicit difference is numerically stable and convergent when the following inequality holds:

$$\frac{\Delta t}{\Delta x^2} \leqslant \frac{1}{2} \tag{2.52}$$

In addition, the numerical errors Δu are proportional to the time step and the square of the space step:

$$\Delta u \propto O(\Delta t) + O(\Delta x^2) \tag{2.53}$$

Implicit Finite Difference

The implicit finite difference or the backward finite difference can be defined in the same manner as the forward scheme:

$$D_t^- u_m^n = \frac{u_m^n - u_m^{n-1}}{\Delta t} \approx u_t(t) + O(\Delta t) \tag{2.54}$$

This continuous heat equation discretized using the backward scheme is shown as follows:

$$\frac{u_m^{n+1} - u_m^n}{\Delta t} = \frac{u_{m+1}^{n+1} + u_{m-1}^{n+1} - 2u_m^{n+1}}{(\Delta x)^2} \tag{2.55}$$

Let us rearrange Eq.(2.58) as follows:

$$-\frac{\alpha}{\Delta x^2}u_{m-1}^n + (\frac{1}{\Delta t} + \frac{2\alpha}{\Delta x^2})u_m^n - \frac{\alpha}{\Delta x^2}u_{m+1}^n = \frac{1}{\Delta t}u_m^{n-1} \tag{2.56}$$

Eq.(2.56) can be represented as:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & 0 & ... & ... & ... & 0 \\ 0 & 0 & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & a_N & b_N \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ ... \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ ... \\ d_{N-1} \\ d_N \end{bmatrix} \tag{2.57}$$

where the coefficients in the matrix are:

$$\begin{aligned} a_i &= -\tfrac{\alpha}{\Delta x^2} \qquad\qquad i = 2,3,...,N-1 \\ b_i &= (\tfrac{1}{\Delta t}) + (\tfrac{2\alpha}{\Delta x^2}) \\ c_i &= -\tfrac{\alpha}{\Delta x^2} \\ d_i &= (\tfrac{1}{\Delta t}) u_i^{n-1} \end{aligned}$$

Eq.(2.57) can also be simplified to:

$$(\mathrm{Id} - \Delta t A) u^{n+1} = u^n$$

Using the backward scheme to numerically solve the heat equation requires solving a system of equations at each time step. The matrix $\mathrm{Id} - \Delta t A$ is a tridiagonal matrix, and this equation can be solved by using the tridiagonal matrix algorithm (also called Thomas algorithm), and the solutions could be obtained in $O(n)$ rather than $O(n^3)$ required by Gaussian elimination. Compared with the forward scheme, the backward scheme has almost the same accuracy, the numerical error is also the same as the forward scheme Eq.(2.59). Furthermore, the main advantage of the backward scheme is that it is unconditionally stable for all time steps.

Crank Nicolson scheme

The Crank Nicolson scheme is implicit as the backward scheme. In the forward and backward scheme, we use the central difference at time $t = n$. If we use the central difference at $t = n + 1/2$, the heat equation is as follows:

$$\frac{u_m^{n+1} - u_m^n}{\Delta t} = \frac{1}{2}\left[\frac{u_{m+1}^{n+1} + u_{m-1}^{n+1} - 2u_m^{n+1}}{(\Delta x)^2} + \frac{u_{m+1}^n + u_{m-1}^n - 2u_m^n}{(\Delta x)^2}\right] \tag{2.58}$$

The system of the Crank Nicolson scheme is the same as the backward scheme Eq.(2.57), but with different coefficients:

$$
\begin{aligned}
&a_i = -\tfrac{\alpha}{2\Delta x^2} && i = 2,3,...,N-1\\
&b_i = (\tfrac{1}{\Delta t}) + (\tfrac{\alpha}{\Delta x^2})\\
&c_i = -\tfrac{\alpha}{2\Delta x^2}\\
&d_i = (\tfrac{1}{\Delta t})u_i^{n-1} - a_i u_{m-1}^{n-1} + (a_i + c_i)u_i n - 1 - c_i u_(m+1)^{n-1}
\end{aligned}
$$

The numerical errors Δu of the Crank Nicolson scheme are:

$$\Delta u \propto O(\Delta t^2) + O(\Delta x^2) \tag{2.59}$$

The computational model of these three schemes are displayed in Figure.2.10.

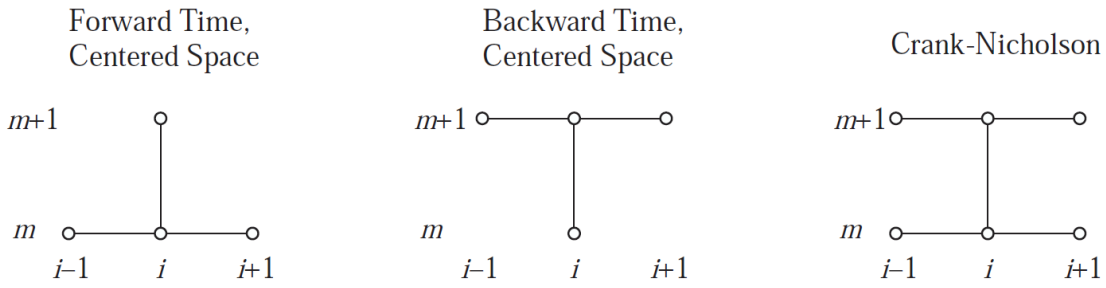


Fig. 2.10 The minimum calculation unit of different finite differences

2D Heat Equation

For the 2D heat equation $u_t = \alpha(u_{xx} + u_{yy})$, the forward scheme is:

$$\frac{u_{m,p}^{n+1} = u_{m,p}^n}{\Delta t} + \alpha(\frac{\Delta t}{\Delta x^2}(u_{m+1,p}^n - 2u_{m,p}^n + u_{m-1,p}^n) + \frac{\Delta t}{\Delta y^2}(u_{m,p+1}^n - 2u_{m,p}^n + u_{m,p-1}^n)) \tag{2.60}$$

The corresponding CFL condition is:

$$\alpha(\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2}) \leqslant \frac{1}{2} \tag{2.61}$$

Assume that $\Delta x = \Delta y$, then the CFL condition is:

$$\alpha(\frac{\Delta t}{\Delta x^2}) \leqslant \frac{1}{4} \tag{2.62}$$

In this thesis, we use a grid step $\Delta x = 1$ and the heat conductivity α is a scalar function, with $0 \leqslant \alpha \leqslant 1$. So when we use the forward scheme, the time step $\Delta t \leqslant 1/4$.

2.4.3 Maximum Principle

In mathematics, the maximum principle is a property of solutions to certain partial differential equations, of the elliptic and parabolic types. The maximum principle applies to the heat equation in domains bounded in space and time. Roughly speaking, it says that the maximum of any solution to the heat equation occurs either initially ($t = 0$), or on the boundary of that domain.

Specifically, the strong maximum principle says that if $u(x,t)$ achieves its maximum in the interior of the domain $\Omega \times [0,T]$, then u must be uniformly a constant. The weak maximum principle says that the maximum of the heat equation is to be found on the boundary, but may re-occur in the interior as well.

In this thesis, for Chapter.2 and Chapter.3, we use a forward scheme to solve the $2D$ heat equation; for Chapter.4, we use a new backward scheme proposed by Mirebeau *et al.* [35] to solve the $3D$ heat equation. Thanks to the development of CPUs and the algorithms for solving the inversion of large sparse matrix [76, 24, 16, 25], the sparse systems arising from elliptic PDEs can be solved in very close to linear time.

2.5 Heat Method

2.5.1 Heat Equation and Varadhan's Formula

In 1967, Varadhan [104] proposed a formula to approximate the geodesic distance $\phi(p_0, p_x)$ between two points p_0 and p_x on a Riemannian manifold:

$$\phi(p_0, p_x) = \lim_{t \to 0} \sqrt{-4t \log u_{p_0}(p_x, t)} \tag{2.63}$$

where u_{p_0} the solution of Eq.(2.39) under the initial condition that $u_{p_0}(0) = \delta_{p_0}$ within a small time $t \to 0$.

2.5.2 Crane's Heat Method

We thus see that the heat equation can be related to the geodesic distance. Recently, Crane *et al.* proposed a heat method to compute the geodesic distance on the given domain [23]. In spite of using the Varadhan's formula directly, the heat method proposed in [23] is based on solving a pair of standard linear elliptic problems. By comparing the heat method with the state-of-the-art Fast Marching Method [97], Crane *et al.* found that, using the heat method to obtain the geodesic distance is faster than the state-of-the-art Fast Marching Method. In

[95, 96], the authors prove that the sparse systems arising from the elliptic PDEs can be solved in very close to linear time.

The computation of geodesic distance ϕ based on the heat method can be divided into three steps:

1. Integrate the heat flow $u_t = \Delta u$ for a period of time t,
2. Compute the vector field $X = -\Delta u/|\Delta u|$,
3. Solve the Poisson equation $\Delta\phi = \nabla \cdot X$

Figure.2.11 shows the outline of the heat method:

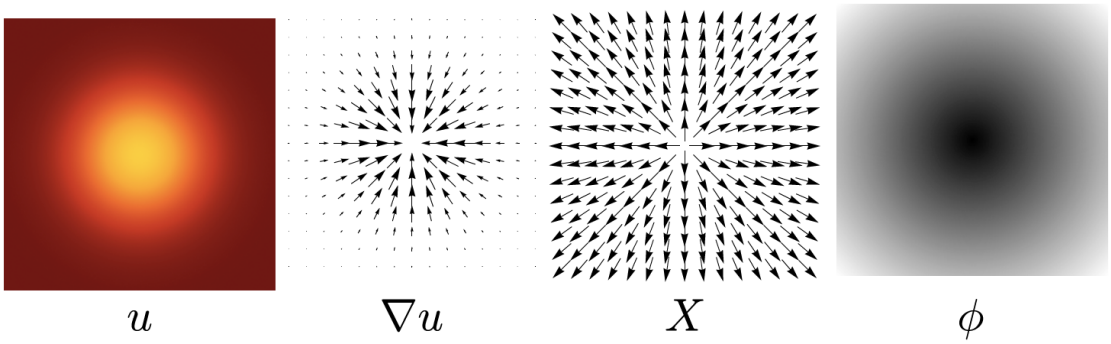


Fig. 2.11 The process of the heat method on a square domain, from left to right are: the heat value u after a period of time diffusion from the center point (source point), the gradient ∇u, the normalized gradient $-\frac{\nabla u}{|\nabla u|}$, the geodesic distance (in this case also the Euclidean distance) from every point on the domain to the center point, [23].

Compared with using Varadhan's formula directly, the heat method has several advantages: the Varadhan's formula is highly sensitive to the time for diffusion, a slight difference in time would result in quite different distance result. While the heat method avoids this by using the gradient of the heat instead of the heat itself. The gradient of heat points in the same direction as the heat flows, and the magnitude can be ignored here.

The discretization of the heat equation in [23] is a single backward Euler step.

The advantage of using the heat method is that the Laplace-Beltrami operator could be precomputed, so that the fundamental solution of the heat equation can be acquired in a single step no matter where the initial point p_{s_0} is, thus it saves a lot of time. While the Fast Marching Method does not reuse information: once the geodesic distance ϕ_{s_0} from the initial source point p_{s_0} is obtained, the distance from another source point p_{s_1} needs to be recomputed from scratch.

The key advantage of the heat method is that the linear systems in steps 1 and 3 can be precomputed. During their process of implementation, they use sparse Cholesky factorization. In addition, evidences show that the sparse systems from elliptic PDEs can be solved in a very short time that is close to linear time.

Compared with the fast marching method, the heat method provides results as good as by using the fast marching method. Moreover, the heat method is faster than the fast marching method.

In this way, the approximation of ϕ can be obtained once the heat equation is solved. Then the geodesic γ could be obtained by solving the ODE Eq.(2.10) depending on the heat diffusion being isotropic or anisotropic.

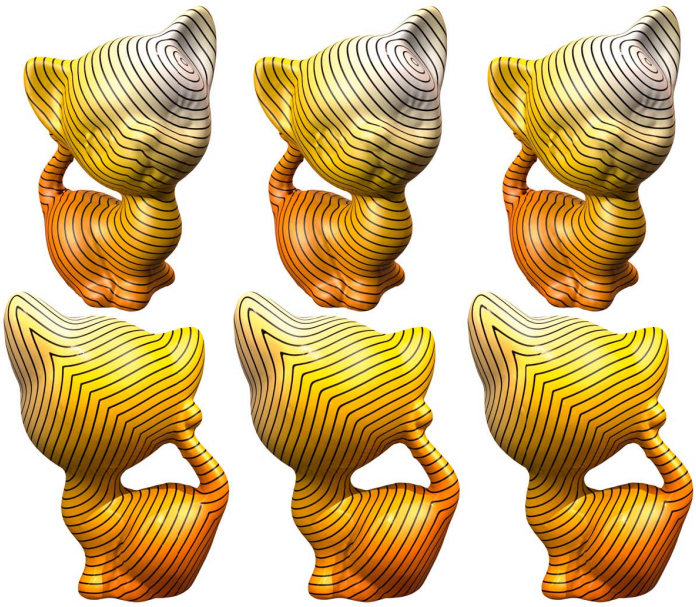


Fig. 2.12 Experiment on a mesh by the heat method, from left to right are the levelset of exact geodesic distance, the heat method distance and the fast marching distance [23].

2.6 Conclusions

In this chapter, we introduced the geodesic method in computer vision. We also introduced several state-of-the-art methods for segmentation in images, such as the active contour model, the minimal path model, the fast marching method and the heat method. In this thesis, we are interested in the heat method due to its advantages: effective, robust and time-saving.

Chapter 3

Geodesic Distance and Curves through Isotropic and Anisotropic Heat Equations on Images and Surfaces

Abstract

This chapter proposes a method to extract geodesic distance and geodesic curves using heat diffusion. The method is based on Varadhan's formula that helps to obtain a numerical approximation of geodesic distance according to metrics based on different heat flows. The heat equation can be utilized by regarding an image or a surface as a medium for heat diffusion and letting the user set at least one source point in the domain. Both isotropic and anisotropic diffusions are considered here to obtain geodesics according to their respective metrics. Our algorithms are tested on synthetic and real images as well as on a mesh. The results are very promising and demonstrate the robustness of the algorithms.

3.1 Introduction

As introduced in Chapter. 1, the heat equation is a partial differential equation that describes the evolution of the distribution of heat (or variation in temperature) in a given region over a certain period of time T. Generally, the form of the heat equation is as follows:

$$\frac{\partial u}{\partial t} - \alpha \Delta u = 0 \tag{3.1}$$

where α, a positive constant, stands for the thermal diffusivity and Δ represents the Laplace operator. In the physical problem of temperature variation, $u(x,t)$ represents the temperature.

More generally, $u(x,t)$ may represent the concentration of a certain substance, like water, whose quantity may vary with time t [78]. Eq.(3.2) gives a more general form of the heat equation.

$$\partial_t u(x,t) = k(x)\mathrm{div}(D \cdot \nabla u) \tag{3.2}$$

The coefficient $k(x)$ is the inverse of the specific heat of the substance multiplied by the density of the substance at that location [112]. In the case of a homogeneous isotropic medium, the matrix D has the form of a constant scalar times I_d, where the scalar represents the conductivity and where I_d is the identity matrix. If this scalar value is varying on the domain, we have a general medium. This case is called the isotropic case. In the anisotropic case, the coefficient matrix D has different eigenvalues. This means conductivity is not the same in different orientations.

The heat equation is widely used in many fields. For example, the HKS (Heat Kernel Signature), which is obtained by restricting the heat kernel to the temporal domain, is based on the properties of the heat diffusion process on a shape [48]. In [86], Raviv *et al* used the heat kernel to compute the diffusion distance for shape matching. As introduced in [118] a long time ago, the notion of Scale-Space is based on the Heat Equation. The non-linear heat diffusion can be also used for filtering problem. For example, in [81], Perona and Malik introduced an anisotropic diffusion approach to reduce image noise without removing prominent parts of the image content, and their non-linear diffusion filter only involves scalar diffusion coefficients. In [35, 73], Fehrenbach and Mirebeau proposed a non-negative numerical scheme called anisotropic diffusion using lattice basis reduction for image filtering and enhancing. It involves constructing the stencils whose geometry is tailored after the local diffusion tensor.

In this chapter, we are interested in the consequences of the work of Varadhan [104], where the author has proposed to approximate the geodesic distance $\phi(p_0, p_x)$ between two points p_0 and p_x on a Riemannian manifold by solving the following equation numerically:

$$\phi(p_0, p_x) = \lim_{t \to 0} \sqrt{-4t \log u_{p_0}(p_x, t)} \tag{3.3}$$

where u_{p_0} is the heat kernel of Eq.(3.1), that is, u_{p_0} is a solution with initial value $u_{p_0}(0) = \delta_{p_0}$. This was used recently in [23] to derive a numerical approximation of the geodesic distance, by solving the heat equation numerically with a small time step t. Diffusion is a process of motion of molecules (mass) moving from a place of high density to a place of low density. Based on this, Crane *et al* [23] proposed a method to extract geodesics on surfaces. Intuitively, one may regard the heat diffusion process as a large collection of hot particles moving from the source point p_0 and to the end points p_x, under the assumption that the domain is homogeneous with unit diffusivity $\alpha = 1$. The heat equation is solved based on standard differential operators. Compared with the state-of-the-art fast marching methods [97], using

heat diffusion to approximate the geodesic distance is computationally more efficient and has comparable accuracy and robustness [23] .

The geodesic distance and geodesic lines on images and surfaces play an important role in computer vision and graphics. They can be applied to vessel segmentation, road extraction, surface remeshing and so on [82]. Generally, the geodesic distance ϕ can be obtained by solving the Eikonal equation numerically using Dijkstra's method [27] or the Fast Marching Method [97, 19]. Once we get an approximation of the geodesic distance ϕ, the geodesic lines γ^* between source point p_0 and other points p_x on the domain are extracted by integrating an ordinary differential equation numerically [19, 82]:

$$\forall s > 0, \frac{\mathrm{d}\gamma^\star}{\mathrm{d}s} = -D^{-1}\nabla\phi, \gamma^\star(0) = p_x \tag{3.4}$$

where D is the metric tensor in the anisotropic case. For the isotropic case, $D = \alpha^2 I_d$, and Eq.(3.4) becomes $\frac{\mathrm{d}\gamma^\star}{\mathrm{d}s} = -\nabla\phi$.

The metrics used in this chapter can also be computed by other distance computation techniques such as the Fast Marching Method [97, 99, 72].

Using a heat method to approximate the geodesic distance has several advantages. The heat method is very fast [23] and also easy to implement. Furthermore, we show that different kinds of features can be extracted by using different diffusion models. Another advantage of this method is that it is not highly sensitive to noise. On the other hand, there are some disadvantages. The heat method is useful within a limited time period. After a long period of time, too much diffusion over the domain will cause blurring and make it hard to sort out the features of interest from all available features.

In this chapter, we go beyond the work of Crane *et al* [23] and introduce different heat flows to find the geodesic distance and lines. These flows can be either isotropic or anisotropic, depending on the needs. Note that all the diffusion models used in this chapter are linear. In order to extract the geodesic lines automatically, we introduce two new approaches based on geodesic voting and key point detection, which are inspired by [91–93] and [4, 45] but adapted to heat diffusion rather than Fast Marching.

This chapter is organized as follows: Section 3.2 presents isotropic and anisotropic heat diffusion models using different potentials and tensors including functions of the image gray values for conductivity, the Perona-Malik (P-M) model, and functions of the image Hessian for the diffusion tensor. We also explain the validity of Varadhan's formula in the cases we consider. Section 3.3 demonstrates the results of using different diffusion models to acquire geodesic distances and paths.

3.2 Varadhan's Formula and Heat Diffusion by Different Potentials and Tensors

Heat diffusion comprises 2 types: isotropic and anisotropic. The distinction is made by determining whether the diffusivity is a scalar or a matrix. When we consider heat diffusing on a $N \times N$ image, the initial condition would be:

$$\begin{cases} u_t - \alpha \Delta u = 0, (x,y) \in [0,N] \times [0,N], t \in [0,R^+] \\ u(x,y,t=0) = \delta_{(x_0,y_0)} \end{cases} \tag{3.5}$$

where α is a constant in the homogeneous domain and δ_{x_0,y_0} is the dirac distribution centered at $p_0 = (x_0, y_0)$. It should be noted that several source points can be used to diffuse simultaneously.

As described in Chapter 2, in Crane *et al*'s method [23], there are three steps to get the geodesic distance ϕ on a surface: 1) compute the heat density: $\partial_t u = \alpha \triangle u$, α is a constant on the whole domain; 2) normalize the gradient: $X = \frac{\nabla u}{|\nabla u|}$; 3) solve the Poisson equation: $\triangle \phi = \text{div}(X)$ to get the distance ϕ. Crane *et al*'s method shows the correlation between the heat density u and the geodesic distance map ϕ. In this chapter, we solve the isotropic (or anisotropic) heat equation to get the heat distribution on images where the heat will flow along the direction of a geodesic. Then we apply Eq. (3.3) directly to get the geodesic distance ϕ.

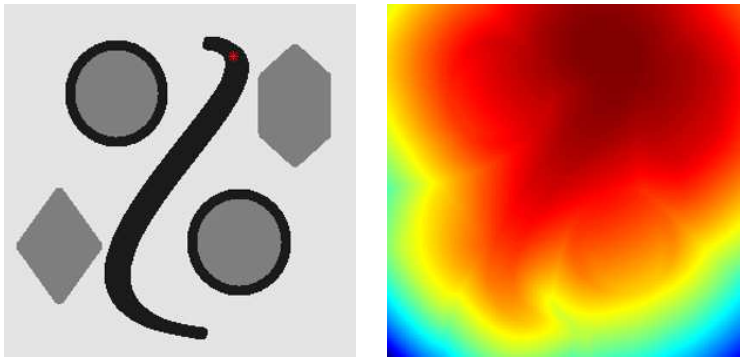
Next, a geodesic curve $\gamma^\star$ between the source point p_0 and another point p_x in the domain can be computed by gradient descent [19, 5], by using $\frac{\mathrm{d}\gamma^\star}{\mathrm{d}s} = -\nabla\phi$. This backtracking becomes Eq. (3.4) in the general anisotropic case.

3.2.1 Isotropic Diffusion

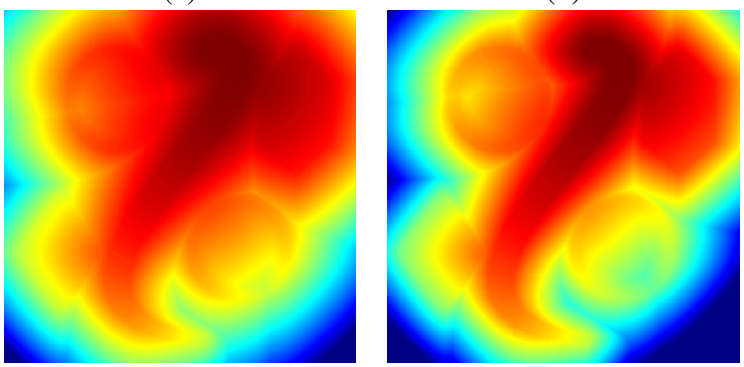
Four situations of isotropic heat diffusion are discussed here: 1) conductivity based on a function of the gray level, 2) P-M model [81] (although Perona and Malik claimed that their model is anisotropic, we still consider it isotropic following [49], since they use a scalar diffusivity and not a tensor), 3) the combination of conductivity and the P-M model, 4) a P-M model using the norm of gradient of the image as the feature.

Conductivity

In Eq.(3.1), α represents thermal diffusivity. In the homogeneous case, α is a constant. Putting the source point in the center of the image, heat will diffuse across concentric circles and geodesics will be straight lines orthogonal to these circles. We are motivated by finding geodesic paths that follow related gray-level values, and therefore use a conductivity which is



(a) (b)



(c) (d)

Fig. 3.1 The example of how heat propagates on a synthetic image and the effect of the power n in Eq. (3.6), (a) is the original 300×300 image composed of several parts including a wide black curve in the middle and its surroundings, the red point which is the source-point $(187, 36)$manually set on top of the curve from which heat can diffuse, (b) is the heat distribution for $n = 1$ in Eq. (3.6), (c) is the result for $n = 2$ in (3.6), and (d) is the result for $n = 3$ in Eq. (3.6). These results are generated in the same period of time T.

a function of the gray-level at each pixel. Given a source point p_0 on an image I, which takes values between 0 and 1, the conductivity of p_0 equals to 1. Point p_x has a higher conductivity α when the gray-value difference between p_0 and p_x is small as shown in Eq. (3.6).

$$\alpha_{p_x} = |(1 - |I(p_0) - I(p_x)|)|^n + \varepsilon \tag{3.6}$$

$n = 1, 2, 3...$, ε is a small positive constant that prevents α from vanishing. The value of n depends on the contrast between the interesting features in the image and the image background. In other words, if there exist fewer differences between the interesting features and their background, we can set a higher n. From Fig.3.1, it can be clearly seen that the black wide curve (the part to be enhanced) is the most visible in (d), compared with the other two results (b) and (c).

Perona-Malik (P-M) model

As stated above, the P-M model [81] is not a real anisotropic model because D used in (3.7) is a scalar and not a tensor.

$$\begin{cases} \frac{\partial u}{\partial t} = \operatorname{div}(D\nabla u) \\ u(x,y,t=0) = \delta_{(x_0,y_0)} \end{cases} \tag{3.7}$$

There are two forms of D usually used, both being positive decreasing functions of the gradient, which are given by

$$D = e^{-(\|\nabla I\|/K)^2} \quad \text{or} \quad D = \frac{1}{1+(\|\nabla I\|/K)^2} \tag{3.8}$$

where K is the contrast parameter and $\|\nabla I\|$ is the norm of the gradient of the image. Diffusion processes in Sect.3.2.1 tend to equilibrate the concentration differences in the materials, while the feature (Eq.(3.8)) used in the P-M model can constrain the diffusion process inside the homogeneous regions. From Eq.(3.8) we can see that wherever there is higher gradient there is lower diffusivity, which indicates that P-M model inhibits heat from leaking outside a homogeneous region. Note that since the goal here is different from the usual P-M equation where the heat density is the image I itself, the initial value of u here is different from the usual case. We use a Dirac distribution as the initial density. Eq.(3.7) can be also written as Eq.(3.9)

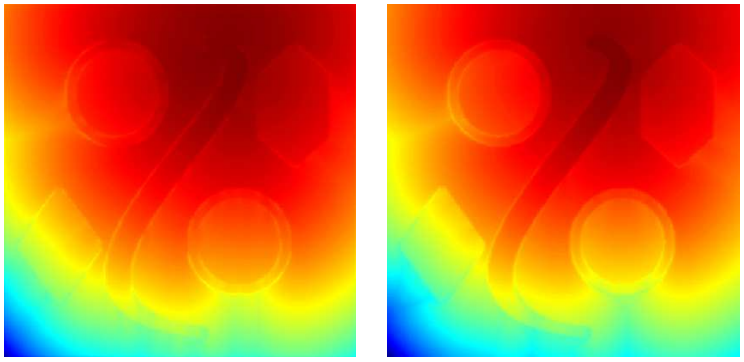
$$\frac{\partial u}{\partial t} = D\Delta u + \nabla D \cdot \nabla u \tag{3.9}$$

Compared with Eq. (3.1), there exists an additional first derivative term in Eq. (3.9): $\nabla D \cdot \nabla u$. Fig.3.2 shows the experimental result on a synthetic image. From this result, it can be seen that there is a difference between (a), the form the result takes without adding the item, and (b), the result with the item considered. Compared to (b), we can see that (a) has more heat around the edges, which shows that adding this item $\nabla D \cdot \nabla u$ helps to restrain the heat from leaking out of a region slightly.

Combination of Conductivity and P-M Model

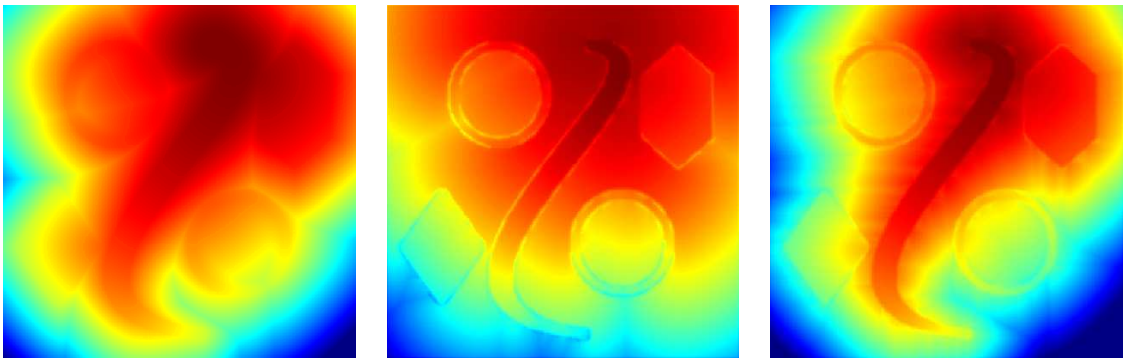
As we can see above, both methods have their own advantages. Using conductivity is direct and useful in simple scenes, but it becomes insufficient when dealing with more complicated scenes. The P-M method helps to weaken heat diffusion on the edges and boundaries. Thus it can be used as an auxiliary factor. This is the reason why we combine the two methods together:

$$\frac{\partial u}{\partial t} = \alpha \cdot \operatorname{div}(D\nabla u) \tag{3.10}$$



(a) (b)

Fig. 3.2 Example for showing effect of the item $\nabla D \cdot \nabla u$, (a) is the result without the item, and (b) is the result with the item.



(a) (b) (c)

Fig. 3.3 Example on the same synthetic image showing the combination of conductivity and P-M diffusivity. A source point settles on top of the curve. After the same period of time, (a) is the result of using the cubic form of Eq. (3.6), (b) corresponds to the PM model Eq. (3.8), (c) is the result of using the combination of conductivity and P-M model Eq. (3.10).

Here, α is defined by Eq. (3.6). Thus diffusion depends on both region and edge based features. The advantage can be seen in Fig.3.3 where the heat becomes more concentrated along the central curve as a result of this combination.

Another advantage of using this combination is that it can get the centerline automatically. According to Eq. (3.3), it is indicated that wherever there is a larger heat density there is a smaller distance between the points on the image and the source point p_0. By using Eq. (3.10), heat will be mostly concentrated in the center of the region containing the source point. Fig.3.8 gives an example of centerline extraction in a vessel image.

A P-M Model that Follows the Edges

In fact, besides the features in Eq. (3.8), there are other features that can also be used in the P-M model. Contrary to the features in Eq. (3.8) that keep the heat inside a region, we

propose to define features that help heat focus on the edges or boundaries by enhancing the potential on the edges:

$$\begin{cases} \partial_t u = \operatorname{div}(D\nabla u) \\ D = \|\nabla I\|^2 + \varepsilon \end{cases} \tag{3.11}$$

In this model, heat diffuses faster wherever the gradient gets higher such as on an edge or across thin structures. In order to keep the diffusion coefficient strictly positive, we add a small positive constant ε to $\|\nabla I\|$. Examples are shown in Fig.3.4. In the row above, without enhancing $\|\nabla I\|$, the very thin line can hardly be taken into account, as shown in (b), while in (c), the heat travels mostly along the thin curve in the middle, using Eq. (3.11). In addition, a much wider line of interest and its background with several polygons is shown in (d). Both (e) and (f) are the results of diffusion using Eq. (3.11) in P-M model where the diffusion starts from two different positions of the source point. When the heat starts diffusing from the red point, it almost goes along the boundary of the hexagon and then heat diffuses to the curve. The heat also concentrates on the boundary of the curve. The same phenomenon can be seen in (f) where heat starts diffusing in the center of the curve and then goes along the double edge. Thus Eq. (3.11) is good at extracting features such as edges and boundaries.

3.2.2 Anisotropic Diffusion

Anisotropic diffusion has a form as follows:

$$u_t = \operatorname{div}(D\nabla u) \tag{3.12}$$

where D is a diffusion tensor rather than a scalar. It is a tensor field of symmetric positive matrices that encodes the local orientation and anisotropy of an image. This anisotropic diffusion makes heat propagate in the direction that we design [5, 47] by defining the relevant tensor D. Weickert [115] proposed a coherence enhancing diffusion method, using a nonlinear anisotropic diffusion equation for filtering problems. A symmetric and positive definite diffusion tensor is used in this method. It is obtained by the tensor product of ∇I: $J_\rho(\nabla I_\sigma) := K_\rho * (\nabla I_\sigma \otimes \nabla I_\sigma)$, where K_ρ is a Gaussian kernel. In [115], the eigenvectors are the same as in $J_\rho(\nabla I_\sigma)$ and the eigenvalues are chosen to make diffusion act mainly along the direction with the highest coherence. An improved structure tensor is proposed in [56] to get an integrated edge and junction detection method. The structure tensor is calculated by means of Gaussian derivative filters of the image I. The authors proposed multiple ways to improve the structure tensor including using a higher sampling rate, improving corner localization etc. Generally, the gradient ∇I of the image is usually taken into account to measure the local direction of edges or texture [82]. In [5], the authors proposed an interactive vessel segmentation method to extract the centerlines as well as the boundaries of the vessels. In this method, they defined

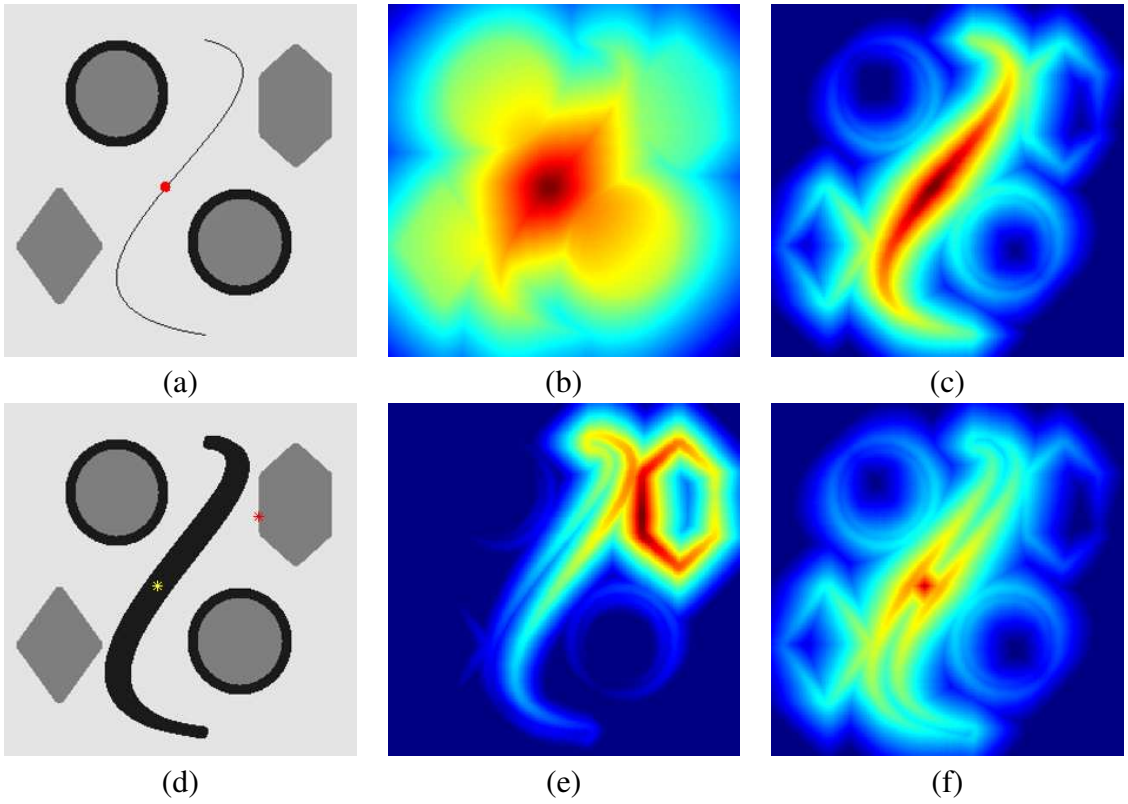


Fig. 3.4 Example on a synthetic image to illustrate how Eq. (3.11) works. The source point is placed on the very thin line in (a) and a wider curve in (d), the red point in (a) is the source point for (b) and (c). (b) is the result of diffusing on (a) by a method discussed in (3.6), (c) is the diffusion result of (a) using (3.11). The red point on (d) indicates the position of the source point for (e) and the yellow point indicates the position of the source point for (f). Both (e) and (f) are the diffusion results of (d) generated by using $|\nabla I|^2$ as the feature in the P-M model.

the metric using the eigenvectors and eigenvalues obtained from OOF (optimally oriented flux) [58]. This metric is oriented along the estimated direction of the vessel, allowing a higher velocity on the centerline and with an estimate of the vessel local radius. Since we do not segment the boundaries in this chapter, we just use the Hessian matrix to construct the metric.

Eigenvalues and Eigenvectors

The tensor field can be diagonalized as in [82]:

$$D(x) = \lambda_1(x)e_1(x)e_1(x)^T + \lambda_2(x)e_2(x)e_2(x)^T \tag{3.13}$$

The normalized vector fields $e_i(x)$ are orthogonal eigenvectors of the symmetric matrix $D(x)$, and the $\lambda_i(x)$ are the corresponding eigenvalues, with $0 < \lambda_1(x) \leq \lambda_2(x)$. Following [82],

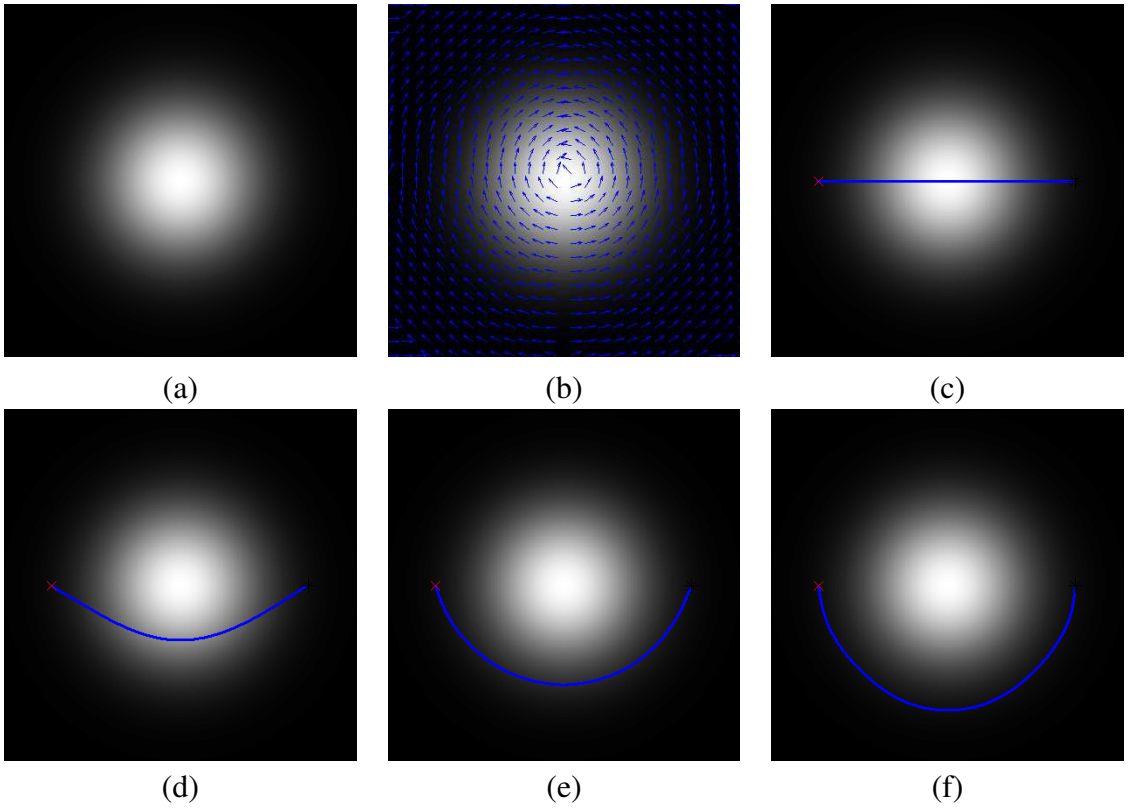


Fig. 3.5 (a) a Gaussian image, (b) tensor field by gradient, (c) to (f) are the shortest paths between the two user-chosen points, the corresponding anisotropies are 0, 0.5, 0.8 and 0.99

the anisotropy $\mu(x)$ is defined as:

$$\mu(x) = \frac{\lambda_2(x) - \lambda_1(x)}{\lambda_2(x) + \lambda_1(x)} \tag{3.14}$$

When $\lambda_1(x) = \lambda_2(x)$, the anisotropy $\mu(x)$ is 0, and the tensor is in fact a scalar metric which makes geodesics the shortest paths according to the isotropically weighted distance.

Anisotropic Diffusion Tensor

When $\lambda_1 \neq \lambda_2$, it is anisotropic. As mentioned before, $\lambda_2 \geq \lambda_1$, λ_2 controls the direction of the heat flow. When λ_2 is far larger than λ_1, the heat flows in the direction of e_2 while only a little goes into the orthogonal direction. Fig.3.5 depicts the effect of the change of anisotropy in detecting a shortest path. We define the gradient direction of the image as e_1, which is the radial direction, and its orthogonal direction as e_2 (in fact, the eigenvectors are the same as those used in [115]). When $\lambda_1 = \lambda_2$, the heat diffusion begets the Euclidean distance map, and the shortest path is a straight line. As the anisotropy grows, the direction of heat flow travels more along the tangent direction and the geodesic lines get closer and closer to a half circle.

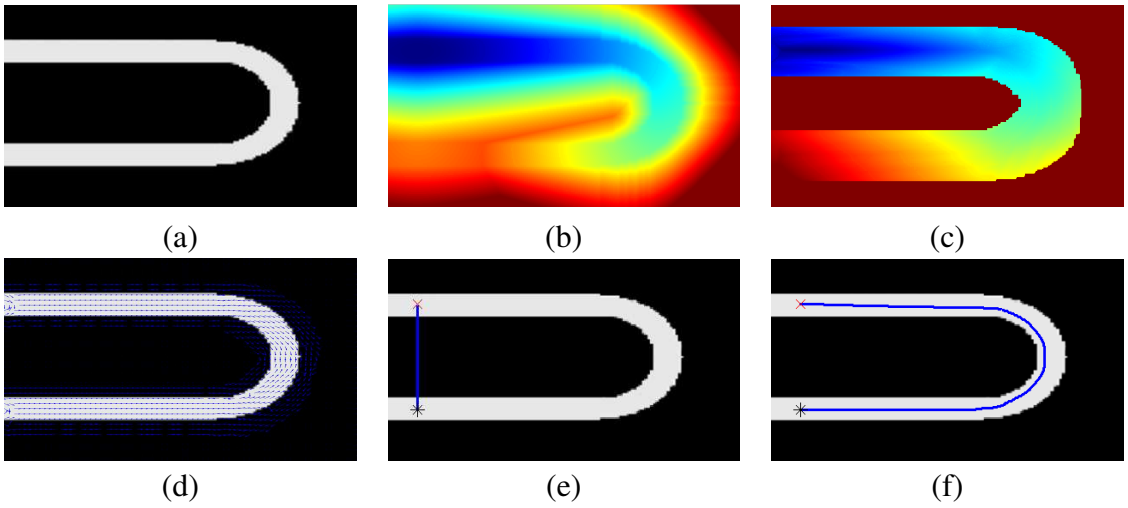


Fig. 3.6 Experiment on a U-tube structure: (a) original image, (d) tensor field, (b) and (c) are the distance maps obtained by isotropic and anisotropic diffusions respectively, (e) and (f) are the corresponding geodesic lines.

In [38], the authors introduced a multi-scale vessel enhancement method by interpreting geometrically the eigenvalues of the Hessian matrix. Using the Hessian eigenvectors, the local orientation of the image can be estimated, allowing to find out where there are tubular structures like vessels. Here we use a fixed Gaussian kernel and compute the 2D Hessian matrix, then compute the eigenvalues and eigenvectors. We use the eigenvectors as a tensor field and control the anisotropy.

Fig.5.3 shows the isotropic and anisotropic heat diffusions respectively on a U-tube image. The U-tube image is given by (a). (b) is the distance map obtained by using conductivity Eq. (3.6), $n = 3$. (c) is the distance map obtained by using anisotropic diffusion. (d) shows the tensor field that is used in the anisotropic diffusion, and it is obtained by Hessian matrix. (e) is the geodesic line obtained by backtracking in (b), we can see that the geodesic line takes a shortcut. When the heat diffuses by taking the local orientation into account, this shortcut is avoided. (f) shows geodesic line by backtracking in (c). As seen, the heat travels along the tensor field, and by backtracking, the line is exactly located on the tube in the correct direction.

3.2.3 Heat Diffusion on Meshes

The heat equation on meshes is similar to the one on images though the heat is transferred from one vertex to another, not from pixel to pixel. We introduce a numerical method to solve the heat equation on triangle meshes. The mesh is composed of faces $\{f_m\}_{1\leq m\leq M}$ and vertices $\{v_n\}_{1\leq n\leq N}$ where M and N are the numbers of faces and vertices respectively.

First, we need to compute the cotangent Laplacian W of the mesh. It can be obtained by using Eq. (3.15) [83, 89]:

$$W_{i,j} = \cot(\alpha_{i,j}) + \cot(\beta_{i,j}) \tag{3.15}$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ are the two angles opposite to the edge (v_i, v_j), which connect two vertices. Next, we need to compute the symmetric Laplacian matrix $L = D - W$, where $D = \mathrm{diag}_i(\sum_j W_{i,j})$. At last, we get the normalized operators $\tilde{W} = D^{-1}W$ and the Laplace operator is $\tilde{L} = D^{-1}L$.

The heat diffusion on a mesh solves: $u_t = -\tilde{L}u$. Here the conductivity of the domain is assumed constant. When it comes to enhancing other special features, for example, the curvature in order to extract the edges on mesh, or the texture on the surface in order to find characteristic lines on the surface, we incorporate these features into the heat equation. It yields:

$$\frac{\partial u}{\partial t} = -\tilde{L}(u * P) \tag{3.16}$$

where P plays a role similar to heat conductivity α in (3.6) and determines the evolution and distribution of heat on the surface.

3.2.4 The Applicability of Varadhan's Formula

As explained in the introduction, the main relation between heat diffusion and distance maps comes from the Varadhan's formula. In this section, we give a closer look to this formula and its extensions.

In [21], the authors introduced approximate solutions for the Green function of uniformly parabolic second-order operators with variables, by using expansions. Let L be the general Laplacian operator which comprises the second and first order terms:

$$Lu := \sum_{i,j}^{n} a_{i,j}(x)\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{j}^{n} b_j(x)\frac{\partial u}{\partial x_j} + c(x)u \tag{3.17}$$

Our heat equation with the boundary condition becomes:

$$\begin{cases} \partial_t u - Lu = 0, & \text{in}(0,\infty)\times\mathbb{R}^N \\ u(x,0) = \delta_{x_0}, & \text{on}\mathbb{R}^N \end{cases} \tag{3.18}$$

According to [106, 113, 21], the operator L can be interpreted as a Laplace-Beltrami operator on a manifold with lower order terms, thus, we can obtain the Green function

(or fundamental solution, heat kernel) $\mathcal{G}_t(x,y)$ of Eq. (3.18) represented by an asymptotic expansion of the form:

$$\mathcal{G}(x,y,t) = \frac{e^{-\frac{\phi(x,y)^2}{4t}}}{(4\pi t)^{N/2}}(\sum_{k=0}^{\infty}\mathcal{G}^{(k)}(x,y)) \tag{3.19}$$

as $t \to 0^+$, where $\phi(x,y)$ is the geodesic distance induced by a Riemannian metric derived from the coefficients $\{a_{i,j}\}$ between points x and y, $\mathcal{G}^{(k)}(x,y)$ are smooth functions.

The fundamental solution of Eq. (3.19) satisfies the formula introduced by Varadhan [104]:

$$\lim_{t\to 0}[-4t\log u_x(y,t)] = \phi^2(x,y) \tag{3.20}$$

where here $u_x(y,t) = \mathcal{G}(x,y,t)$. For example, this formula is well understood in the case of the homogeneous heat equation posed on the whole domain $\mathbb{R}^2$. Then, the Green function is in fact a Gaussian function and the explicit solution is

$$u_x(y,t) = (2\pi t)^{-k/2}\exp\{-\frac{1}{4t}\|x-y\|^2\} \tag{3.21}$$

It can be easily seen that this function satisfies formula Eq. (3.20).

In this chapter, we introduce three kinds of heat flows.

1. The heat equation for the conductivity case is obtained from Eq. (3.18),

   taking $a_{i,j}(x) = \alpha_{p_x}\delta_{i,j}$, where α_{p_x} is defined in Eq. (3.6) and $\delta_{i,j}$ is the kronecker symbol, equal to 1 or 0 depending on i equal to or different from j. And we have $b_j(x) \equiv 0$, $c(x) \equiv 0$. Therefore, the distance ϕ is a weighted distance. The geodesic paths correspond to minimal paths relative to isotropic potential equal to the conductivity α_{p_x}.

2. For the P-M case, of section 3.2.1, we have $a_{i,j}(x) = D(x)\delta_{i,j}$ and $(b_j(x))_j = \nabla D$, $c \equiv 0$ in Eq. (3.18).

3. For the anisotropic case, of section 3.2.2, we can take, by developping $\mathrm{div}(D\nabla u)$ from Eq. 3.12, $a_{i,j}(x) = D_{i,j}(x)$, $b_j(x) = div((D_{i,j}(x))_i)$ and $c(x) \equiv 0$. Since for the general anisotropic case the distance ϕ derives from a Riemanian metric, the orientation of the geodesic lines have to agree as much as possible with the eigenvectors of the metric.

In all cases above, Varadhan's formula means that the corresponding heat flow allows to find an estimate for the distance map according to the Riemannian metric derived from the coefficients $\{a_{i,j}\}$.

3.3 Experiments and Analysis of Different Heat Flows

3.3.1 Experiments Data and Settings

We test the different heat diffusion models on several datasets:

Isotropic diffusion on real images. In the experiment of road extraction (Fig.3.7), the data is an optical remote sensing image and is resized to 300×300. In the experiment of vessel extraction (Fig.3.8), the image size is 512×512. The conductivity is given by Eq. (3.6), and $n = 3$, the coefficient K in Eq. (3.8) is 0.03.

Isotropic diffusion on a noisy image. Fig.3.9 is an example on a noisy image (a), with a given percentage of corrupted pixels, ξ equals to 0.133.

Anisotropic diffusion on a synthetic images. In the spiral experiment in Fig.3.10, we use the Hessian matrix to define the diffusion tensor and the anisotropy is set to 0.9.

Isotropic diffusion on a mesh. The block of fig.3.11 has 57184 faces and 25894 vertices. Curvature in the mesh was computed following the method in [20] and [2].

In addition, it should be guaranteed that the heat has spread all over the domain (i.e. the image), where the structures to be extracted are all included. On the other hand, the heat is not supposed to propagate for a long time because the image will eventually get blurred. In order to achieve that, we must manage to set an iteration number in accordance with the problem data (size of the image, position of the source point) while managing control of the heat flows. Therefore, the diffusion has to take place during a limited and short period of time. In addition, to make sure that the values are computed within a reasonable number of time iterations, the time step of each iteration τ in $\frac{\partial u}{\partial t} - \tau \cdot \alpha \Delta u = 0$ is set to $\tau = 0.2$, which satisfies the CFL condition [22].

3.3.2 Results and Analysis

To evaluate the performance of the centerline extracted by different methods, we compute the *precision* and *recall* criteria given by the following formula:

$$\begin{cases} recall = \frac{TP}{TP+FN} \\ precision = \frac{TP}{TP+FP} \end{cases} \tag{3.22}$$

where TP is the length of extracted centerline that matches the manually labeled ground truth, FP represents the length of extracted centerline which are not on the ground truth, and FN is the length of the ground truth but that is not extracted.

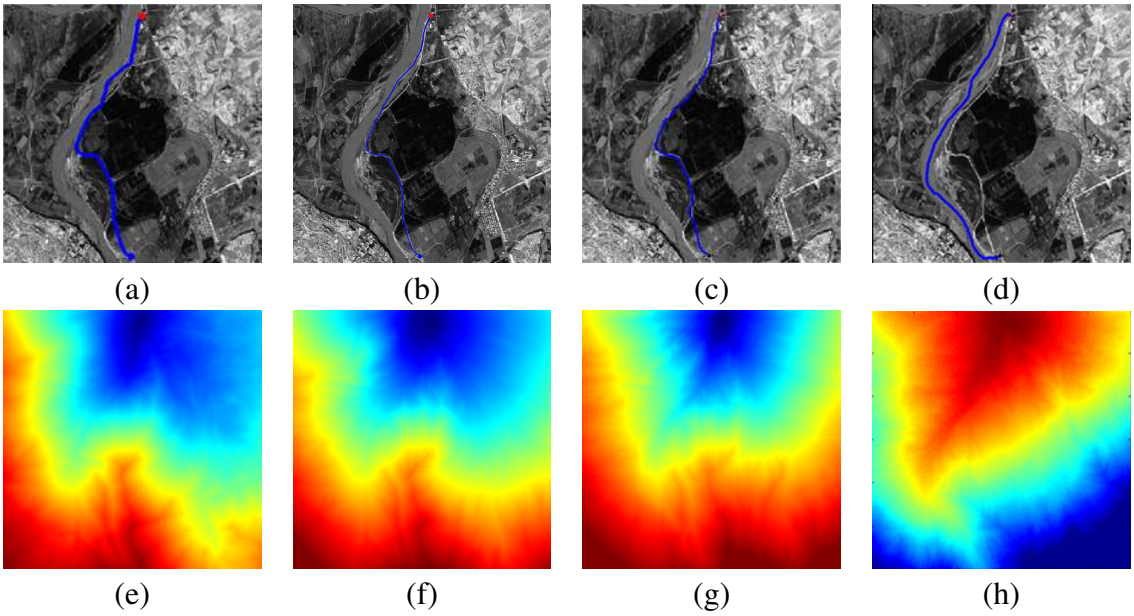


Fig. 3.7 Experiment on the image of a road: the top row from left to right displays the corresponding path extracted by using Fast Marching, the heat method by using respectively Eq. (3.6) as conductivity, $n = 2$ and $\epsilon = 0$ Eq. (3.11) and Eq. (3.8) in the P-M model, the bottom row displays the distance maps by using these methods respectively.

Isotropic diffusion on real images.

In Fig.3.7, the source point is set on the top of the very thin white line (which is a road)with the endpoint at the bottom. The blue curves in (a) to (d) are the paths extracted by backtracking from the end point to the source point. It is based on the distance maps obtained respectively by using isotropic Fast Marching, by using the heat method with conductivity Eq. (3.1), by using P-M diffusion model with Eq. (3.11) and by using P-M model using Eq. (3.8) as shown from (e) to (h). In (a) and (b), we use the same metric for the Fast Marching Method and heat diffusion.

From Table 5.1, the road extracted in (a) and (b) has a similar recall and precision. But the road extracted in (b) is much smoother than the one from Fast Marching (a). In addition, the P-M model based on Eq. (3.11) is good at extracting thin structures, see (c). Further, in (d), by using Eq. (3.8) in the P-M diffusion model, the more homogeneous parts can be easily distinguished. This indicates that the heat diffusion by the use of Eq. (3.8) in the P-M model is likely to present good results when there is a relatively larger part to be extracted. Moreover, heat in such a case is easily diffused in places where a few changes exist.

The experiment of vessels is shown in Fig.3.8, in which the source point (marked as red cross) and end points (marked as black) are given by the user. By using isotropic Fast Marching (a), the extracted lines do not exactly follow the centerline, especially in the center. By using the isotropic linear heat diffusion Eq. (3.1) with Eq. (3.6) as the conductivity, due

to directionally independent heat scattering, the centerline of the vessels is not completely right. (c) is the result that combines the P-M model with the conductivity Eq. (3.10), using Eq. (3.8) in P-M model, the heat is more concentrated in a homogeneous area where the source point stays. From (d), we can see that the centerlines are well extracted despite the position of the manually set source point, which means that the point of most concentrated heat moves to the centerline in the process of diffusion. In this test, compared to the center line ground truth, it can be seen in Table.5.1 that Eq. (3.10) is effective in extracting the center line of the vascular-like structure.

Table 3.1 (the indexes of evaluation%).

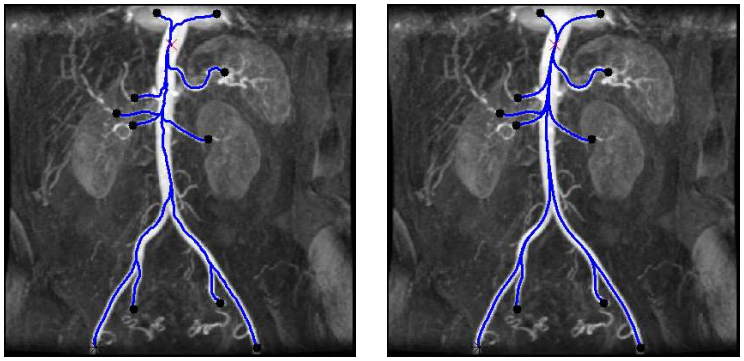
data	method	recall	precision
Road	Fast Marching	79.59	72.87
Fig.3.7	Heat Diffusion(3.6)	76.32	71.00
	Heat Diffusion(3.11)	98.11	91.00
Vessels	Fast Marching	84.73	63.68
Fig.3.8	Heat Diffusion(3.6)	91.78	67.01
	Heat Diffusion(3.10)	92.26	70.09

Isotropic diffusion on a noisy image

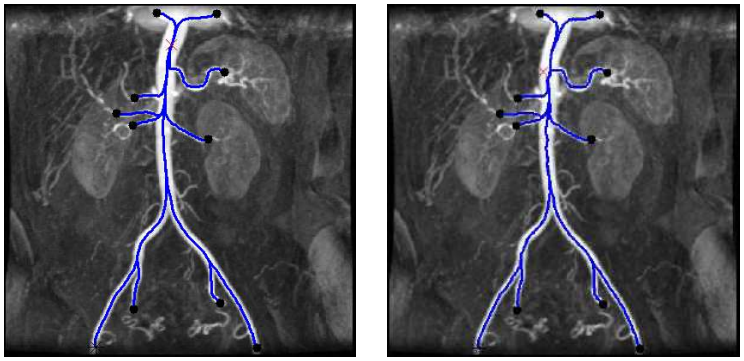
The diffusive nature of the heat equation causes instant smoothing. Even if there is a temperature discontinuity at initial time $t = t_0$, the temperature becomes smooth as soon as $t > t_0$. Solutions of the heat equation are characterized by a gradual smoothing process from the initial temperature distribution by the flow of heat from warmer to colder areas of an object, and this can be considered as a blurring process. This is why the heat equation is used for filtering problems. And also in our case, the geodesic curves that are extracted are not so affected by noise. Fig.3.9 is an example on a noisy image (a), with a percentage 0.133 of corrupted pixels. Given a source point and an end point on both ends of the black curve, the red lines on (b) and (c) are obtained by the isotropic Fast Marching Method and heat method using P-M model respectively. The potential used in Fast Marching and the scalar used in P-M model are the same. Both are the norm of gradient of the image. From the result we can see that the heat method gives a better result than Fast Marching despite the noise, which indicates that the heat method is less sensitive to noise.

Anisotropic diffusion on a synthetic images

Fig.3.10 illustrates an experiment using anisotropic heat diffusion. The results are as follows: (a) is the original spiral image, (d) is its corresponding tensor field by using the Hessian matrix. (b) is the distance map obtained via the isotropic heat diffusion (Eq. (3.1) as



(a) (b)



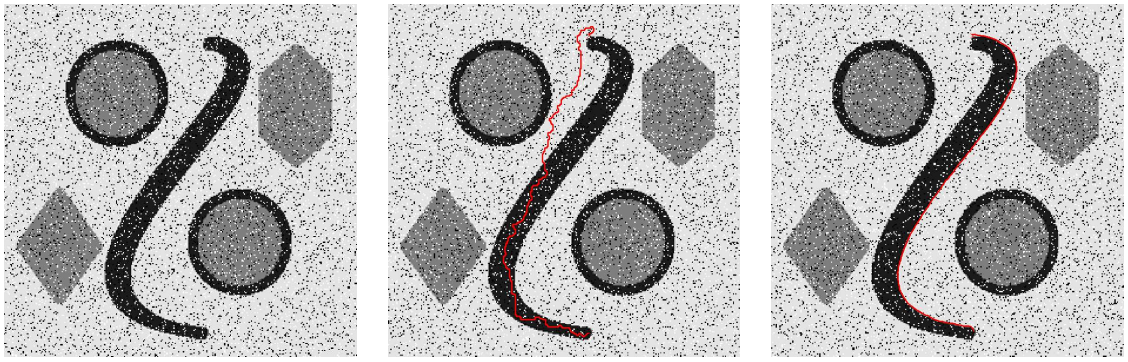
(c) (d)

Fig. 3.8 Experiment on real vessel image: the red cross is the manually set source point, the black spots are the end points provided by the user, and the blue curves are the extracted geodesics (a) is the result by isotropic Fast Marching (b) is the result by only using the conductivity (Eq. 3.6), (c) and (d) are the results by using the combination (Eq. 3.10), but with different source points.

the diffusion model, and Eq. (3.6) as the conductivity with $n = 3$) and (c) is the distance map obtained via the anisotropic heat diffusion, (e) and (f) are the extracted lines. From the results, we can see that in (b) and (e), using isotropic diffusion, the temperature blurs in the process of diffusion, and the path extracted takes the shortcut from the end point to the source point, while in (c) and (f), using anisotropic diffusion, the path backtracks along the spiral line, and the heat diffuses predominantly along the spiral apparently.

Isotropic diffusion on a mesh

To illustrate the method of section 3.2.3, Fig.3.11 uses the curvature of the block as diffusivity. Furthermore, the higher the curvature at a point, the higher the probability that at this point the amount of heat received is larger than at the neighboring points: (a) is the original block structure; (b) is the distance map, as shown in (b), the distances on the edges are smaller than the flat surfaces, which means heat is more concentrated on the edges of the block, in comparison with the smooth and flat parts. (c) shows the result of the minimal paths



(a) (b) (c)

Fig. 3.9 Experiment on a noisy image (a), the red line on (b) is obtained by Fast Marching Method, the red line on (c) is the result of P-M heat method (3.11).

between the ten end points and three source points. It is very conspicuous that that all paths go along the edges.

As is introduced above, the heat flows are not strongly affected by noise. For the different heat flows, there are other advantages and disadvantages which we list in Table.3.2, where IHF and AHF are abbreviations for the isotropic heat flow and anisotropic heat flow.

Table 3.2 (Comparison of different flows).

	Metrics	Advantages	Disadvantages
IHF	Conductivity P-M Eq.(3.8) P-M Eq.(3.11)	$\surd$ Convenient Intuitive $\surd$ Homogeneous regions $\surd$ Edges Boundaries	$\times$ Complicated scenes $\times$ For edges $\times$ For regions
AHF		$\surd$ Specified directions	$\times$ Time Consumption

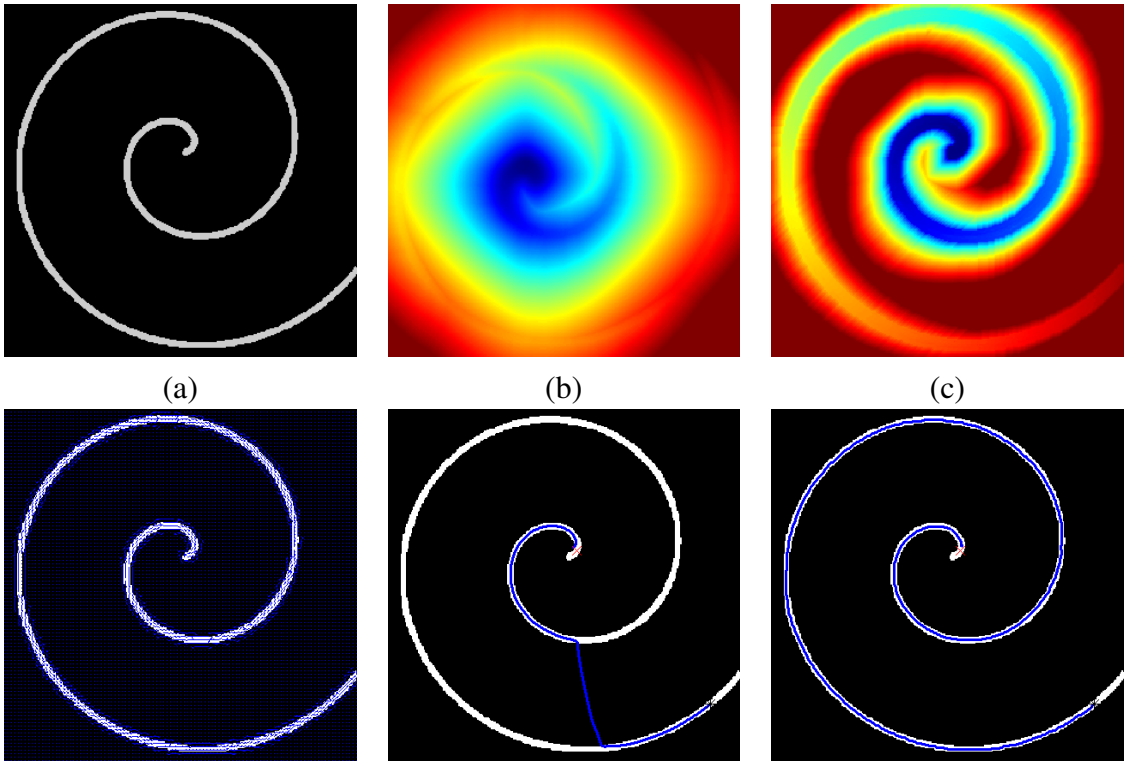
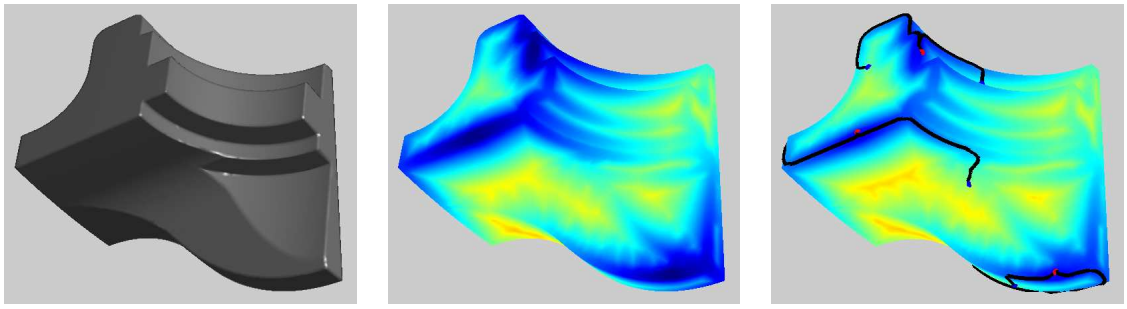


Fig. 3.10 Experiment on a spiral: (a) original spiral image; (d) tensor field; (b) distance map by isotropic heat diffusion; (e) geodesic line extracted corresponding to (b); (c) distance map by anisotropic heat diffusion; (f) geodesic line extracted corresponding to (c).

3.4 Conclusion and Future work

In this chapter, we proposed new methods using the isotropic and anisotropic heat diffusions to get the geodesic distance and geodesic lines for image segmentation purposes. Using different kinds of diffusivity, models and tensors, the methods work well for different types of images and features of interest. For example, the P-M model can either try to hold the heat within the boundary of a region or make the heat flow along the edges. By using different diffusion tensors, the anisotropic heat diffusion will flow along the direction that we design. The biggest advantage of using heat flow is that it is very fast and robust as well, and also easy to implement. Furthermore, heat diffusion is not very sensitive to noise, a little noise will not affect its performance.



(a) (b) (c)

Fig. 3.11 Experiment on the wedge-like block, three points on the edges are chosen to be the source points, and 10 points are randomly chosen as the end point. (a) is the data, (b) the distance map, (c) the paths extracted along the edges of the block.

Chapter 4

Automatic Segmentation Based on Heat Diffusion

Abstract

In the previous chapter, we introduced how to obtain the geodesic distance and curves by using different heat diffusion equations. However, in all of these methods, the user has to provide at least one source point and several end points to extract the curves. This extensive manual intervention makes it very tedious to extract complex curves. To address this issue, we propose two algorithms for extracting the geodesic lines automatically without having to provide the end points.

4.1 Introduction

The first algorithm is realized by the extension of the method of geodesic voting, which was proposed in [92]. The authors first use the Fast Marching Method to get the distance map, and subsequently use the boundary of the image domain, or randomly selected points, as the end points. Backtracking from these end points to the source point, there will be numerous paths. At each point of the image domain, the geodesic density is defined as the number of paths that go through that point. By thresholding the density, an automatic segmentation of the desired structures can be obtained. The general idea of our voting method is that we set some time $\Delta T > 0$ for the diffusion, which depends on the size of the image. After time ΔT, heat can pervade a certain region surrounding the source point p_0. The pixels on the front (boundary) of this region are then used as the end points for backtracking to p_0. Then, setting a cutoff-threshold ϵ on the geodesic density, we retain only those pixels, which have geodesic densities above this threshold.

The second method is inspired by the key point method described in [4]. In this method, a set of contour curves or thin structures are obtained as a set of minimal geodesic paths connecting successive keypoints. These keypoints are defined in an iterative way by selecting the first point on the Fast Marching front for which the minimal path reaches a given curve length. We refer to [4] for the rationale and details on the method. Here, in our own method, we first let the heat diffuse for some time ΔT_1 and use the current source point as the center and r as radius to form a circle. We identify the points on the circle with the largest temperature values (note that there might be more than one large value, and we are looking for the peaks of the heat density larger than a prescribed threshold). The new source points are located at these peak points. After obtaining the new source points from the peak points, we let them begin to diffuse one after the other until a stopping criterion is met. This is a particular topic of discussion in the keypoint section of the chapter.

Section 4.2 details how the voting method is adapted to the heat method; Section 4.3 displays the keypoint method; Section Section 4.4 shows experimental results for these automatic methods on images. Section 4.5 provides some concluding remarks.

4.2 Voting Method

In [92], the authors present a novel method for automatic segmentation of tree structures, named geodesic voting. First, the authors obtained the distance map by using the Fast Marching Method, then they use some endpoints chosen automatically to backtrack to the source point. Thus there will be a series of paths extracted. The points located on these paths can be used to define the geodesic density:

$$\mu(p) = \sum_{n=1}^{N} \delta_p(l_n) \tag{4.1}$$

where $\delta_p(l) = 1$ if pixel p is crossed by path l, N being the number of paths. A threshold-cutoff for the geodesic density is also set. We retain only the pixels with a number of paths above the prescribed threshold in the final result.

In this section, we introduce three voting methods. They will be detailed in the ensuing subsections. One method is to vote from the front of heat distribution directly. As stated above, the paths are extracted from the boundary of the heat distribution after time ΔT. They consist in joining each end point to the source point by backtracking. Fig.4.1 shows how the time ΔT affects the results. In the first row, we choose a smaller time $\Delta T'$, which is half of the ΔT in the second row. (a) and (d) show the heat distribution after $\Delta T'$ and ΔT and the blue curve surrounding the regions are the fronts which are considered as the end points. (b) and (e) are the paths obtained by backtracking from the front points to the source point. (c)

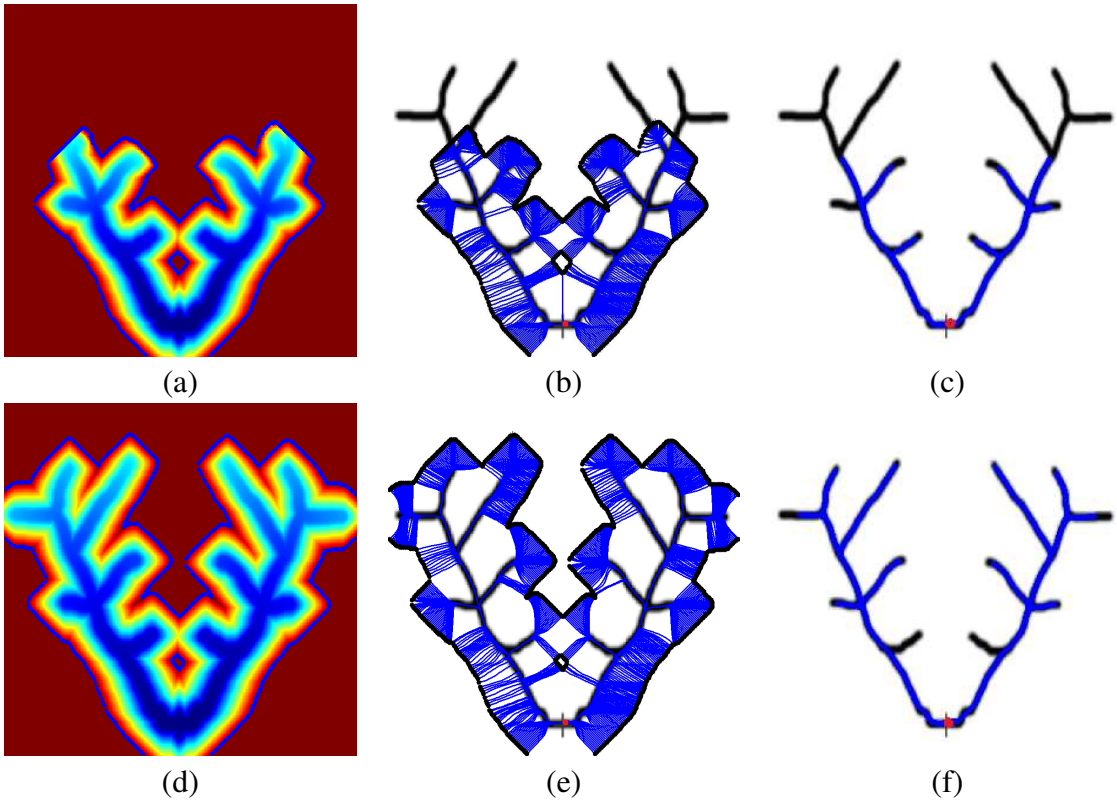


Fig. 4.1 Experiment on a tree structure by voting from front, from left to right: the first column shows the distance maps, the second one shows the paths obtained by backtracking from the heat front to the source point, and the third one presents the result by voting.

and (f) give the structure extracted by voting. Clearly we can see that in Row2, the method gives a more thorough result compared to Row1, which emphasizes the importance of how we choose ΔT.

Another voting method is to add new voting points within a smaller period time ΔT_1. We then track back from the front to the source point and vote for the first result. Next, we let the heat continue to diffuse for another time ΔT_1 to track back and vote again, adding the new result to the first result. This process of letting the heat diffuse within a period of time ΔT_1 is reiterated as many times as need be until the stopping criterion is fulfilled.

The third voting method is realized by resetting the source point. After time ΔT_1, the first result can still be obtained by voting. Then, we use these points as the new source points, and let the heat diffuse for another ΔT_1. The new results then add-up as the collection of source points for the following diffusion. This process is repeated until the stopping criterion is met. Experiments of these three voting methods are compared in the following section 4.4 dedicated to the experiments.

4.2.1 Voting from the Front

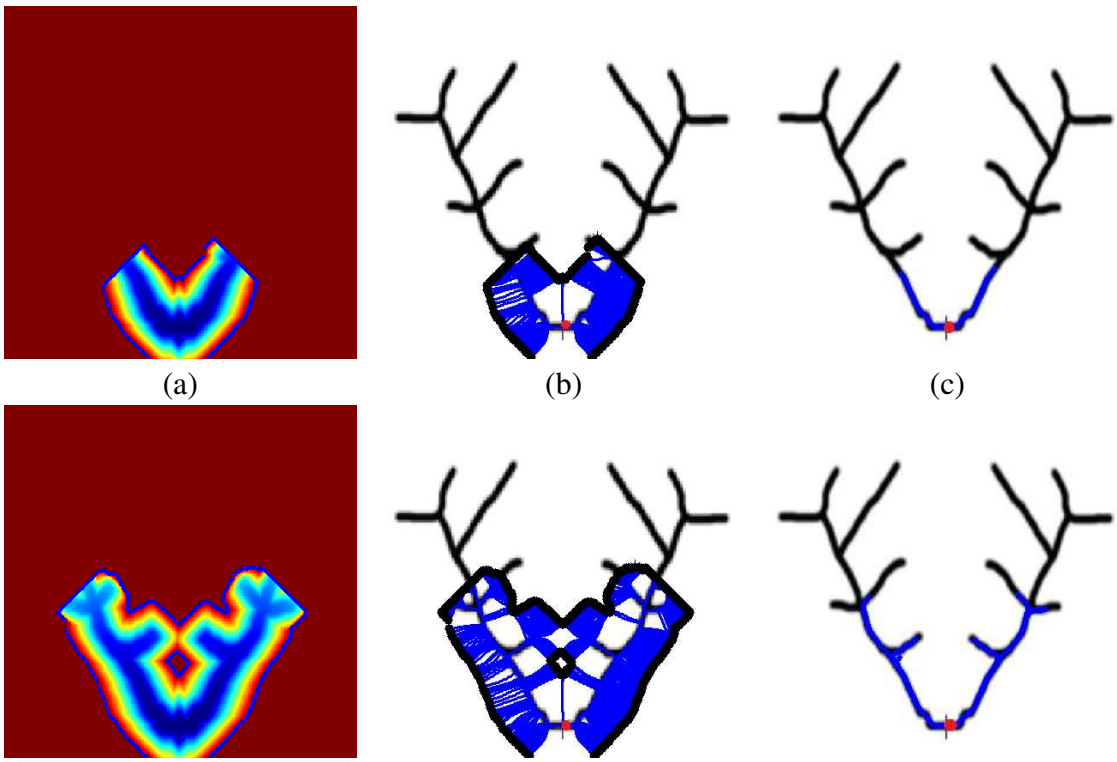
First of all, we set a source point p_0 at the location of interest. After $\Delta T > 0$, the distance map ϕ is obtained by Eq.(3.3). The positions on the boundary of the region covered by heat are considered as end points. By backtracking from these end points back to the source point, we get many geodesic lines. Setting up an appropriate threshold ϵ for the geodesic density β can be useful for getting the right path. Generally, we choose $\epsilon = \max(\beta)/M$, where M is a constant. For example, in Fig.4.1(f), M equals 30 in this case, but M can be different in different cases.

Voting directly from the heat front is easy to implement. When we compare the use of heat front to get end points with the use of boundary points of the image or randomly chosen end points as the end points, we find that the heat front is more suited to get the centerline. This is because the way heat diffuses depends a lot on the geometry of the shapes of the structure. Yet, as heat propagates, the heat front does not provide the exact shape of the structure to be extracted. This makes it easy to mix-up two paths that make a small angle between each other like in the case of tree structures, making it more difficult to recover all the paths. As shown in Fig.4.1(f), there are two segments inside the tree structure that are missing, whereas in (c), these same segments are extracted. For this reason, we propose two other methods for more complicated scenes.

4.2.2 Multiple Voting Method

As mentioned above, given a more complicated scene, such as a tree structure, there will be branches missing even if we use the boundary of the region of heat diffusion. Besides, the diffusion time ΔT should be re-selected when the image size is different. For these reasons we propose the idea of the multi-voting method.

First, a source point p_0 is given. Within a smaller period of time ΔT_1, we see a reduction in the magnitude of the region pervaded by the heat as shown in Fig.4.2(a). Using the boundary of this region as the end points and backtracking to p_0, as shown in (b), we vote and set a threshold ϵ of the voting score Eq.(7.6). Here we choose $\epsilon = \max(\beta)/7$ in the experiments. Note that we use 7 rather than 30 here because it is different from voting directly: there are fewer paths extracted at each iteration. The points $\{p_{s_1}\}$ with a higher density value than ϵ are retained, as shown in (c). Then, we let the heat continue to diffuse for the same period of time ΔT_1 and get the distance map (d). We vote again from the points on the new front of the region of heat and the newly obtained points $\{p_{s_2}\}$ are saved again as shown in (f). Heat diffuses in this way and we keep all the points $\{p_{s_1}\}$ to $\{p_{s_n}\}$ together until the stopping criterion is met (see section 4.2.4). We show the two first steps here and the final result is shown in the experiment section.



(a) (b) (c)

(d) (e) (f)

Fig. 4.2 Experiment on the tree structure by multiple voting method, the top row is obtained by the first step of Sect.4.2.2, the bottom row is the second step, from left to right are the distance map, the paths obtained by backtracking from the front to the source points and the result after voting.

4.2.3 Accumulation of Source Points

Resetting source points method is based on a procedure similar to that of adding voting points of the previous subsection. Both of them need a multi-voting process. The resetting source points method takes advantage of diffusion and makes the process more exact because each time there will be more source points that cover the structure.

The first step of voting is the same as the one in Sect.4.2.2. After time ΔT_1 of diffusion from source point p_0, we get the points $\{p_{v1}\}$ by thresholding the geodesic density with ϵ and save the points $\{p_{v1}\}$. We then reset the temperature to zero everywhere on the image and set all points of p_{v1} as the source points that have the same temperature. Next, we let these source points diffuse for the same time ΔT_1 and we vote again and get points $\{p_{v2}\}$. Add $\{p_{v2}\}$ to $\{p_{v1}\}$ and repeat resetting the temperature to zero on the whole domain and adding new points to $\{p_{v1}\}$ as the source points, until the stopping criterion is met. The stopping criterion here is the same as in 4.2.2 and will be detailed in sect.4.2.4.

Resetting source points method is different from Sect.4.2.2 because the source points keep changing every time there is a vote. It is better at controlling the direction of the heat flow because increasing the number of source points leads to higher accuracy. As is shown in Fig.4.3, these are two first intermediate steps of this algorithm. After time ΔT_1, the distance map (a) is obtained, by voting we can get (c). In the second step, we use the points in (c) as the source points, and let the heat diffuse from scratch. After the same time ΔT_1, we get the result in (f). In the next steps, we use the points obtained from its previous step as new source points until the stopping criterion is met. Compared to Fig.4.1, by using resetting the source points method, the segments on the tree structure are almost all extracted, and heat is more concentrated on the boundary of the region that it covers. The final result is shown in the experiment part.

4.2.4 Stopping Criterion

We define two criteria for the heat to stop diffusing. Once at least one of them is satisfied, the heat diffusion stops, see Algorithm 2. Take Fig.4.3 as an example.

1. At the beginning, one point p_0 is chosen as the source point in an image (with a size $M \times N$). Save p_0 into a list L, which is initialized as empty. After one step of voting, we get the points $\{p_{v1}\}$, as shown in Fig.4.3 (c), save them into L. Denote the number of pixels in L by N_L. If $N_L/(M \times N) > \eta$, the heat diffusion is terminated. Otherwise, it continues to diffuse. In our experiments, according to our experience, there is a rule of thumb and this magnitude η is set to $1/30$.

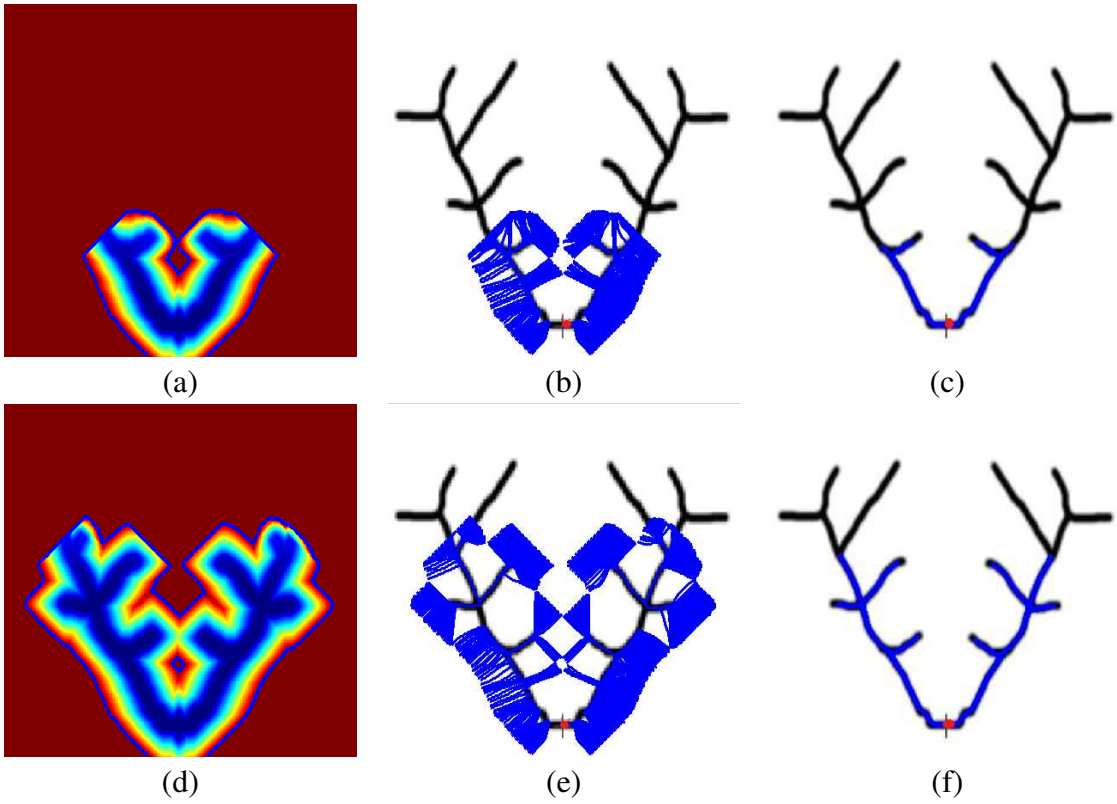


Fig. 4.3 Experiment on the tree structure by accumulation of source points, the above row is obtained by the first step of Sect.4.2.3, the below row is the second step, from left to right are the distance map, the paths obtained by backtracking from the front to the source points and the result after voting.

2. After the ith step of diffusion, new points $\{p_{vi}\}$ should be added into list L, if $N_L/(N_L + N_{\{p_{vi}\}}) > 95\%$, where $N_{\{p_{vi}\}}$ is the number of the set $\{p_{vi}\}$, the heat diffusion can also be stopped.

Algorithm 2 Stopping Criteria

Initialization: p_0 chosen as the source point, $L \leftarrow p_0$
repeat
    $L \leftarrow p_{v_n}$, p_{v_n}: the n-th voting;
until $N_L/(M \times N) > \eta$ or $N_L/(N_L + N_{\{p_{v_i}\}}) > 95\%$

4.3 Key Points from Heat

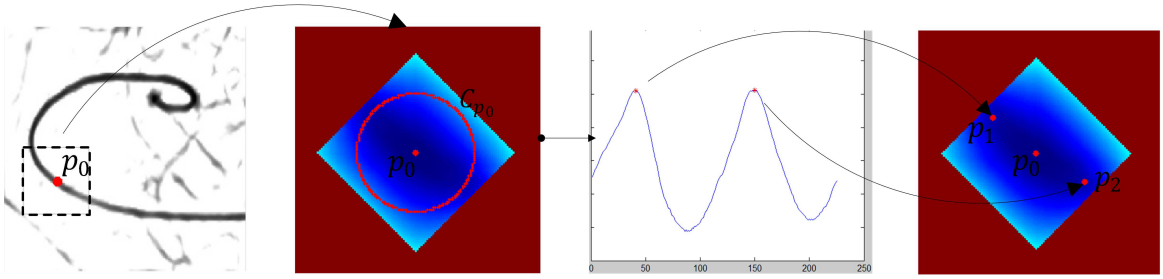
In [4], the authors introduced a method for segmentation using Fast Marching Method by growing minimal path and detecting key points on the curves of interest recursively. First, the user provides an initial point on the desired object. Then, starting from the initial point, a front is propagated and the key points are detected iteratively. These key points are almost

equi-distributed along the curve of interest, and thus are detected based on the Euclidean lengths of the minimal paths. The whole process can be described as follows. First, start from at least one single point p_0 to initiate the Fast Marching Method. Every time we compute the geodesic distance $U_{p_0}(p_x)$ from a point p_0 to p_x on the image, we also need to compute the Euclidean length $L_{p_0}(p_x)$ of the geodesic path from p_0 to p_x. When a point p_1 satisfies that $L_{p_1} \geq \gamma$, where γ is a threshold given by the user, p_1 is considered as the first so-called key point. As soon as p_1 is detected, it is considered as a new source of propagation. The same process is used to define the successive key points $\{p_k\}$. Front propagation is let to continue until a stopping criterion is met. Refer [4] for details.

Since the key point method is efficient and robust in [4], we adapt it to heat diffusion. However, using the heat diffusion, it is not as easy to compute the Euclidean distance together with geodesic distance as it is in the case of Fast Marching in [4, 51]. This is because when using Fast Marching, the geodesic and Euclidean distances are updated simultaneously at every iteration when the status of a pixel is updated, by propagation, while in heat diffusion, we do not compute the distances by the same kind of front propagation. Therefore, we propose a new method to detect key points using heat diffusion without computing the Euclidean length of each minimal path during every detection.

First, we set a source point p_0 on the curve of interest, and p_0 can be considered as the first key point. For every key point, there are two states, $s(p_x) = 0$ and $s(p_x) = 1$, where $s(\cdot)$ is the state function. As will be made clear below, points with state 0 are used to compute the geodesic distance and lines, as well as finding the next key point, while points with state 1 are points that have already been used for computing its neighbor key points. Now let the heat diffuse from p_0, where $s(p_0) = 0$. We stop the heat from diffusing after a certain time Δt. The distance map d_{p_0} can be obtained by Eq.(3.3). In the region R_{p_0} of heat diffusion, we make a circle C_{p_0} where p_0 is the center and r is the radius. It should be guaranteed that the circle is located within this region R_{p_0}. Then we find the peaks among heat density $u_{C_{p_0}}$ on the circle C_{p_0} and set a threshold ϵ_1, save the positions $\{p_{s1}\}$ of the peaks whose values are larger than ϵ_1, and define $\{p_{s1}\}$ as the new source points. Fig.4.4 depicts the flowchart of choosing the new source point in the first Δt. We try to find the position of the peaks of the heat density. Two points p_1 and p_2 are found here, $\{p_{s1}\} = \{p_1, p_2\}$, and they are considered as the key points which are found at the first diffusion.

After the first time of diffusion Δt, we get $\{p_{s1}\}$ which denotes the new collection of source points. Since p_0 has been already computed for finding its neighbor key points, its state is changed to 1, and we use a list L to save p_0. Now it is turn for p_1 to diffuse and get its neighboring key points. We empty the heat density everywhere on the image and let heat diffuse from p_1, and use the same technique as shown in Fig.4.4. We get a distance map d_{p_2}. Yet, note that here we only use the subpart where $C_{p_1} \subset R_{p_1} \backslash R_{p_0}$. After finding its neighboring key point p_3, the state of p_1 is changed to 1, and p_1 is saved in the collection L. Point p_2 goes



(a) (b) (c) (d)

Fig. 4.4 Flowchart of how to choose the new source points in the first Δt. From left to right, (a) is the original image with a source point p_0, (b) is the distance map D_1 of the dashed box in (a) after time t, the red circle is C_{p_0}, (c) plots the heat density on the points of C_{p_0} and there are two peaks which represent the new source points p_1 and p_2 in (d).

the same way until we get to p_n, where the maximal value of heat of $C_{p_n} \subset R_{p_n} \backslash \{R_{p_0}, ..., R_{p_{n-1}}\}$ is less than the threshold ε_1 we set. Algorithm.3 is the key point algorithm using the isotropic heat diffusion, where ϵ_1, ϵ_2 and ε_1 are three parameters. Once we stop the process, we have a set of key points and we obtain a set of paths, where each keypoint is linked by a geodesic path to a previously obtained key point. This is made by backtracking from a keypoint to the previous key point from which it takes its origin. Each path is obtained in the algorithm at the time the key point is added in the list.

Algorithm 3 Keypoint Algorithm

Initialization:
$s(p_0) = 0, L \leftarrow p_0, u = 0, \phi = 0, \Gamma = \{0\}$; % Γ saves the geodesic paths
repeat:
  $p_{curr} = SearchState(L)$, % find the first point whose state is 0;
  $u(p_{curr}) = 1$; % initialize the heat of current point
  $[u, \phi] = HeatDiff(u)$;
  $\{p_m\} = GetPeaks(C_{p_{curr}}(r))$; % $C_{p_{curr}} \subset R_{p_{curr}}$,
  $s(p_{curr}) = 1$;
  if $\max\{l(p_m)\} < \epsilon_1$
    break;
  end if
  for i = 1:m; % m is number of peak points;
    if $\alpha(p_i) > \epsilon_2$; % α is the conductivity in Eq.(3.6).
      $L(end+1) = p_i$;
      $\Gamma(end+1) = \gamma(p_i, p_{curr})$;
      $s(L(end)) = 0$;
    end if
  end for
until: $\max(C_{p_n} \subset R_{p_n} \backslash \{R_{p_0}, ..., R_{p_{n-1}}\}) < \varepsilon_1$

4.4 Experiments and Analysis on Automatic Segmentation of Geodesic Curves

4.4.1 Experiment Data and Settings

We test on three images: 1) a synthetic tree structure image in Fig.4.5 and Fig.4.9; 2) a real vessel image with many branches in Fig.4.6, it is obtained by maximum intensity projection of a real 3D vessel data and 3) a real medical image with a catheter which is highly curved in Fig.4.8 (a), the size of all three images is 300×300. In Fig.4.5(b), (c), (d), we use isotropic heat equation Eq.(3.1) with the conductivity in all the three methods, Eq.(3.6), $n = 3$ here. In Fig.4.6 , Fig.4.7 and Fig.4.8, we use Eq.(3.10), the combination of P-M model and conductivity for the heat diffusion. In the process of voting directly from the front, the time ΔT is controlled by the iteration times of heat diffusion, we set the iteration times to $2N$, where N is the length of image, here $N = 300$. In the process of multiple voting methods, including Sect.4.2.2 and Sect.4.2.3, the smaller time ΔT_1 is controlled by the iteration time of each step, which is set to 100 here. In detecting key points, for each step, the iteration time is 50, and the radius r is 30, the threshold ϵ_1 we set here is 80% of the highest heat density on C_{p_0} during the first iteration.

4.4.2 Results and Analysis

Here we extract the centerlines by voting method or using the key point method.

In Fig.4.5, (e) is the result obtained by [92] and 500 end points are randomly chosen on the image. (b), (c) and (d) are the voting maps obtained by using directly voting from the front, multi-voting by adding voting points and multi-voting by accumulating of source points. (f), (g) and (h) show the corresponding structures that are extracted by different methods, and the black part are mis-extracted part. In (e), there are some parts missing in the terminals in the tree branches, the same phenomenon takes place in (f) and (g). (h) has the best result because all branches and details, as well, are extracted.

Fig.4.6 shows the results of Fast Marching (a) and the three automatic voting methods (b) (c) (d) with heat on real medical images. From the results, we can see that the blue lines in (b) (c) (d) go along the centerline of the vessels, but (a) fails to follow the centerline. And for all of (a), (b) and (c), there are segments missing. However, by using resetting source point method nearly all vessels are extracted in (d).

In Fig.4.7, the terminals of the catheter are hard to extract. Results are shown superimposed on the potential image built from the Laplacian. Fig.4.7(a) is the result by [92] and the end points are randomly selected, and from (b) through (d) are the results from using directly

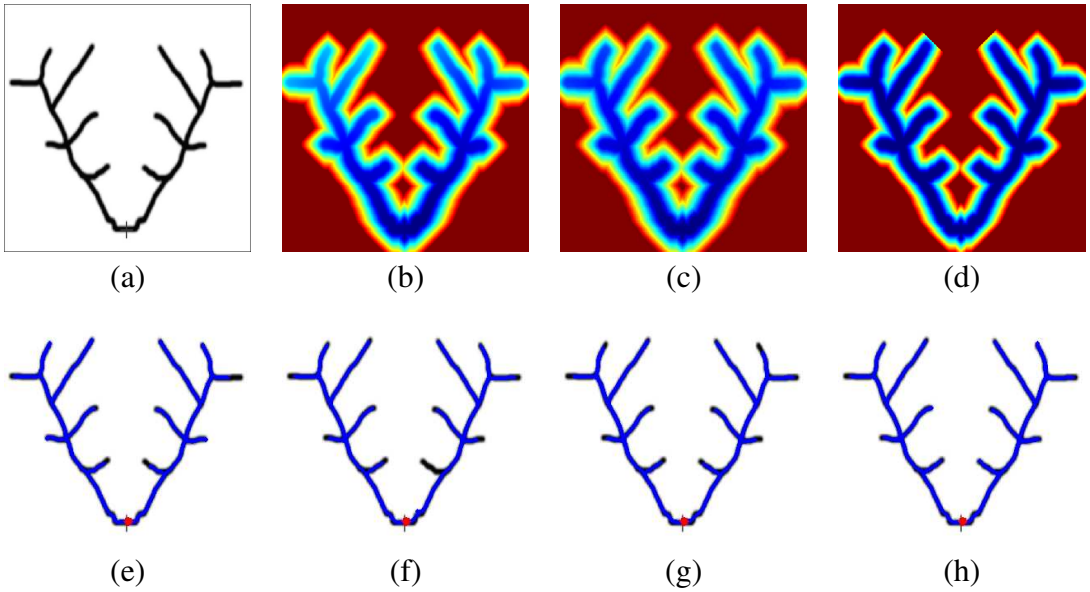


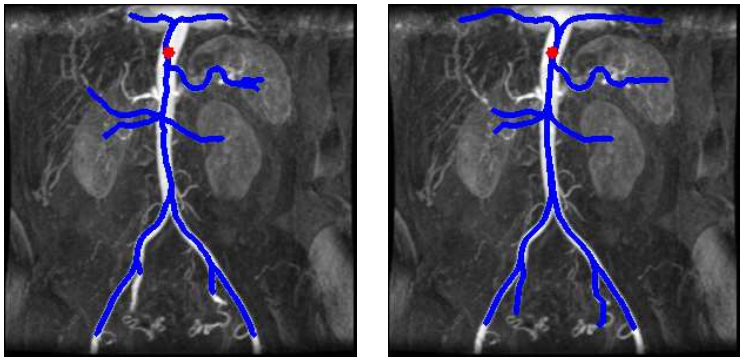
Fig. 4.5 Voting experiment on the tree structure, (a) is the original image, (e) is the result by [92] by using randomly chosen end points, from the second column to the fourth, the above row shows the voting map by using respectively voting from the front, multi-voting by adding voting points and multi-voting by accumulating of source points. And on the bottom row are the corresponding results extracted by the three automatic methods.

voting from the front, multi-voting by adding voting points, and multi-voting by accumulation of source points. All of the four results give the contour of the curve-of-interest.

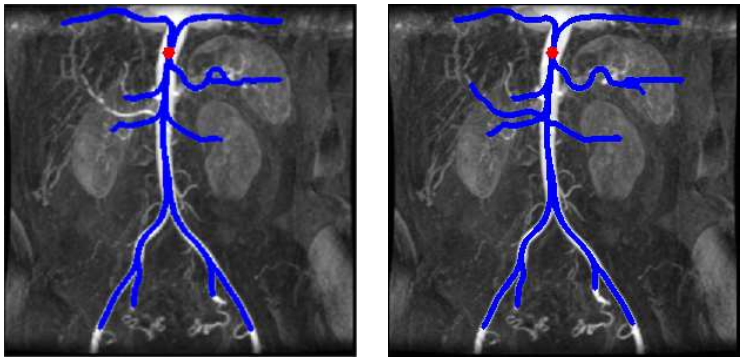
The automatic method in Sect.4.3 is tested in Fig.4.8 and Fig.4.9. Fig.4.8(a) is the original medical image, where there is a curve with some high curvature; (b) is the potential built according to the Laplacian; (c) is the first step through detecting key points, and we can see that there are two points (yellow) that are detected; (d) is the step following (c), and it can be found that there is another key point detected; (e) is the next step and it is also the third step in the whole process of detection; (f) is the result after several steps and (g) is the final result in the case where the blue paths that are extracted by backtracking from each key point one after the other, and they cover exactly the curve; (h) is the traveling map of the key points, which means the geodesic distance map to each key point in its neighbourhood during the whole process of heat diffusion.

The key point from heat method does better than voting methods for this example. Computationally, comparing the heat density becomes much easier and less time-consuming than computing the Euclidean length of the geodesic curves each time in [4]. Furthermore, it also provides satisfactory results.

Another important issue in using key points from heat is how to choose an appropriate radius r for defining the circles every time. Fig.4.9 are experiments on the tree structure using the key points from heat. From (a) through (d), the radius are 45, 30, 15, 10 respectively. And from the extracted paths, we can see that (d) gives the most complete structure among the



(a) (b)



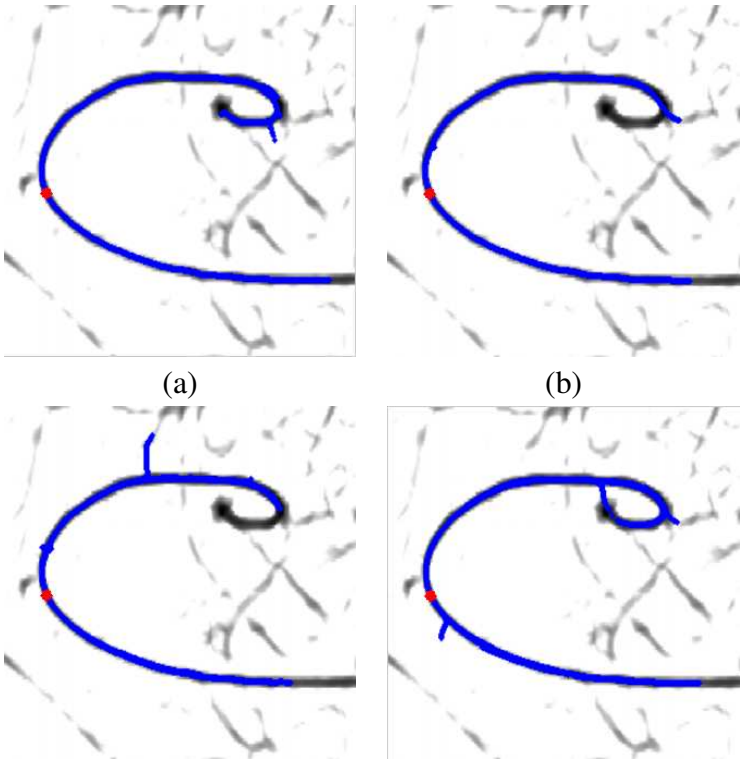
(c) (d)

Fig. 4.6 Voting experiment on the real vessel image, (a) is the original image with classical voting from Fast Marching, (b) (c) (d) are the results by using directly voting from the front, multi-voting by adding voting points and multi-voting by accumulating of source points.

four results. In this case, it can be seen that when a smaller radius r is used, the result is better. But this does not prove that it is better using a smaller r in every situation. In fact, how to choose an appropriate radius depends on the size of the feature that we want to extract. Here in the tree structure, the width of the branches of the tree have no more than 10 pixels, so a smaller radius is more effective in key points detection in this particular case. However, as in the classical key point method [4], using a larger radius makes the method less sensitive to noise, and this is why the radius should not be chosen too small.

4.5 Conclusion

In this section, we apply two automatic ways to heat method to extract geodesic curves: one is based on voting and the other is based on finding keypoints. Experiments show that using the proposed automatic methods, the results are satisfactory and robust as well as time-saving and these two methods help to reduce human intervention effectively.



(a) (b) (c) (d)

Fig. 4.7 Voting experiment on the real medical image with catheter, from left to right: the results of [92] are displayed by using randomly chosen end points; using directly voting from the front; multi-voting by adding voting points and multi-voting by accumulating of source points.

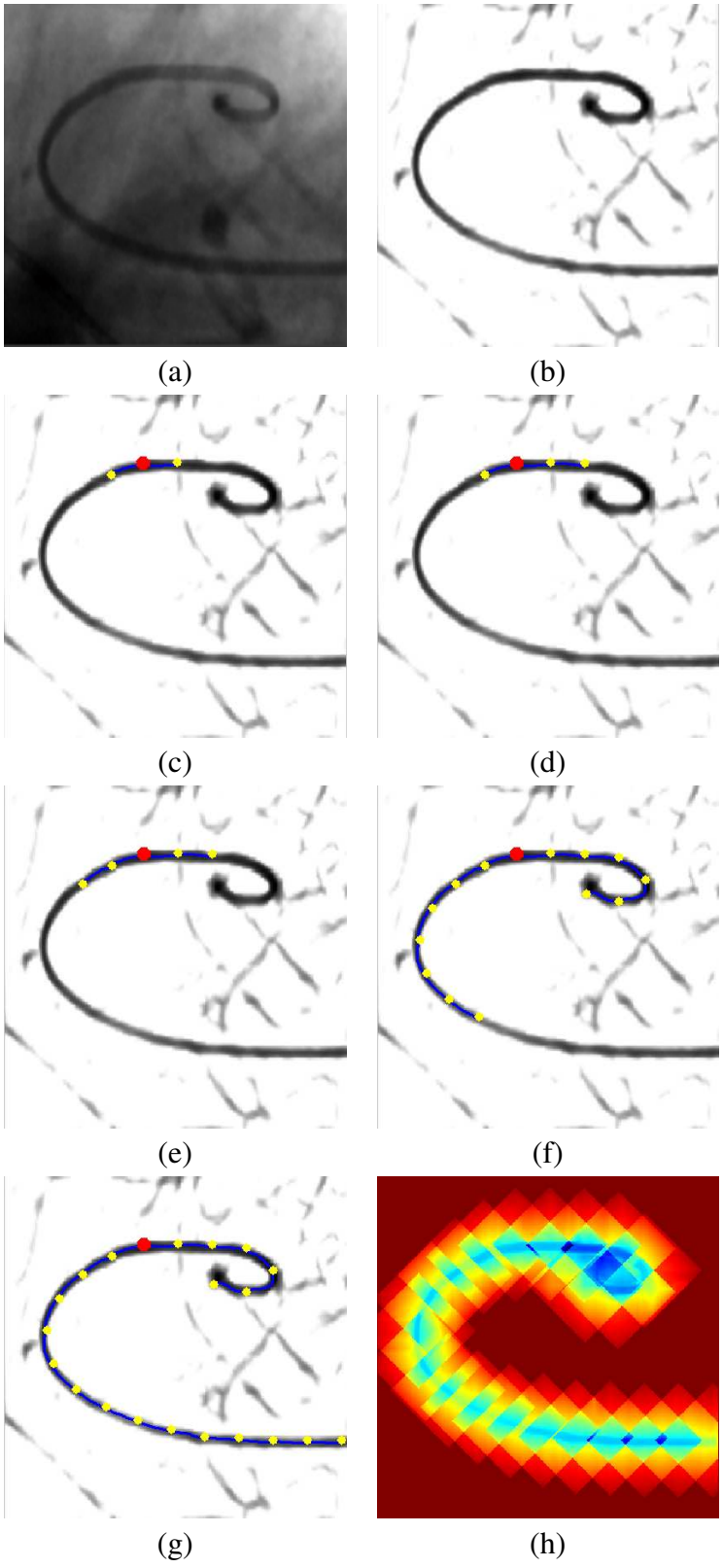
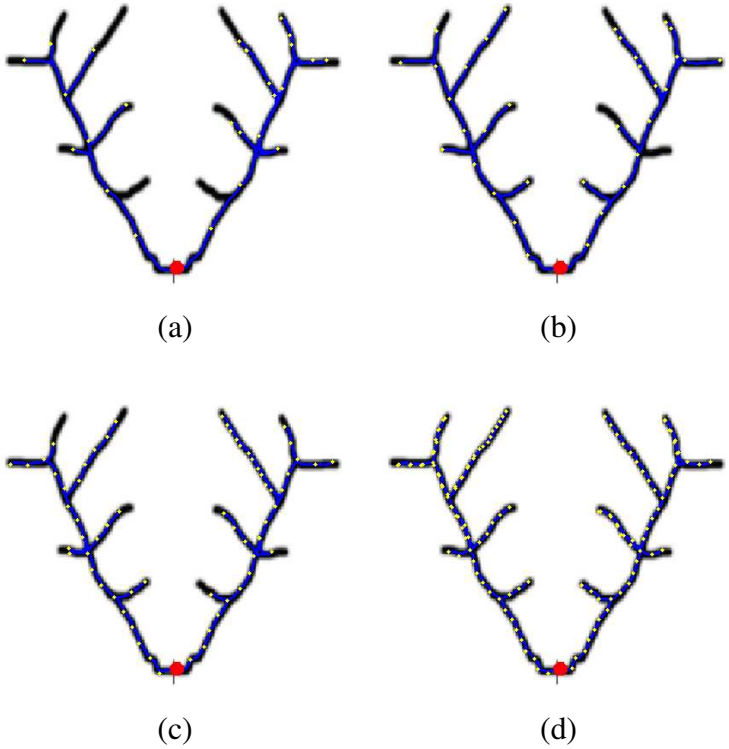


Fig. 4.8 Keypoint experiment: (a) is the original image; (b) is the Laplacian of the original image; (c) the red point is the source point given by user, and the two yellow points are the first key points detected; (d) is the second step of detecting key point from heat; (e) is the next step, and (f) the result after several steps in search for key points; (g) shows the final result of all the key points and paths detected and (h) is the traveling map of key points.



(a) (b)

(c) (d)

Fig. 4.9 Keypoint experiment on a tree structure by using key point method from heat, the red point is the source point given by the user, and the yellow points are the key points detected. From left to right are displayed the results when using different radii r in defining the circles, they are 45, 30, 15 and 10 pixels respectively from (a) to (d).

Chapter 5

Tubular Structure Segmentation based on Heat Diffusion

Abstract

As shown in Chapter 3 and Chapter 4, we applied the different heat equations to extract the centerlines of tubular structures. In this chapter, an interactive method for tubular structure segmentation is proposed. The method is based on the minimal paths obtained from the geodesic distance solved by the heat equation. This distance can be based both on isotropic or anisotropic metric by solving the corresponding heat equation. Thanks to the additional dimension added for the local radius around the centerline, our method can not only detect the centerline of the structure, but also extracts the boundaries of the structures. Our algorithm is tested on both synthetic and real images. The promising results demonstrate the robustness and effectiveness of the algorithm.

5.1 Introduction

As introduced in Chapter 2, generally, the geodesic distance ϕ could be acquired via Dijkstra's method [27] or solving the Eikonal equation $\|\nabla\phi\|_{T^{-1}} = 1$, where T is a tensor metric computed from the image I in anisotropic cases. When in isotropic cases, $T = \mathcal{P}^2 \mathrm{I}d$, the Eikonal equation becomes: $\|\nabla\phi\| = \mathcal{P}$. Please refer Chapter 2.1 for details. The Fast Sweeping Method [125] and the Fast Marching Method [97, 19] are quite often used to solve the Eikonal equation. For the extraction of the geodesic lines $\gamma^\star$ between the initial point p_{s_0} and the endpoint p_x, it can be achieved by solving an ordinary differential equation after the computation of ϕ.

$$\forall s > 0, \frac{\mathrm{d}\gamma^\star}{\mathrm{d}s} = -\frac{T^{-1}\nabla\phi}{\|T^{-1}\nabla\phi\|}, \gamma^\star(0) = p_x \tag{5.1}$$

Let us recall the heat equation: the heat equation is a partial differential equation (PDE) that describes the evolution of the distribution of heat on a domain within time T. The general form is:

$$\frac{\partial u}{\partial t} = \alpha \Delta u \tag{5.2}$$

where u stands for the heat, and α, a positive constant, represents the thermal conductivity, Δ is the Laplace operator.

Let us also recall the Varadhan's formula: in 1967, Varadhan [104] proposed to approximate the geodesic distance $\phi(p_0, p_x)$ between two points p_0 and p_x on a Riemannian manifold:

$$\phi(p_0, p_x) = \lim_{t \to 0} \sqrt{-4t \log u_{p_0}(p_x, t)} \tag{5.3}$$

where u_{p_0} the solution of Eq.(5.2) under the initial condition that $u_{p_0}(0) = \delta_{p_0}$ within a small time $t \to 0$. In Chapter. 3, we have proved that the Varadhan's formula can be applied to either the isotropic heat diffusion or the anisotropic heat diffusion.

Recently, Crane *et al*. [23] proposed a heat method to estimate the geodesic distance. Their heat method is extremely fast because some crucial steps can be precomputed. It is also proved that the sparse systems arising from the elliptic PDEs can be solved in very close to linear time [95, 96]. Details can be found in the original paper [23] or Chapter. 2.

In Chapter 3, we have extended and gone beyond the work of Crane *et al*. . We introduced isotropic and anisotropic heat flows to approximate the geodesic distance by using Varadhan's formula Eq.(5.3). It is shown in Chapter 3 and 4 that using the heat method to approximate the geodesic distance is not only fast and efficient, but also less sensitive to noise. Despite the efficiency and robustness of the heat method, according to previous work, it could only provide the centerline for the tubular structure. We would like to integrate the segmentation methods to the heat equation and make it extract the centerline and segment the tubular structure simultaneously.

In the past few decades, numerous segmentation methods based on minimal paths have been proposed, such as [19, 60, 7, 6, 123]. In [60], Li and Yezzi have incorporated an additional non-spatial dimension to the traditional minimal path technique [19](where there are only spatial information). This method can measure the thickness (radius) of the structures in space. In other words, this additional dimension can help to extract the boundaries and surfaces of the structures in 2D and 3D spaces. And in the meantime, their method can also detect a precise centerline of the structures. But the potential $\mathcal{P}$ that is used in [60] is isotropic and depends on the positions. The orientations of the tubular structures are ignored. Later on, Benmansour *et al*. [7] have proposed an anisotropic minimal path model which also takes the additional dimension into account, so the potential depends both on the positions and the tangent directions. The way they built the metric was based on the anisotropic Optimally Oriented Flux (OOF) descriptor proposed by Law and Chung [58]. The OOF descriptor

makes the propagation faster along the tubular structures. The advantage of the anisotropic model is its ability to avoid the shortcut issues effectively. Then Benmansour and Cohen have extended their method into $3D$ vessel extraction [6].

In [60, 6], the authors use the Fast Marching Method to get the numerical solution of the Eikonal equation, while in this chapter, we are interested in the segmentation of tubular structures by using the heat method. Here, we use the same way to construct the metric tensor as the authors do in [7, 6]. This is called *2D + Radius* model in heat.

The contribution of this work is that we add a non-spatial third dimension in both isotropic and anisotropic heat diffusion to segment tubular structures. We use the OOF descriptor [58] to build the metric. Therefore, the heat method can be used to detect the centerlines and boundaries simultaneously. Besides, we use the scheme proposed in [35] to discretize the heat equation, for each image, the Laplacian operator can be precomputed. Additionally, under the same conditions, compared with the Fast Marching Method, the heat method presents good results within less time.

This chapter is organized as follows: in Section 5.2, we give some background on the minimal path, the heat diffusion, and the OOF descriptor; in Section 5.3, how to construct the metric and the way to solve the heat equation are presented; in Section 7.4, we test our method on some synthetic and real data. Section 5.5 provides some concluding remarks and possible directions for future work.

5.2 Background

5.2.1 Minimal Paths

Let us recall the minimal path model introduced in Chapter 2, given an image $I : \Omega \to \mathbb{R}$ and two points p_{s_0} and p_x, the geodesic γ is a curve connecting these two points that globally minimizes the following energy functional $E : \mathcal{A}_{p_{s_0},p_x} \to \mathbb{R}^+$:

$$E(\gamma(s)) = \int_\Omega \{\mathcal{P}(\gamma(s))\}ds, \quad \gamma(s) \in \mathcal{A}p_0, p_x \tag{5.4}$$

where $\mathcal{P}$ is a potential cost function computed from I. $\mathcal{A}_{p_{s_0},p_x}(s)$ is the set of all the curves linking p_{s_0} and p_x, s is the arclength.

This is the isotropic case. To solve this minimalization problem, Cohen and Kimmel [19] proposed a Hamiltonian approach: Find the minimal action map $\phi : \Omega \to \mathbb{R}$ that solves the Eikonal equation with an isotropic metric:

$$||\nabla\phi|| = \mathcal{P} \tag{5.5}$$

with the boundary condition $\phi(p_{s_0}) = 0$. Popular ways to solve the Eikonal equation such as the Fast Marching [97, 19] and Fast Sweeping [125] are quite often used. But these methods do not reuse information [23]: once the geodesic distance ϕ_{s_0} from the initial source point p_{s_0} is obtained, the distance from another source point p_{s_1} needs to be recomputed from scratch. According to Eq.(5.3), ϕ can be also approximated by the heat kernel. The advantage of using the heat kernel is that the Laplace operator could be precomputed, so that the fundamental solution of the heat equation can be acquired in a single step no matter where the initial point p_{s_0} is. In this way, the approximation of ϕ can be obtained once the heat equation is solved. Then the geodesic γ could be obtained by solving Eq.(2.13) or Eq.(2.10) depending on the heat diffusion being isotropic or anisotropic.

5.2.2 Isotropic and Anisotropic Heat Diffusion

Eq.(5.2) presents a homogeneous form of heat equation. When it comes to isotropic and anisotropic heat diffusion, the heat equation could be written as:

$$\frac{\partial u}{\partial t} = k\text{div} \cdot (D\nabla u) \tag{5.6}$$

where k is the diffusivity, it can be a constant (as α in Eq.(5.2)) or a scalar function, and D can be a scalar function or a diffusion tensor. According to [49], when D is a scalar function, the heat diffusion is isotropic. And when D is a diffusion tensor, it is a tensor field of symmetric positive matrices that can encode the local orientation and anisotropy of an image [123]. Then the heat diffusion becomes anisotropic. For the 2D heat diffusion, the tensor field D can be decomposed as shown in Eq.(5.7):

$$D = \lambda_1 \mathbf{e}_1 \mathbf{e}_1^T + \lambda_2 \mathbf{e}_2 \mathbf{e}_2^T \tag{5.7}$$

λ_1 and λ_2 are the eigenvalues, $\lambda_2 \geq \lambda_1 > 0$, $\mathbf{e}_1$ and $\mathbf{e}_2$ are the the corresponding orthogonal eigenvectors. A measure of the local anisotropy can be defined as $\mu = (\lambda_2 - \lambda_1)/(\lambda_2 + \lambda_1)$. When $\lambda_1 = \lambda_2$, the heat diffusion becomes isotropic.

The main difference between isotropic and anisotropic diffusion lies in the fact that isotropic diffusion does not include the local orientation, by using the anisotropic diffusion, heat could be more concentrated on the directions that the users design.

5.2.3 Optimally Oriented Flux

In order to use the relevant anisotropic heat equation, we need to find some estimates for the local orientation and scale to describe the tube-like structures. In fact, many classical enhancers like the Hessian-based vesselness mesures have been proposed [38, 62]. The

Hessian-based enhancers include adjacent features, while the OOF descriptor [58] avoids this problem.

Given an image $I : \Omega \to \mathbb{R}^2$, the oriented flux is the amount of the image gradient projected along the axis $\mathbf{p}$ flowing out from a $2D$ circle $\mathcal{S}_r$ at point $\mathbf{x}$ with radius r:

$$f(\mathbf{x}; r, \mathbf{p}) = \int_{\partial \mathcal{S}_r} (\nabla (G_\sigma * I)(\mathbf{x} + r\mathbf{n}) \cdot \mathbf{p})(\mathbf{p} \cdot \mathbf{n}) ds \tag{5.8}$$

where G_σ is a Gaussian with some variance σ, and empirically σ is set to 1. $\mathbf{n}$ is the outward unit normal of $\partial \mathcal{S}_r$. ds is the infinitesimal length on $\partial \mathcal{S}_r$. Based on the divergence theorem, the oriented flux can be written as:

$$f(\mathbf{x}, r; \mathbf{p}) = \mathbf{p}^T \mathbf{Q}_{r,\mathbf{x}} \mathbf{p} \tag{5.9}$$

where $\mathbf{Q}_{r,\mathbf{x}}$ is a symmetric matrix and we have:

$$\mathbf{Q}_{r,\mathbf{x}} = \begin{pmatrix} \partial_{xx} G_\sigma & \partial_{xy} G_\sigma \\ \partial_{yx} G_\sigma & \partial_{yy} G_\sigma \end{pmatrix} * \mathbb{1}_r * I(\mathbf{x}) \tag{5.10}$$

where $\mathbb{1}$ is an indicator function, which means that inside the circle, it equals to 1 and outside the circle it equals to 0. We denote the eigenvectors and eigenvalues of $\mathbf{Q}_{r,\mathbf{x}}$ by $\mathbf{v}_i$ and λ_i, $i \in \{1, 2\}$. The entry at ith row and jth column of $\mathbf{Q}_{r,\mathbf{x}}$ is:

$$q^{i,j}_{r,x} = \int_\Omega b_r(y)(g_{\hat{a}_i \hat{a}_j, \sigma} * I)(x + y)) dS = ((b_r * g_{\hat{a}_i \hat{a}_j, \sigma}(x) * I(x)) \tag{5.11}$$

where $b_r(\mathbf{x})$ is a step function and we have $b_r(\mathbf{x}) = \begin{cases} 1, & \|\mathbf{x}\| \leq r \\ 0, & \text{otherwise} \end{cases}$, and $g_{\hat{a}_i \hat{a}_j, \sigma}$ is the second derivative of Gaussian along $\hat{a}_i$ and $\hat{a}_j$.

To acquire each entry of $\mathbf{Q}_{r,\mathbf{x}}$, a Fourier transform on Eq.(5.11) is used. To deal with the tube-like shapes with different radii r, a multi-scale method is applied with the OOF, i.e., different r is used as the scale for different width of the shapes.

In [58], the authors used only the eigenvalues λ_i of $\mathbf{Q}_{\mathbf{x},r}$ for the vessel enhancement. In this chapter, we use both the eigenvalues λ_i and the eigenvectors $\mathbf{e}_i$ to form the diffusion tensor, thus the heat could be more concentrated on the tube-like structures.

5.3 Construction of the Metric and Numerical Solutions of the Heat Equation

5.3.1 Construction of the Metric

Now we are considering building a $(d+1)$D metric, d is the dimension of the image, in our case, $d = 2$, and the $3rd$ dimension is not spatial but a radius dimension. We use the same way as described in [6] to construct the metric.

$$D(x,r) = \begin{bmatrix} \hat{D}(x,r) & \mathbf{0} \\ \mathbf{0} & \mathcal{P}_r(x,r) \end{bmatrix} \tag{5.12}$$

where $\hat{D}(x,r)$ is a 2×2 symmetric matrix, this entry is used to describe the spatial anisotropy. In addition, $\mathcal{P}_r(x,r)$ is the isotropic radius potential entry. For a certain scale r, the anisotropic entry $\hat{D}$ can be constructed by the eigenvalues λ_i $(i \in 1,2)$ $(\lambda_2 > \lambda_1)$ and the eigenvectors $\mathbf{v}_i$ of the OOF descriptor:

$$\hat{D}(x,r) = \eta_1(\exp(\beta\cdot\lambda_1(x))\mathbf{v}_1(x)\mathbf{v}_1(x)^T + \exp(\beta\cdot\lambda_2(x))\mathbf{v}_2(x)\mathbf{v}_2(x)^T) \tag{5.13}$$

The radius potential entry can be described by the eigenvalues of the OOF descriptor.

$$\mathcal{P}_r(x) = \eta_2 \exp(\beta\frac{\lambda_1(x)+\lambda_2(x)}{2}) \tag{5.14}$$

Here β is a constant that is controlled by the maximal spatial anisotropic ratio μ, which is defined as:

$$\mu = \max_{x,r} \sqrt{\exp(\beta\cdot(\lambda_2(x,r)-\lambda_1(x,r)))} \tag{5.15}$$

By choosing the maximal spatial anisotropy ratio μ, β is then fixed. $0 \le \eta_1, \eta_2 \le 1$ are two constants that control the space and radius speed. If we would like the heat to propagate faster on the radius dimension, we could choose a bigger $\eta_2 > \eta_1$. Empirically, in this chapter, η_1 and η_2 are always set to be 1. Using Eq.(5.12) as the diffusion tensor in Eq.(5.6), the heat equation can be written as:

$$\frac{\partial u(x,r,t)}{\partial t} = \mathrm{div}\cdot(D(x,r)\nabla u(x,r,t)) \tag{5.16}$$

For the isotropic diffusion, the metric D becomes:

$$D(x,r) = \mathcal{P}_r(x,r) I_d \tag{5.17}$$

I_d is an 3×3 identity matrix.

5.3.2 Solving the Heat Equation

After the construction of the metric, we now solve the heat equation. Generally, the numerical approximation to the solution of the discrete heat equation could be achieved by different schemes [79, 31, 116, 35].

Given the image $I : \Omega \to \mathbb{R}$, suppose that the domain is discretized into $M \times N$ grids and the scale of the third dimension $r \in [R_{min}, R_{max}]$ is K. Then the initial condition becomes: $u^0_{i,j,k} = 1$, $u^0_{i',j',k'} = 0$, $(i', j', k') \neq (i, j, k)$, and (i, j, k) is the initial point given by the users.

Numerical Solution of Isotropic Diffusion

For the isotropic diffusion, we use a backward finite differences scheme, which is also called implicit finite differences scheme. Taking the $3D$ heat equation Eq.(5.2) into consideration, the backward finite difference scheme would be:

$$(\mathrm{Id} - \tau\alpha\Delta)u^t = u^0 \tag{5.18}$$

Id is the identity matrix, τ is the diffusion time, u^t is the heat value after time τ. The Laplace operator Δ can be easily discretized as an $(N \times M \times K)^2$ block penta-diagonal sparse matrix. After the discretization of the Laplace operator Δ, the heat distribution u^t can be acquired by setting an appropriate time step τ.

Numerical Solution of Anisotropic Diffusion

For the anisotropic diffusion, we use a backward discretization scheme designed by Fehrenbach and Mirebeau [35]. The scheme is called Anisotropic Diffusion using Lattice Basis Reduction (AD-LBR). The advantages of this scheme are its non-negativity and sparsity, thus making the solution robust and fast.

For Eq.(5.16), the backward scheme is:

$$\frac{u^t - u^0}{\tau} = \mathrm{div} \cdot (D\nabla u^t) \tag{5.19}$$

To acquire the fundamental solution of Eq.(5.19) within a small time τ, we have:

$$(\mathrm{Id} - \tau \mathrm{div} \cdot (D\nabla))u^t = u^0 \tag{5.20}$$

The symmetric operator $A = \mathrm{div} \cdot (D\nabla)$, with Neumann boundary conditions, can also be defined through the identity

$$\int_\Omega u(x) A u(x)\, dx = \int_\Omega \nabla u(x)^{\mathrm{T}} D(x) \nabla u(x)\, dx, \tag{5.21}$$

for all $u \in H^1(\Omega)$. In order to discretize A, the AD-LBR approximates the contribution of each grid point $x \in \Omega$ to the r.h.s. of (5.21) using a sum of squared finite differences

$$\nabla u(x)^{\mathrm{T}} D(x) \nabla u(x) \approx \sum_{v \in V(x)} \omega_x(v) \left(\frac{u(x+hv) - u(x)}{h} \right)^2, \tag{5.22}$$

where $h > 0$ is the grid scale, $V(x)$ is a set of vectors referred to as the *stencil* of the point x, and $\omega_x(v)$ is the *weight* of the vector v at x. From these stencils and weights, the sparse symmetric matrix of A is then easily assembled. The specificity of the AD-LBR numerical scheme is that the stencils are sparse, with at most 12 elements in 3D, which limits the numerical cost of the method, and that the weights are non-negative, which guarantees discrete maximum principle as well as the robustness of the method.

Their computation involves the construction at each grid point $x \in \Omega$ of an obtuse superbase with respect to the matrix $D(x)$, which is a family $(e_i)_{i=0}^d$ of vectors with integer coordinates such that $|\det(e_1, \cdots, e_d)| = 1$ and $e_i^{\mathrm{T}} D(x) e_j \leq 0$ for all $0 \leq i < j \leq d$. The stencil is then $V(x) = \{e_i \times e_j ; i \neq j\}$ and the corresponding non-negative weights are $\omega_x(e_i \times e_j) = -\frac{1}{2} e_k^{\mathrm{T}} D(x) e_l$ whenever (i, j, k, l) are pairwise distinct, $i, j, k, l \in \{0, 1, 2, 3\}$. The stencil construction is cheap and efficient thanks to arithmetic techniques, thus computation time is dominated by solving the linear systems. See [35] for details.

5.4 Experiments and Results

5.4.1 Experiment Data and Settings

We have tested our method both on synthetic and real images:

Figure.5.1 is an example of a noisy synthetic image (a) of size 100×100. This image is obtained by corrupting the original image with 35% pepper & salt noise. The ground truth in (d) is the original image without adding the noise. Figure.7.12(a) is a 300×300 vessel image and Figure.7.12(e) is a 200×160 road image. The ground truth Figure.7.12(d) and Figure.7.12(h) are labelled manually.

In Figure.5.3, to illustrate the advantage of anisotropic diffusion, we use a 100×100 image with a tube-like structure which has sharp corners.

Figure.7.11a demonstrates a medical image with a catheter. And before diffusion, to preprocess the image I, we first build a potential $\mathcal{P}$ based on the image Laplacian, then we use a sigmoid function Eq.(5.23) on $\mathcal{P}$.

$$I_m(x) = 1 - \frac{1}{1 + e^{\lambda(\mathcal{P}(x)-k)}} \tag{5.23}$$

I_m is the result after preprocessing. Here we set $\lambda = 10$ and $k = 0.5$.

For all the experiment results, the red points and blue points represent the initial points and endpoints respectively. We use the closed red curves to segment the structures boundary. And the blue curves stand for the centerlines. Moreover, the green blocks on some images are used to mark the difference between our method and the Fast Marching Method. In addition, the radius setting for Figure.5.1, Figure.7.12(a) and Figure.7.11 ranges from 0.5 pixel to 5 pixels, and the interval is 0.5 pixel. For Figure.7.12(e) and Figure.5.3, the radius setting is from 1 pixel to 10, with 1 pixel interval. For the first three experiments, the same diffusion time $\tau = 0.01$ is employed.

The time consumption of the heat method is composed by two terms: the time of factorization (the disretization of the Laplacian operator) and the time of solving the heat equation. The factorization process was implemented in C++. Then we solve the heat equation under the scheme of Crane's [23]. We compare our approach with the Fast Marching implementation of Peyré *et al.* [82]. Performance was measured based on a quad core of a 2.8 GHz Intel Core i7 (Table I).

To evaluate the performance of our method, we compute *precision* and *recall*:

$$\begin{cases} recall = \frac{TP}{TP+FN} \\ precision = \frac{TP}{TP+FP} \end{cases} \tag{5.24}$$

Here TP represents the segmentation part which matches the ground truth (GT), FP is the part that do not coincide with the GT, FN stands for the part that is not extracted, i.e. falsely labelled as negative.

5.4.2 Results and Analysis

Isotropic Diffusion on a Noisy Synthetic Image

Figure.5.1 is an example of a noisy synthetic image (a), with a percentage 0.35 of corrupted pixels. (d) is the ground-truth obtained by using the Fast Marching Method on the image without adding noise. From the two results (e) and (f), compared with the ground-truth (d), we can see that the heat method outperforms the Fast Marching Method, because not only

Table 5.1 time consumption and indexes of evaluation (precision & recall)%.

data	heat method				fast marching method		
	factor	solve	precision	recall	time	precision	recall
noisy curve	0.06s	**0.06s**	**94.24**	**97.58**	0.16s	92.97	93.54
vessel	0.48s	**0.73s**	89.54	**90.31**	1.18s	**91.26**	88.62
road	0.2s	**0.42s**	**93.40**	**99.82**	0.69s	92.51	97.91

the centerline but also the boundaries extracted by heat method are smoother than the ones extracted by the Fast Marching Method. Table.5.1 presents the *precision* and *recall* and the corresponding time consumption of these two methods. The total time of heat method is less than the Fast Marching. And the precision and recall index of heat method in higher than the Fast Marching. Additionally, the distance map obtained by Fast Marching (b) is more noisy then the one by heat diffusion (c). The distance maps (b) and (c) illustrate that the heat method is robust in noisy circumstances. This is due to the fact that the heat equation can get fast smoothing by nature. With the initial condition $u(x_0, y_0, t_0) = \delta_{x_0,y_0}(t_0)$, the heat becomes smooth as soon as $t > t_0$. Additionally, in the sense of mathematics, the solutions of the heat equation are characterized by a Gaussian kernel, this can be regarded as a blurring process. This is also the reason why the heat method can be used for filtering issues.

Isotropic Diffusion on a Vessel Image and a Road Image

Figure.7.12 demonstrates the experiments on a vessel image and a road image [1]. We are using the same isotropic metric for the Fast Marching Method and the heat method. For the vessel image (a), one initial point and several endpoints are selected manually. From Table.5.1, we can see the result by the heat method is comparable with the Fast Marching Method. First, the time consumed by both methods remains almost the same. But when we choose other source point, for heat method, there is no need to recompute the factor again because that the factorization is already pre-computed, which saves some time for the heat method, while for the Fast Marching, all data should be recomputed. For the road image (e), there are many abandoned cars on both sides of the road, which may cause much influence in boundary detection. From the results (f) and (g), we can see that the boundaries (highlighted in the green rectangles) that extracted by our method is with higher precision and recall. In other words, our result is less influenced by the cars than Fast Marching. In addition, our method gives a smoother result than Fast Marching.

[1] This image is obtained from the website of GettyImages, it is a DigitalGlobe Worldview-1 satellite image, showing abandoned cars on the road that leads to the top of the Sinjar Mountain Range.

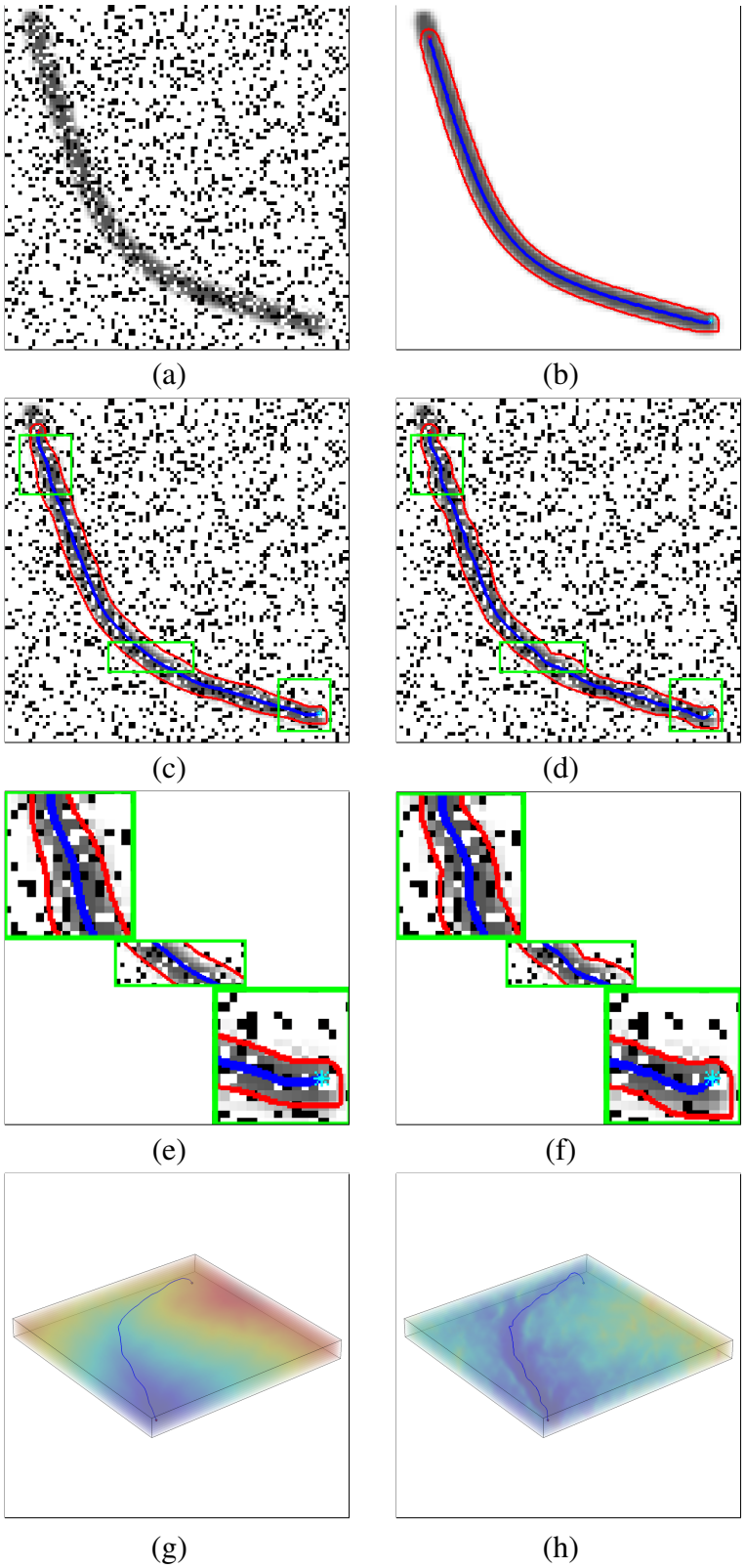


Fig. 5.1 Experiment on a noisy image: (a) original noisy image; (b) and (c) are the distance maps ϕ and the $3D$ minimal path between the seed point and endpoint (transparent visualization) by using the isotropic Fast Marching Method and isotropic Heat Method. (d) is the ground-truth; (e) and (g) are the results by the Fast Marching Method and Heat Method. (f) and (h) are the zoom-in results of the corresponding green boxes in (e) and (g).

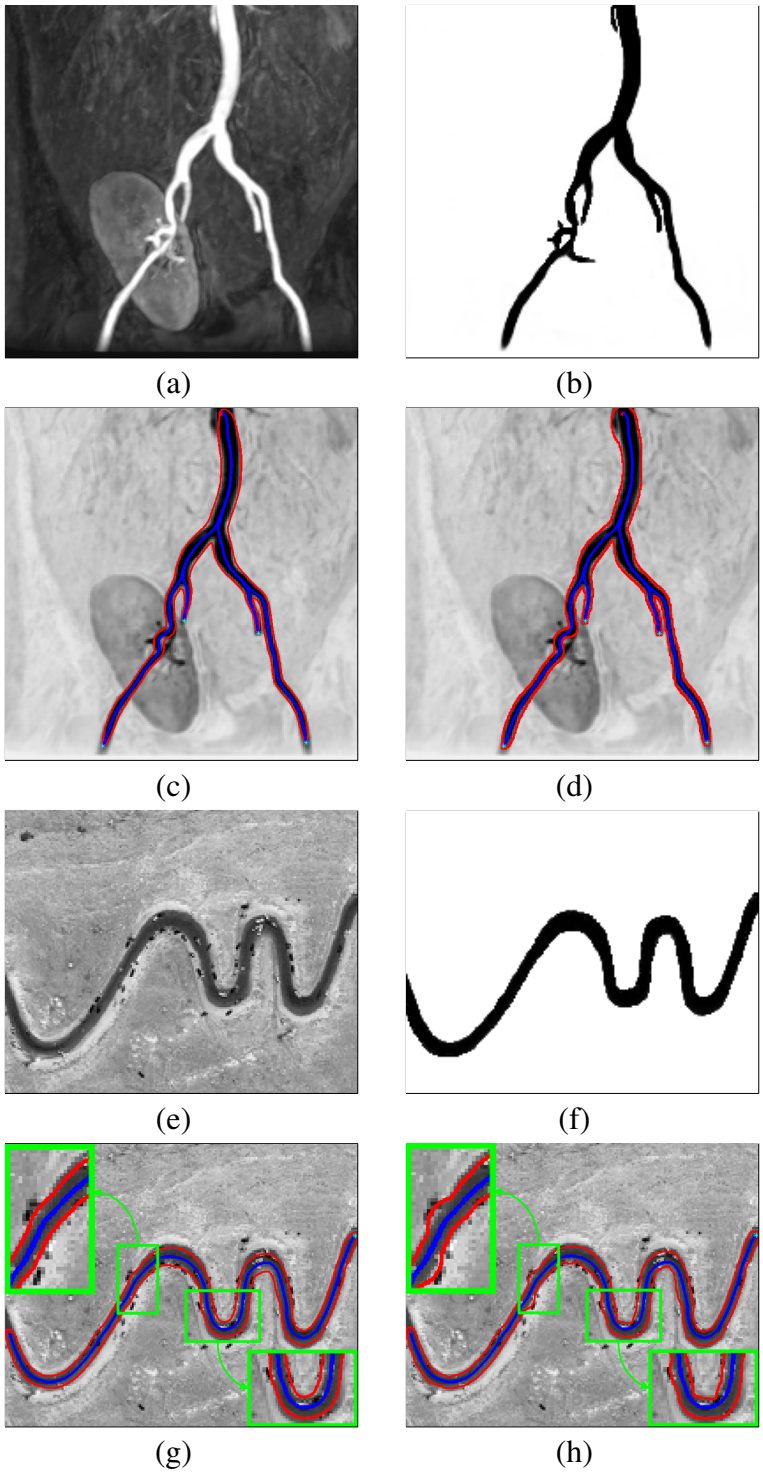
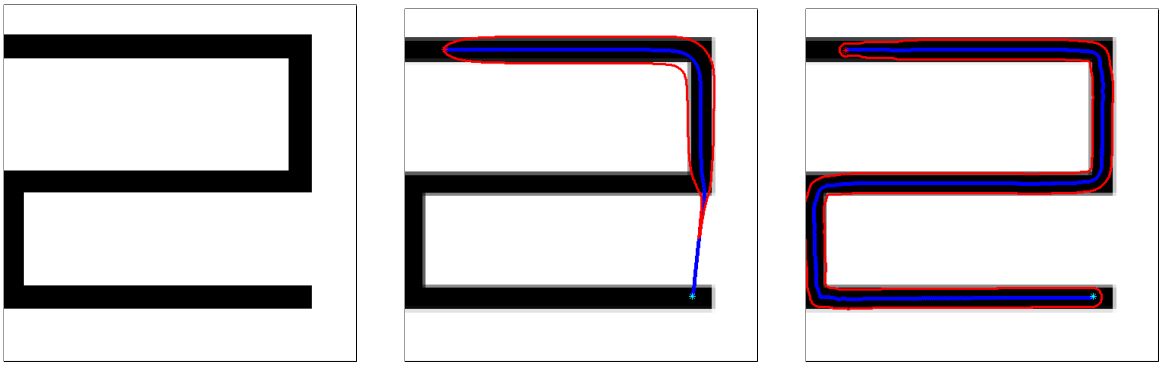


Fig. 5.2 Example on a medical image with several endpoints (row above) and on a road image (row below), from left to right, they are the original images, the result by the Fast Marching Method and the result by the heat method. The green rectangles illustrate the places that the heat method surpasses the Fast Marching Method.

Isotropic and Anisotropic Diffusion on a Tube-like Structure

In Figure.5.3, there is a tube-like structure with several sharp corners. Here we test the difference between isotropic and anisotropic methods. From (b), it is obvious that there is a short-cut on the way back to the initial points, in (c), the backtracking process is totally along the structure without any short-cut. This indicates that by using the anisotropic heat diffusion, the heat can be more concentrated on the direction that the users design.



(a) (b) (c)

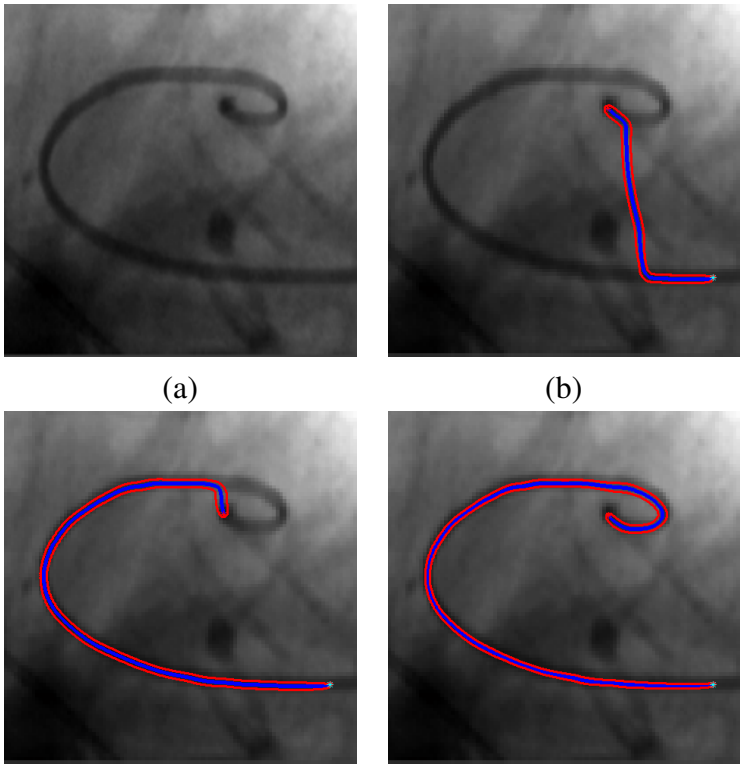
Fig. 5.3 Example on a tube-like structure image, (a) the original image, (b) the result by the isotropic heat method, (c) the result by the anisotropic heat method. There is a short-cut in (b). The radius becomes zero and the centerline in blue is partly superimposed on the boundary in red. In (c), the detection result of centerline and boundary is along the structure without short-cut.

Diffusion on a Medical Image within different time step

The time step τ is an important factor for the heat method. It decides the time for diffusion. According to Eq.(5.3), the distance map ϕ could be approximated only when the diffusion time t is as small as possible. Fig.5.4 demonstrates the effect of diffusion time τ. Different τ are applied. From (b) to (d), τ equals to $0.1, 0.01$ and 0.001 respectively. From the results, it can be seen clearly that the longer the diffusion time is, the more the distance map gets blurred, thus leading to the shortcut on the way when backtracking to the initial point.

5.5 Conclusion

We have proposed a *2D + Radius* model in heat diffusion to extract the centerlines as well as the boundaries of the tubular structures in 2D images. This model integrate the OOF descriptor and diffusion tensor. From the results, we can see that the *2D + Radius* model in heat is very robust and efficient. Besides, our model is less influenced by noise compared with



(a) (b)

(c) (d)

Fig. 5.4 Experiment on a real medical image: (a) original image; (b) (c) (d) are the results generated by isotropic heat diffusion with different time step τ.

the Fast Marching Method. In addition, the anisotropic diffusion does better in controlling the direction than isotropic diffusion. It is fit for detecting the very curved lines and structures.

Part II

Point Flow Model

Chapter 6

A Review of Edge Detection

In the field of image processing and computer vision, edge detection remains a fundamental task which can date back to the 60's. In the views of image processing, an edge is the place where there is a sharp change or discontinuity of intensity or brightness. A discontinuity in image brightness can be assumed to correspond to a discontinuity in either depth, surface orientation, reflectance or illumination [62]. A higher level understanding of edge is defined as a contour that represents a change in pixel ownership from one object to another [71]. Edges play essential roles and act as the basis of a number of tasks such as image segmentation and object recognition [111, 36, 129]. Generally, edge detection focuses on finding the sharp discontinuities and aims at capturing places where important changes occur in images. These discontinuities and changes do not necessarily occur at places where there exists sudden changes in intensity, so it is not sufficient to detect edges and contours by simply using the gradient of image intensity.

Over the past decades, a large number of methods were proposed to detect edges. Chronologically, these methods can be basically sorted into three classes: early and traditional methods, recent approaches, learning-based methods. The early approaches to edge detection aim at identifying points where the image brightness changes sharply. In [11], the authors evaluated a number of low-level edge detectors. In [128], the authors presented an overview of these methods. However, many visually salient edges or contours do not simply correspond to gradients, but also to the texture edges or illusory contours. These complex situations disable the above methods to detect boundaries under more challenging conditions such as textured regions. Therefore, researchers try to apply multiple features such as brightness, color, texture and so on to their edge detectors [94, 117, 71, 3]. By combining these features, the performance of recent edge detectors is significantly improved. Besides, since its occurrence, deep learning has been the hottest topic in computer science. Around the 2010's, with the rapid increase of computer performance and the emergence of big data, the learning-based,

especially deep learning methods have achieved great success for numerous applications in computer vision systems, and edge detection is no exception [28, 30, 8].

In this chapter, we discuss the existing well-known edge/boundary/countour detectors from early stage to recent works. In addition, we also describe the methods for evaluation of the different methods in brief.

6.1 Different Edge Detectors

In this section, we discuss the well-known edge detectors chronologically. Early stage dates back from the 60's to the 90's, a large number of works have been done during this time: [90, 84, 33, 37, 68, 14, 26, 81, 39]. These works are generally based on computing the intensity of the images and they can only find the edges where there is high gradient. Generally, they can be divided into two classes: the gradient-based and Laplacian-based. The gradient-based methods focus on detecting edges via finding the extremal, positive and negative values of the first derivative. The Laplacian-based methods are based on the zero crossings in the second derivative. The methods of early stage are easy to implement but could not satisfy the needs for detecting edges in complex images. Then people are interested in detecting and finding the contours that separate two or more objects which are next to each other or overlapped partially by one or more objects [71, 28, 64, 55, 3, 59, 126, 119, 61]. Among these approaches, some machine learning techniques, such as the clustering methods, are used to define features. The performance of these methods are much better than the classical edge detectors. Nearly at the same time, the neural networks re-rise in the name of deep learning. Almost all of the computer vision tasks can be done by deep learning. The edge detection is no exception [54, 40, 30, 102, 8, 120]. And the edge detection results generated by the deep learning methods achieve the best among all the edge detectors.

6.2 Traditional Edge Detectors

6.2.1 Robert cross operator

Speaking of traditional edge detectors, the first one should be the Roberts cross operator, which was proposed by Lawrence Roberts in his thesis in 1963 [90].

Theoretically, the Robert cross operator consists of a pair of 2×2 convolutional kernels, shown in Figure.6.1. These two kernels are designed to maximize the response to an edge at 45° to the pixel grid, and the two kernels are perpendicular to each other. To detect edges in images, these two kernels should be convolved separately with the input image, thus producing

the gradient in each orientation ($G_x(x,y)$ and $G_y(x,y)$). The magnitude of the gradient at each point can be also obtained. This idea of Robert cross detector is to approximate the image gradient by calculating the sum of the squares of the differences between the diagonally adjacent pixels. The small kernel size makes the Robert cross detector very sensitive to noise.

+1	0
0	-1

(a) G_x

0	+1
-1	0

(b) G_y

Fig. 6.1 The two kernels of Robert cross detector

6.2.2 Prewitt edge detector

Similar to the Robert cross detector, the Prewitt edge detector, proposed and developed by Judith M.S.Prewitt [84] is also a detector based on convolving the original image with a pair of filter both in horizontal and vertical direction. Generally, two perpendicular 3×3 kernels, shown in Figure.6.3, are convolved with the source image to approximate the derivatives in different directions, thus obtaining the gradient of each pixel and its corresponding magnitude.

+1	0	-1
+1	0	-1
+1	0	-1

(a) G_x

+1	+1	+1
0	0	0
-1	-1	-1

(b) G_y

Fig. 6.2 The two kernels of Prewitt operator

The Robert cross detector, Prewitt detector or the Sobel detector *etc.* are based on the first derivative of the images by convolving the original images with different simple masks. They do not consider the nature of edges, and their approximation of the image gradient is rather unrefined. Then methods that are based on the second derivative of the image are proposed, as well as a pre-processing to reduce noise.

6.2.3 Marr-Hildreth operator

The Marr-Hildreth algorithm [68] is based on searching for the zero crossings of the second order derivative of an image. It is realized by convolving the image $I(x,y)$ with the Laplacian of the Gaussian (LoG) function (or kernel) shown as in Eq.(6.1). After getting the filter result,

$g(x,y)$, one can obtain the edges by detecting the zero crossings.

$$g(x,y) = [\nabla^2 G(x,y)] * I(x,y) \tag{6.1}$$

The Marr-Hildreth algorithm do not behave well at the places where the intensity varies. And it generates localization errors, which are problems that users want to avoid.

6.2.4 Haralick Algorithm

The main idea of the Haralick's algorithm [42] is similar to the Harr-Hildreth: both of these algorithms aim to find zero crossings of the second order derivatives. The difference is that the former one smooth the input image by a local bi-cubic polynomial fitting. The advantage of using this technique lies in that it is possible to find an equivalent expression as a function of its parameters during the process of computing the second order derivative of the polynomial.

In 2003, Kimmel and Bruckstein [52] proved that finding the zero crossings of the second derivative along the gradient direction, is in fact the result of minimizing a Kronrod−Minkowski functional while maximizing the integral over the alignment of the edge with the gradient field.

6.2.5 Canny detector

When John Canny proposed the Canny detector in 1986 [14], this edge detector has quickly become the most popular way to detect edges till today. It uses a multi-stage algorithm to detect a wide range of edges in images. Canny detector aims at finding an optimal edge detection algorithm, which means that the algorithm 1) could label as many real edges as possible in images; 2) could localize the edges accurately on the edge; 3) should mark no more than once for an edge and should not mark the possible noise as the edges. No doubt that even today, these properties of optimal edge detectors are still considered as the standard for defining a good detector.

The process of Canny edge detection algorithm contains the following steps:

1. Reduce the image noise. Nearly none of the edge detection algorithm can achieve good result without a filtering pre-process. This process convolves the original data with a Gaussian smooth kernel, and the obtained image is slightly blurred compared to the original data. In this way, single pixel noise would not affect the detection result.
2. Find the intensity gradient of the images. The edges in images are with different directions, so the Canny algorithm uses 4 masks to detect the horizonal, vertical and the two diagonal edges.

3. Apply non-maximum suppression to eliminate the spurious response to edge detection. For each pixel, We retain the highest response and the corresponding direction, thus we obtain the gradient magnitude map, as well as the direction.
4. Track the edges in images. The places where there are high intensity of gradient magnitude are likely to be edges, however, no exact threshold can be applied to threshold the gradient magnitude, so Canny uses hysteresis threshold. There are two threshold for the hysteresis threshold: a high and a low threshold. Suppose that the important edges in images are continuous curves, by using an hysteresis threshold, the Canny detector can track the fuzzy part in these important edges. Starting from the high threshold, the real edges can be labeled. By using the direction information from the above step, we can track all the edges in the whole image. During the process of tracking, the lower threshold is used to track the fuzzy edges.

6.2.6 Canny-Deriche detector

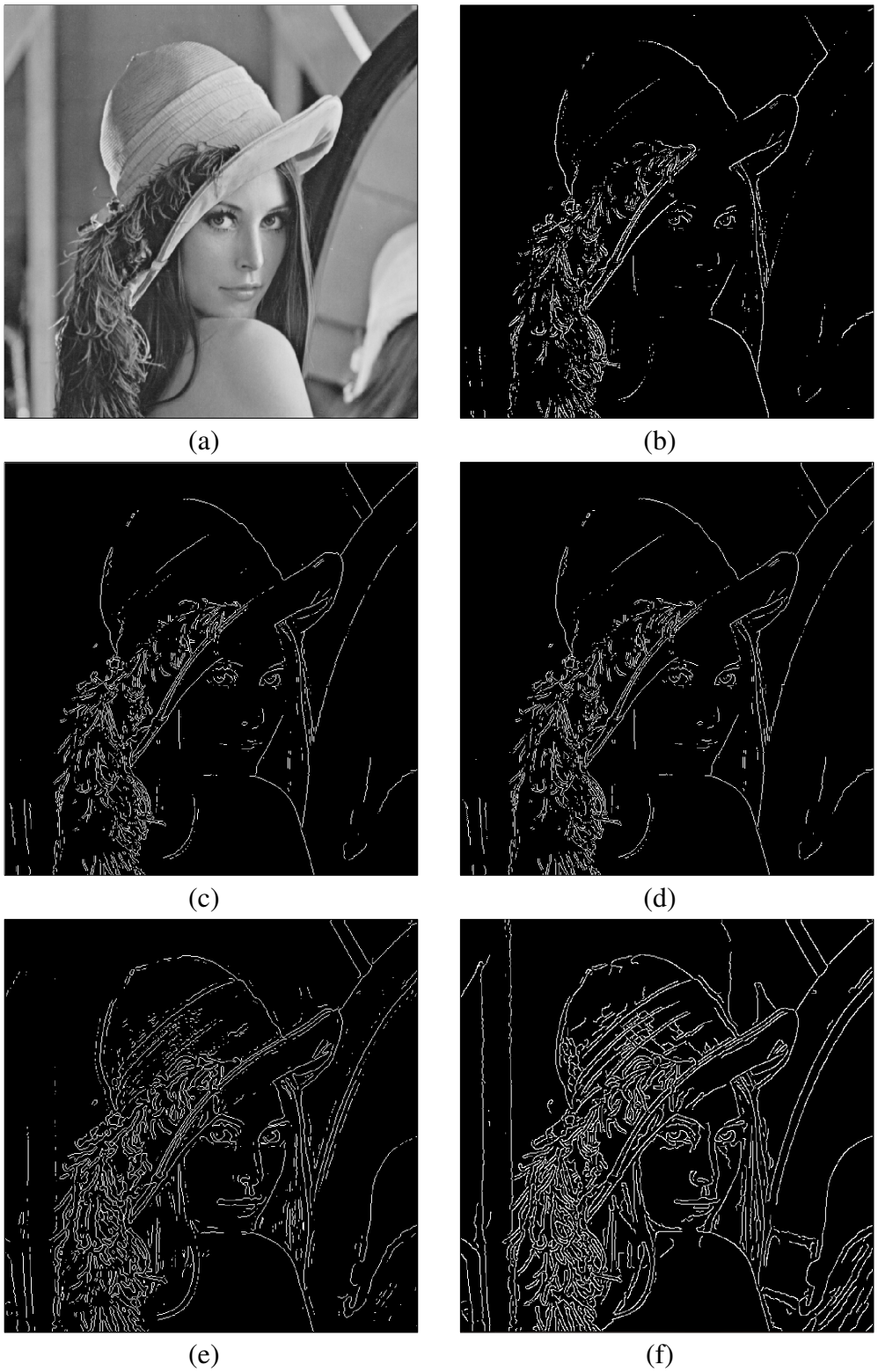
The Deriche algorithm [26] was proposed and developed by R.Deriche in 1987. It is often referred to as the Canny-Deriche edge detector because this algorithm considered the definition of optimal edge detection proposed by J.Canny and was based on the work [14] related to the edge detection. Generally, there are four steps which are similar to the steps in the Canny algorithm. The main differences between the original Canny detector and the Canny-Deriche detector lie at the noise-reducing step. In the first step, the Canny detector utilizes a Gaussian filter to smooth the image but the Canny-Deriche detector uses the IIR filter. The advantage by using this filter is that it is adapted to the characteristics of the image by using only one parameter.

Despite the fact that the above edge detectors are easy to implement, most of their approximation of the image gradient is rather unrefined, especially for the high-frequency places in the images. So nowadays, most of these methods are not being used. But these early approaches have opened up the land for edge detections in image processing.

Next we are going to introduce some recent and modern edge detectors, which behave better than the traditional methods on finding object boundaries of natural images.

6.3 Modern detectors

The incapability (on detecting boundaries of objects in natural images) of the above brightness/intensity/gray level edge models has driven researchers to design detectors which are able to detect the boundaries between textured regions [94].



(a) (b) (c) (d) (e) (f)

Fig. 6.3 From (a) to (f) are the original Lena image, the result of Robert cross operator, the result of Sobel operator, the result of Prewitt operator, the result of Marr-Hildreth operator, the result of Canny detector.

To evaluate the performance of the existing methods, in 2001, Martin *et al.* [69] has built a database, containing 200 training images and 100 test images. Later on, this database was updated and extended, and 200 new test images are provided. This is one of the most widely used database for edge detection and image segmentation. In this thesis, we have also experimented and tested our method on the same database. In addition, most of the modern detectors have more or less related to machine learning methods, such as the Pb detector [71], the gPb detector [3] and so on. So it is necessary to provide data with the corresponding well labeled ground-truth.

Among all the modern methods, several contour detectors stand out and become popular because of their applicability and good performance. We will introduce four representative contour detectors in below.

6.3.1 Probability of Boundary contour detector

In [71], the authors differentiate the problem of boundary detection from the edge detection. In their opinion, a boundary is a contour which distinguish two object, while an edge is usually described as an abrupt change in some low-level feature such as gray level or color level. To detect image boundary, the authors propose an approach which combines different image features. This approach aims at looking for local discontinuities in several feature channels, with a series of orientations and scales, including two intensity features: oriented energy and brightness gradient; one color feature: color gradient; one texture feature: texture gradient. This detector is known as the Probability of Boundary (*Pb*) contour detector.

Speaking of the *Pb* detector, the way to obtain different features should be discussed. First we talk about the oriented energy. The oriented energy was proposed in [74] and extended to detect edges [80]. It is defined as:

$$OE_{\theta,\sigma} = (I * f^e_{\theta,\sigma})^2 + (I * f^o_{\theta,\sigma})^2 \tag{6.2}$$

where $f^e_{\theta,\sigma}$ and $f^o_{\theta,\sigma}$ are two filters at orientation θ and scale σ. This pair of filters are called even and odd symmetric filters. The even filter is actually a Gaussian second-derivative and the odd one is the Hilbert transform of the even one.

Next comes the multiple gradient features. For the *Pb* detector, in [71], the authors introduced a way to compute gradient by setting discs on each pixel (x, y) and comparing the histogram difference of each half-disc g and h using different orientations θ.

$$\chi^2 = \frac{1}{2}\sum_i \frac{(g(i) - h(i))^2}{g(i) + h(i)} \tag{6.3}$$

To compute the texture gradient of a image, the corresponding texton map is necessary. To obtain the texton map, each pixel should be assigned a texton id. In their paper, to obtain the

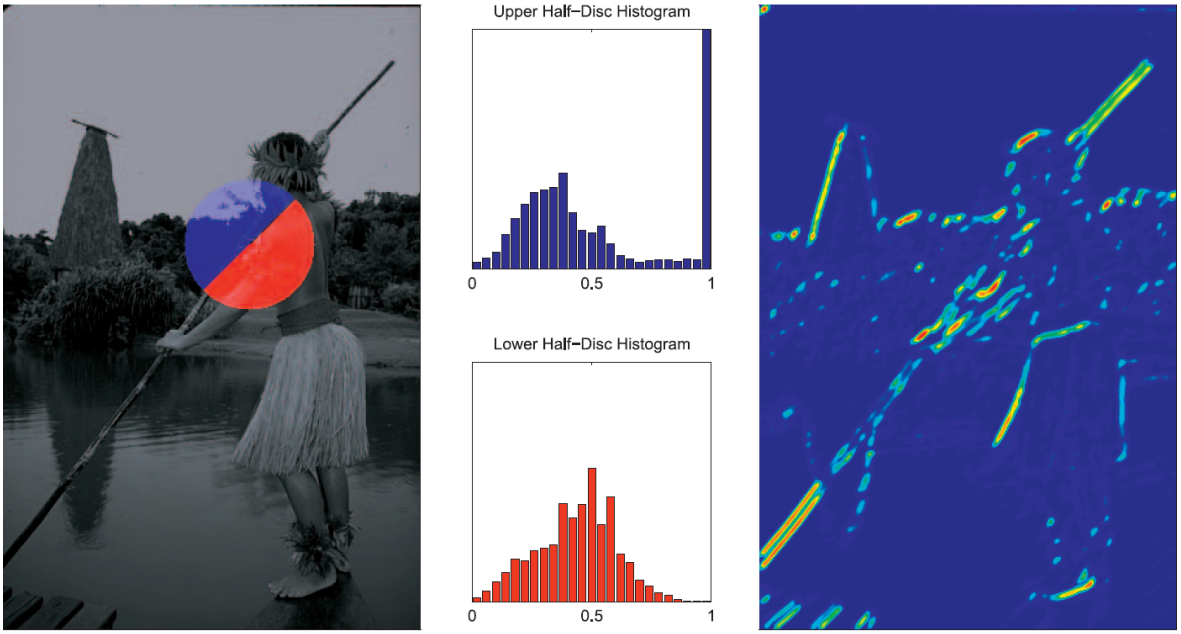


Fig. 6.4 Oriented gradient example [71]

texton id, Martin *et al*. utilized a set of 13 filters to convolve with the training images. This set of filters contains six pairs of elongated, oriented, odd/even filters and an additional center-oriented (difference of Gaussian) filter, shown in Figure.6.5. The even filters are generated by using the Gaussian second derivative and the odd ones are generated by the Hilbert transform of the even ones. For each pixel, they are associated with a 13-dimensional (*13D*) vector of response, of which each dimension stands for the response of its corresponding filter. After obtaining all the responses of the training images, the authors cluster the set of *13D* vectors by using K-means, where $K = 64$ here. The cluster centers define the texton id, which ranges from 1 to K.

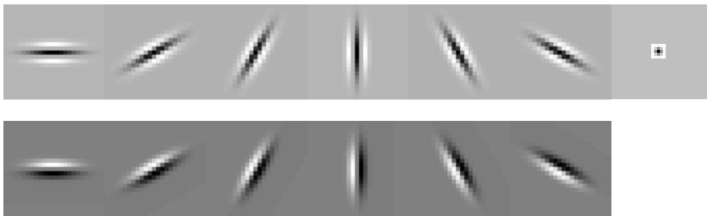


Fig. 6.5 13 filter patterns for creating textons [71]

After obtaining the four feature cues, the authors optimized each cue by using coordinate ascent. Then the optimized cues are combined by using a classifier for modeling the posterior probability of a boundary.

The *Pb* detector does not include the scale information, which is a concern of the edge detectors. Later on, Ren [88] integrates the information from both large-scale detection and small-scale detection on the *Pb* detector, also called *mPb*. This is realized by using multiple disk sizes. The authors in [88] explored a number of multi-scale cues, such as the boundary

contrast, localization and so on. The experimental results on the BSDS500 show that the multi-scale processing improves the performance.

The boundary detection results obtained by [71] and [88] are probability of maps of edges but not real boundaries. In [127], the Zhu *et al.* proposed a grouping method called untangling cycles by introducing a topological formulation. The idea is to exploit the 1D topological structure of salient contours from the 2D image clutter, which in this paper is obtained by thresholding the result of pb detector. Though that the method does not improve the precision-recall performance on the BSDS500 dataset much, but it gives much cleaner detection results.

6.3.2 Globalized Probability of Boundary contour detector

To achieve better performance, the same group as in [71] proposes to combine the multiple local cues into a globalization framework based on spectral clustering [65, 3], called globalized probability of boundary (*gPb*).

Firstly, an oriented gradient signal $G(x,y,\theta))$ from an image I is obtained by using the *Pb* contour detector. Then a multi-scale extension of the Pb detector is introduced. The authors compute the gradient for brightness, color and texture at three scales (the radius of the disc for computing the histogram difference): $[\frac{\sigma}{2},\sigma,2\sigma]$. Then these multi-scale signals are combined linearly as follows:

$$mPb(x,y,\theta) = \sum_s \sum_i \alpha_{i,s} G_{i,\sigma}(x,y,\theta) \tag{6.4}$$

where $\alpha_{i,s}$ are the weights at different scale s and different feature channel i, and it is learned by gradient ascent on the F-measure on the BSDS500.

In [3], the authors use eight orientations of θ range from 0 to π. The maximum response among the eight orientations could be regarded as the probability of boundary:

$$mPb(x,y) = \max_\theta \{mPb(x,y,\theta)\}$$

After combining the multi-scale cues, a spectral clustering technique is used to obtain the signals of the most salient curves. Here the spectral clustering works as the most important step for globalization. A sparse symmetric matrix which describes the affinity of two pixels of which the distance is within a fixed radius r was designed as follows:

$$W_{i,j} = \exp(-\max_{p\in\bar{ij}}\{mPb(p)\}/\rho) \tag{6.5}$$

$\bar{ij}$ is the line segment connecting i and j and ρ is a constant.

Let $D_{ii} = \sum_j W_{ij}$ and solve the system $(D - W)\mathbf{v} = \lambda D\mathbf{v}$. The generalized eigenvectors carry the contour information and each eigenvector $\mathbf{v}_k$ can be regarded as an image. Convolve $\mathbf{v}_k$ with a series of Gaussian filters at different orientations θ and the signals are $\{\nabla_\theta \mathbf{v}_k(x,y)\}$. These signals are then combined to offer the spectral boundary detector:

$$sPb(x,y,\theta) = \sum_{k=1}^{n} \frac{1}{\sqrt{\lambda_k}} \cdot \nabla_\theta \mathbf{v}_k(x,y) \tag{6.6}$$

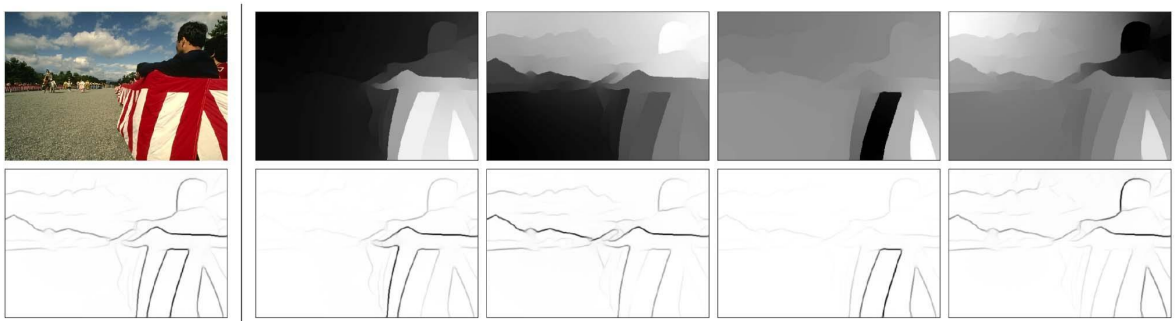


Fig. 6.6 Top row from Column.2 to Column.5 are the first four generalized eigenvectors that are used in Eq.(6.6), bottom row from Column.2 to Column.5 are their corresponding maximum gradient response. Left-bottom shows the maximum response of sPb over all orientations [3].

The signal *mPb* contains edge information on the whole domain, and *sPb* conveys different information. The signal *sPb* only cares about the most salient curves. A simple combination is sufficient to benefit from both signals. And the final globalized probability of boundary is as follows:

$$gPb(x,y,\theta) = \sum_s \sum_i \rho_{i,s} G_{i,\sigma(i,s)}(x,y,\theta) + \gamma \cdot sPb(x,y,\theta) \tag{6.7}$$

6.3.3 Sketch Tokens

Sketch tokens [61] are a series of learned features using supervised mid-level information. The idea is interesting: 1. it converts the contour detection (binary classification) into a multi-classification problem, which reduces the difficulties for classifiers. 2. it transfers the large number of types of contours into a limited number of labels. The tokens are in fact a set of representative features describing the distribution of edges in a $M \times M$ region, where in the original paper, $M = 35$. The method contains two steps: 1. defining sketch tokens; 2. detecting contours by using sketch tokens.

Defining Sketch Tokens

Given a set of images I and their corresponding well labeled GroundTruth images GT, to define the sketch token classes, one should first divide each GT image into $M \times M$ pieces (or patches). Only those patches whose center pixel is covered by an edge could be taken into account. To avoid deviations caused by slight shift in edge placement, the authors use the Daisy descriptors [107] to compute the features of each pixel. Then they cluster the features by using the K-means algorithm, where $K = 150$. These classes identify the different shapes or distributions of sketches in the patches.

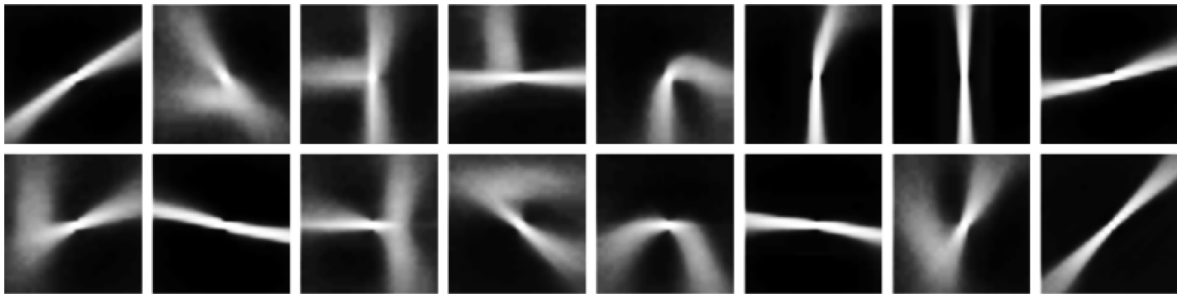


Fig. 6.7 Sketch Tokens examples [61]

Detecting Sketch Tokens

After defining and obtaining the sketch token classes, one wants to detect the frequency of occurrence of the sketch tokens in images. The detecting process can be done by two steps: feature extraction and classification.

For feature extraction, the authors use two types of features, one is channel features and the other is self-similarity features. The authors use a set of feature channels in each patch x_i, including color, gradient, as well as the oriented gradient. To compute the three color channels, they use the CIE-LUV color space. And three normalized gradient with different scales and orientations are also utilized. All the channels are post-blurred with a $\sigma = 1$ Gaussian kernel.

Since the edges do not only occur at places where there are changes in intensity, but also occur at the boundary of textured regions, the texture information should be taken into consideration. This feature is realized based on self-similarity proposed in [101]. For a 35×35 patch x_i, the texture information is computed on a $m \times m$ grid cell, $m = 5$. Therefore, there are 7×7 grid cells for a patch x_i. The self-similarity feature is defined as:

$$f_{i,j,k} = s_{j,k} - s_{i,k} \tag{6.8}$$

where i and j are grid cells and $s_{i,k}$ is the sum of the grid i of the kth channel. The last step for feature extraction is to combine the channel features and similarity features for every image patch.

Then a random forest is used as classifier. The aim is to compute the probability of a patch of image belonging to each tokens.

The probability of the center pixel of a patch x_i belonging to an edge is computed as:

$$e_i = \sum_j t_{ij} = 1 - t_{i0} \tag{6.9}$$

where t_{ij} means the probability of x_i belonging to label (token) j, $t_{i,0}$ means that x_i belongs to no token.

Finally, a non-maximal suppression is used to find the contours with maximal response.

6.3.4 Structured Forest

In [29, 30], the authors proposed a generalized structured learning method for edge detection and the structure information is obtained by a random forest framework. The structured labels which are learned by decision trees are used to determine the split function. The final edge map is computed by predicting a patch of edge pixel labels which converges on the image domain.

Similar to the sketch token work of Lim *et al.* [61], this work also converts the binary classification work into a multi-classification task. The difference is that they are predicting the local structure of image patches directly.

Before introducing the structured forest, we would like to briefly review the decision tree and random forest.

Decision Trees

A decision tree works as a classifier. Generally, a decision tree contains one root node, several inner nodes and leaf nodes. The leaf nodes determine the detection result and the other nodes correspond to different tests of attributes. Apparently, the generation of a decision tree is recursive.

For a sample $x \in \mathcal{X}$, a decision tree $f_t(x)$ classifies it by branching down until getting to a leaf node. Each inner node in the tree has a split function:

$$h(x, \theta_j) \in 0, 1 \tag{6.10}$$

$h(x,\theta_j) = 0$ means that on the node j, the sample x should be sent to the left, otherwise, it should be sent to the right.

Random Forest

A random forests is an ensemble of a number of independent decision trees. The predictions from each decision tree is combined by using an ensemble model into a single output.

Decision Tree Training

The aim of training a decision tree is to obtain the parameters θ_j of the split function in Eq.(6.10). The information gain criterion is defined as follows:

$$I_j = I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R) \tag{6.11}$$

where $\mathcal{S}_j^L = \{(x,y) \in \mathcal{S}_j | h(x,\theta_j) = 0\}$, $\mathcal{S}_j^R = \mathcal{S}_j \setminus \mathcal{S}_j^L$. The parameters θ_j are trained to maximize the information gain I_j. The training process ends when a maximum depth is reached.

The information gain of multi-class classification problem is defined as:

$$I_j = H(\mathcal{S}_j) - \sum_{k \in \{L,R\}} \frac{|S_j^k|}{|S_j|} H(S_j^k) \tag{6.12}$$

where $H(\mathcal{S}) = -\sum_y p_y \log(p_y)$ is the Shannon entropy and p_y is the frequency of $\mathcal{S}$ of label y.

Figure.6.8 illustrates the decision tree splits, where a set of structured labels are classified into two labels.

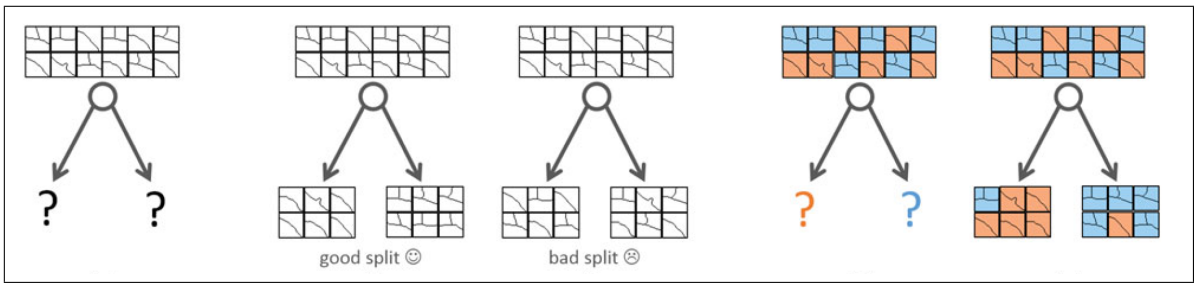


Fig. 6.8 Decision tree node splits according to structures [29]

The goal of structured forest is to label each pixel a signal and see if it is belongs to an edge or not. It is realized by using the structured forest to predict the local segmentation masks for given image patches.

Testing on the BSDS dataset, it can be concluded that the feature-based methods have some limitations: they do not include object-level (higher-level) information, thus result in a

lot of false detection. This is the main reason that researchers try to use the Convolutional Neural Networks (CNN) to search for high-level, object-level and multi-scale information.

6.4 Edge Detectors by Deep Learning

As we mentioned before, motivated by the overwhelming trend of deep neutral networks, many researchers begin to use the deep features to do image processing and computer vision tasks, such as image classification [57], object detection [41] and so on.

Recently, there are also a lot of CNN-based edge detectors, such as the N^4-Fields [40], DeepEdge [8], DeepContour [102], HFL [9], HED [120] and so on.

Before getting into the edge detectors trained by deep learning models, let us give a brief introduction of the convolutional neural networks (CNNs). The CNN is a kind of feed-forward artificial neural network, it is highly favored because of its excellent performance on processing large number of images.

CNN is a multi-layer neural network, every layer has multiple 2D surfaces which are composed by independent neurons. Generally, CNN has two main types of layers: convolutional layer and pooling layer.

The convolutional layer is composed by several convolutional units. The parameter of every convolutional unit is obtained by backward propagation optimization. The use of convolutional layer is that it can be used to extract different features of images, multiple convolutional layers can extract more complicated features from lower features.

The pooling layer is a kind of downsampling process and is used to reduce the data space thus reduce the parameters and computation, in the meanwhile, it controls the overfitting to some certain extent. In general, there are two ways of pooling: max-pooling and mean-pooling. Max-pooling uses the biggest value of the pooling window as the sample value and mean-pooling takes the average value of the pooling window as the sample value.

6.4.1 Neural Network Nearest Neighbor Fields (N^4-Fields)

It is intuitive to think of using the local gradient, texture changes to find edges in images. The idea of N^4-field is also intuitive [40]: there are numerous patches in images, the features of each patches can be obtained by using the CNN; then by looking up to the dictionary and searching for the similar edge patches and ensembling the similar edge information, the final result is formed. Figure.6.9 shows the flowchart of N^4 field. Because of the stronger features, the performance is improved.

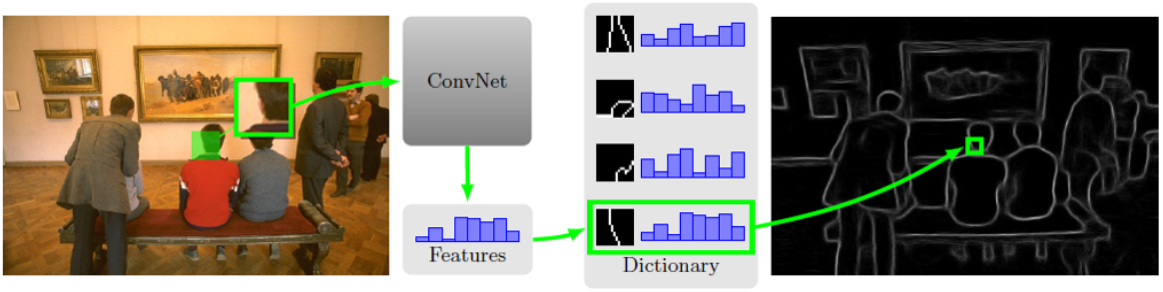


Fig. 6.9 CNN architecture of DeepEdges [40]

6.4.2 DeepEdge

The DeepEdge [8] extended the N^4 fields. First, the candidate contours are obtained by Canny edge detector. Then multi-scale (4-scales) patched are built on these candidates. Take these patches as input and run through the 2-branch CNN: one branch is for classification and the other is used for regression. Finally, each candidate will get a probability of being on the true edges. The model for training the dataset used in consists of five convolutional layers and a bifurcated fully-connected sub-network. Figure.6.10 shows the multi-scale DeepEdge network architecture.

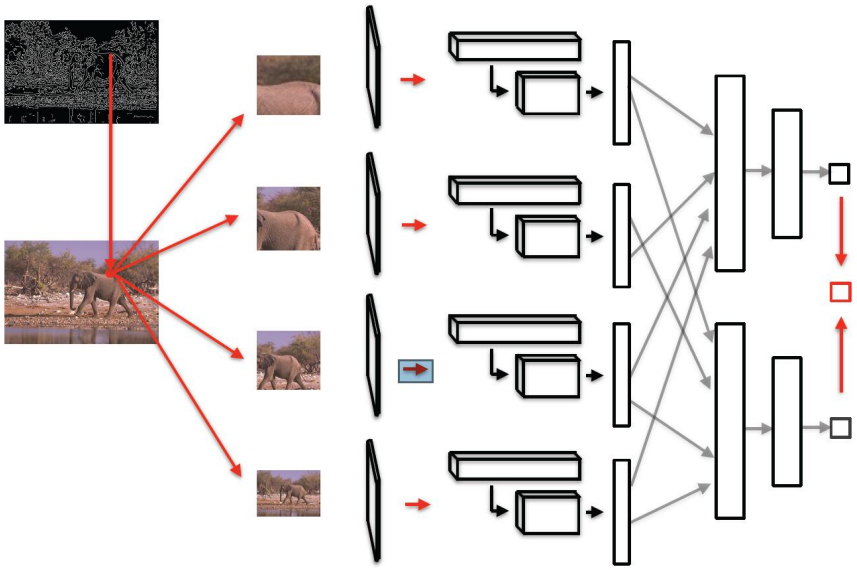


Fig. 6.10 Architecture of DeepEdge [8]

6.4.3 DeepContour

The work of DeepContour [102] is also based on image patches. It is worth mentioning that the authors utilized the sketch tokens [61] to obtain the structure of contours and use these tokens to classify image patches. The CNN architecture is composed by six layers: the

first four are convolutional and the rest two are fully-connected. Figure.6.11 displays the architecture of DeepContour.

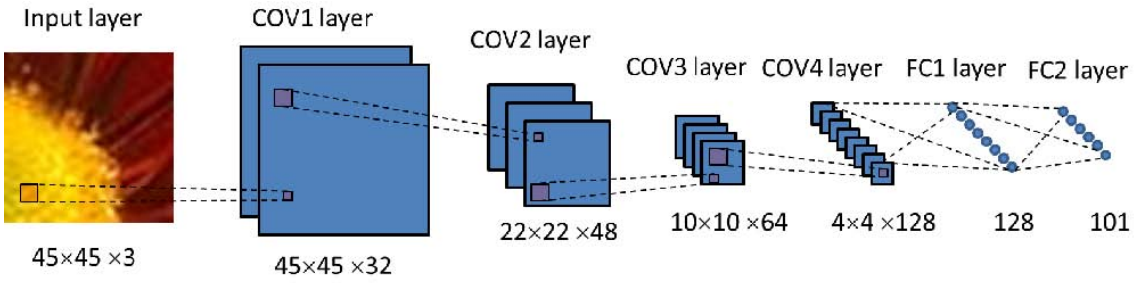


Fig. 6.11 CNN architecture of DeepContour [102]

6.4.4 High-For-Low (HFL)

The work of [9] high-for-low (HFL) also use CNN to differentiate the possible edge candidate points. The authors use VGGNet model (a deep CNN model proposed by the visual geometry group of Oxford) which is trained by high-level semantic information. It improves the quality of edge detection. Figure.6.12 shows the architecture of HFL.

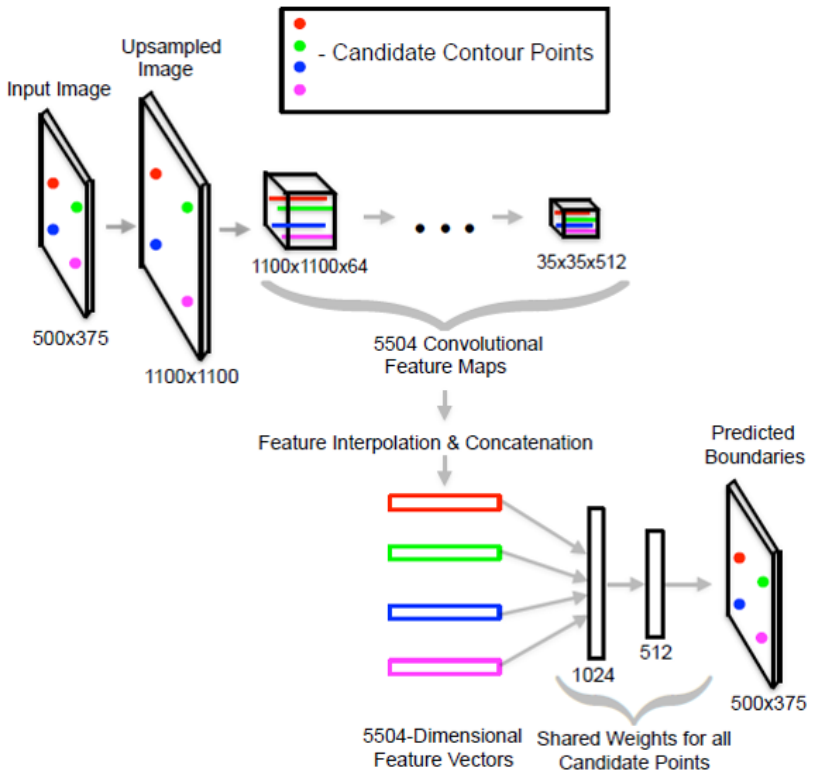


Fig. 6.12 CNN architecture of HFL [9]

The experiments on the BSDS dataset shows that the above CNN-based edge detectors make some progresses, but they do not really achieve the human behaviors. The above methods are based on local strategy and aim at one patch, they do not necessarily take full advantage of higher-level information. In addition, this kind of pixel-level image patch classification methods is very time-consuming. Limited by the image patches, they do no model large context, thus affect the performance of algorithms.

6.4.5 Holistically-Nested Edge Detection (HED)

Instead of using local patches-based method as the above methods, the holistically-nested edge detection (HED) [120] utilize a global image-to-image fashion to detect edges. In other words, this method no longer operates on patches but on the whole images, thus facilitate to acquire high-level information.

Meanwhile, the HED utilizes a deep supervision process which is realized by inserting a side output layer at the side of the convolutional layer. The side output layer is obtained by upsampling the output of convolutional layer to get a map which is the same size as the original image, shown in Figure.6.13. Then the obtained maps are used to compute the cost with the groundtruth. The loss of multiple side output are reconverted to their corresponding convolutional layer thus avoid the loss of gradient.

So far, the HED outperforms all the other edge detectors on the BSDS Dataset.

6.5 Evaluation Methodology

After obtaining the edges, how to evaluate the performance of the edge detectors becomes a problem. Some people compare the detection result in a subjective way: the edges look better or worse than the other ones, but this is not convincing. So it is very necessary to propose some evaluation methods that compare the quality of different edge detectors. During decades, a lot of researches on the evaluation of edge detectors have been done.

One of the earliest evaluation of edge detector was done by Fram and Deutsch [37]. In their paper, Fram and Deutsch proposed several characteristics for the quantitative evaluation of edge detection algorithms, such as the edge orientation biases, the edge detection under noise, the ability to detect blurred edges and curved edges, and the complexity of the algorithm. In their opinion, the principle feature for edge detection evaluation is that the comparative results must be quantitative. To evaluate quantitatively, they proposed a measure that:

$$F = \frac{\sum_{i=1}^{I_A} \frac{1}{1+\alpha(d(i))^2}}{\max\{I_A, I_I\}} \tag{6.13}$$

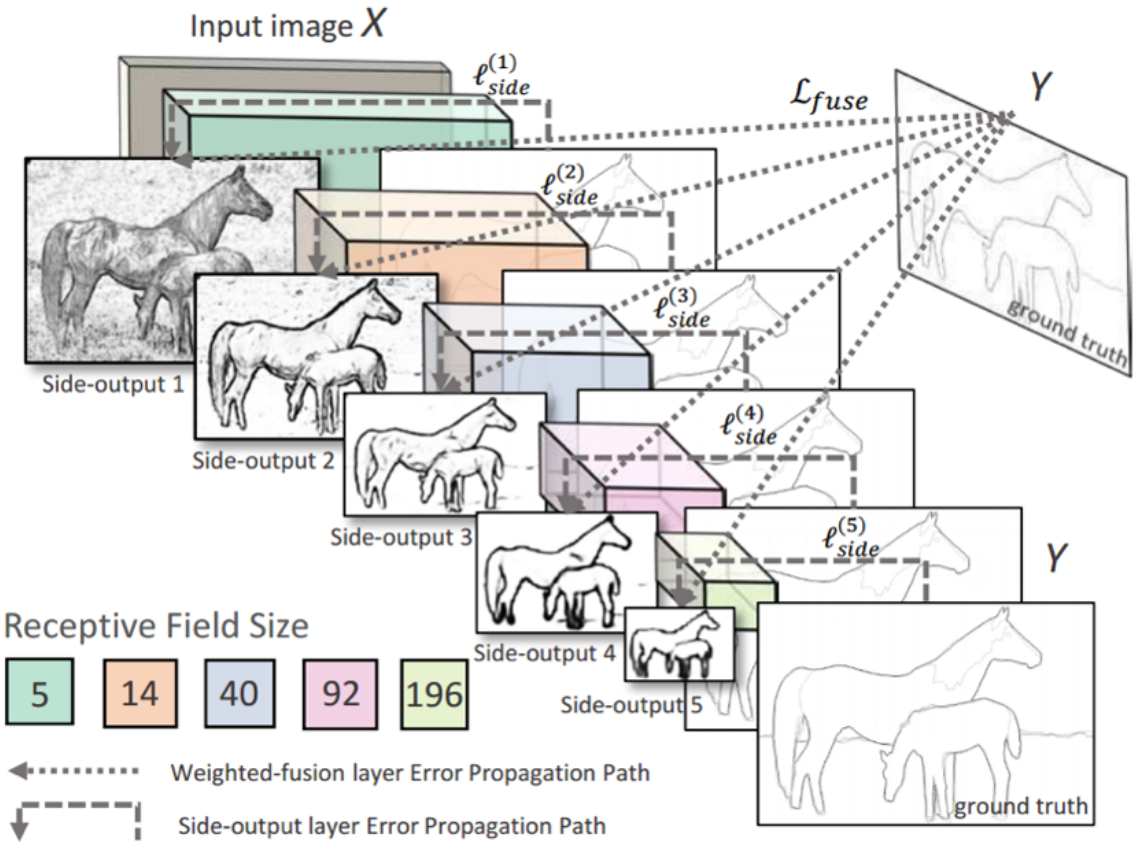


Fig. 6.13 CNN architecture of HED [120]

where I_A is the edge pixels that are detected, and I_I are edge pixels of the groundtruth.

Abdou and Pratt [1] proposed Pratt's figure of Merit criterion for either synthetic or real images evaluation. Their proposed technique can be used optimally to design a variety of small and large mask edge detectors. In [53], Kitchen and Rosenfeld evaluated the edge detectors by using a criterion based on edge coherence. There is no need of the ideal edge position. Ramesh and Haralick [85]'s evaluation method is based on the probabilities of mis-detection and false alarm.

The evaluation method proposed by Heath *et al.* [44] was motivated by: 1 the experiments should be done on real images, 2. the evaluation results should also comply with our subjective cognition.

The precision and recall are general indexed for edge detection. Normally, it is pixel-level evaluation:

$$\begin{cases} precision = I_{correct}/I_{detect} \\ recall = I_{correct}/GT \end{cases} \tag{6.14}$$

where $I_{correct}$ represents the sum of the pixels that are correctly detected by the detector, I_{detect} stands for the total number of pixels that are detected, and GT means the groundtruth. The F-measure is also widely used for edge detection.

$$F = PR/(\alpha R + (1-\alpha)P) \tag{6.15}$$

where P stands for precision and R represents recall, α is set to be 0.5 generally.

The evaluation methods above are reasonable in certain cases, so far there is not a rigorous theory for the evaluation method. In Chapter. 8 of this thesis, we also proposed our own evaluation method that is suitable for the dataset that we build.

Chapter 7

A Model for Automatically Tracing Object Boundaries

In this chapter, we propose a novel method for tracing object boundaries automatically based on a process called "PointFlow" in image induced vector fields. An ordinary differential equation describes the movement of points under the action of an image-induced vector field and generates induced trajectories. The trajectories of the flows detect and integrate edges and determine object boundaries. In addition, the PointFlow process can be applied to infer certain illusory contours.

We tested our method on a real image dataset and synthetic images. Results show that the PointFlow method is not only good at providing precise and continuous curves, but also has an ability to infer illusory contours. The experimental results clearly exhibit the robustness and effectiveness of the proposed method.

7.1 Introduction

We have introduced in Chapter 6 that edge detection focuses on finding the sharp discontinuities and aims at capturing places where important changes occur in images. It plays an important role in image processing and computer vision. Over the past few decades, numerous methods were proposed to detect edges, such as Sobel operator, Prewitt operator, Canny and the Haralick edge detector, and so on [128]. After edge detection, an edge integration process is needed in order to find object boundaries for segmentation and analysis.

In [34], a semi-automatic method to detect boundaries of objects is proposed by using simulation of particle motion in an image induced vector field. Users of the method were required to provide the location of starting points and the number of time steps to be carried out. In addition, users needed to adjust some parameters to achieve good results. In [66],

a similar method is used to track and detect the most important edges, in order to produce artistic one-liner renderings of objects appearing in images. In [63] a *c*-evolute model is presented for the particle motion in [34] to approximate the edge curves. More recently, Kimmel and Bruckstein [52] proposed to incorporate the Haralick/Canny edge detector into a variational edge integration process.

In this chapter, we are interested in providing a process of tracking object boundaries automatically. Imagine that a magnetic vector field is induced by the image. As the input, a number of points which are placed randomly in the image and are considered as tiny magnetized iron pieces. The iron pieces move following the direction of the magnetic field. We record the trajectories of these points and use them to obtain the edges on the images. The movement of the points can be described by an ordinary differential equation, which describes the "point flow".

After suitably designing the vector field, we initiate the flow from a number of random points in the image plane. Guided by the vector field, the trajectories of these points are attracted towards and along the significant edges in the image. Note that the flow process will not end unless a stopping criterion is met. An iterative process allows to refine the trajectories and make the result more robust.

Additionally, the PointFlow process can be extended to infer illusory contours under the assumption that the illusory contours are simply missing parts of the borders of regular shapes and it can be approximated by arcs of certain adapted curvatures.

The contribution of this paper is a new way based on point flow for automatical edge detection and integration. The edges that are extracted are continuous and precise. Moreover, the PointFlow process can also be applied to the inference of illusory contours.

The chapter is organised as follows: Section. 7.2 introduces the core algorithm of the point flow method, including the construction of the vector field, the edge detection and integration modules. Section. 7.4 presents some experiments results on BSD500 dataset [70] and compares with classical edge detection methods. Section. 7.5 describes how the PointFlow model can be used to infer the illusory contours. Section. 7.6 displays some illusory contour inference results by using the point flow method.

7.2 Point Flow Method

The PointFlow process is based on an ordinary differential equation (ODE) which describes the trajectory of a moving point driven by a velocity vector field $\mathbf{V}$. In a $2-dimensional$

domain, the PointFlow process is defined as follows:

$$\frac{d(P(t))}{dt} = \mathbf{V}(P(t)) \tag{7.1}$$

where $P(t) = (x(t), y(t))$ is a curve which describes the location of a moving point at time t and starting from a given point p_0 at time $t = 0$. Within a certain time ΔT, the trajectory of P driven by the vector field $\mathbf{V}$ will be recorded. Here $\mathbf{V}$ is the vector field that controls the speed and direction of the points.

In order to use the PointFlow to trace the object boundaries, the vector field $\mathbf{V}$ is induced by a given input image and its design is crucial factor for the correct tracing of object boundaries. The details of designing the vector field are given in the following section. The processing chain of PointFlow method as applied to a very simple input image is shown in Fig.7.6.

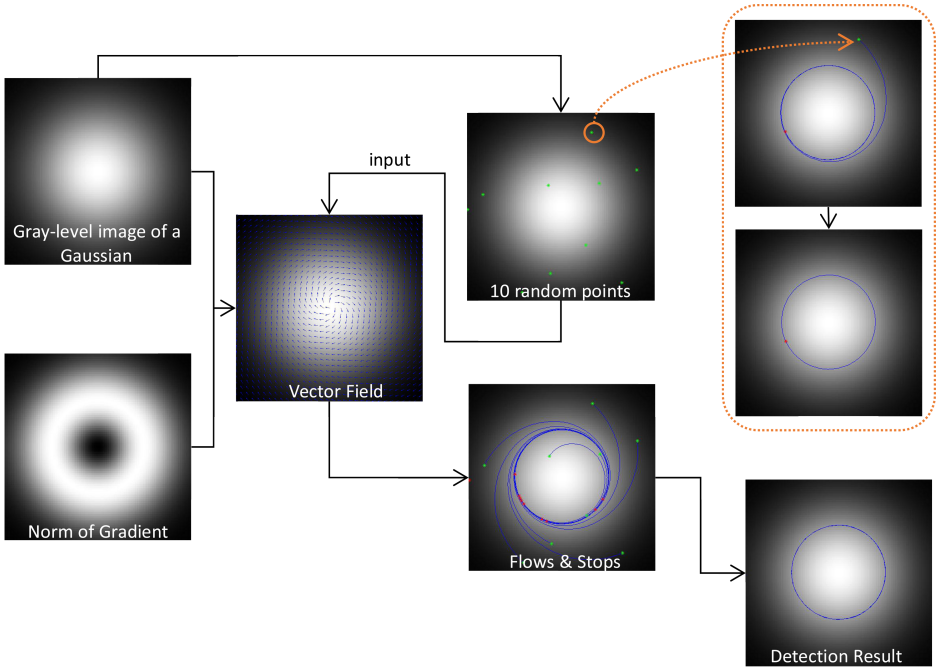


Fig. 7.1 The flowchart of the PointFlow. The left column presents a 2D Gaussian image and the magnitude of its gradient. A linear combination of the gradients of both these images are used to design the vector field $\mathbf{V}$. In the example above, 10 random points are used as the starting points, moving under the effect of the vector field. The top-right dashed-box shows the movement of a single point, the blue curve is the trajectory and the red point is the end point. The flow terminates a point hits its own trajectory. Then, we re-initiate the flow from the end (red) point to obtain a complete and precise contour of the Gaussian image. All other points move in the same way. Right below shows the result.

7.2.1 The Design of the Vector Field

Given an image $I : \Omega \to \mathbb{R}$, based on Eq.(7.1) and a starting point $p_0 = P(t_0)$, under the action of the vector field $\mathbf{V}$, we wish to have a trajectory l_{p_0}. To draw l_{p_0} towards and on the object boundaries or edges, we need to construct a vector field which not only directs the flow towards the image edge, but also keeps the flow on the edges. Hence we design two components $\mathbf{V}_1$ and $\mathbf{V}_2$ to form the vector field: $\mathbf{V}_1$ must push the points towards high gradient places, and once near an edge, $\mathbf{V}_2$ should cause the points to move along the edge. The resulting vector field will combine the two vector fields $\mathbf{V}_1$ and $\mathbf{V}_2$: $\mathbf{V} = \zeta \cdot \mathbf{V}_1 + \xi \cdot \mathbf{V}_2$.

Since the gradient vector is always orthogonal to the edge orientation, $\mathbf{V}_2$ can be defined as a vector orthogonal to the gradient of the image I, because it is directed along edges, so we take $\mathbf{V}_2 = \nabla I^{\perp}$. For $\mathbf{V}_1$, a second-order derivative of I will be used, namely the gradient of the gradient of the image $\mathbf{V}_1 = \nabla(\|\nabla I\|)$. The advantage of the second-order derivative lies in that it generates a field that drives points towards the maxima of the gradient from both sides of the edge. Lower-order derivatives clearly can not provide such information. This term is similar to one of the terms in the snake model evolution equation [50]. In the snake model this term is called "image force" that pushes the given curve towards the significant edges, which correspond to the desired features. The combination of these two terms is the vector field desired. Hence $\mathbf{V}$ can be written as follows:

$$\mathbf{V} = \zeta \cdot \nabla\|\nabla I\| \pm \xi \cdot \nabla I^{\perp} \tag{7.2}$$

where ζ and ξ are two constants: $0 \le (\zeta, \xi) \le 1$.

To be specific, we can write $\mathbf{V}$ explicitly as follows:

$$\mathbf{V} = \zeta \cdot \frac{1}{\sqrt{{I_x}^2 + {I_y}^2}} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix} \pm \xi \cdot \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{7.3}$$

The constants ζ and ξ control the relative strengths of the two terms $\mathbf{V}_1$ and $\mathbf{V}_2$, and can be used to balance these two components to make $\mathbf{V}$ appropriate and suitable for detecting edges. Empirically we took $\zeta = \xi = 0.5$, which fit for most of the cases. $\mathbf{V} = \frac{1}{2}(\mathbf{V}_1 \pm \mathbf{V}_2)$. The signs $\pm$ can be used to change the direction of $\mathbf{V}$ by 180°, which turns out to be an important step for edge integration. We shall use $\mathbf{V}$ to represent $\frac{1}{2}(\mathbf{V}_1 + \mathbf{V}_2)$ and $\mathbf{V}'$ to represent $\frac{1}{2}(\mathbf{V}_1 - \mathbf{V}_2)$. Hence:

$$\mathbf{V} = \frac{1}{2}(\mathbf{V}_1 + \mathbf{V}_2) \tag{7.4}$$

$$\mathbf{V}' = \frac{1}{2}(\mathbf{V}_1 - \mathbf{V}_2) \tag{7.5}$$

7.2.2 Edge Integration by Voting

After building the vector field, we choose N random points from the image as starting points and let them move according to Eq.(7.1) for a same period of time ΔT. After ΔT, we get N trajectories: $\{l_n\}_{\{1\leq n\leq N\}}$, each l_n represents one trajectory. Then a voting process can be employed to extract the edges.

In terms of the voting process, we are inspired by [92], where the essence of the voting process is to extract geodesic paths from the end points chosen automatically to the source point provided manually by the user. The points located on these paths can be used to define the density:

$$\mu(p) = \sum_{n=1}^{N} \delta_p(l_n) \tag{7.6}$$

where $\delta_p(l) = 1$ if pixel p is crossed by path l, or $\delta_p(l) = 0$, N being the number of paths. Pixels which have higher densities are more likely located on the edges.

$\mu(p)$ determines the importance of each pixel on the trajectories. A threshold β for $\mu(p)$ is also set. We retain only as elements in the final result the pixels with a number of paths above the prescribed threshold β, where $\beta \in [0, \max \mu(p)]$. The pixels with higher densities $\{p_i\}_{0\leq i\leq M}$ have more possibilities lying on the edges, and they are called the important pixels, M is the number of these pixels.

After acquiring the most important pixels, we need to integrate them to form the complete edges. It can be realized by cutting off the extra edges from the result of $\{l_n\}_{\{1\leq n\leq N\}}$. This process is called "Prune". Let us take a single trajectory l_n as an example. The part before it goes into and merge with the correct ones (the important pixels) can be cut under the premise that once upon a trajectory finds the edge, it will not deviate from the edge. The integrity of the final result depends heavily on how complete the N trajectories are. The flowchart of edge detection and integration by voting is shown in Figure..7.2.

7.2.3 Edge Integration by Reflowing

The result by the integration method by voting then pruning the trajectories is full of redundant edge segments, then we put up with another method by reflowing.

As stated above, after the construction of the vector field, a set of random points are chosen to flow as the starting points in the image plane. At the beginning, all starting points move in accordance with $\mathbf{V}$. The trajectories of the movement of these point are recorded in $P(\cdot)$. And they will stop moving until a stopping criteria is met.

Here we define three stopping criterions for the flow:

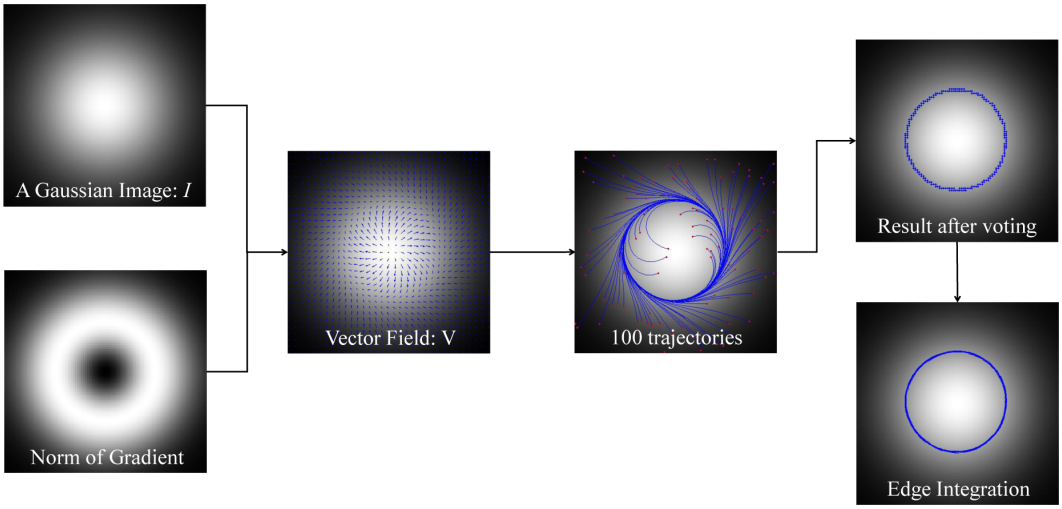


Fig. 7.2 The flowchart of using vector field to track trajectories from random starting points. The left column present a Gaussian image and the magnitude of its gradient. They form the vector field **V** together. Starting from 100 random points(labelled as red), under the action of vector field **V**, 100 trajectories are recorded correspondingly, shown in the third column. A voting process is used to determine the importance of the points on these trajectories. The points with higher importance are left to depict the highest gradient magnitude, shown as the circle in the column right above. Finally, we use an integration method to integrate these points and obtain the complete edges, the result is shown in the column right below.

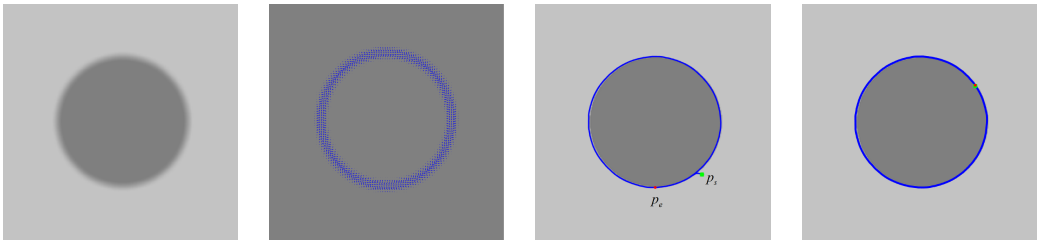
1. When the flow hits itself. This kind of end points is the first type of end points, labelled as "*E1*".

2. When the flow hits the boundary of the image. This kind of end points is the second type of end points, labelled as "*E2*".

3. When the flow hits a pixel where the gradient is zero. If it is at the source point where the gradient is zero, we will remove this source point. For the other cases, we will not consider the flow unless it merges with other flows. This case is detailed in the next section.

Endpoints labelled as "*E1*"

For a trajectory l_{p_s}, it starts from p_s and ends at p_e. If p_e is labelled as "*E1*", we re-initiate the movement from p_{e_1} in the same vector field **V**. In this situation, l_{p_e} will surely hit itself and according to the stopping criteria 1, it will stop and form a closed curve. This closed curve is the boundary of a shape. Figure.7.3 illustrates the integration process of the first kind of end points.



(a) (b) (c) (d)

Fig. 7.3 From left to right are: the original image; the vector field $\mathbf{V}$; a trajectory from the green point to the red point; the final result.

Endpoints labelled as "*E2*"

Suppose the boundary of the objective image ∂I, when a trajectory l_{p_s} moves to a point p_e and $p_e \in \partial I$, the motion stops. In order to extract a complete edge, it should reflow from p_e, while in an opposite direction, namely, $\mathbf{V}'$.

Let us take Figure.7.4 as an example. We initiate the point flow from 50 random points under the vector field Figure.7.4(c), shown in Figure.7.4(e), many points end at the time when they start to move, because they satisfy the third stopping criteria. These points are removed in Figure.7.4(f) and they will no longer play an role in the next steps.

For the two numbers (or shapes) "1" and "2" in the image, obviously, the boundaries are closed curves. The way to detect and integrate the edges on these two shapes are in the same situation as "*E1*".

For the slash which separates the black and white region, the two terminals of this slash lie at the top and bottom boundary of the image, we call this slash the "split line". From Figure.7.4(f) we can see that there is a short curve starting from the nearby of the "split line" and ends at the top boundary of the image. To extract the whole edge, the flow restarts from this end point under $\mathbf{V}'$ (Figure.7.4) until it meets the bottom boundary of the image, the endpoint $p_{e'} \in \partial I$. Thus, all the edges on the image are extracted.

Junction Points

This part is designed for the situation when two or more different streams converge and flow to the same endpoint p_e. Obviously, for close curves, this case can be excluded.

Figure.7.5 serves as an example for this case. For the starting (green) points in Figure.7.5(e), they flow under the action of $\mathbf{V}$ (Figure.7.5(c)). After removing the third type of endpoints (which the source and end points are at the same location), all flows merge into one stream and come to the same end point p_e, shown in Figure.7.5(f). According to the integration method mentioned in the part above, we need to re-initiate the flow from the endpoint on the vector field $\mathbf{V}'$ (Figure.7.5(d)). On its way back, it will meet the intersection, and here comes

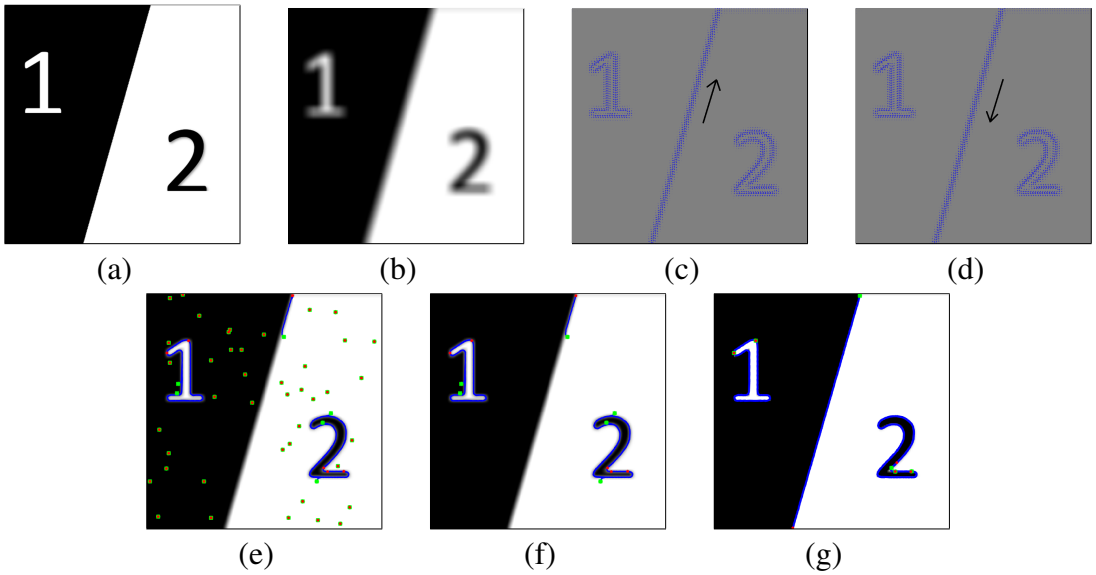


Fig. 7.4 from left to right, above to below, these figures are: (a) the original image; (b) image filtered by a 2D Gaussian; (c) vector field $\mathbf{V}$; (d) vector field $\mathbf{V}'$; (e) primary flow result from 50 random points (green points are the starting points and red ones the end points); (f) result after removing the non-edge points; (g) final result. (the black arrows on (c) and (d) illustrate the direction of the vector field of the corresponding boundary)

the dilemma – which way should it choose. Normally, the flow l_{p_e} goes into the part where there is stronger vector field, as shown in Figure.7.5(g). However, this will lead to a loss of important information. Therefore, we propose a way to complete the detection result.

For Figure.7.5(f), there are five starting points, which generate five trajectories. When any two trajectories intersect, we record the first point where they meet, as the cyan points in Figure.7.5(f). These points are recorded in $\{p_{c_{i,j}}\}$, where i and j stand for the ith or jth trajectory, $(i,j) \in \{1,...,N\}$ and $i \neq j$ ($N = 5$ in this case). If a point $p_{c_{i,j}}$ is on or very close to l_{p_e}, we will not consider it. Only those points which are far away from l_{p_e} will be taken into account. So the intersection points that on or close to the trajectory on Figure.7.5(g) will be removed. The two points on the lower edge are regarded as two new starting points for another flow process. This time, it will flow on both vector field $\mathbf{V}$ and $\mathbf{V}'$ until a stopping criteria is met. In addition, in order to avoid repetition, if one new starting point is covered by the trajectories of the previous ones, it will be removed from the set of new starting points, and so forth. Figure.7.5(h) shows the final complete integration result.

Algorithm.4 describes the whole process of the algorithm of the point flow model.

The processing chain of point flow method is shown in Fig.7.6.

Algorithm 4 Point Flow Algorithm

Initialization:
p_{s_i}, $i \in \{1, ..., N\}$, % N starting points;
$\mathbf{V}$, $\mathbf{V}'$, %vector fields;
k = 1, % counter of the number of end points;
new = 1; % counter of the number of new starting points;

---

for i = 1:N
  if $\|V(p_{s_i})\| \neq 0$
    $\frac{p_{s_i+d} - p_{s_i}}{d} = \mathbf{V}(p_{s_i})$ % d is the space step
    if $p_{s_i+d} \in l_{p_{s_i}}$
      $p_{e_j} = p_{s_i+d}$; $p_{e_k \cdot flag} = E1$; k++;
    else if $p_{s_i+d} = m$ or n; % meet the boundary of the image
      $p_{e_j} = p_{s_i+d}$; $p_{e_k \cdot flag} = E2$; k++;
    end if
  end if
end for

---

for i = 1:k
  if $p_{e_k \cdot flag} = E1$
    $\frac{p_{e_i+d} - p_{e_i}}{d} = \mathbf{V}(p_{e_i})$; % to get a closed curve
  else if $p_{e_k \cdot flag} = E2$
    $\frac{p_{e_i+d} - p_{e_i}}{d} = \mathbf{V}'(p_{e_i})$;
  end if
end for

---

for i = 1:k-1
  for j = i+1:k
    if IsIntersect($l_{p_{e_i}}$**,** $l_{p_{e_j}}$**)** % when two flow merge
      $p_{c_{i,j}} = (l_{p_{e_i}} \& l_{p_{e_j}})._{first}$ % the first location that they meet
      if $p_{c_{i,j}} \in \{l_{p_e}\}$
        continue;
      else
        $p_{new} = p_{c_{i,j}}$; % new starting points
      end if
    end if
  end for
end for

---

for i = 1:k-1
  if $p_{s_i+d} \neq p_{s_i}$
    $p'_{s_i} = p_{s_i}$
    $\frac{p_{s_i+d} - p_{s_i}}{d} = \mathbf{V}(p_{s_i})$;
    $\frac{p'_{s_i+d} - p'_{s_i}}{d} = \mathbf{V}'(p'_{s_i})$;
  end if
end for

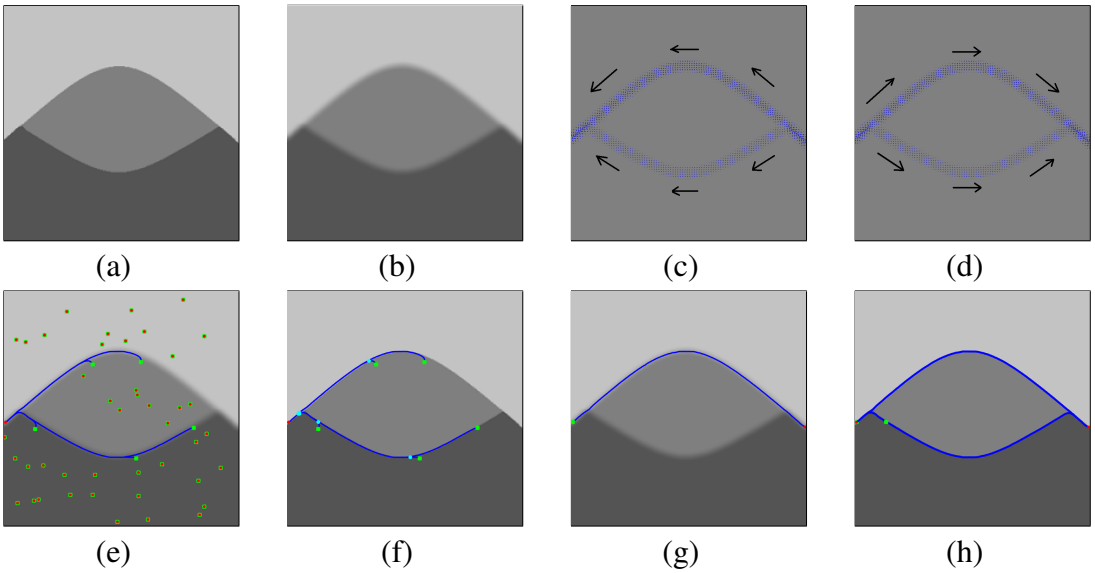


Fig. 7.5 from left to right, above to below, these figures are: (a) the original image; (b) image filtered by a 2D Gaussian; (c) vector field $\mathbf{V}$; (d) vector field $\mathbf{V}'$; (e) primary flow result from 50 random points (green points are the starting points and red ones the end points); (f) result after removing the non-edge points; the cyan points are the intersections between any two flows; (g) result by simply flowing back from a "*E2*" endpoint; (h) final result. (the black arrows on (c) and (d) illustrate the direction of the vector field of the corresponding boundary)

7.3 Combination of Multiple Features

7.3.1 Construction of vector Field

As described above, in natural images, it is not always sufficient by only using the gradient of the grayvalue to form the vector field. It is necessary to combine some other features to form the vector field. The high-level feature are always extracted by machine learning techniques, so a dataset containing the original images and their corresponding groundtruth is necessary. In this section, the BSDS500 dataset [69] is used for extracting the high level feature such as the texture and spectral features.

In [71], the authors proposed a *Pb* (probability of boundary) edge detector where they introduced several features, i.e. color, texture to detect the contours. In [3], the authors not only use the features in [71] but also introduced a spectral feature and combine them into a global feature to detect the contours, the detector is known as the *gPb* (global probability of boundary) contour detector. To compute the gradient on each pixel of each feature channel, it is realized by setting discs on each pixel (x,y) of each feature channel and comparing the histogram difference of each half-disc g and h using different orientations.

$$\chi^2 = \frac{1}{2}\sum_i \frac{(g(i)-h(i))^2}{g(i)+h(i)} \tag{7.7}$$

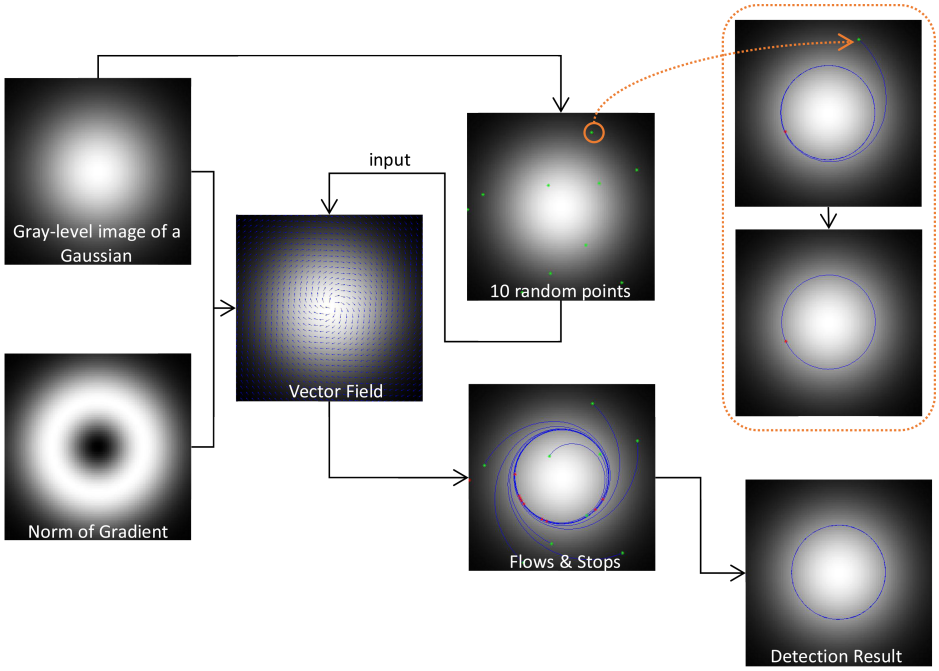


Fig. 7.6 The flowchart of the point flow algorithm. The left column presents a 2D Gaussian image and the magnitude of its gradient. A linear combination of their gradients are used to form the vector field $\mathbf{V}$. 10 random points are used as the starting points, moving under the effect of the vector field. The top-right dashed-box shows the movement of a single point, the blue curve is the trajectory and the red point is the end point. The flow terminates when it hit its own trajectory. Then, we re-initiate the flow from the end (red) point to obtain a complete and precise contour of the Gaussian image. The other points move in the same way. Right below shows the result.

Fig. 7.7 17 filters for computing the texture [3]

Normally, eight orientations are used to describe the directions of the gradient, where the orientation $\theta = [0, \pi/8, \pi/4, 3\pi/8, \pi/2, 5\pi/8, 3\pi/4, 7\pi/8]$.

To improve the performance of PointFlow in complicated scenes, we use the same feature in [3]. The color channels contain three channels which correspond to the CIE Lab colorspace including the brightness, color a and color b channels. The fourth channel is the texture channel, to obtain the texture feature, the primary thing is to compute the texton map. The texton map defines the texton label on each pixel of the image. Firstly, each input training image from the BSDS dataset is convolved with a set of 17 Gaussian derivative, shown in Figure.7.7.

Then each pixel is associated with a 17-dimensional vector of responses. A K-means cluster is then used to define the clustering center, with $K = 64$. For the test image, we convolve them with the set of 17 Gaussian derivatives, too. The pixel which is nearest to one of the 64 cluster center will be given the corresponding texton label.

Let us denote the gradient signal $G_i(x, y, \theta)$, where i represents the feature channel and θ is the orientation. We also use the multi-scale and spectral features that were described in [3].

For the multi-scale features, each channel gradient G_i is computed at three scales $s = [\frac{\sigma}{2}, \sigma, 2\sigma]$, where s controls the radius of the disc for computing the histogram difference in Eq. (7.7). $\sigma = 5$ pixels for brightness channel and $\sigma = 10$ pixels for color a, color b and texture channel. The multi-scale gradient signals at each orientation can be combined linearly as follows:

$$mG(x, y, \theta) = \sum_{s} \sum_{i} \alpha_{i,s} G_{i,s}(x, y, \theta) \tag{7.8}$$

where $\alpha_{i,s}$ are the weights at different scale s and different feature channel i and it is learned by gradient ascent on the F-measure on the BSDS500 dataset.

The spectral features are used to obtain the most salient curves and it is obtained by a spectral clustering technique. A sparse symmetric matrix W which describes the affinity of two pixels of which the distance is within a fixed radius r is provided according to mG. The spectral signals sG then are obtained by solving a Laplacian system of W, details can be found in [3].

By combining the multi-scale and spectral features, we can get the global signal at each orientation $gG(x, y, \theta)$.

Figure.7.8 shows the feature map gG of eight orientations.

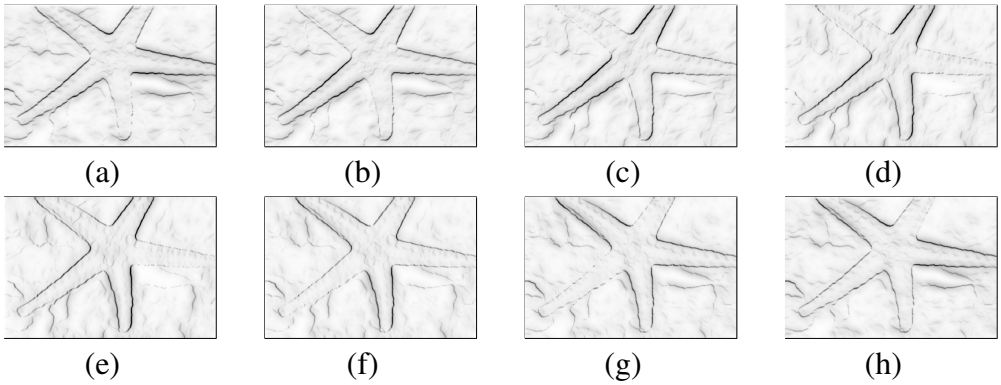


Fig. 7.8 Gradient magnitude of eight orientations, from (a) to (h), the orientation $\theta = [0, \pi/8, \pi/4, 3\pi/8, \pi/2, 5\pi/8, 3\pi/4, 7\pi/8]$

After computing the multiple features at each orientation, we can construct the vector field based on these features. For each single pixel, we will retain the largest value among the eight values as the gradient magnitude, and the corresponding orientation denotes the direction of the gradient:

$$\mathbf{v}(x,y) = gG_{\max}(x,y,\theta) * (\cos(\theta), \sin(\theta)) \tag{7.9}$$

so that: $\mathbf{V}_2 = \mathbf{v}^{\perp}$. To compute the second-order derivative, we use the gradient of $gG_{\max}(x,y)$, that is: $\mathbf{V}_1 = \nabla(gG_{\max})$.

Figure.7.8 shows an example of using multiple features to construct the vector field and its corresponding contour detection result. Compared with Figure.7.13 (d), the result in Figure.7.9 (c) is more clean and complete.

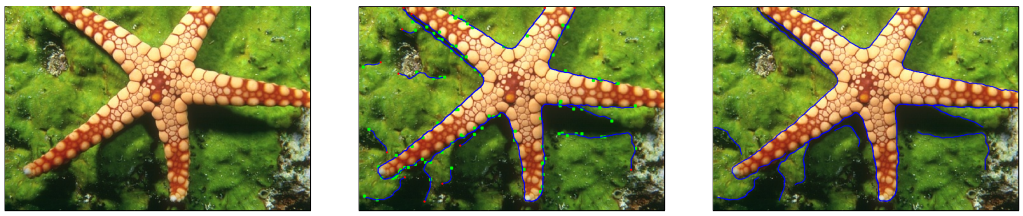


Fig. 7.9 Experiment on a real color image. From left to right are the original color image, the trajectories generated from 2000 random points and the result of detection of contours.

7.4 Experiment Result and Analysis on PointFlow

7.4.1 Experiment Result and Analysis by Voting

Figure.7.10, Figure.7.11 and Figure.7.12 are results synthetic and real medical images respectively by using the voting method to integrate the edges.

Fig.7.10 display the results of the above four synthetic images adding noise. From the top to bottom, they are: the original noisy image (after preprocessing by a Gaussian filter); the vector field; the trajectories from N starting points; the results after voting; and the results after Pruning. From these results, we can see that under a noisy circumstance, the PointFlow is still robust and effective.

We test our method on a real medical image with catheter, shown in Fig.7.11. In Fig.7.11(d), a lot of curves have been detected. As we introduced above, the points on the main catheter have higher importance than points in other parts, so in the final result, the main catheter is extracted while the paths left are filtered.

Here we show another example on a real vessel image, see Fig.7.12. Since the original vessel image has complicated background, we filter it using a sigmoid function, Fig.7.11(b). From the results, we can see that nearly all the boundaries of the vessel are detected and only a few are missed.

Despite the advantage of using voting method to integrate the edge segment, the results are full of redundancy. In addition, in order to achieve good performance, a large number of initial points are needed, which leads to more computations.

7.4.2 Experiment Result and Analysis by Reflowing

Here we have tested our point flow method by using reflowing to integrate the edges on the images from the famous Berkeley Segmentation Dataset[69], and we have also evaluated our method by using the 100 validation images from the same dataset. The number of random source points is set to be $N = 2000$ for each image. For most of the images, we use the grayscale to compute the vector field.

For color images, we use the highest gradient at each pixel from the three channels RGB as the gradient of that pixel. For certain images, we use the HSV color space to compute the vector field. Before flowing, we use a Gaussian filter to smooth the images, the standard deviation of the filter is $\sigma = 2$, and the size is 4×4.

In addition, there are numerous ways to approximate numerically the solution to the first-order ODE Eq.(7.1), such as the Euler method, backward Euler method, first-order exponential integrator method and so on [13]. To make the solutions smoother and with less oscillations, we used the Runge-Kutta algorithm, also known as RK4, to solve Eq.(7.1).

Our detection provides a "hard" boundary result, not a probability map. So on the precision-recall curve map, what we present is a single point. We compare our method with the classical canny detector [14], the state-of-the-art probability of boundary (pb) detector [71], and the graph-based segmentation method untangling cycle [127].

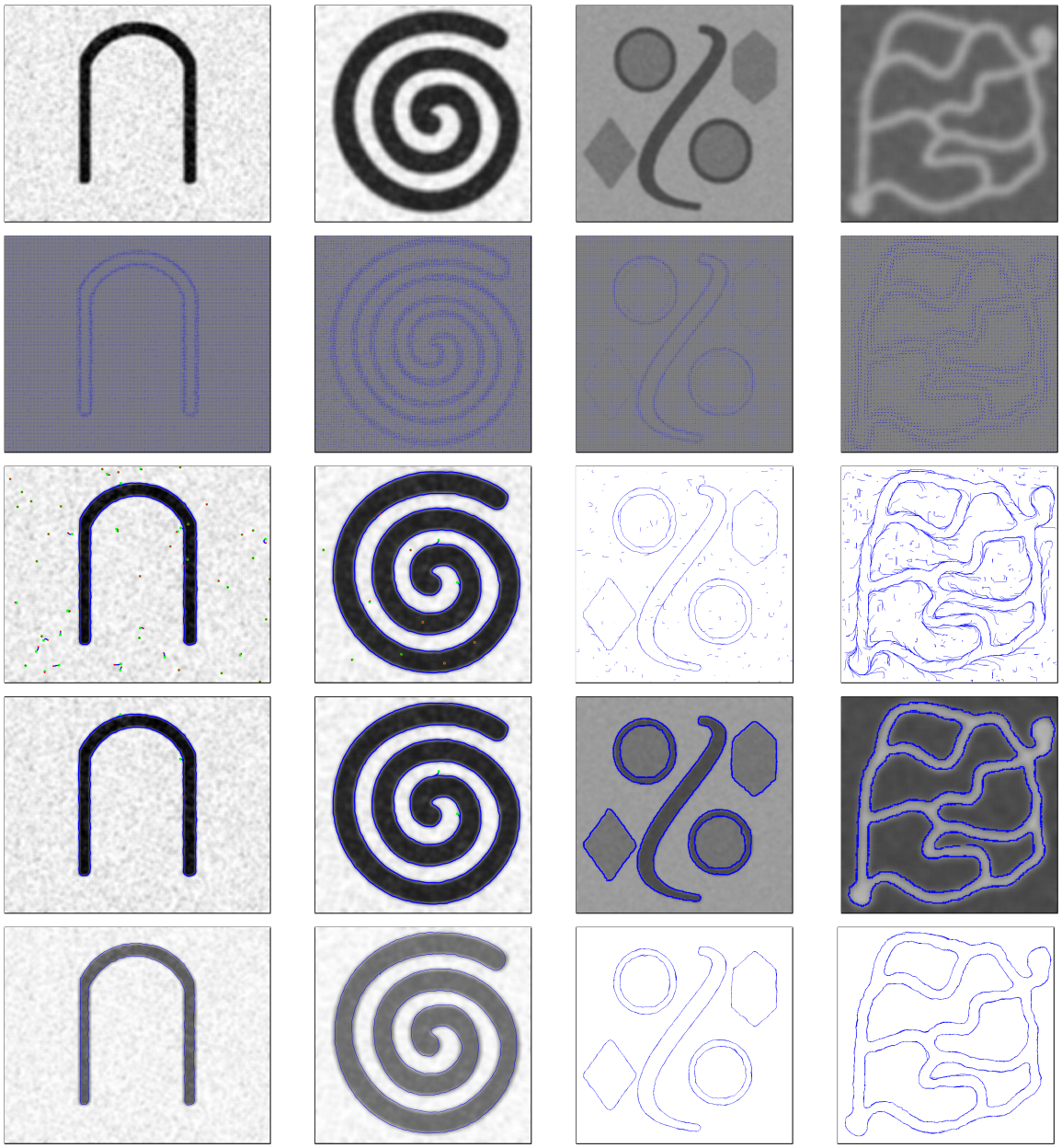


Fig. 7.10 From the above to below, the first row are four noisy synthetic images after filtering, the second shows the vector field obtained from Eq.(7.3), the third row displays the trajectories by using point flow, the number of starting points are 500, 300, 1000 and 500, the forth represent the results by setting a threshold β, which equals to 3, 5, 3, 2 respectively. The five shows the integration results by voting.

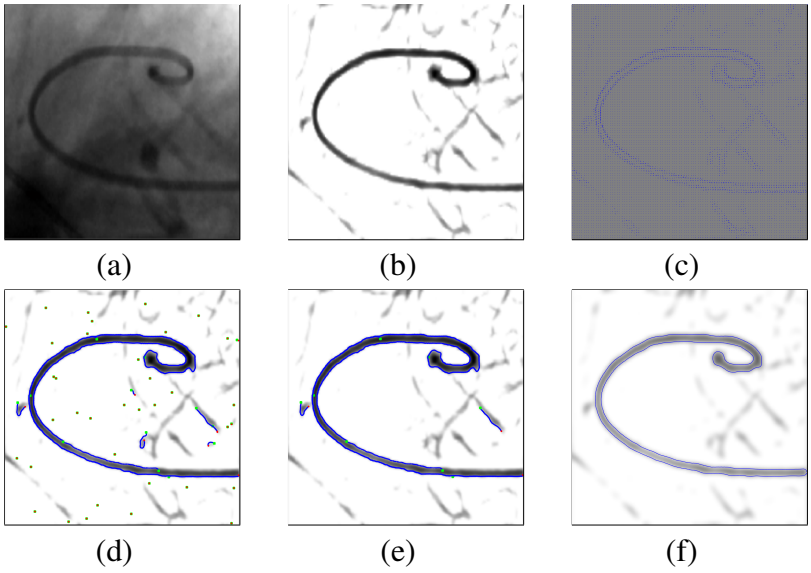


Fig. 7.11 Experiment on a real medical image with catheter, the above from left to right: the original image, the Laplacien of the original image, the vector field of the image; the below from left to right, the trajectories starting from 50 points, the trajectories after pruning, the result after integration.

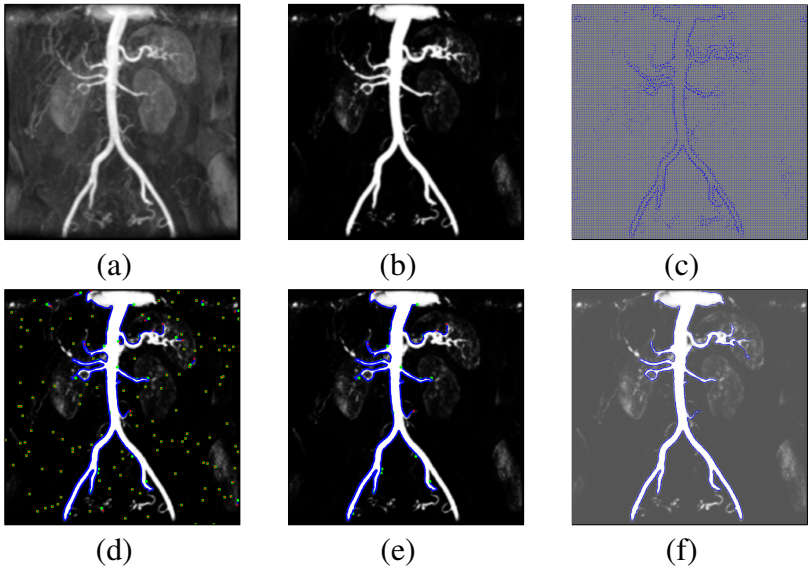
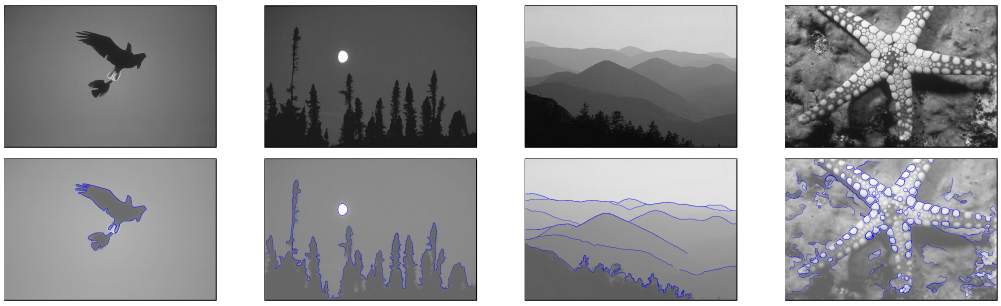


Fig. 7.12 Experiment on a real vessel image, the above from left to right: the original vessel image, the preprocessing of the original image by Eq.(5.23), the vector field of the image; the below from left to right, the trajectories starting from 150 points, the trajectories after pruning, the result after integration.

Last but not least, we note that our method presents a sub-pixel level detection result. During the evaluation process, we must assign each point on the trajectory a precise pixel location. This clearly will lead to a loss of available sub-pixel information.



(a) (b) (c) (d)

Fig. 7.13 Experiment on some gray images, the row above display the original images and the row below show the edge detection results.

We have tested our method on the widely known BSD dataset. Figure.7.13 and Figure.7.14 show some detection results using our proposed method, and the results are very encouraging. In terms of the evaluation, shown in Figure.7.15, our method nearly performs as the best results obtained with non-learning based methods. Note that we are only use the color or graylevel information to construct the vector field. Clearly, the BSD dataset are images selected from a wide range of images, they do not usually comply with our model. Additionally, note that our method provides precise sub-pixel level result, however, this advantage could not be reflected by the metric in the Berkeley benchmark.

7.4.3 Experiment by Reflowing Using Multiple Features

Figure.7.16 shows some results obtained by using different vector fields. The left column displays the results by only using color feature to construct the vector field and the right column shows the results by using multiple features to construct the vector field. From Figure.7.16, we can see that by using the multiple features (obtained from the gPb detector), we can get more complete and clean results, while by using the single feature (graylevel or color), we are able to obtain more details.

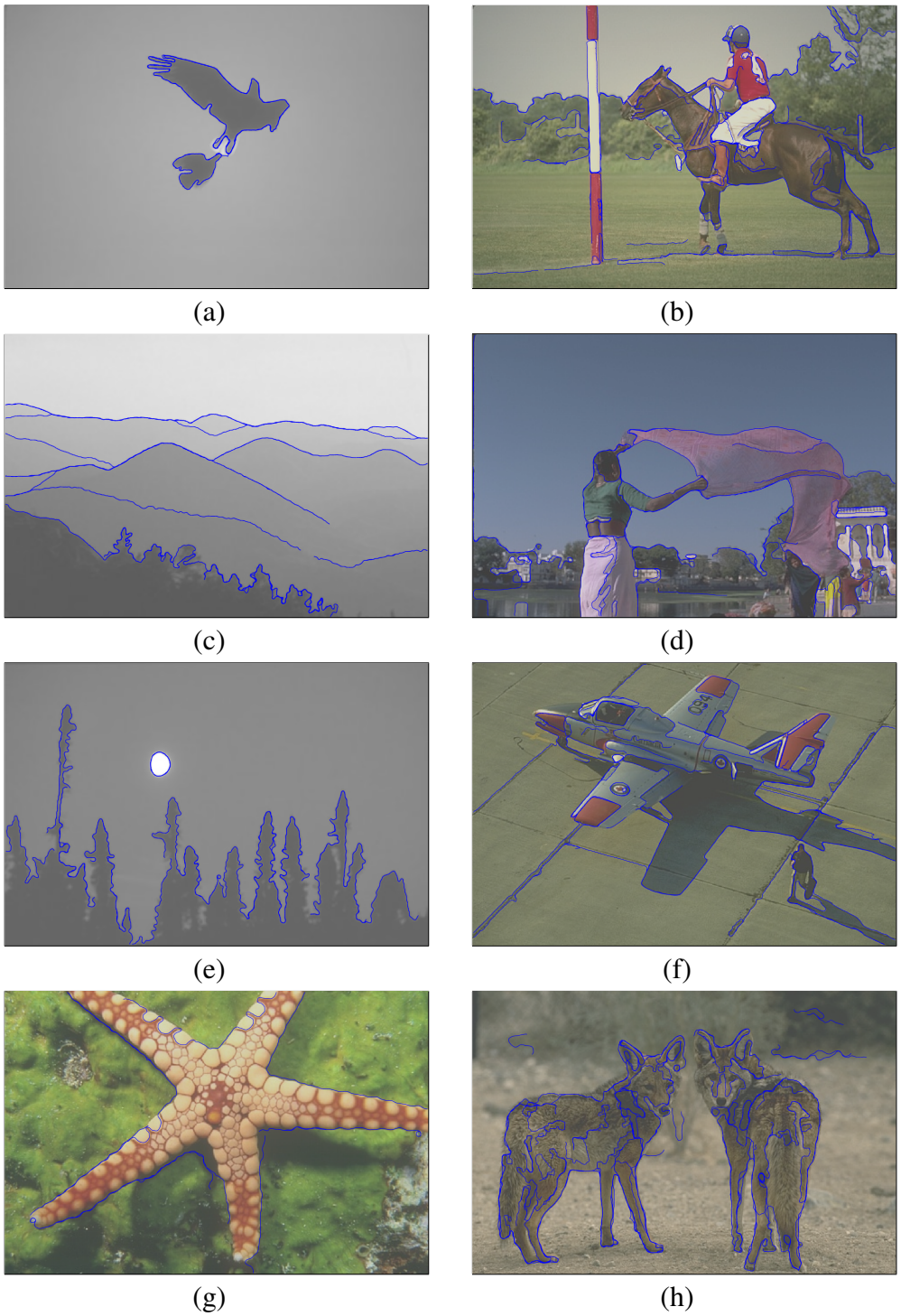


Fig. 7.14 some results on the BSD Dataset. (a), (c) and (e) are tests on gray images; (b), (d) and (f) are tests on color images; (g) and (h) are tests on the first and second channel of HSV space seperately.

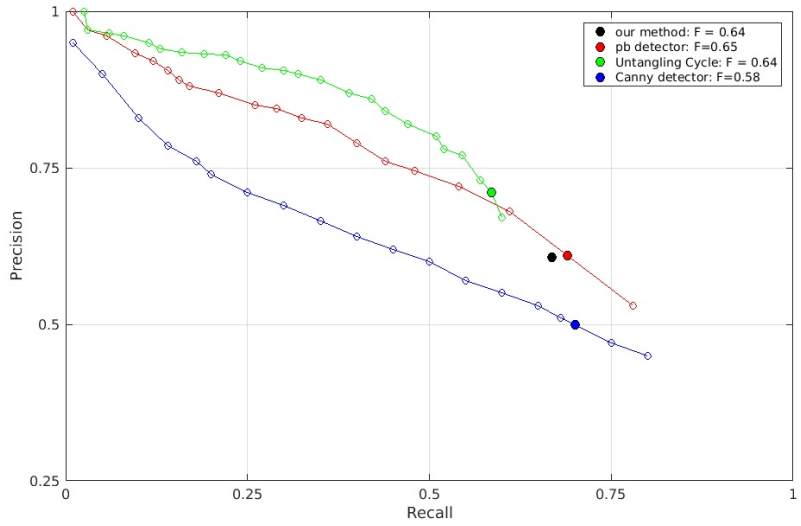


Fig. 7.15 Precision recall curve on the Berkeley benchmark, compared to Pb detector, untangling cycles and Canny detector.

7.5 Inference of Illusory Contours

The illusory contours (or subjective contours) are visual illusions that evoke the perception of an edge without brightness or color changes on that edge. Speaking of the illusory contours, the first image comes into our mind would be the Kanizsa's Triangle, shown in Figure.7.17. We can perceive from the spatially separated fragments in Figure.7.17, that there are hidden triangles and circles, but in the view of images, there are no complete triangles or circles. Then we can not help to ask that why we can perceive these unreal outline, can we make the computers sense these phenomena. According to [114], it is believed that for us human beings, the early visual cortical regions such as the primary visual cortex (V1) and secondary visual cortex (V2) are responsible for forming illusory contours.

For the computers, how can they perceive the illusory contours like humans remains our problem. By observing the illusory contours shown in Figure.7.17, we can find that the illusory contours are usually an unseen curve that connects two corner points. Here comes the question: is it possible for a point to flow on the illusory contour rather than on the boundary of an object when it meets a corner point? The answer is yes.

7.5.1 Inertia-Driven PointFlow and Hough Transform

In our opinion, the illusory contours are always small missing parts of regular and predictable shapes. For these kind of illusory contours, it can be completed by an arc with a specific radian. Even for a straight line, it can be described as an arc of a circle of which the radius is

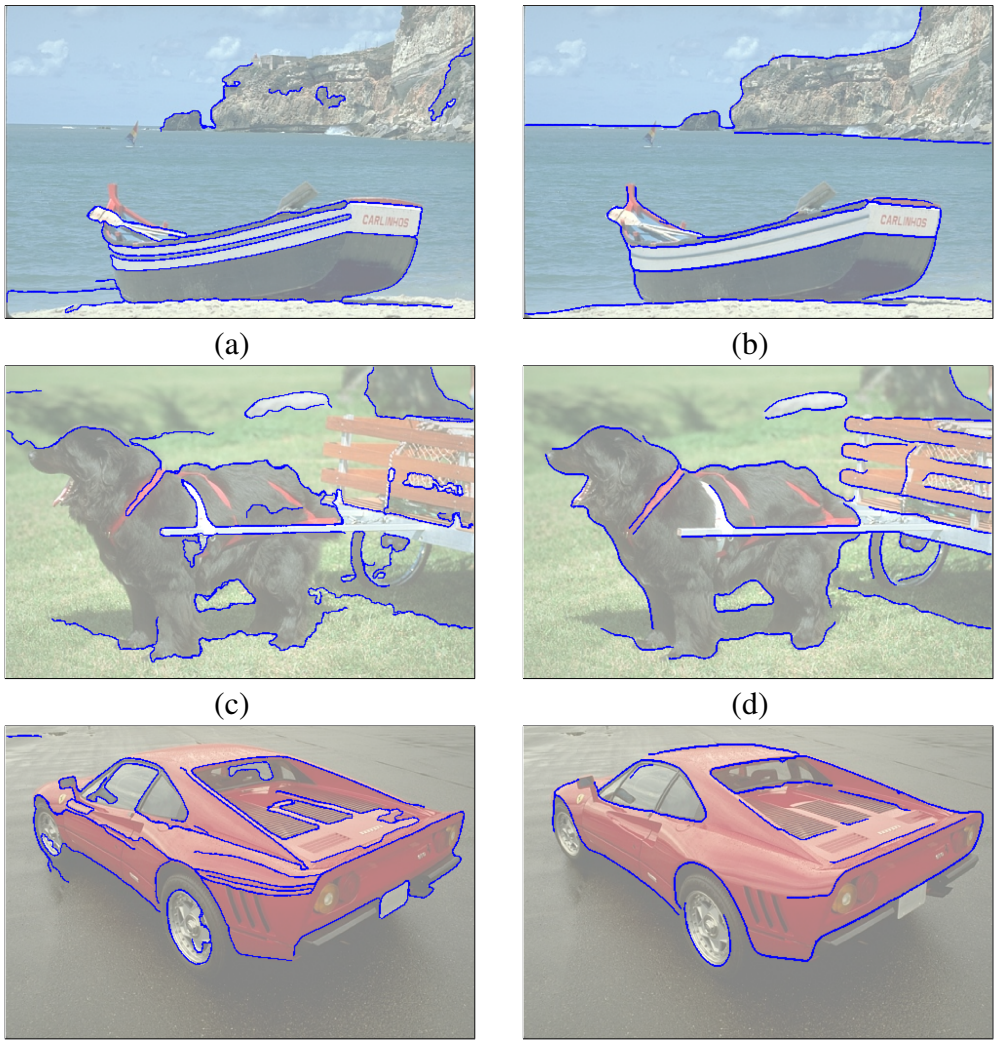


Fig. 7.16 some comparisons between using single feature (color) and multiple features . Top row displays the results by only using color feature to construct the vector field; bottom row shows the results by using multiple features to construct the vector field.

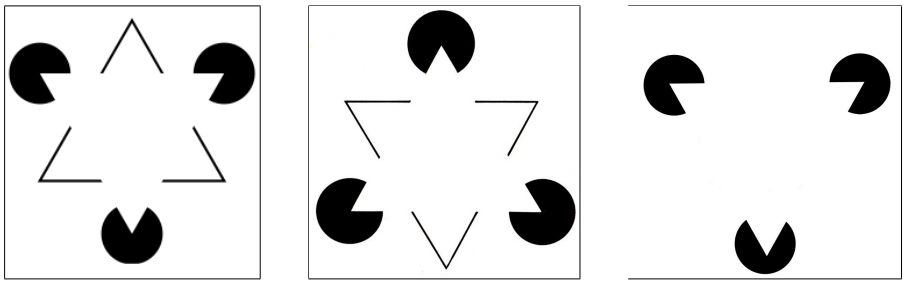


Fig. 7.17 Some examples of the Kanizsa's Triangle

infinite. So the main task of inferring the illusory contours is to find the two terminals of an appropriate arc.

In this paper, we assume that the terminals of an illusory contour are always caused by abrupt changes in curvature of a closed curve. These terminals are also called the corner points.

After obtaining the corner points, we make the point flow at the same velocity (angular velocity) **v** as it did before it hits the corner point, instead of flowing on the image induced vector field.

Due to the fact that the point flow on a field without any forces after hitting the corner point, the behavior of the point is like inertia-driven, so we call this process "Flow with inertia".

The idea that a missing curve in illusory contour images can be completed by an arc reminds us of the Hough Transform (HT) and Circle Hough Transform (CHT)[32].

Hough Transform

The classical Hough Transform is used to detect straight line in images. In the parameter space, a straight line can be represented as a point (θ, r):

$$r = x\cos\theta + y\sin\theta \tag{7.10}$$

where r is the distance from the origin to the line and θ is the angle between x axis and the line. For a single line, the parameter θ, r is fixed. The Hough Transform takes advantage of this characteristic to detect straight lines in images. The idea is to project the input image into a parameter space and find the corresponding coordinates with highest values. Figure.7.18 shows an example of line detection by the classical Hough Transform.

(a)the original image (b)the projection on (θ, r) space (c) Line detection result

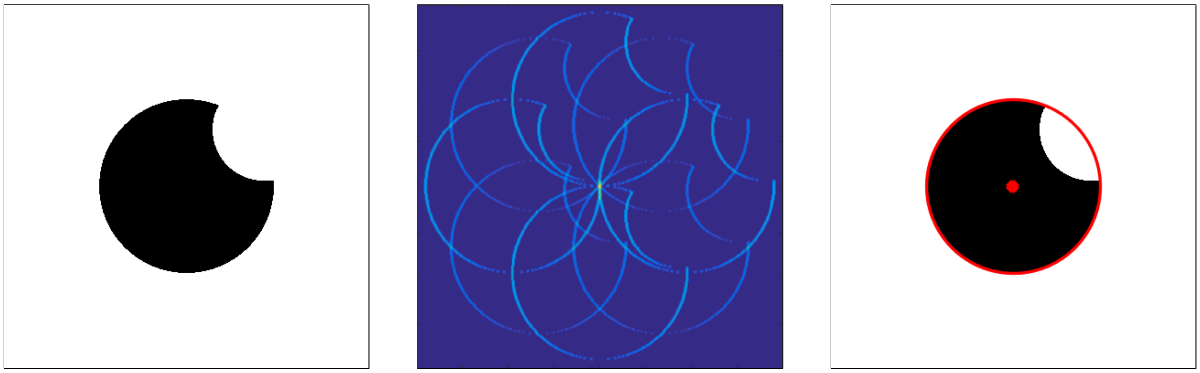
Fig. 7.18 Detection of straight lines by Hough Transform

Circle Hough Transform

The Circle Hough Transform (CHT) is a technique for detecting circular objects in images. The idea is similar to the classical Hough Transform. In a 2D space, a circle of which the center locates at (a,b) with a radius r can be described by:

$$(x-a)^2+(y-b)^2=r^2 \tag{7.11}$$

The parameter space is 3D, (a,b,r). And the corresponding circle parameters can be identified by the intersection of a number of conic surfaces which are caused by the circle on the image.



(a) (b) (c)

Fig. 7.19 Detection of circles by CHT, from left to right, (a) the original image; (b) projection on (a,b,r) space ($r=95$); (c) circle detection result, shown in red.

From the example Figure.7.19, we can see that the CHT has the ability to infer obvious circles. But the users have to provide the range of radii, and a large radius will result in a bad detection. We will compare the circle detection results by our inertia-driven flow and the circle hough transform in the experiment subsection.

7.5.2 Finding Corner Points

To infer the illusory contours using point flow with inertia, the detection of the corner points should be the primary task. Due to that there are two ways of integrating the edges: voting and reflowing, there are corresponding two ways to obtain the corner points.

Corner Points via Voting

To obtain the corner point, we need first use the PointFlow model to detect the boundaries in the images. The input of this process are the important points (points on the boundaries or edges) obtained from voting method in Section.7.2.2, denoted as $\{p_i\}_{0\le i\le M}$, M is the number

of these points. Then we start the point flow from every p_i on the same vector field $\mathbf{V}$ in Eq.(7.3). To make the results more complete, we initiate the point flow using both sign in the first term of $\mathbf{V}$.

For a single point p_i, after flowing for time ΔT_2, a trajectory l_i is obtained. For l_i, the starting point is p_i, and the end point is p_{e_i}. Then the middle point p_{m_i} on l_i should be the point at time $\Delta T_2/2$.

Once the three points p_i, p_{m_i} and p_{e_i} are obtained, we need to compute the angle θ_i between $\overrightarrow{p_i p_{m_i}}$ and $\overrightarrow{p_{m_i} p_{e_i}}$ and decide whether p_{m_i} is a corner point or not.

$$\theta_i = \text{acos}\left(\frac{\langle \overrightarrow{p_i p_{m_i}}, \overrightarrow{p_{m_i} p_{e_i}} \rangle}{\|\overrightarrow{p_i p_{m_i}}\| \|\overrightarrow{p_{m_i} p_{e_i}}\|}\right) \tag{7.12}$$

Here we set a threshold θ_{thr}, if $\theta_i > \theta_{thr}$, then p_{m_i} is considered as a corner point.

After the search for all the M points, we get J middle points $P_{m_j}, j \in [0, J]$. The corresponding J trajectories $\{L_j\}$ are also saved.

Fig.7.20 presents the flowchart of the inference of illusory contours via the voting method to obtain the corners and .

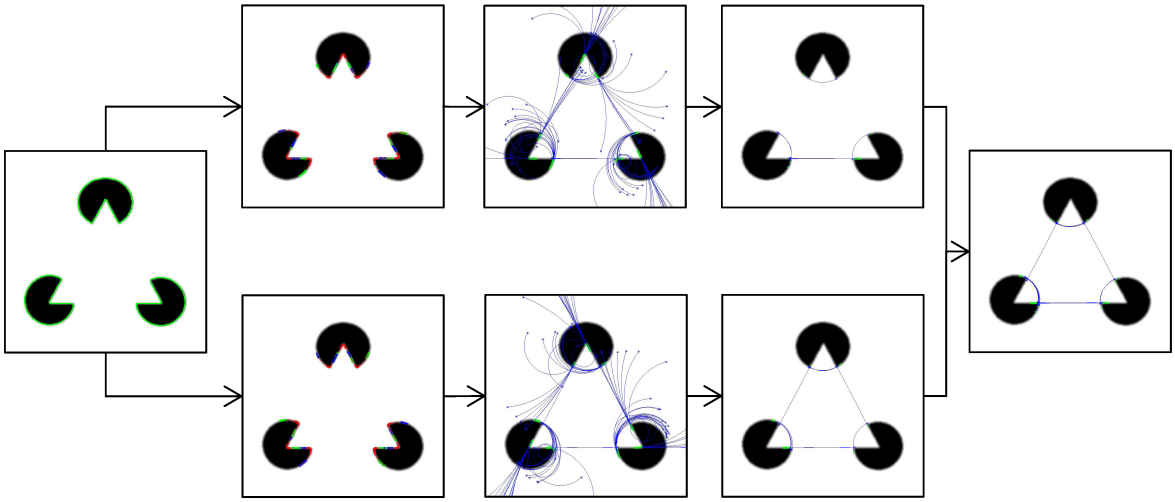


Fig. 7.20 Flowchart of the inference algorithm, from left to right, the original image with the detected edge points (labelled as green), the corner points (labelled as red stars), the trajectories generated by inertia, the inferred trajectories, the result obtained by combining the inferred trajectories.

Corner Points via Flowing

As stated above, the PointFlow model is a dynamic model, the points driven by this model on the image tend to trace the edges and object boundaries on the images. The trajectories

formed by the motion of these points are continuous, thus we can detect the corner points by taking the change of curvature into consideration. That is to say, where there is a high change of curvature of a trajectory, there is a potential corner point. We can determine that whether the potential corner point is a true corner point or not by setting a threshold for the curvature. we will make the point move at the same velocity **v** as it did before it hits the true corner point. Figure.7.21 demonstrates the process of finding corners by taking the curvature into consideration.

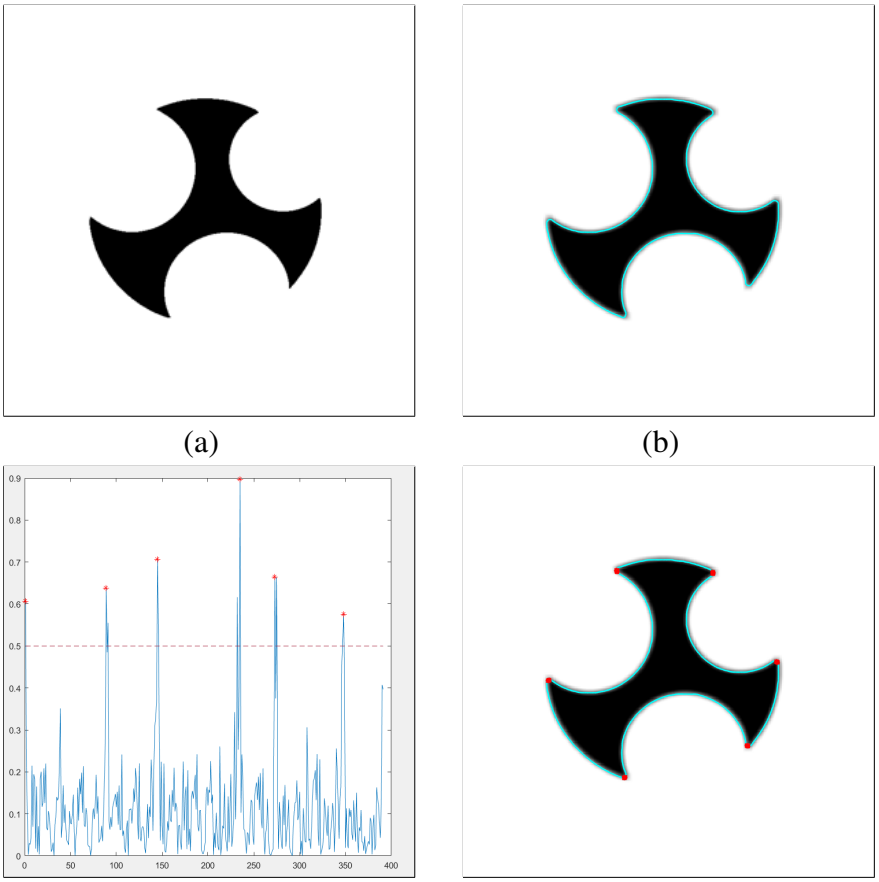


Fig. 7.21 An example for detecting the corner points. (a) the original image; (b) the cyan curve is the detection result by using PointFlow model; (c) the plot of curvature of the detected curve; (d) the corresponding corner points.

7.5.3 Flow with inertia

After finding the corner points, we can infer the illusory contours by deciding that whether two corner points can be connected by an arc. As mentioned above, each movement of a single point p takes a unit time, then after ΔT, there should be $\Delta T + 1$ points on the trajectory. In addition, we are introducing a kind of "inertia" here, which means that when a point flows

to a corner point, it will no longer move according to the vector field, but continues to flow according to the trajectory it just passed through.

The following three steps describe how to infer the illusory contours by PointFlow in detail.

1. Compute the angular velocity. For a trajectory L_j which passes through two corner points p_{cor_1} and p_{cor_2} successively, in order to guarantee the accuracy and robustness, N points $\{p_n\}, n = [1, ..., N]$ are chosen on L_j to compute the angular velocity. First, the angular displacement from these N points to corner point p_{cor_2} is computed.

$$\phi(l_n) = \text{acos}\left(\frac{\langle \vec{a}, \vec{b} \rangle}{||\vec{a}||||\vec{b}||}\right).$$

where l_n is a piece of L_j from p_n to p_{cor_2}, $\vec{a}$ stands for the tangent vector of the curve at the corner point p_{cor_2} and $\vec{b}$ represents the tangent vector at nth point. A toy model can be found in Figure.7.22 (a). So the angular velocity for l_n is:

$$\omega(l_n) = \frac{\phi(l_n)}{\text{length}(p_{cor_2}, p_n)} \tag{7.13}$$

2. Flow with inertia. Based on the assumption that the missing part of the illusory contours can be simulated by an arc, we propose a kind of "inertia" which contains an angular velocity $\omega(l_n)$ and the initial speed $\mathbf{v}_0 = \mathbf{V}(p_{cor_2})$. Now p_{cor_2} is considered as the starting point of the trajectory generated by the inertia. With the inertia, the velocity $\mathbf{v}$ of the point can be obtained by a rotation matrix:

$$\mathbf{v}_t = \mathbf{v}_0 \begin{bmatrix} \cos(\omega(l_n)) & \sin(\omega(l_n)) \\ -\sin(\omega(l_n)) & \cos(\omega(l_n)) \end{bmatrix}^t \tag{7.14}$$

where t is the time of flowing, and it also represents the number of flowing steps. When $\omega(l_n) = 0$, the inertia will generate a straight line, when $\omega(l_n) \neq 0$, the inertia will generate an arc with corresponding radian.

3. Let us denote the trajectory generated by inertia l_{ine}, and the starting point of l_{ine} is the corner point p_{cor_2}. After time ΔT_1, which is a time limit provided by the users, if l_{ine} is still not connected to any corner point, l_{ine} will be ignored. If it is connected to some corner point according to the three conditions below, it is an illusory contour. Decisions will be made at each move of l_{ine}, and the end point of l_{ine} at each step is labeled as p_{cur}. To decide whether l_{ine} can be an illusory contour or not, there are three conditions:

- The length of l_{ine} is larger than a threshold that we set:

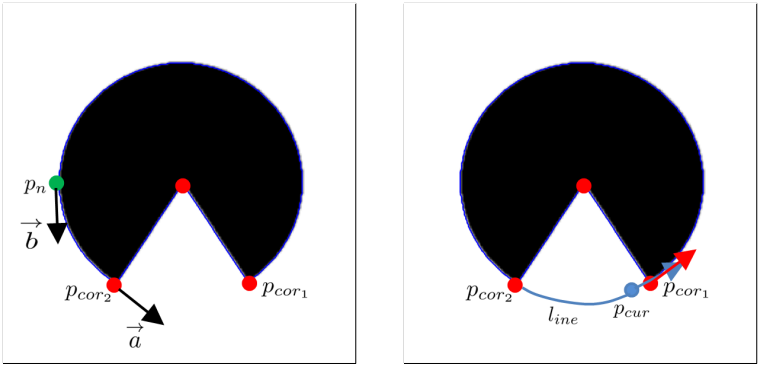
$$\text{len}(l_{ine}) > \beta$$

- The distance between p_{cur} and a corner point p_{cor} should be small enough:

$$\text{len}(p_{cur}, p_{cor}) < \epsilon$$

- l_{ine} can be connected to p_{cor} in a smooth enough way. Let us take Figure.7.22 (b) as an example. Let $\vec{c}$ be the tangent vector of l_{ine} at p_{cur} (the blue arrow), $\vec{d}$ the tangent vector at p_{cor_2} (the red arrow). The angle φ between $\vec{c}$ and $\vec{d}$ can be represented by the cosine value: $\varphi = (\frac{\langle \vec{c}, \vec{d} \rangle}{\|\vec{c}\|\|\vec{d}\|})$. If $\|\varphi\| > \delta$, $\delta = 0.9$ empirically, then the direction of the flow l_{ine} is in accordance with L_j.

If the above three conditions are satisfied, l_{ine} can be considered as the arc which fill in the missing part of the illusory contours.



(a) (b)

Fig. 7.22 An example for the flow driven by inertia, (a) describes how the angular displacement is obtained in Step.1; (b) shows how l_{ine} can be an illusory contour.

7.6 Experiment on the Inference of Illusory Contours

7.6.1 Data Settings

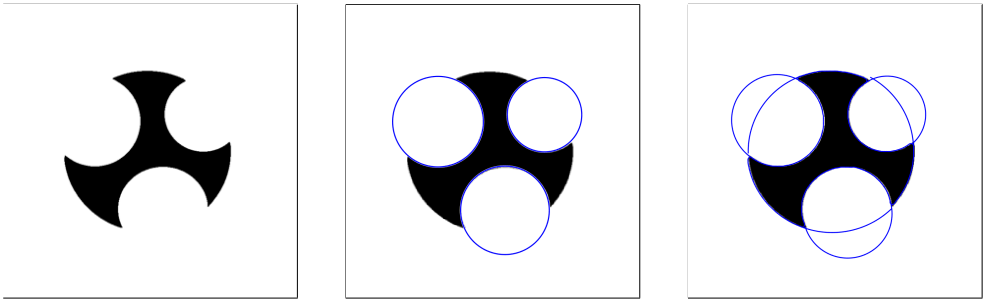
We test our method for inferring the illusory contours on three illusory images.

The first image is shown in Figure.7.23(a), the size is 400×400. We can perceive that there are 4 circles in this image: a big black one and three small white ones. For inferring the circles, we use twice the Circle Hough Transform (of which the radius ranges from 30 to 100 pixels and 50 to 120 pixels) and our inference method respectively.

The other two images are shown in Figure.7.24 (a) and (d), the size of both images is 400×400. It can be perceived by users that in Figure.7.24 (a) there exist a hidden triangle and three small circles and in Figure.7.24 (d) there are four circles as well as a square.

7.6.2 Experiment Result and Analysis

Figure.7.23 (b) and (c) display the circle detection result by using the CHT and PointFlow respectively. From the result (b), we can see that the CHT detect only the small circles. Due to that the big black circle has a lot of missing parts, no matter what radius we use, it cannot be detected. While in (c), our method has inferred and completed all the four circles. By using the CHT to detect circles, users have to provide a range of radii and a very large radius may result in bad results. While inference by PointFlow is automatical and effective.



(a) (b) (c)

Fig. 7.23 Experiment on a synthetic image, from left to right, (a) the original image (b) the circle detection result by using CHT (c) the inference result by using our method.

From Figure.7.24(c) and (f), it can be seen that all the illusory parts (either the straight lines of the triangle and the square or the curved arcs of the circles) are inferred by using our inertia-driven PointFlow method.

The results in Figure.7.23 and Figure.7.24 demonstrate that our inference algorithm is efficient at inferring the straight and smooth arcs. The disadvantage of this method lies in that it will fail when there is an obstacle on an illusory contours, or the illusory contours could no longer be represented by an arc.

7.7 Conclusion

This chapter proposes a PointFlow method for automatically modeling the process of tracking object boundaries in images. In this chapter, on the stage of using the color (or graylevel) feature to construct the vector field, our method is comparable among the numerous edge detection methods. In addition, the PointFlow method can be extended to infer illusory contours, and it is robust and efficient finding the arcs or straight lines.

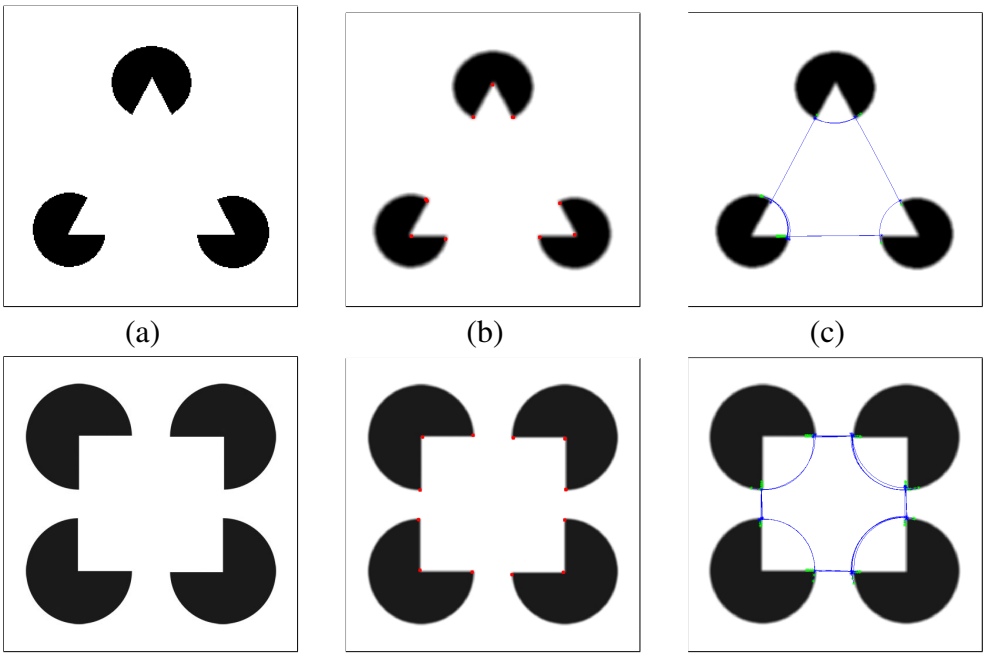


Fig. 7.24 Experiment on synthetic images, from left to right: (a) and (d) are the original image, (b) and (e) are the corresponding detected corner points, (c) and (f) the inferred illusory contours.

Chapter 8

A Model is Worth Hundreds of Examples

Abstract

Nowadays, in the field of computer vision, Deep Learning have become the hottest issue in both the academia and industry with the development of science and technology. For applications such as image segmentation, image classification, deep learning has done an excellent job. In terms of the performance, deep learning methods exceeds the other methods. Take the boundary detection methods on the famous Berkeley BSDS500 dataset as an example, so far the best detection results are generated by using deep learning methods. In this chapter, we build a dataset which contains 100 synthetic images in total. We use 50 images as validation data, and the rest as training data. We aim at figuring out that for an edge detector trained by deep learning, how many training images should be used that it can achieve a comparable validation result with an edge detection model proposed by an expert.

8.1 Introduction

As described in Chapter. 6, edge detection is the most fundamental task in computer vision and it is a significant preprocessing step for numerous applications, such as object recognition, segmentation and so on [103, 77, 36, 10]. According to Chapter. 6, edge detectors can be divided into three types chronologically, they are: 1. Traditional edge detectors such as the Robert cross detector [90], Sobel detector or Canny detector [14] aim at finding abrupt changes in image intensity [71]. But for real images, many salient edges do not necessarily correspond to changes in low-level features like color or brightness. 2. Modern edge detectors no longer only focus on the intensity changes, but take the changes between textured regions into consideration [67], such as the pb detector [71] and gpb (global pb) detector [3] and so on. Compared with the traditional operators, these kind of modern detectors care more

about the completeness of contours of objects. 3. The third kind of edge detector are trained by using deep learning techniques [102, 8]. Thanks to the huge database of images and their corresponding groundtruth, these detectors nearly perform best among the current edge detectors.

It is widely known that the major difference between the deep learning technology and traditional methods lies in that whether it uses expert experience. Besides, the deep learning methods utilize a lot of training images to achieve a satisfactory result. It is a very interesting question in our opinion that how many examples of images generated according to a given model are necessary for the learning process to achieve performance equivalent to segmentation process designed specifically for the given model.

In the past three decades, a lot of model based edge detectors have been proposed. For example, the active contour models [50, 19], where the authors use an energy-minimizing method to find the image edges. In [62], Lindeberg presented a mechanism for automatic selection of scale levels when detecting edges. In [52], Kimmel and Bruckstein proposed an optimal edge integrator by taking the edge integration problem for object segmentation as a geometric variational framework. The advantage of using the model base edge detectors is that theese techniques could be well described by reasonable mathematical methods and they give very intuitive detection process. More importantly, they give optimal detection results under certain conditions.

In this chapter, we compare two methods, one is the PointFlow which is proposed by ourselves and the other one is the DeepContour which is proposed in [102]. According to the experiments on synthetic and real data, the PointFlow can achieve a better performance than the traditional edge detectors. And it is also possible to combine the other features such as texture feature with the PointFlow. The DeepContour detector achieve nearly the best on the BSDS dataset [69]. To the best of our knowledge, the existing edge detectors aim at comparing the performance and no other research has been done on this work.

The current existing dataset are mostly composed by real images and the groundtruth are always labeled by humans, which remains subjective. This is because that the edges in real images are hard to be precisely localized by humans. The groundtruth labeled by humans may be close and similar but not exactly and necessarily identical. In those dataset, even there exist differences, all of the hand labeled results are considered as groundtruth. Hence, the evaluation processes tolerate a lot of deviations sometimes. In this chapter, to achieve our goal, it is expected to generate a dataset which contains a large set of synthetic images. For these images, the edges and contours can be precisely obtained by simply calculating the gradient or using a threshold. By doing so, there will be no ambiguous results in the groundtruth. For evaluation, we will set up a range of tolerance from 0 to 2 pixels and observe the performance of each method.

The contribution of this chapter is that, 1) we build a database which contains 300 artificial original images and their corresponding groundtruth; 2) we quantitatively analyze how many training images are necessary for the DeepContour detector to achieve a good or even better result compared with the PointFlow model.

The chapter is organized as follows: Sect.8.2 introduces the two methods: DeepContour and PointFlow; Sect.8.3 details the data preparation work; Sect.8.4 illustrates how we evaluate the performance of the two methods and Sect.8.5 shows the comparison results. Sect.8.6 concludes the chapter.

8.2 Related Work

To answer the question above, we shall choose at least two edge detectors separately from traditional detectors and deep learning detectors. In this chapter, we use the PointFlow model as the representative of traditional edge detector and the DeepContour detector as the representative of the deep learning ones. According to the experiments on both artificial and real data, the PointFlow model outperforms the other traditional detectors.

8.2.1 PointFlow Model

The PointFlow model is presented is Chapter. 7. For the purpose of detecting edges on images, the designed field is a field oriented around the edges. In this chapter, it is simply obtained by combining the first- and second- order derivatives of the image. It can be described by the following Ordinary Differential Equation:

$$P'(t) = \mathbf{V}(P(t)) \tag{8.1}$$

where $P(t)$ is a point function which represents the position at time t, and $\mathbf{V}(\cdot)$ is the vector field generated from the image $I \in \mathbb{R}$:

$$\mathbf{V} = \zeta \cdot \nabla ||\nabla I|| \pm \xi \cdot \nabla I^{\perp} \tag{8.2}$$

where we use $\zeta = \xi = 0.5$.

The PointFlow model is an effective and robust edge detector. The edge detection result by this model is continuous and sub-pixel level, which means that the detection result is precise.

8.2.2 DeepContour

The DeepContour method is also introduced in Chapter .7 and it is achieved by using a multi-scale deep network which is made up of five convolutional layers and a bifurcated fully-connected sub-network [8]. The architecture of the CNNs of DeepContour can be found in Chapter 6. For training, we are using the same parameters as the original paper did on the BSDS500 dataset.

8.3 Preparation of Dataset

In [43], the authors propose that the edge characterization should rely on the the goal of what we want to do with the edges, such as object recognition or image segmentation. Current dataset for image segmentation, classification or edge detection such as the BSDS segmentation dataset [69], the drive database for vessel segmentation[105] and the NYU depth dataset [75] are composed by real images, such as natural images, medical images, or satellite images, and so on. The groundtruth of these images are labeled manually, which more or less remains subjective and may result in false for validations. Hence, these dataset do not conform with our goal: we want to compare the detection results exactly and precisely.

To achieve our goal, we are urged to build a dataset which is made up of artificial and simple image examples so that the edges are obvious and unique. To meet our needs, we determine to generate circles images, because circles are smooth to trace, and their groundtruth is very easy to obtain. More importantly, we can get as many examples as we want.

The groundtruth of this dataset is precise because we know exactly the center and radius of each circle when we generate the circles. But we will have to round up and down the position of the boundaries. Figure.8.1 gives us some examples of the dataset and the corresponding groundtruth.

8.4 Evaluation With Tolerance

The evaluation method in this paper is pixel level. We use recall and precision as evaluation metrics. The accuracy of edge location will affect the evaluation results, so a small tolerance of deviation is possible. Basically, the tolerance permits a displacement Δd between the groundtruth edge and the detected edge.

Take Figure.8.2 as an example. The blue segment is the groundtruth, and the purple ones are the detected edges. If $\Delta d = 0$, the purple edges do not coincide with the groundtruth, so it will be regarded as a false detection. When tolerance $\Delta d = 1$, which is illustrated by the

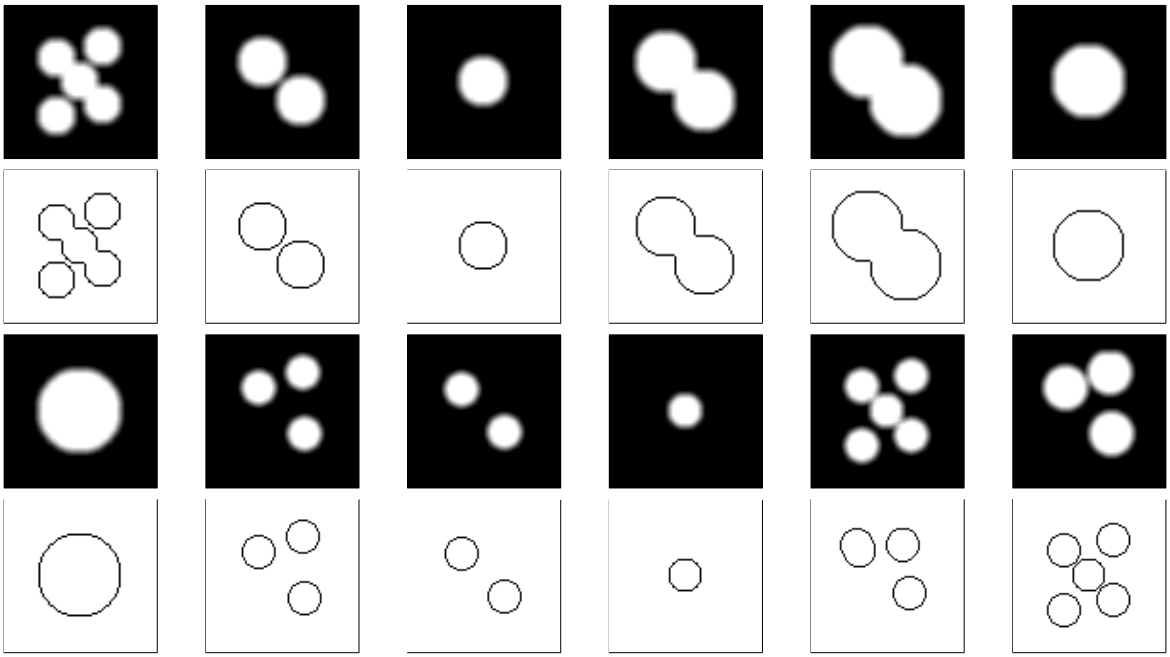


Fig. 8.1 Some examples of the dataset and their groundtruth

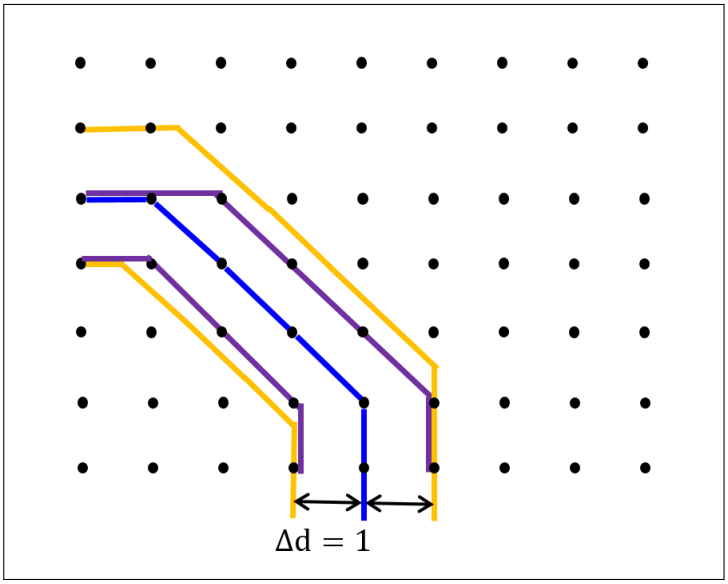


Fig. 8.2 The sketch map of using a tolerance = 1.

region limited by the two yellow line, any detected edges (purple segments) located in this region are considered as the corrected detection results. In this paper, the tolerance ranges from 0 to 1 pixel, $\Delta d = \{0, 1\}$. The precision and recall can be obtained by using Eq.(8.3).

$$\begin{cases} \text{precision} = \frac{\sum \mathbb{1}([P_{GT} * T_{\Delta d}] \cap P_{detect})}{\sum P_G T} \\ \text{recall} = \frac{\sum \mathbb{1}([P_{GT} * T_{\Delta d}] \cap P_{detect})}{\sum P_{detect}} \end{cases} \tag{8.3}$$

where P_{GT} and P_{detect} respectively denote the pixels of the groundtruth and the detected edge. $\mathbb{1}$ is an indicator function, when its input is null, $\mathbb{1}(\cdot) = 0$, otherwise, $\mathbb{1}(\cdot) = 1$. In addition, $T_{\Delta d}$ is a template, when $T_{\Delta d} > 0$, it is a matrix of size $(2\Delta d) \times (2\Delta d)$. When $\Delta d = 0$, $T_{\Delta d} = 1$.

8.5 Comparison

In this part, we test the above two methods on our dataset. We do not only compare the results on the original data, we also test the two methods by adding noise to the data increasingly.

8.5.1 Data settings

For the original data, on each image plane, there are n discs with different centers, different radius. n is the number of circles and it can be arbitrary. In our cases, we set $n = \{1, 2, 3, 4, 5\}$. The size of the image is also different, ranging from 50×50 to 100×100. For the noisy data, we use two kinds of noise on the original data, the pepper & salt noise and the Gaussian noise. For the former one, 10% and 20% noise are used respectively. For the latter one, the variances are 0.1 and 0.2.

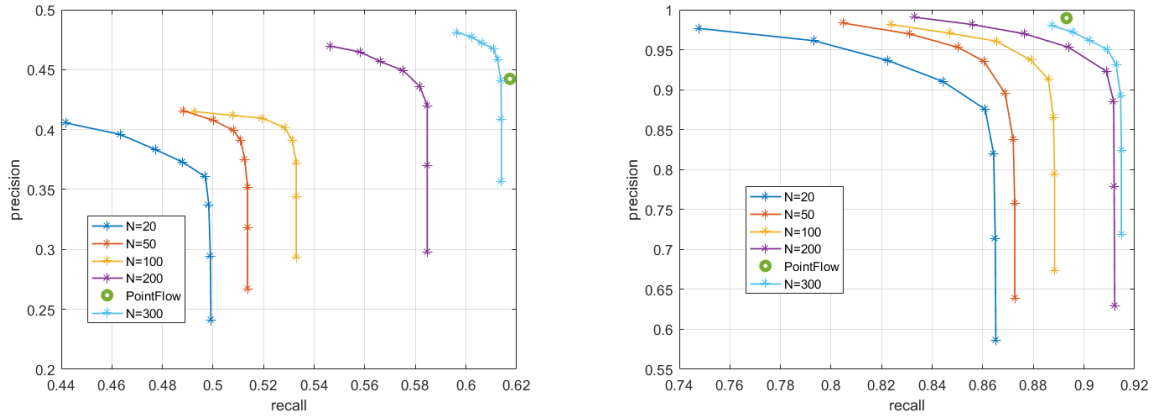
The mean of the Gaussian filter for the original circle data is 4 and the variance is 2 when the size is smaller than 80×80, otherwise it is 5 and 3 respectively.

We use the same parameter setting as the DeepContour is trained in [102]. For the PointFlow method, 200 random source points are selected. Due to that the results by PointFlow is "hard" boundary, the precision and recall will be a single point on the figure.

In addition, we generate 100 circle images for testing.

8.5.2 Result and analysis

Figure.8.3 shows the precision and recall curve of the original data. It clearly indicates that with the increase of the training images, the performance of DeepContour is also increasing. When the training examples $N \approx 300$, the detection result of DeepContour is competitive against the PointFlow model with a tolerance.



(a) No tolerance $\Delta d = 0$ (b) $\Delta d = 1$

Fig. 8.3 The precision-recall curve of different deviation tolerance by using DeepContour and PointFlow.

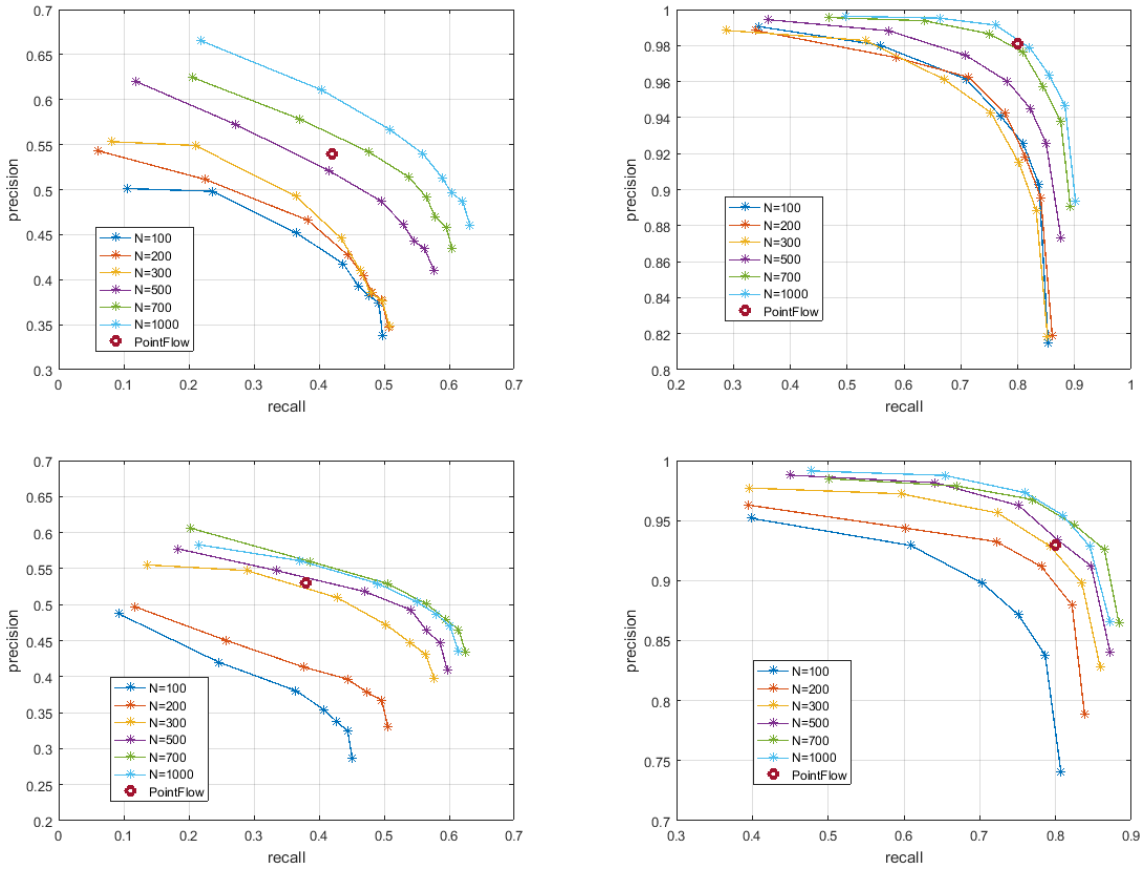
When the original data is damaged by 10% of pepper & salt noise, the precision & recall is shown in the above row of Figure.8.4. We can see clearly that more than 500 training examples should be used for the DeepContour that it can exceed the PointFlow method. But when we increase the percentage of the noise (20%), the PointFlow method will be more influenced by the noise, the DeepContour will give a comparable result when the number of training images $N > 300$, see the bottom row of Figure.8.4.

Figure.8.5 shows the results on Gaussian noise images. It further proves the previous opinions: with the increase of training images, the DeepContour behaves better. As for the noise, the PointFlow method is more affected by the noise than the DeepContour. But hundreds of examples are still needed for the DeepContour to achieve a satisfying result (result obtained by PointFlow method).

8.6 Conclusion

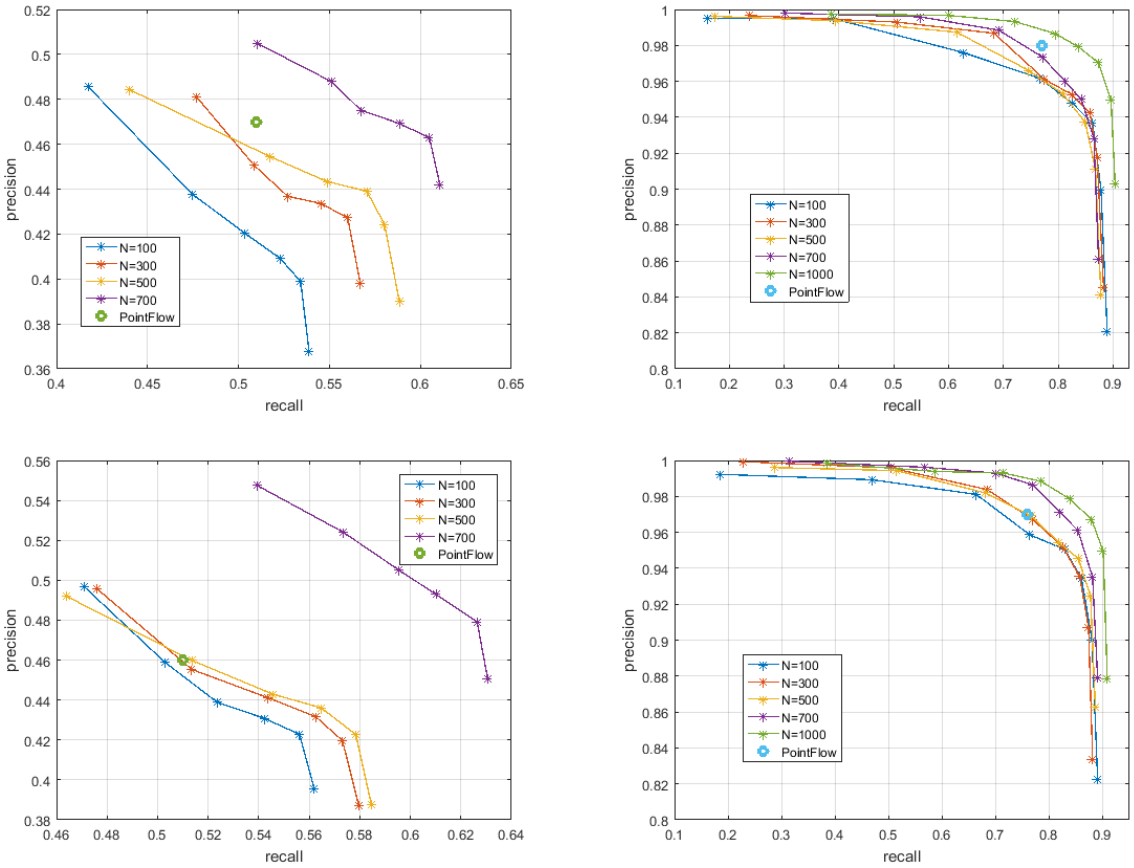
Nowadays the deep learning techniques have influenced the computer vision field tremendously. Thanks to these techniques, the performance for different applications has been improved a lot.

In this paper, we compare two edge detectors, one is PointFlow which is designed based on human experience, the other one is DeepContour which is trained by deep learning. We want to find out that for a deep learning edge detector, how many training images are needed to achieve a desired result. In order to achieve this purpose, we build a dataset composed of thousands of artificial images in this paper and we qualitatively analyze the detection result on this dataset.



(a) No tolerance $\Delta d = 0$ (b) $\Delta d = 1$

Fig. 8.4 The precision-recall curve of different deviation tolerance by using DeepContour and PointFlow on noisy (pepper & salt) images, 10% and 20% damaged above and below respectively



(a) No tolerance $\Delta d = 0$ (b) $\Delta d = 1$

Fig. 8.5 The precision-recall curve of different deviation tolerance by using DeepContour and PointFlow on noisy (Gaussian noise) images, and damaged above and below respectively

From the experiment result based on the dataset that we build, we can see that at least 300 examples should be used for training the DeepContour model to let it behave as good as the PointFlow model. It can be deduced that for the real images, more training images are needed to generate a rather good result by using the deep learning method. In addition, the training and testing process of the DeepContour model takes really long time. Our key point is that on the process of chasing for good result, we should also pay attention to the traditional methods. After all, a model is worth more than hundreds of examples.

Chapter 9

Conclusion

This chapter concludes this thesis by summarizing and discussing the results obtained in the development of the different research topics. At the end, it gives an outlook for the future work.

9.1 Summary

This thesis mainly focuses on two works: 1). applying the heat method to image segmentation; 2). detecting and tracing edges by using the PointFlow model.

- **Image segmentation by heat method**

  Heat diffusion can be used to approximate the process of computing geodesic distance on Riemannian manifolds by applying Varadhan's formula proposed in 1967. With the geodesic distance, the users can obtain the geodesic curves easily by applying an ordinary differential equation. In this thesis, we firstly testify the availability of Varadhan's formula for both isotropic and anisotropic heat equation. Then we apply different metrics (isotropic of anisotropic) to the heat equation to obtain the desired geodesic distance and curves on images or surfaces. For the purpose of reducing the manual intervention, we propose two automatic methods that are specially designed for the segmentation by heat method: the first one is a voting method and the second one is a keypoint method. Users will no longer have to provide the endpoints. Finally, we extend the heat method on a 2D image plane to 3D, where the 3rd dimension in fact describes the width of the tubular structures to be extracted. The experimental results show that the segmentation by using heat method is not only fast and effective, but also remains very robust.

- **Edge detection by PointFlow model**

The edge detection remains the most fundamental problem in computer vision. It is a very important preprocessing step for higher-level applications such as image segmentation or object recognition. In this thesis, we classify the edge detectors into three types. We first introduce some representative edge detectors for each type. Then we propose our own model: the PointFlow model. This model is designed by considering the image plane a vector field (based on the first- and second- order derivative of the image) and the pixels are particles which can move according to the vector field. This model can trace the boundaries of objects on image planes. It can also combine the low level feature, such as the brightness or color with high-level feature such as the texture, spectral features to achieve a better result. Except for detecting edges, we also extend the PointFlow model to infer illusory contours. Finally, we compare the PointFlow model with the DeepEdge detector by testing on our own dataset and figure out how many examples of images should be used that a deep learning edge detector can have an equivalent results as the PointFlow model.

9.2 Perspective

- **Heat Diffusion for Automatic 3D Medical Image Vessel Segmentation**

  The vessel segmentation methods proposed in this thesis focus on detecting and segmenting the tubular structures in 2D images. With the development of imaging technology, such as the MRA (Magnetic resonance angiography) based on the MRI (Magnetic Resonance Imaging) and the CTA (Computed tomography angiography) based on the CT (X-Ray Computed Tomography), 3D medical images can be obtained and help to diagnose the diseases. So the segmentation of vessels in 3D images are really important. In fact, there are a lot of researches on the 3D vessel segmentation, but to the best of our knowledge, no one has applied the heat method to do it. Due to the advantages of the heat method on 2D images, it is interesting to see how it will perform on the 3D images.

- **Heat Diffusion on Asymmetric Manifold**

  Inspired by the work of [15], we are interested in applying the asymmetric metric to the heat method.

- **PointFlow Model to Infer High Level Information**

  The edges and boundaries detected by the PointFlow model can be exploited to do high level tasks, such as the inference of elevation information. Take a 2D satellite road network image as an example, the road network can be detected by using the PointFlow model. When there is a overcross on the road image, we may find out it is a junction or

an overpass. In addition, we are also interested in distinguishing the arteries from the veins in retinal images.

References

[1] Abdou, I. E. and Pratt, W. K. (1979). Quantitative design and evaluation of enhancement/thresholding edge detectors. *Proceedings of the IEEE*, 67(5):753–763.

[2] Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., and Desbrun, M. (2003). Anisotropic polygonal remeshing. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 485–493. ACM.

[3] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916.

[4] Benmansour, F. and Cohen, L. D. (2009a). Fast object segmentation by growing minimal paths from a single point on 2D or 3D images. *Journal of Mathematical Imaging and Vision*, 33(2):209–221.

[5] Benmansour, F. and Cohen, L. D. (2009b). Tubular anisotropy segmentation. In *Scale Space and Variational Methods in Computer Vision*, pages 14–25. Springer.

[6] Benmansour, F. and Cohen, L. D. (2011). Tubular structure segmentation based on minimal path method and anisotropic enhancement. *International Journal of Computer Vision*, 92(2):192–210.

[7] Benmansour, F., Cohen, L. D., Law, M., and Chung, A. (2009). Tubular anisotropy for 2D vessel segmentation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2286–2293. IEEE.

[8] Bertasius, G., Shi, J., and Torresani, L. (2015a). Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4380–4389.

[9] Bertasius, G., Shi, J., and Torresani, L. (2015b). High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 504–512.

[10] Borenstein, E. and Ullman, S. (2008). Combined top-down/bottom-up segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 30(12):2109–2125.

[11] Bowyer, K., Kranenburg, C., and Dougherty, S. (1999). Edge detector evaluation using empirical roc curves. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1, pages 354–359. IEEE.

[12] Bruckstein, A. M. (1988). On shape from shading. *Computer Vision, Graphics, and Image Processing*, 44(2):139–154.

[13] Butcher, J. C. (2000). Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics*, 125(1):1–29.

[14] Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698.

[15] Chen, D. (2016). *New Minimal Paths Models for Tubular Structure Extraction and Image Segmentation*. PhD thesis, Université Paris Dauphine.

[16] Chen, Y., Davis, T. A., Hager, W. W., and Rajamanickam, S. (2008). Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22.

[17] Cohen, L. D. (1991). On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218.

[18] Cohen, L. D. and Cohen, I. (1993). Finite-element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Transactions on Pattern Analysis and machine intelligence*, 15(11):1131–1147.

[19] Cohen, L. D. and Kimmel, R. (1997). Global minimum for active contour models: A minimal path approach. *International journal of computer vision*, 24(1):57–78.

[20] Cohen-Steiner, D. and Morvan, J.-M. (2003). Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 312–321. ACM.

[21] Constantinescu, R., Costanzino, N., Mazzucato, A. L., and Nistor, V. (2009). Approximate solutions to second order parabolic equations i: analytic estimates. *arXiv preprint arXiv:0910.1562*.

[22] Courant, R., Friedrichs, K., and Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234.

[23] Crane, K., Weischedel, C., and Wardetzky, M. (2013). Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, 32(5):152.

[24] Davis, T. A. (2006). *Direct methods for sparse linear systems*. SIAM.

[25] Davis, T. A. and Hager, W. W. (2009). Dynamic supernodes in sparse cholesky update/downdate and triangular solves. *ACM Transactions on Mathematical Software (TOMS)*, 35(4):27.

[26] Deriche, R. (1987). Using canny's criteria to derive a recursively implemented optimal edge detector. *International journal of computer vision*, 1(2):167–187.

[27] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

[28] Dollar, P., Tu, Z., and Belongie, S. (2006). Supervised learning of edges and object boundaries. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1964–1971. IEEE.

[29] Dollár, P. and Zitnick, C. L. (2013). Structured forests for fast edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1848.

[30] Dollár, P. and Zitnick, C. L. (2015). Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1558–1570.

[31] Douglas, J. and Rachford, H. H. (1956). On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439.

[32] Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

[33] DUDA/HART (1973). *Pattern classification and scene analysis*. John Wiley.

[34] Eua-Anant, N. and Udpa, L. (1999). Boundary detection using simulation of particle motion in a vector image field. *Image Processing, IEEE Transactions on*, 8(11):1560–1571.

[35] Fehrenbach, J. and Mirebeau, J.-M. (2014). Sparse non-negative stencils for anisotropic diffusion. *Journal of Mathematical Imaging and Vision*, 49(1):123–147.

[36] Ferrari, V., Fevrier, L., Jurie, F., and Schmid, C. (2008). Groups of adjacent contour segments for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 30(1):36–51.

[37] Fram, J. R. and Deutsch, E. S. (1975). On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE Transactions on Computers*, 100(6):616–628.

[38] Frangi, A. F., Niessen, W. J., Vincken, K. L., and Viergever, M. A. (1998). Multiscale vessel enhancement filtering. In *Medical Image Computing and Computer-Assisted Interventation MICCAI 98*, pages 130–137. Springer.

[39] Freeman, W. T., Adelson, E. H., et al. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906.

[40] Ganin, Y. and Lempitsky, V. (2014). N^4-fields: Neural network nearest neighbor fields for image transforms. In *Asian Conference on Computer Vision*, pages 536–551. Springer.

[41] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[42] Haralick, R. M. (1984). Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):58–68.

[43] Heath, M., Sarkar, S., Sanocki, T., and Bowyer, K. (1996). Comparison of edge detectors: a methodology and initial study. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 143–148. IEEE.

[44] Heath, M. D., Sarkar, S., Sanocki, T., and Bowyer, K. W. (1997). A robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1338–1359.

[45] Hu, J., Razdan, A., Femiani, J. C., Cui, M., and Wonka, P. (2007). Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE Transactions on Geoscience and Remote Sensing*, 45(12):4144–4157.

[46] Hysing, S.-R. and Turek, S. (2005). The eikonal equation: numerical efficiency vs. algorithmic complexity on quadrilateral grids. In *Proceedings of ALGORITMY*, volume 2005, pages 22–31.

[47] Jbabdi, S., Bellec, P., Toro, R., Daunizeau, J., Pélégrini-Issac, M., and Benali, H. (2008). Accurate anisotropic fast marching for diffusion-based geodesic tractography. *Journal of Biomedical Imaging*, 2008:2.

[48] J.Sun, M.Ovsjanikov, and L.Guibas (2009). A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum*, 28(5):1383–1392.

[49] J.Weickert (1998). *Anisotropic diffusion in image processing*. Teubner-Verlag.

[50] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331.

[51] Kaul, V., Yezzi, A., and Tsai, Y. (2012). Detecting curves with unknown endpoints and arbitrary topology using minimal paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1952–1965.

[52] Kimmel, R. and Bruckstein, A. M. (2003). Regularized laplacian zero crossings as optimal edge integrators. *International Journal of Computer Vision*, 53(3):225–243.

[53] Kitchen, L. and Rosenfeld, A. (1981). Edge evaluation using local edge coherence. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(9):597–605.

[54] Kivinen, J. J., Williams, C. K., Heess, N., et al. (2014). Visual boundary prediction: A deep neural prediction network and quality dissection. In *AISTATS*, volume 1, page 9.

[55] Kokkinos, I. (2010). Boundary detection using f-measure-, filter-and feature-(f3) boost. In *European Conference on Computer Vision*, pages 650–663. Springer.

[56] Köthe, U. (2003). Edge and junction detection with an improved structure tensor. In *Pattern Recognition*, pages 25–32. Springer.

[57] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[58] Law, M. W. and Chung, A. C. (2008). Three dimensional curvilinear structure detection using optimally oriented flux. In *Europeen Conference on Computer Vision, ECCV 2008*, pages 368–382. Springer.

[59] Leordeanu, M., Sukthankar, R., and Sminchisescu, C. (2014). Generalized boundaries from multiple image interpretations. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1312–1324.

[60] Li, H. and Yezzi, A. (2007). Vessels as 4-D curves: Global minimal 4-D paths to extract 3-D tubular surfaces and centerlines. *Medical Imaging, IEEE Transactions on*, 26(9):1213–1223.

[61] Lim, J. J., Zitnick, C. L., and Dollár, P. (2013). Sketch tokens: A learned mid-level representation for contour and object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3158–3165.

[62] Lindeberg, T. (1998). Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):117–156.

[63] Lu, C., Chi, Z., Chen, G., and Feng, D. (2006). Geometric analysis of particle motion in a vector image field. *Journal of Mathematical Imaging and Vision*, 26(3):301–307.

[64] Mairal, J., Leordeanu, M., Bach, F., Hebert, M., and Ponce, J. (2008). Discriminative sparse image models for class-specific edge detection and image interpretation. In *European Conference on Computer Vision*, pages 43–56. Springer.

[65] Maire, M., Arbeláez, P., Fowlkes, C., and Malik, J. (2008). Using contours to detect and localize junctions in natural images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[66] Makhervaks, V., Barequet, G., and Bruckstein, A. (2002). Image flows and one-liner graphical image representation. *Annals of the New York Academy of Sciences*, 972(1):10–18.

[67] Malik, J., Belongie, S., Leung, T., and Shi, J. (2001). Contour and texture analysis for image segmentation. *International journal of computer vision*, 43(1):7–27.

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

[69] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001a). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 416–423. IEEE.

[70] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001b). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423.

[71] Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549.

[72] Mirebeau, J.-M. (2014). Anisotropic fast-marching on cartesian grids using lattice basis reduction. *SIAM Journal on Numerical Analysis*, 52(4):1573–1599.

[73] Mirebeau, J.-M., Fehrenbach, J., Risser, L., and Tobji, S. (2015). Anisotropic diffusion in ITK. *arXiv preprint arXiv:1503.00992*.

[74] Morrone, M. C. and Burr, D. (1988). Feature detection in human vision: A phase-dependent energy model. *Proceedings of the Royal Society of London B: Biological Sciences*, 235(1280):221–245.

[75] Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*.

[76] Ng, E. and Peyton, B. W. (1993). A supernodal cholesky factorization algorithm for shared-memory multiprocessors. *SIAM Journal on scientific computing*, 14(4):761–769.

[77] Opelt, A., Pinz, A., and Zisserman, A. (2006). A boundary-fragment-model for object detection. *Computer Vision–ECCV 2006*, pages 575–588.

[78] Pardoux, E. (1980). Stochastic partial differential equations and filtering of diffusion processes. *Stochastics*, 3(1-4):127–167.

[79] Peaceman, D. W. and Rachford, Jr, H. H. (1955). The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for industrial and Applied Mathematics*, 3(1):28–41.

[80] Perona, P. and Malik, J. (1990a). Detecting and localizing edges composed of steps, peaks and roofs. In *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 52–57. IEEE.

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

[82] Peyré, G., Péchaud, M., Keriven, R., and Cohen, L. D. (2010). Geodesic methods in computer vision and graphics. *Foundations and Trends® in Computer Graphics and Vision*, 5(3–4):197–397.

[83] Pinkall, U. and Polthier, K. (1993). Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36.

[84] Prewitt, J. M. (1970). Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19.

[85] Ramesh, V. and Haralick, R. M. (1992). Random perturbation models and performance characterization in computer vision. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 521–527. IEEE.

[86] Raviv, D. and Kimmel, R. (2015). Affine invariant geometry for non-rigid shapes. *International Journal of Computer Vision*, 111(1):1–11.

[87] Recktenwald, G. W. (2004). Finite-difference approximations to the heat equation. *Class Notes*.

[88] Ren, X. (2008). Multi-scale improves boundary detection in natural images. In *European conference on computer vision*, pages 533–545. Springer.

[89] Reuter, M., Biasotti, S., Giorgi, D., Patanè, G., and Spagnuolo, M. (2009). Discrete laplace–beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3):381–390.

[90] Roberts, L. G. (1963). *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology.

[91] Rouchdy, Y. and Cohen, L. D. (2008). Image segmentation by geodesic voting. application to the extraction of tree structures from confocal microscope images. In *19th International Conference on Pattern Recognition, ICPR 2008.*, pages 1–5. IEEE.

[92] Rouchdy, Y. and Cohen, L. D. (2011). A geodesic voting method for the segmentation of tubular tree and centerlines. In *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 979–983. IEEE.

[93] Rouchdy, Y. and Cohen, L. D. (2013). Geodesic voting for the automatic extraction of tree structures. *Computer Vision and Image Understanding*, 117(10):1453–1467.

[94] Rubner, Y. and Tomasi, C. (1996). Coalescing texture descriptors. In *ARPA Image Understanding Workshop*, pages 927–935.

[95] Schmitz, P. G. and Ying, L. (2012). A fast direct solver for elliptic problems on general meshes in 2D. *Journal of Computational Physics*, 231(4):1314–1338.

[96] Schmitz, P. G. and Ying, L. (2014). A fast nested dissection solver for cartesian 3d elliptic problems using hierarchical matrices. *Journal of Computational Physics*, 258:227–245.

[97] Sethian, J. A. (1996). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595.

[98] Sethian, J. A. (1999). Fast marching methods. *SIAM review*, 41(2):199–235.

[99] Sethian, J. A. and Vladimirsky, A. (2000). Fast methods for the eikonal and related hamilton–jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences*, 97(11):5699–5703.

[100] Sethian, J. A. and Vladimirsky, A. (2003). Ordered upwind methods for static hamilton–jacobi equations: Theory and algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363.

[101] Shechtman, E. and Irani, M. (2007). Matching local self-similarities across images and videos. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[102] Shen, W., Wang, X., Wang, Y., Bai, X., and Zhang, Z. (2015). Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3982–3991.

[103] Shotton, J., Blake, A., and Cipolla, R. (2005). Contour-based learning for object detection. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 503–510. IEEE.

[104] S.R.S.Varadhan (1967). On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics*, 20(2):431–455.

[105] Staal, J., Abràmoff, M. D., Niemeijer, M., Viergever, M. A., and Van Ginneken, B. (2004). Ridge-based vessel segmentation in color images of the retina. *IEEE transactions on medical imaging*, 23(4):501–509.

[106] Taylor, T. J. (1989). Off diagonal asymptotics of hypoelliptic diffusion equations and singular riemannian geometry. *Pacific journal of mathematics*, 136(2):379–399.

[107] Tola, E., Lepetit, V., and Fua, P. (2008). A fast local descriptor for dense matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[108] Tsai, Y.-h. R. (2002). Rapid and accurate computation of the distance function using grids. *Journal of Computational Physics*, 178(1):175–195.

[109] Tsai, Y.-H. R., Cheng, L.-T., Osher, S., and Zhao, H.-K. (2003). Fast sweeping algorithms for a class of hamilton–jacobi equations. *SIAM journal on numerical analysis*, 41(2):673–694.

[110] Tsitsiklis, J. N. (1995). Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538.

[111] Ullman, S. and Basri, R. (1991). Recognition by linear combinations of models. *IEEE transactions on pattern analysis and machine intelligence*, 13(10):992–1006.

[112] Unsworth, J. and Duarte, F. (1979). Heat diffusion in a solid sphere and fourier theory: An elementary practical example. *American Journal of Physics*, 47(11):981–983.

[113] Vassilevich, D. V. (2003). Heat kernel expansion: user's manual. *Physics reports*, 388(5):279–360.

[114] Von Der Heyclt, R., Peterhans, E., and Baurngartner, G. (1984). Illusory contours and cortical neuron responses. *Science*, 224.

[115] Weickert, J. (1999). Coherence-enhancing diffusion filtering. *International Journal of Computer Vision*, 31(2-3):111–127.

[116] Weickert, J. and Scharr, H. (2002). A scheme for coherence-enhancing diffusion filtering with optimized rotation invariance. *Journal of Visual Communication and Image Representation*, 13(1):103–118.

[117] Will, S., Hermes, L., Buhmann, J. M., and Puzicha, J. (2000). On learning texture edge detectors. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 3, pages 877–880. IEEE.

[118] Witkin, A. P. (1984). Scale-space filtering: A new approach to multi-scale description. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'84.*, volume 9, pages 150–153. IEEE.

[119] Xiaofeng, R. and Bo, L. (2012). Discriminatively trained sparse code gradients for contour detection. In *Advances in neural information processing systems*, pages 584–592.

[120] Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403.

[121] Xu, C. and Prince, J. L. (1998). Snakes, shapes, and gradient vector flow. *Image Processing, IEEE Transactions on*, 7(3):359–369.

[122] Yang, F., Bruckstein, A. M., and Cohen, L. D. (2017). A model for automatically tracing object boundaries. In *International Conference on Image Processing*. IEEE.

[123] Yang, F. and Cohen, L. D. (2016). Geodesic distance and curves through isotropic and anisotropic heat equations on images and surfaces. *Journal of Mathematical Imaging and Vision*, 55(2):210–228.

[124] Yang, F. and Cohen, L. D. (2017). Tubular structure segmentation based on heat diffusion. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 54–65. Springer.

[125] Zhao, H. (2005). A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627.

[126] Zheng, S., Yuille, A., and Tu, Z. (2010). Detecting object boundaries using low-, mid-, and high-level information. *Computer Vision and Image Understanding*, 114(10):1055–1067.

[127] Zhu, Q., Song, G., and Shi, J. (2007). Untangling cycles for contour grouping. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[128] Ziou, D. and Tabbone, S. (1998). Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559.

[129] Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer.

Résumé

Cette thèse vise à fournir des méthods pour détécter des contours dans les images. Elle se compose par deux parties: 1, la distance et les courbes géodésique(s) obtenu par la diffusion de chaleur; 2, une méthode s'appelle l'ecoulement de point pour tracer les bords d'images.
Dans la première partie de cette thèse, nous appliquons des métriques différentes sur l'équation de chaleur pour rapprocher la distance géodésique. De plus, nous proposons deux algorithmes automatiques pour extraire des lignes géodsiques. Enfin, nous proposons une façon pour la segmentation de la structure tubulaire en ajoutant une autre dimension à l'équation de chaleur de 2D. Les résultats expérimentaux témoignent de la robustesse de notre méthode à base de chaleur.
Pour la deuxième partie de cette thèse, nous sommes intéressés par la détéction de bord. Nous proposons une méthode qui peut simuler le processus de tracer les bords d'objets automatique. Nous combinons des caractéristiques différentes pour détécter les bords. En outre, ce modèle est capable de déduire les contours illusoires. Nous testons notre méthode sur un dataset bien connu et les résultats sont comparable aux autres détecteurs de bords classiques.

Mots Clés

Chemins Minimal, Geodésique, Diffusion de Chaleur, Segmentation d'images, Equation Différentielle Partielle Eikonal, Isotrope et Anisotrope, Structure Tubulaire, Ecoulement de Point, Détéction de Bord, Equation Différentielle Ordinaire

Abstract

This thesis aims at providing methods for finding contours in images and is composed by two parts: geodesic distance and curves computations using heat diffusions and a boundary-tracking method named point flow.
In the first part of this thesis, we apply different geodesic metrics for heat equation to approximate the geodesic distance. Additionally, we propose two algorithms for extracting the geodesic lines automatically. At last, we come up with an approach for the segmentation of tubular structure by adding an additional dimension to the 2D heat equation. The experimental results testify the robustness and effectiveness of our heat-based method.

For the second part of this thesis, we are interested in one of the most fundamental problems in computer vision: edge detection. We propose a model called point flow which can simulate the process of tracking object boundaries automatically. On the way of finding contours, we do not only consider the gray-level information, but also combine different features to detect the edges and contours. Moreover, this model can be also used to infer the illusory contours.
We test our edge detection method on a well known dataset and the result is comparable to the other classical edge detectors.

Keywords

Minimal Paths, Geodesic, Heat Diffusion, Image Segmentation, Eikonal Partial Differential Equation, Isotropic and Anisotropic, Tubular Structure, Point Flow, Edge Detection, Ordinary Differential Equation