Mathematical morphology and graphs: application to interactive medical image segmentation

Jean Stawiaski

MINES ParisTech

N attribué par la bibliothèque

THÈSE

pour obtenir le grade de

Docteur de l'École des Mines de Paris

Spécialité "Morphologie Mathématique"

Présentée et soutenue publiquement
par

JEAN STAWIASKI

le 13 Octobre 2008

Mathematical Morphology and Graphs : Application to Interactive Medical Image Segmentation

Directeurs de thèse: Etienne Decencière et Dominique Jeulin

Jury:

Jean Marie Becker	*Président du Jury*
Marc Van Droogenbroeck	*Rapporteur*
Philippe Salembier	*Rapporteur*
Francois Bidault	*Examinateur*
Thomas Sangild Sørensen	*Examinateur*
Etienne Decencière	*Directeur de thèse*
Dominique Jeulin	*Directeur de thèse*

Remerciements

"Plus on partage, plus on possède. Voilà le miracle"

Léonard Nimoy (Mr Spock).

Je voudrais avant toute chose remercier mes parents qui ont permis, à travers leur soutien, de me pousser constamment à fournir le meilleur de moi-même et à me motiver pour arriver jusqu'à l'écriture de cette thèse. Je les remercie de n'avoir jamais douté de moi, même dans les moments les plus difficiles. Leur confiance a été mon moteur principal durant ces années d'études. Je suis fier de pouvoir écrire aujourd'hui ces quelques lignes, et de leur dédicacer cette thèse.

J'aimerais également remercier mon frère, qui a été et qui reste pour moi un modèle de réussite. J'espère que l'avenir nous donnera l'occasion de mettre nos forces dans des projets communs.

Je voudrais en suite remercier mes directeurs de thèse, Dominique Jeulin et Etienne Decencière. Travailler à leur cotés a été un grand honneur. Leurs conseils et leurs aides furent précieux pour mener à bien cette thèse. Au-delà de leurs qualités scientifiques, ces personnes sont avant tout des hommes généreux, avec qui il est agréable de discuter, d'échanger des idées, et de plaisanter. J'espère que ces immenses qualités continueront à m'influencer pour longtemps.

Travailler au Centre de Morphologie Mathématique a été un vrai plaisir. Je voudrais remercier tout le personnel de ce centre, dans lequel règne une atmosphère agréable où il est plaisant de venir tout les matins. Ce lieu de travail a réellement une âme qui continuera, je l'espère, d'influencer de nombreuses générations de jeunes chercheurs comme moi. J'aimerais particulièrement remercier Catherine Moysan, sa patience à mon égard, et son aide m'a permis de passer ces trois années à Fontainebleau sans soucis pratiques. Je veux aussi adresser mes remerciments à tout les doctorants, post-docs et ingénieurs, nouveaux et anciens, avec qui j'ai eu l'honneur de travailler pendant ces années, particulièrement Guillaume Noyel avec qui j'ai passé une grande partie de mes études.

Je suis particulièrement redevable à Fernand Meyer, directeur du centre, ces conseils et son talent m'inspireront certainement tout au long de ma carrière. J'ai aussi été particulièrement heureux de pouvoir partager avec lui certaines particularités de nos origines alsaciennes. J'espère que nos collaborations seront fructueuses et dureront pour encore plusieurs années.

J'aimerais aussi adresser mes remerciements à Jean-Marie Becker, qui m'a en quelques sortes donné l'envie de préparer cette thèse. Il fait partie des professeurs qui ont joué un grand rôle dans mes choix d'études. Je voudrais remercier Thomas Sangild Sørensen et Jesper Mosegaard qui m'ont accueilli chaleureusement au Danemak. Les quelques mois que j'ai passé à leurs cotés m'ont appris énormément.

J'aimerais finalement remercier mes amis. Je garde de ces trois dernières années à Fontainebleau et à Paris des souvenirs impérissables. Je remercie Xavier Courtial pour m'avoir nourri pendant la grande partie de ma thèse. C'était "franc bien" d'avoir un cuisinier à portée de main même s'il est finalement

la raison principale des nombreux kilos que j'ai accumulé ces dernières années. Je remercie aussi Gilles Pelfrennes, Marco De Lucia, Gonzague Du Suau de La Croix, Aymen Selmi, Pedro Da Costa et Salim Bensmina pour les nombreuses parties de belote, de pes, de poker et autres divertissements qui ont rythmé nos soirées à la résidence universitaire. Je remercie particulièrement Marco pour m'avoir acceuilli plusieurs fois et fait découvrir les beautés de sa région en Italie. J'aimerais remercier Rose Bensimon, avec qui j'ai partagé des moment inoubliables de Fontainebleau à Ravenne. Son enthousiasme, son énergie et son humour m'ont donné le sourire un nombre incalculable de fois, pourvu que cela dure encore longtemps.

La liste des gens que j'ai rencontré à Fontainebleau est trop longue pour que tous soient cités ici, je remercie ainsi mes collègues footballeurs qui se reconnaîtront sans aucun doute.

Je voudrais aussi remercier ici une amie de longue date, Sophie Pabst, qui m'a fait découvrir les longues soirées parisiennes et qui m'a accueilli de nombreuses fois chez elle. Je la remercie d'être là pour moi après toutes ces années. Je me souviendrais longtemps des excellentes soirées que j'ai passé à Paris en companie de Sophie, Maria et Bruno. Je profite de cette occasion pour remercier chaleureusement Maria Belen Migone qui m'a gentillement acceuilli et fait découvrir son beau pays qu'est l'Argentine.

J'adresse finalement un remerciement particulier à Hellen Altendorf pour m'avoir tout d'abord soutenu professionnellement ; Hellen est une brillante scientifique, son remarquable sens logique m'a aidé à surmonter certains passages techniques de ce manuscrit. Mais son aide la plus précieuse m'est parvenue de sa gentillesse et de sa bienveilance. J'espère pouvoir lui rendre en double dans le futur et je la remercie d'avoir partager avec moi ce moment important de mon parcours.

Ces dernières années m'ont donné l'opportunité de connaitre des gens formidables, j'espère garder ces liens d'amitiés qui m'ont permis de traverser ces trois années dans la joie et la bonne humeur.

Abstract

Medical imaging is one of the most active research topics in image analysis. Analyzing and segmenting medical images in a clinical context remains a challenging task due to the multiplicity of imaging modalities and the variability of the patients characteristics and pathologies. Medical image processing also requires a human supervision for interpretation and validation purposes. The numerous applications and the huge amount of medical image data need complex software that combine high level graphical user interfaces as well as robust and fast interactive image analysis tools.

Recent research in image segmentation has highlighted the potential of graph based methods for medical applications. These new tools have generated a great interest in the imaging community. Graph theory is the framework used in this thesis to propose innovative image segmentation tools. The focus of this thesis is both theoretical and practical. On the theoretical level, we study properties of minimal spanning trees, shortest paths trees and minimal cuts and we revisit these notions for image segmentation purposes. It allows us to derive new theoretical links between minimal spanning trees, shortest paths trees and minimal multi-terminal cuts. These results highlight the possibilities and the limitations of each technique for image segmentation problems.

In a second step, we propose some theoretical and practical improvements of image segmentation based on graph cuts. This structure is particularly interesting for solving a fairly large variety of energy minimization problems. Our strategy is based on the use of the region adjacency graph produced by the watershed transform from mathematical morphology. The combination of morphological and graph cuts segmentation permits us to speed up and define new classes of energy functions that can be minimized using graph cuts. The use of region graphs gives promising results and can potentially become a leading method for interactive medical image segmentation.

The third point of this thesis details some qualitative and quantitative studies of medical image segmentation problems, which is the initial motivation of this work. We show that the developed methods are well suited for various medical image segmentation problems. We study applications to surgery simulation, radiotherapy planning and tumors delineation. We show by a quantitative analysis that the combination of morphological and graph cuts segmentation methods outperforms several recent and state of the art tools. This study shows that our methods can be used in a clinical context.

The last point of the thesis revisits and extends some classical graph based image segmentation tools. We revisit the well known watershed transform from the point of view of path optimization and path algebra. We recall existing properties of the watershed transform and propose some clear definitions of the watershed transform on graphs. Finally we also propose new extensions of the minimal graph cut problem for image segmentation purposes. New types of constraints are included in the classical minimal graph cut problem, which bring this standard problem into the linear programming framework. This class of combinatorial optimization problems is particulary interesting for image segmentation purposes since it provides a general framework for various constrained image segmentation problems such as boundary constrained minimal cuts and various geometric constrained minimal cuts. These new methods show great potential for various image segmentation scenarios.

Keywords : Image Segmentation, Graphs, Watershed Transform, Tumors Delineation, Linear Programming.

Résumé

La recherche en imagerie médicale est une des disciplines les plus actives du traitement d'images. La segmentation et l'analyse d'images dans un contexte clinique reste un problème majeur de l'imagerie médicale. La multiplicité des modalités d'imagerie, ainsi que les fortes variabilités des structures et pathologies à analyser rendent cette tâche fastidieuse. Dans la plupart des cas, la supervision de spécialistes, tels que des radiologistes, est nécessaire pour valider ou interpréter les résultats obtenus par analyse d'images. L'importante quantité de données, ainsi que les nombreuses applications liées à l'imagerie médicale, nécessitent des outils logiciels de très haut niveau combinant des interfaces graphiques complexes avec des algorithmes interactifs rapides.

Les récentes recherches en segmentation d'images ont montré les avantages et les qualité des méthodes à base de graphes. L'intérêt suscité dans la communauté scientifique a permis de développer et d'utiliser rapidement ces techniques dans de nombreuses applications. La théorie des graphes constitue aussi la base de nos travaux. Nous avons étudié les arbres de recouvrement minimaux, les coupes minimales ainsi que les arbres de chemins les plus courts. Notre étude a permis de mettre en lumière des liens entre ces structures a priori très différentes. Nous avons prouvé que les forêts des chemins les plus courts, ainsi que les coupes minimales convergent toutes les deux, en appliquant une transformation spécifique du graphe, vers une structure commune qui n'est autre qu'une forêt de recouvrement minimale. Cette étude nous a permis de souligner les limitations et les possibilités de chacune de ces techniques pour la segmentation d'images.

Dans un second temps, nous avons proposé des avancées théoriques et pratiques pour l'utilisation des coupe minimales. Cette structure est particulièrement intéressante pour segmenter des images à partir de critères de minimisation d'énergie. D'une part, nous avons montré que l'utilisation de graphes de régions d'une segmentation morphologique permet d'accélérer les méthodes à base de coupes minimales. D'autre part nous avons montré que l'utilisation de graphes de régions permet d'étendre la classe d'énergie pouvant être minimisée par coupe de graphes. Ces techniques ont toutes les caractéristiques pour devenir des méthodes de référence pour la segmentation d'images médicales.

Nous avons alors étudié qualitativement et quantitativement nos méthodes de segmentation à travers des applications médicales. Nous avons montré que nos méthodes sont pertinentes pour la détection de tumeurs pour la planification de radiothérapie, ainsi que la création de modèles pour la simulation et la planification de chirurgie cardiaque. Nous avons aussi mené une étude quantitative sur la segmentation de tumeurs du foie. Cette étude montre que nos algorithmes offrent des résultats plus stables et plus précis que de nombreuses techniques de l'état de l'art. Nos résultats ont finalement été comparés à des segmentations manuelles de radiologistes, prouvant que nos techniques sont adaptées à être utilisées en routine clinique.

Nous avons aussi revisité une méthode classique de segmentation d'images : la ligne de partages des eaux. La contribution de notre travail se situe dans la re-définition de cette transformation dans le cas des graphes et des images multi spectrales. Nous avons utilisé les algèbres de chemins pour montrer que la ligne de partages des eaux correspond à des cas particuliers de forêt des chemins les plus courts dans un graphe.

Finalement, nous proposons quelques extensions intéressantes du problème des coupes minimales. Ces extensions sont basées sur l'ajout de nouveaux types de contraintes. Nous considérons particulièrement les coupes minimales contraintes à inclure un ensemble prédéfini d'arrêtes, ainsi que les coupes minimales contraintes par leur cardinalité et leurs aires. Nous montrons comment ces problèmes peuvent être avantageusement utilisés pour la segmentation d'images.

Mot Clés : Segmentation d'images, Graphes, Lignes de partage des eaux, Segmentation de tumeurs, Programmation Linéaire.

Acknowledgements

We would like to thank Région Île-de-France and the Cancéropôle Île-de-France for funding our research on medical image segmentation.

We also would like to thank the "Institut Gustave-Roussy" and especially professor François Bidault for providing medical images and useful advice for developing our segmentation software.

Contents

Introduction

Motivations

Medical image segmentation remains a challenging problem of image analysis. The continuous improvements of the imaging technologies as well as the huge amount of medical image data and applications require new algorithmic solutions. Medical image analysis has shown numerous new applications in the last years: computer assisted diagnosis and intervention, surgery planning and simulation, radiotherapy planning, etc. Most of these applications require the images to be segmented, i.e. the zones of interest, the pathologies, have to be extracted from the whole image. In these scenarios, the image segmentation task is only the first step that allows later quantitative measurements, diagnosis, treatment planning, and modeling of anatomical structures. Our work is motivated by the need to offer fast and robust image segmentation tools for a clinical routine context. Keeping in mind this constraint, we developed a whole software for image visualization, analysis and segmentation. This software has been tested for various medical applications and quantitative studies proved the efficiency of the developed tools. We have especially proposed new interactive image segmentation tools thanks to the funding of the "Région Île-de-France" and the "Cancéropôle Île-de-France". All the algorithmic solutions were developed in cooperation with the "Institut Gustave-Roussy [1]".

General Framework : Graphs

Our image segmentation tools are based on graphs, which is a rather recent and rapidly expanding area of image processing. The focus of this thesis work is both theoretical and practical. On the theoretical point of view, we study properties of minimal spanning trees, shortest paths trees and minimal cuts and we revisit these notions for image segmentation problems. We detail the properties, the characteristics and the limitations of these optimal graph structures for image segmentation purposes. We establish in the first chapter some theoretical links between these structures and give a general framework unifying several graph based image segmentation methods. We describe then briefly the direct use of these optimal graph structures for image segmentation purposes and precise which method should be used in which situations. We address the usual problems met when using graph based segmentation tools and propose solutions in concordance with our theoretical results.

Combining Morphological and Graphs Based Segmentation

The second part of the thesis is dedicated to innovative methods combining graph cuts and morphological low level segmentations. We propose in this chapter some practical improvements of image segmentation based on graph cuts. We detail some mathematical methods commonly used in the image segmentation community, namely minimal surfaces and Markov random fields. We show how these technique can be

[1] The Institut Gustave-Roussy pioneered the multidisciplinary approach to the treatment of cancer. It is an undisputed expert in all the conceptual and technical fields related to cancer.IGR's activity is centred on rapidly transforming progress in basic research into clinical applications for the benefit of patients.

efficiently combined with a morphological segmentation in order to speed up the use of these models. We detail the assumptions and the hypotheses used to approximate solutions of these minimization problems by considering the region graph of a watershed transform. We also show that the use of a region adjacency graph of a low level morphological segmentation has a richer potential than a pixel adjacency graph approach. We describe some new extensions that permit to add shape and statistical constraints to graph cuts methods. These extensions can only be used on region adjacency graphs, which highlight the important properties of this representation.

Applications to Medical Image Segmentation

We detail in chapter 4 the medical applications of our segmentation methods. We consider in this chapter three different applications that are nowadays considered as important challenges of medical image analysis. The first application aims at creating patient specific models of the heart for surgery planning and simulation. This first application summarizes four months of collaboration with the University of Aarhus (Denmark) funded by a European Marie Curie fellowship. The second application was realized in collaboration with the "Institut Gustave-Roussy", and aims at segmenting and tracking lung tumors for radiotherapy planning. This application summarizes a long cooperation with the imaging department of the "Institut Gustave-Roussy", which helped us to develop a software for medical images visualization and segmentation. Finally, the last application details a quantitative study of the quality of our segmentation tools. This application consists in delineating liver tumors. The segmentation results are compared to segmentation results provided by experienced radiologists. We show, by this application, that the proposed tools are well suited for the segmentation of liver tumors and moreover we show that our methods outperform many state of the art methods.

Perspectives

We present in the two last chapters some interesting perspectives of graph based techniques. We first revisit the well known watershed transform from the point of view of path optimization and path algebra. We recall existing properties of the watershed transform and propose some definitions of the watershed transform on graphs. We give a general algorithm that can be used to compute the watershed transform on graphs based on any optimal path property. These last definitions are then extended to deal with vector valued images, such as color images.

Finally we also propose new extensions of the minimal graph cut problem for image segmentation purposes. New types of constraints are included in the classical minimal graph cut problem, which bring this standard problem into the linear programming framework. We show that this framework is adapted to solve complex image segmentation problems, such as contours constrained image segmentation (segmentations that have to contain a set of user specified contours), area and perimeter constrained image segmentation (segmentations that are constrained by geometric properties such as a maximal area and perimeter limits). These extensions, based on the linear programming formulation of the minimal cut problem, offer interesting perspectives and opportunities for future work on graph based image segmentation techniques.

1

Graph Theory

Graph theory appears in modern Mathematics as a branch of Combinatorics and Computer Science. A graph is a fundamental mathematical structure used to model pairwise relations between objects from a set. Leonhard Euler is considered as the father of graph theory since his seminal paper on the Seven Bridges of Königsberg, published in 1736, and illustrated in figure 5.1. Later on, the fusion of ideas coming from Mathematics with those coming from Chemistry is at the origin of a part of the standard terminology of graph theory. First, graph theory is extensively used in Chemistry for molecular modeling; an example is the study of isomers developed by Caley (1875). In particular, the term "graph" was first introduced by Sylvester in a paper published in 1878 in Nature [108].

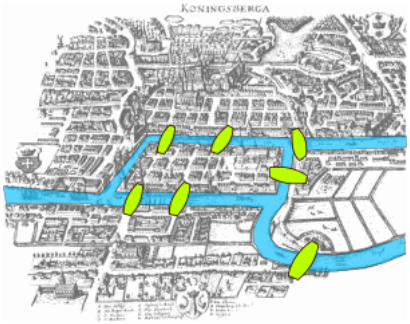


Figure 1.1: The Königsberg bridges problem. There is an island in Königsberg, called "Kneiphoff", with the two branches of the river flowing around it. There are seven bridges crossing the two branches. The question, solved by Euler, was whether a person could plan a walk in such a way that he would cross each of these bridges once but not more than once.

Graph theory is then mainly developed as mathematical games during the second half of the 19th century. Sir William Rowan Hamilton (1805 - 1865), mostly known for the creation of the "quaternions" proposed some ingenious mathematical games related to modern graph theory such as the famous "Icosian Game". The idea for the game was first exhibited by Hamilton during a meeting of the British Mathematical Association in Dublin [51]. Later on, he sold the game for 25 pounds to "John Jacques and Son", a wholesale dealer in games. The "Icosian Game", illustrated in figure 1.2, is a strategy game involving two players. The main problem of the game is to find a "Hamiltonian circuit", i.e. a loop, along the edges of a icosahedron.

Interest in graph theory has also been guided by real life applications such as the study of electrical networks and circuits. Historically, it was initiated in 1847 with Kirchhoff with an abstract picture providing the whole information about an electrical network geometry (an electric scheme). Kirchhoffs rules

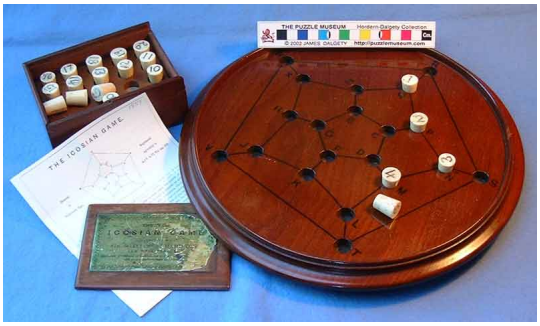


Figure 1.2: An original copy of Sir William Rowan Hamilton's famous "Icosian Game". Hamilton gave his name to a large variety of graph structures.

provide a basis for establishing the numbers of linearly independent currents and voltage equations. In addition, such a graph representation enabled Kirchhoff to formulate, for a network with a finite number of linear resistors, some methods for computing the current in any of the resistors, knowing the voltage in the network. Kirchhoff's work is now considered as the basis of network flow theory and related algebraic methods.

Graph Theory was then constantly improved since the seminal works of mathematicians of the 19th century. Graph Theory was also linked to other branches of Mathematics such as Algebra. For instance algebraic graph theory is a branch of Mathematics in which algebraic methods are applied to graph problems. Especially, it studies the spectrum of some particular matrices representing graphs. This part of algebraic graph theory is called spectral graph theory in the literature and has shown to be of great interest for unsupervised image segmentation.

Graph theory has boomed since 1930 with the work of König, Hall, Erdös, Tutte, Edmonds, Berge, and many other researchers. Hundreds of papers about graphs have been published since 1930, covering a large spectrum of real life applications as well as theoretical considerations. Graph is now clearly linked to Algebra, Topology, and other topics from Combinatorics. Moreover graph theory provides leading approaches to solve Computer Science, Operations Research, Game Theory and Decision Theory problems. Two centuries after Euler's seminal work, the Hungarian mathematician Denés König published in 1936 the first unified graph theory approach in his book "Theorie der endlichen und unendlichen Graphen" (Theory of finite and infinite graphs). Later on, a new interest in graph theory took place in the 1950's and culminated with the books of Ford and Fulkerson (1962 [36]) as well as the work of the French mathematician Claude Berge [7]. A major contribution was proposed through the generalization of the concept of graph by Claude Berge, where a graph may model high order relations between objects (not only pairwise relations as indicated earlier).

A new interest in graph theory has again appeared in the last years with the development of the Internet and other communication networks. New problems and applications have emerged from these technologies and graph theory has again proved to be well suited to solve these new problems. The structure of many real life applications can thus be described in the terminology of graph theory, such as:

- the representation of networks: computer networks (Internet), telecommunication network, social networks, electrical networks, etc.,
- the representation of evolutionary systems and the description of transitions in genetics, demography, etc..
- the representation of binary (or high orders) relations in algebra and combinatorics.

Graph structure is fundamental throughout this thesis to describe images and to treat segmentation problems. The first chapter of this thesis is an introduction to graph theory and presents the state of the art methods for graph based image segmentation. This chapter reviews the standard definitions and notations for graphs and details some basic theorems as well as fundamental algorithms. We first give our notations in section 1.1. Sections 1.2, 1.3 and 1.4 define the fundamental objects that are used throughout the thesis, namely graphs, paths, trees, forests and cut sets. Finally some important properties of these structures as well as classical algorithms of graph theory are detailed. We give some new results concerning the multi-terminal cut problem, also named multiway cut in the literature. We detail a post-optimality lemma concerning this problem which is known to be NP-hard. Finally, in section 1.5, we highlight new theoretical links between some optimal graph structures. We give in this last section some explicit methods that establish an equivalence between classical optimal structures used for image segmentation.

1.1 Notations and Definitions

This first section defines the notion of graph and some important related structures.

Definition 1.1.1 (Graph). *A graph G is a pair $G = (V, E)$, where V and E are both finite sets. The elements $v \in V$ are called vertices and the elements $e \in E \subset \{\{i, j\}, i, j \in V, i \neq j\}$ are called edges. An edge $\{i, j\}$ of E is denoted $e_{i,j}$.*

Definition 1.1.2 (Adjacency). *We say that an edge $e_{i,j}$ connects i and j. In this case vertices i and j are adjacent and i is a neighbor of j (and vice versa).*

Definition 1.1.3 (Subgraph). *Given a graph $G = (V, E)$, the graph $G' = (V', E')$ is called a subgraph of G if and only if $V' \subset V$ and $E' \subset E$.*

Definition 1.1.4 (Edge Weighted Graph). *An edge weighted graph G is a triplet $G = (V, E, W)$, where (V, E) is a graph and W is a mapping of the set of the edges E into $\mathbb{R}^+$. For each edge $e_{i,j}$ of G we write $w_{i,j} = W(e_{i,j})$; $w_{i,j}$ is called the weight of the edge $e_{i,j}$.*

Definition 1.1.5 (Weight Map). *The mapping W is called the weight map of $G = (V, E, W)$. The mapping W^n will denote the mapping of the set of edges E such that the weight of the edge $e_{i,j}$ is equal to $w^n_{i,j}$.*

We will sometimes distinguish undirected graphs from directed graphs. In the last terminology, edges $e_{i,j}$ and $e_{j,i}$ are considered as distinct edges. However in the following we are mainly going to consider undirected graphs. Our only use of directed graphs is dedicated to the study of graph flows in section 1.3.

Note also that vertices are sometimes called nodes or points. Edges can also be called arcs (especially for directed graphs) and in this context graphs are called networks. We often use "graphical" representations of a graph, as illustrated in figure 1.3.

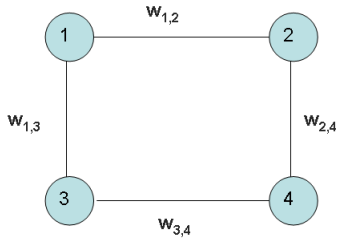


Figure 1.3: Graphical representation of a graph. Vertices are represented by disks and arcs by lines.

The notion of adjacency in a graph enables us to derive some geometrical properties of graphs. Imagine that a graph represents a road network. Adjacency relations model road connections between cities. It is thus possible to derive the notions of travels, walks and paths on a graph.

Definition 1.1.6 (Path). *A path π of the graph $G = (V, E)$ is a sequence of edges written $\pi = (e_{i,j}, e_{j,l}, ...e_{k-2,k-1}, e_{k-1,k})$. π is also called a path from i to k. The set of all paths from i to k is denoted $\Pi_{(i,k)}$.*

Definition 1.1.7 (Generalized Path Length). *The n-generalized length of a path π is defined by:*

$$n \in \mathbb{N}^*, L_n(\pi) = \sqrt[n]{\sum_{e_{i,j} \in \pi} w^n_{i,j}} \tag{1.1.1}$$

Moreover:

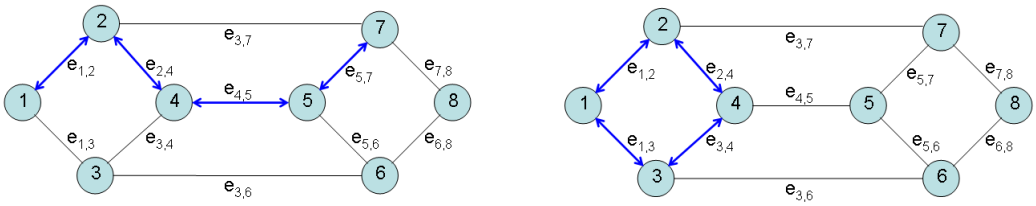
$$\begin{cases} L_\infty(\pi) = \max\limits_{e_{i,j}\in\pi}(w_{i,j}) = \lim\limits_{n\to\infty}\left(\sqrt[n]{\sum\limits_{e_{i,j}\in\pi} w_{i,j}^n}\right) \\ L_0(\pi) = \sum_{e_{i,j}\in\pi} 1 \end{cases} \tag{1.1.2}$$

Note that for a path π, $L_1(\pi)$ is simply the sum of edges weights along the path (also called the "length" of the path). $L_\infty(\pi)$ is the maximum edge weight (also called "height" of the path), and $L_0(\pi)$ is by convention the number of edges of the path.

Definition 1.1.8 (Cycle)**.** *A cycle or a circuit is a path $\pi = (e_{i,j}, ..., e_{l,k})$ such that $i = k$.*

Definition 1.1.9 (Connected Component)**.** *A connected component $G' = (V', E')$ of the graph $G = (V, E)$ is a subgraph of G such that there is a path connecting each pair of distinct nodes of G'. A graph G is called connected if G is itself a connected component.*

Definition 1.1.10 (Tree, Forest)**.** *A tree T is a connected graph without cycles. A graph F that has no cycles and that is not connected is called a forest.*



(a) (b)

Figure 1.4: Path and Cycle. (a) A path highlighted in blue from node 1 to 7. (b) A cycle.

Paths and cycles are illustrated in figure 1.4. These notions are very close to the intuitive ideas that one can have about these structures. The path is a fundamental notion that we will meet in many graph problems, such as the Königsberg's bridges problem. In the following chapters, we only consider weighted and connected graphs. This restriction is not important in our case since graphs encountered in imaging applications represent images and are always connected.

1.2 Shortest Paths

A classical combinatorial problem of graph theory is the search of a shortest path between two specified nodes of a graph. Obviously this problem has many practical applications. An example is finding the shortest way from one town to another on a road map. In this case, the nodes represent towns, and the edges represent portions of roads weighted by their lengths, or their traveling times, their costs, etc. This problem can also appear as a subproblem when dealing with more difficult combinatorial problems that we will detail later. Applications of the shortest path problem are numerous: general traveling problems, path planning, roads networks optimization, maze problems, etc.

Definition 1.2.1 (Shortest Path)**.** *Given an edge weighted connected graph $G = (V, E, W)$, the shortest path problem between two nodes v_f and v_b consists in finding a path π^* such that the length of π^* is minimal:*

$$\pi^* \in \underset{\pi\in\Pi_{(v_f,v_b)}}{\operatorname{argmin}} \left(L_1(\pi)\right) . \tag{1.2.1}$$

We recall that $\Pi_{(v_f,v_b)}$ is the set of all paths between v_f and v_b in G.

The problem is called the shortest path problem because one can consider edges weights as geometrical lengths between nodes. In fact, this problem is not easy to solve if we allow arbitrary edges weights. The problem becomes easier if we restrict ourselves to nonnegative weights. Dijkstra's algorithm [28] allows to compute all shortest paths between a source node and all other nodes of a graph weighted by positive real numbers, This problem is known as the single source shortest paths problem.

Definition 1.2.2 (Shortest Path Tree)**.** *Given a source node v_f, a shortest path tree $SP = (V, E^*)$ of $G = (V, E, W)$ is a tree such that $E^* \subset E$ and for each node i the unique path π^* in SP from v_f to i is a shortest path in G:*

$$\pi^* \in \underset{\pi \in \Pi_{(v_f,i)}}{\operatorname{argmin}} \left(L_1(\pi)\right) . \tag{1.2.2}$$

Dijkstra's algorithm wears the name of its discoverer, computer scientist Edsger Dijkstra (1930 - 2002). For a given source node in the graph, the algorithm finds the paths with smallest lengths between the source node and every other node. Moreover, the output of the algorithm has the special property to be a tree graph, where each node is connected to the source through a shortest path, i.e. a shortest path tree.

The algorithm visits first the neighbors of the source node and sorts them depending of their distances to the source node. The algorithm inserts the visited nodes in an ordered queue depending on their distances. Neighbors of the queued nodes are then analyzed, sorted and inserted in the queue. The result of the algorithm is stored as a predecessor map. This map indicates for each node which of its neighbors is the closest to the source. Another output of the algorithm is the distance map which indicates for each node the length of the shortest path to the source node. Note also that a node can be inserted many times in the queue. Predecessors and distances to the source can thus be updated during the process if a shorter path is found. A proof of correctness of the algorithm can be found in [43]. The complexity of the described pseudo-code is $O(|E|log(|V|))$. It is obtained by assuming that the graph is stored as an adjacency list, i.e. for each vertex a list of its neighbors is stored. We also assume that a binary heap is used to sort the edges in the queue. The algorithm can be slightly modified to find a shortest path from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.

Algorithm 1 Dijkstra's algorithm

Input: Graph G, Root Vertex v_f
Output: Predecessor Map $PredMap$, Distance Map $Distmap$

```
for all vertex v in G do
    Distmap[v] := ∞
    PredMap[v] := v
end for
Distmap[v_f] = 0
INSERT(Queue,v_f,Distmap[v_f])
while Queue is not empty do
    u := EXTRACT-MIN(Queue)
    for all vertex v adjacent to u do
        if (w_{u,v} + Distmap[u]) ≤ Distmap[v] then
            Distmap[v] := (w_{u,v} + Distmap[u])
            PredMap[v] := u
            INSERT(Queue, v, Distmap[v])
        end if
    end for
end while
```

Dijkstra Algorithm has also been generalized to deal with a larger class of optimal path problems. This algorithm is also used as a subroutine for many other graph problems such as network flow problems.

1.3 Network Flows and Graph Cuts

Network flow is another common problem arising in the field of connected and weighted graph theory. Like the shortest path problem, edge weights can be interpreted as a physical property. The weights or the capacity of each arc is here interpreted as the maximum amount of some commodity that can "flow" through this arc. A graph is called in this context a network. A network can be used to model currents in an electrical circuit, traffic in a road system, fluids in pipes, or anything similar in which a certain fluid travels through a network of nodes. Such a flow is always directed and we make thus a difference between the flow from node i to j and the flow from node j to i. This assumption forces us to consider in this section weighted directed graphs, i.e. make a difference between edges $e_{i,j}$ and $e_{j,i}$.

1.3.1 Definitions

We now introduce the problem of finding a flow from a source node v_f to a sink node v_b. We call any real function $x : E \to \mathbb{R}^+$ of the arcs, a feasible flow between v_f and v_b in the directed graph G if and only if x verifies the following conditions:

$$\begin{cases} 0 \leq x(e_{i,j}) \leq w_{i,j}, \quad \forall e_{i,j} \in E \quad \text{(capacity constraints)} \\ \sum_j x(e_{j,i}) - \sum_j x(e_{i,j}) = 0, \quad \forall i \in V \setminus \{v_f, v_b\}, \quad \text{(flow conservation law)} \end{cases} \tag{1.3.1}$$

The above conditions imply that:

$$\sum_j x(e_{v_f,j}) - \sum_j x(e_{j,v_f}) = \sum_j x(e_{j,v_b}) - \sum_j x(e_{v_b,j}) = f, \tag{1.3.2}$$

where f is the value of the flow.

We will now focus on the problem of finding a maximal feasible flow from a node v_f, called the source, to a node v_b, called the sink. This is a combinatorial optimization problem in which the objective is to maximize f under the constraints given by equation (1.3.1). Note that both the constraints and the objective function are linear with the variables $x(i, j)$. We first define the maximal flow problem before defining its dual problem called the minimal cut problem. This type of linear problem can be often viewed from two points of views, using the theory of duality.

Definition 1.3.1 (Maximal Flow)**.** *Given an edge weighted connected graph $G = (V, E, W)$, the maximal flow problem between the nodes v_f and v_b consists in finding a maximal feasible flow x^*:*

$$x^* \in \underset{x \in Flow}{\operatorname{argmax}} \left(\sum_j x(e_{v_f,j}) - \sum_j x(e_{j,v_f}) = \sum_j x(e_{j,v_b}) - \sum_j x(e_{v_b,j}) \right) , \tag{1.3.3}$$

where $Flow$ is the set of all feasible flows in G. Note that the maximal value of a feasible flow is not necessarily attached to a unique flow.

We can now define the dual problem called the minimal graph cut.

Definition 1.3.2 (Graph Cut)**.** *Given an edge weighted graph $G = (V, E, W)$ and a set of distinct nodes $T = \{t_1, .., t_k\} \in V$ (called terminals), a cut C is a set of edges such that the removal of C from E separates the terminals in G (i.e. there is no path between each distinct pair of nodes (t_i, t_j) after the removal of edges of C).*

Note that a cut defined in this way is called a "$multi - terminal$" cut in the literature [25], in order to stress the fact that it separates more than two terminals.

Definition 1.3.3 (Generalized Cut Weight)**.** *The generalized weight of a cut C is defined by:*

$$\forall n \in \mathbb{N}^*, L_n(C) = \sqrt[n]{\sum_{e_{i,j} \in C} w_{i,j}^n} \tag{1.3.4}$$

Moreover:

$$\begin{cases} L_\infty(C) = \max\limits_{e_{i,j} \in C}(w_{i,j}) = \lim\limits_{n \to \infty} \left(\sqrt[n]{\sum\limits_{e_{i,j} \in C} w_{i,j}^n} \right) \\ L_0(C) = \sum_{e_{i,j} \in C} 1 \end{cases} \tag{1.3.5}$$

Note that $L_1(C)$ is simply the sum of edges weights along the cut (also called the "weight" of the cut). $L_\infty(C)$ is the maximum edge weight (also called "height" of the cut), and by convention $L_0(C)$ is the number of edges of the cut.

Definition 1.3.4 (Minimal Cut)**.** *Let $G = (V, E, W)$ be an edge weighted graph and $T = \{t_1, .., t_k\}$ be a set of distinct nodes. The minimal cut problem consists in finding a set of edges C^* of E separating each terminal $t_i \in T$, such that the weight of C^* is minimal.*

$$C^* \in \underset{C \in C_{\{t_1,..,t_k\}}}{\operatorname{argmin}} \; (L_1(C)) \; . \tag{1.3.6}$$

where $C_{\{t_1,..,t_k\}}$ is the set of all possible cuts separating each terminal t_i.

Note that the problem of finding a minimal cut is NP-hard (unsolvable in polynomial time) for cuts separating more than two terminals. NP-hardness of the problem of multi-terminal cuts is addressed by Dahlaus in [25]. The multi-terminal cut problem is also our point of interest in section 1.3.3. On the contrary, the problem of finding a minimal cut between two nodes s and t, often called minimal s-t cut problem, is solvable in polynomial time.

A famous theorem due to Ford and Fulkerson [36, 27] establishes an equivalence between the maximal flow between two nodes and the minimal cut value separating these two nodes.

Theorem 1.3.5 (Max-Flow Min-Cut Theorem)**.** *For any network the maximal flow value from v_f to v_b is equal to the minimal capacity of all cuts separating v_f and v_b.*

The initial maximization problem of computing a maximal flow is equivalent to the minimization problem of computing a minimal cut. This theorem is a classical illustration of duality in combinatorial optimization.

1.3.2 (s-t) Cuts and The Ford-Fulkerson Algorithm

The Ford-Fulkerson algorithm computes the maximum flow in a network. It was published in 1956 in [27]. The name "Ford-Fulkerson" is also used for the Edmonds-Karp algorithm, which is a specialization of the Ford-Fulkerson algorithm. The idea behind the algorithm is very simple: as long as there is a path from the source node to the sink node, with available capacity on all edges in the path, we send flow along one of these paths. A path with available capacity is called an augmenting path. The Edmonds-Karp algorithm is identical to the Ford-Fulkerson algorithm, except that the search order for finding an augmenting path is defined as a heuristic [30]. For instance, the augmenting path can be found by first analyzing the set of shortest paths which have some available capacity. An important restriction on these algorithms, is that they are limited to graph with rational edge capacities, $w_{i,j} \in \mathbb{Q}^+$. Different algorithms can also be found in the literature based on other heuristics [42, 29, 15].

By adding some flow along an augmenting path to the flow already established in the graph, the maximum flow will be reached when no more augmenting paths can be found in the graph. However, this situation is not guaranteed to be reached. In this case the algorithm runs forever and the flow might

Algorithm 2 Ford Fulkerson Algorithm

Input: Graph G, Source Vertex s, Sink Vertex t
Output: Flow Map $fMap$

```
for all edge e_{i,j} in G do
    fMap(e_{i,j}) := 0
end for
while There is a path π such that (∀e_{i,j} ∈ π, w_{i,j} − fMap(e_{i,j}) > 0) do
    Find e_{m,n} ∈ argmin_{(e_{i,j}∈π)} (w_{i,j} − fMap(e_{i,j}))
    for all edge e_{i,j} ∈ π do
        fMap(e_{i,j}) := fMap(e_{i,j}) + w_{m,n} − fMap(e_{m,n})
        PredMap[v] := u
    end for
end while
```

not even converge to the maximum flow. This situation can only occur with irrational flow values. When the capacities are integers, the runtime of the Ford-Fulkerson algorithm is bounded by $O(|E| * f)$, where $|E|$ is the number of edges in the graph and f is the value of the maximum flow in the graph. This is because each augmenting path can be found in $O(|E|)$ time and increases the flow by an integer amount which is at least 1. One can also notice that the algorithm is based on a path finding problem and thus the Dijkstra algorithm can be used as a subroutine to find augmenting paths. This heuristic was initially proposed by Dinic [29] and later on rediscovered by Edmonds and Karp [30]. More recently, Kolmogorov and Boykov have also presented a similar algorithm where shortest paths are stored in a suitable tree structure [15]. This last algorithm has been experimentally shown to be very efficient for imaging applications.

1.3.3 Multi-terminal Cuts

We now focus on the generalization of the minimal cut problem between two nodes. In the minimal multi-terminal cut problem we are given a weighted graph and a subset of the vertices called terminals. The problem consists in finding a minimum weight set of edges that separates each terminal from all the others. The multi-terminal cut problem can also be seen as a partitioning problem since the problem is to find a partition $P = \{P_1, ..., P_k\}$ of the graph nodes where P_i is a connected component containing exactly one terminal t_i. Our approach of the problem is the one proposed by Dahlaus in [25].

Definition 1.3.6 (Isolating Cut)**.** *Let $G = (V, E, W)$ be an edge weighted graph and $\{t_1, ..., t_k\}$ a set of k specified nodes (called terminals). The isolating cut C_i is defined by the minimal cut between the two terminals t_i and $T = \bigcup_{j \neq i} \{t_j\}$.*

Definition 1.3.7 (Minimum Multi-Terminal Cuts [25])**.** *Given an edge weighted connected graph $G = (V, E, W)$ and a set $\{t_1, ..., t_k\}$ of k specified nodes (called terminals), the minimum multi-terminal cut problem consists in finding a cut C separating the nodes of $\{t_1, ..., t_k\}$ such that the weight of the cut $L_1(C)$ is minimal.*

We recall that the minimum multi-terminal cut problem is NP-hard as soon as the number k of terminals is larger than two [25]. In the other case ($k = 2$) the problem can be solved in polynomial time using some well known graph cuts algorithms proposed in [15]. In the multi terminal case, approximate bounded algorithms can be used to produce near optimal results [25, 16].

Multi-Terminal Cut Algorithm

The algorithm proposed by Dahlhaus in [25] uses the so-called isolation heuristic, to compute a near optimal multi-terminal cut. It works with a few iterative binary optimizations of the cut weight. The algorithm computes a set of k isolating cuts and selects a posteriori the $(k-1)$ smallest cuts, as illustrated in figure 1.5. One should note that this algorithm is not the most efficient considering the approximation

bounds of the process. However its simplicity and clarity of the algorithm has permitted us to derive some properties concerning the isolating cuts.

Algorithm 3 Dahlaus Multi-Terminal Cut Algorithm

Input: Graph G, Terminals $\{t_i\}$
Output: Multi-Terminal Cut C
  for all $i \in \{1, .., k\}$ **do**
    Compute a minimum isolating cut C_i between the terminal t_i and the super-terminal $T = \bigcup_{j \neq i} t_j$
  end for
  $C = \bigcup_{i \neq m} C_i$, *where* $L_1(C_m) \geq L_1(C_i)$, $\forall i \in \{1, .., k\}$

Theorem 1.3.8 (Approximation Bounds [25]). *The Isolation Heuristic constructs a multi-terminal cut whose weight is guaranteed no to exceed more than* $2(k-1)/k$ *times the optimal weight cut.*

Some algorithms can give a better upper bound. The best bounded algorithm, based on a linear programming relaxation, is presented in [58] with bound ≈ 1.3438. We did not consider this last method since the isolation heuristic gives satisfying results for the aimed applications, i.e. image segmentation. On the other hand, Boykov et al. [16] have proposed two minimization algorithms, α-expansion and $\alpha-\beta$ swap move, inspired by the simulated annealing process. Their algorithms are particularly adapted to the computation of multi-terminal cuts depending on a large number of terminals. Our opinion is that their approach does not improve the results of the isolation heuristic for image segmentation purposes. Moreover, on the theoretical side, their algorithm does not provide better approximation bounds.

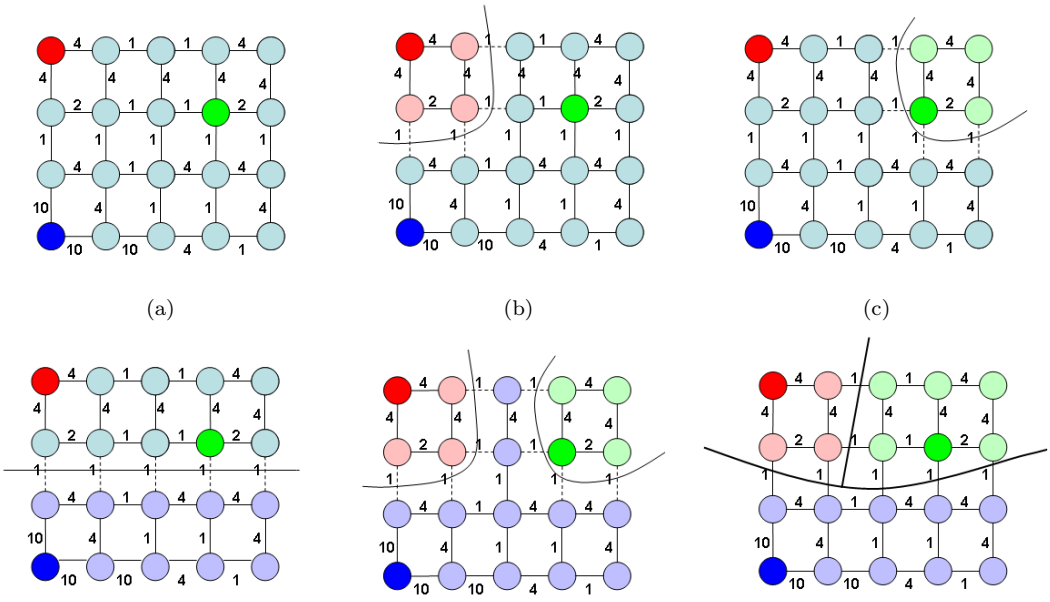


Figure 1.5: Multi-terminal graph cut via an isolation heuristic. (a) A graph with three terminals represented by colored nodes. (b) The minimum cut between t_1 and $(t_2 \cup t_3)$. (c) The minimum cut between t_2 and $(t_1 \cup t_3)$. (d) The minimum cut between t_3 and $(t_1 \cup t_2)$. (e) Final labeling; the largest cut has been removed. (f) The true optimal multi-terminal cut.

Post Optimality

We present a new post optimality result that we have derived from the isolation heuristic. From our experiments, we noticed that the following property can appear when we apply the isolation heuristic algorithm:

$$\forall j \in \{1,..,k\},\ \ C_j \subset \bigcup_{i \in \{1,...,k\} \setminus j} C_i \ . \tag{1.3.7}$$

The property in equation 1.3.7 means that any isolating cut C_j can be ignored without affecting the resulting partition of the graph. It is thus not always necessary to remove the largest isolating cut. We use this property to propose the following post-optimality lemma:

Lemma 1.3.9 (Optimality condition)**.** *Let A be an optimal multi-terminal cut, let C_i be k isolating cuts associated with k distinct terminals, and let C be the union of the $k-1$ smallest isolating cuts, then:*

$$\left(\forall j \in \{1,..,k\},\ \ C_j \subset \bigcup_{i \in \{1,...,k\} \setminus j} C_i \right) \Rightarrow (L_1(C) = L_1(A)) \ . \tag{1.3.8}$$

Proof: Let A be an optimal multi-terminal cut for G. By removing the set of edges A from G, we produce k connected components $\{V_1, V_2, ..., V_k\}$ where V_i contains t_i as its only terminal. Let A_i be the set of edges in G that link V_i and $V - V_i$, so that A_i is an isolating cut for t_i. Note that every edge $e \in A$ must link two components V_i and V_j , so that $e \in A_i$ and $e \in A_j$. It follows that:

$$\sum_{i=1}^{k} L_1(A_i) = 2L_1(A) \ . \tag{1.3.9}$$

Now since A_i is an isolating cut the terminals t_i and since we found the minimum-weight isolating cut for each terminal, we necessarily have:

$$\sum_{i=1}^{k} L_1(C_i) \leq \sum_{i=1}^{k} L_1(A_i) \ . \tag{1.3.10}$$

Moreover the property 1.3.7 implies that:

$$\sum_{i=1}^{k} L_1(C_i) = 2L_1(\cup_{i=1}^{k} C_i) = 2L_1(C) \ . \tag{1.3.11}$$

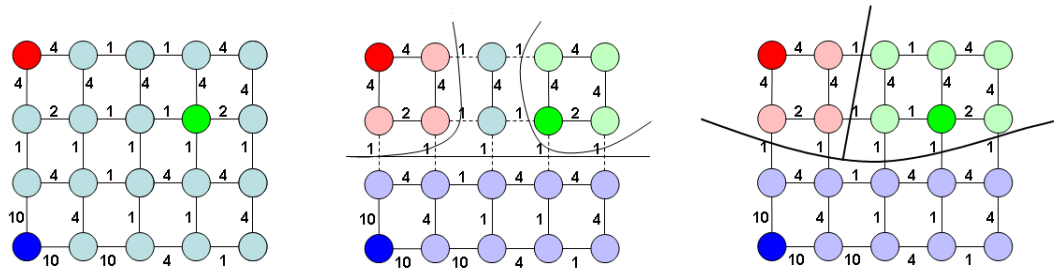
Therefore, we have:

$$2L_1(C) = \sum_{i=1}^{k} L_1(C_i) \leq \sum_{i=1}^{k} L_1(A_i) = 2L_1(A) \ . \tag{1.3.12}$$

Since A is an optimal multi-terminal cut, we necessarily have: $L_1(C) = L_1(A)$, therefore C is also a minimal multi-terminal cut, which concludes our proof.

Lemma 1.3.9 is a sufficient but not necessary condition to check the optimality of a multi-terminal cut computed with the isolation heuristic. This lemma gives us a possibility to characterize optimal multi-terminal cuts a posteriori. By assigning a label to each connected component of an isolating cut, it is possible to check the optimality of the multi-terminal cut by simply checking that all nodes of the graph are labeled. This process is illustrated in figure 1.6.

We will use the labeling procedure in our examples to highlight that the computed multi-terminal cuts are optimal. We will also use this procedure for image segmentation.



(a) (b) (c)

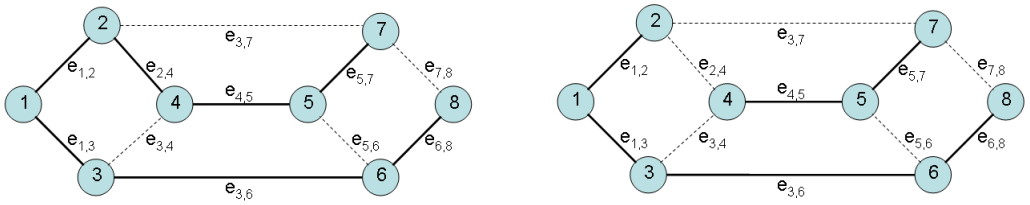
Figure 1.6: Multi-terminal graph cut via an isolation heuristic. (a) A graph with three terminals represented by colored nodes. (b) The optimality of the multi-terminal cut is checked by labeling the nodes of each isolating cuts. In this case, some nodes remain unlabeled after the computation of the isolating cuts and thus the multi-terminal cut obtained by the Isolation heuristic is not guaranteed to be optimal and in fact the cut is not optimal, see figure 1.5. (c) The true optimal multi terminal cut.

1.4 Trees and Forests

Forests and trees are some fundamental structures used in many various scientific fields. We have already mentioned this structure earlier in connection with the shortest paths problem. Nature commonly exhibits this structure and many situations can be represented by tree graphs: family hierarchies, river structures, etc.. Their usefulness in Combinatorics comes from important properties that are enumerated in this section. Section 1.4.1 presents the basic notions related to trees and forests. Section 1.4.2 refers to classical optimization problems related with trees, namely, minimal spanning trees and minimal spanning forests. Finally section 1.4.3 details the algorithm used to compute a minimal spanning tree of a graph.

1.4.1 Definition

Definition 1.4.1 (Tree, Forest)**.** *A tree T is a connected graph without cycles. A graph F that has no cycles and that is not connected is called a forest.*



(a) (b)

Figure 1.7: Tree and forest. (a) A tree: the graph has no cycles and is connected. (b) A forest: the graph has no cycles and it is not connected (there is no path from the node 2 to 4).

The extensive use of trees in Combinatorics comes from their particular structure. For instance this structure is also the output of the Dijkstra algorithm described in section 1.2.

Theorem 1.4.2 (Tree properties [62])**.** *Let $T = (V, E)$ be a graph of n vertices ($|V| = n$), then the following statements are equivalent:*
(a) T is a spanning tree.
(b) T has $(n-1)$ edges and no cycles.

(c) T has $(n-1)$ edges and is connected.
(d) T contains a unique path between any pair of nodes.

Definition 1.4.3 (Spanning trees and forests)**.** *Given a connected graph $G = (V, E)$, a spanning tree $T = (V, E_T)$ of G is a connected graph without cycles such that $E_T \subset E$. A spanning forest $T = (V, E_F)$ of G is a non-connected graph without cycles such that $E_F \subset E$.*

1.4.2 Minimal Spanning Tree

The minimal spanning tree is an optimal structure that we will often meet throughout this thesis. We especially highlight the usefulness of this structure for image segmentation. We define in this section the minimal spanning problem and give some important properties of this structure.

Definition 1.4.4 (Minimal Spanning Tree)**.** *Given an edge weighted connected graph $G = (V, E, W)$, the minimal spanning tree problem consists in finding a spanning tree $T^* = (V, E_{T^*})$ of G such that the sum of the edges weights of T^* is minimal.*

$$T^* \in \underset{T \in ST}{\operatorname{argmin}} \left(\sum_{e_{i,j} \in E_T} w_{i,j} \right) . \tag{1.4.1}$$

Where ST is the set of all spanning trees of G.

Note that a minimal spanning tree of a graph is in general not unique. A rather restricting condition of uniqueness is that each edge of the graph has a distinct weight. The number of minimal spanning tree of a given graph may thus be large. Note also that the set of minimal spanning trees of the graph $G = (V, E, W)$ is also the set of minimal spanning trees of $G' = (V, E, f(W))$, where $f : \mathbb{R} \to \mathbb{R}$ is a strictly increasing function, because the ordering of edges weights is preserved. An algorithm for computing a minimal spanning tree is given in section 1.4.3.

A slightly different problem from the minimal spanning tree is the minimal spanning forest problem, where each distinct tree of the forest is constrained to contain a given set of nodes called the roots.

Definition 1.4.5 (Rooted Minimal Spanning Forest)**.** *Given an edge weighted connected graph $G = (V, E, W)$, the minimal spanning forest rooted on a set of k distinct nodes $\{t_1, ..., t_k\}$ consists in finding a spanning forest $F^* = (V, E_{F^*})$ of G such that each distinct tree of F^* contains exactly one root t_i, and such that the sum of the edges weights of F^* is minimal.*

$$F^* \in \underset{F \in SF}{\operatorname{argmin}} \left(\sum_{e_{i,j} \in E_F} w_{i,j} \right) . \tag{1.4.2}$$

Where SF denotes the set of all spanning forests of G rooted on the set $\{t_1, ..., t_k\}$.

Construction of Optimal Spanning Forests

A minimal spanning forest rooted on a set of specified distinct nodes $\{t_1, ..., t_k\}$, called roots or terminals, can be obtained from a minimal spanning tree by several methods.

First, a minimal spanning tree $T^* = (V, E_{T^*})$ of the graph $G = (V, E, W)$ is computed, then for each pair of distinct terminals (t_i, t_j), an edge of maximum weight lying of the unique path from (t_i, t_j) in T^* is deleted, if such a path exists. After the deletion of this edge, a minimal spanning forest is obtained; moreover each tree of the forest contains exactly one terminal t_i. Similarly a maximal spanning forest can be computed from a maximal spanning tree by deleting the edge of minimum weight along the unique path between each pair of terminals. This procedure is illustrated in figure 1.8 for the determination of

a maximal spanning forest.

Another method to obtain a minimal spanning forest consists in adding an extra node A to the graph $G = (V, E, W)$. This new node A is then exclusively connected to the terminals $\{t_1, ..., t_k\}$. These new edges are assigned a null weight, in order that they belong to be in all minimal spanning trees of the new constructed graph. Finally the minimal spanning tree of the constructed graph induces a minimal spanning forest in G such that each tree of the forest contains exactly one terminal t_i. This procedure is illustrated in figure 1.9 for the computation of a maximal spanning forest.

Note also that the last procedure can be used to compute shortest path forests. The shortest path tree of the constructed graph (by adding an extra node and some null weighted arcs connected to the terminals) induces a shortest path forest in the original graph.

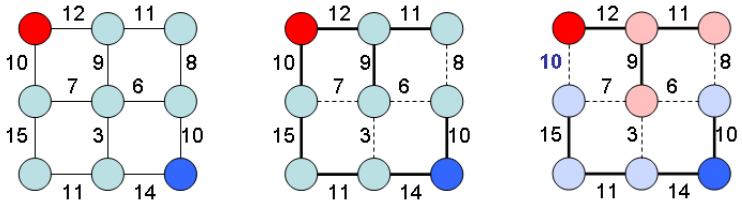


Figure 1.8: Construction of maximal spanning forests. (a) The original graph G with two terminals represented by colored nodes (red and dark blue). (b) A maximal spanning tree of the graph G. (c) The maximal spanning forest (rooted on the red node and the dark blue node) induced by deleting the edge of minimum weight along the path between the terminals.

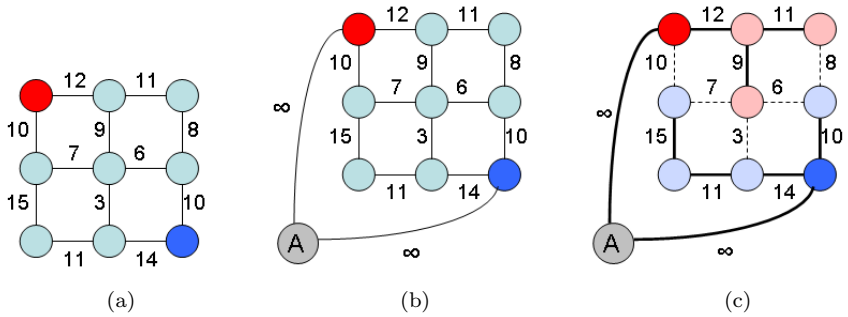


Figure 1.9: Construction of maximal spanning forests. (a) The original graph G with two terminals represented by colored nodes (red and dark blue). (b) An extra node A is added to the graph G and exclusively connected to the terminals with infinite weight. (c) The maximal spanning tree of the modified graph induces a maximal spanning forest of G rooted on the terminals.

Characterization of Optimal Spanning Trees and Forests

We present now some useful characterizations of optimal spanning trees and forests. The following theorems are important and allow us to derive new results highlighting some links between optimal structures appearing in graph optimization problems. Optimal spanning trees and forests present some interesting properties concerning the paths and the cuts induced by these structures. These theorems highlight some optimal properties of the paths included in a spanning tree and the cut sets induced by optimal spanning forests. The problems linked to spanning trees are among the oldest problem studied in graph theory due to the numerous tree structures appearing in real life problems. We refer the readers

to the following textbook of Gondran and Minoux [43] for proofs and detailed explanations of the results presented here.

Theorem 1.4.6 (Path Optimality [53]). *A spanning tree $T^* = (V, E_{T^*})$ of $G = (V, E, W)$ is a minimal spanning tree if and only if it satisfies the following path optimality condition:*

$$\forall e_{k,l} \notin E_{T^*}, \forall e_{i,j} \in \pi, \quad w_{k,l} \geq w_{i,j} \ , \tag{1.4.3}$$

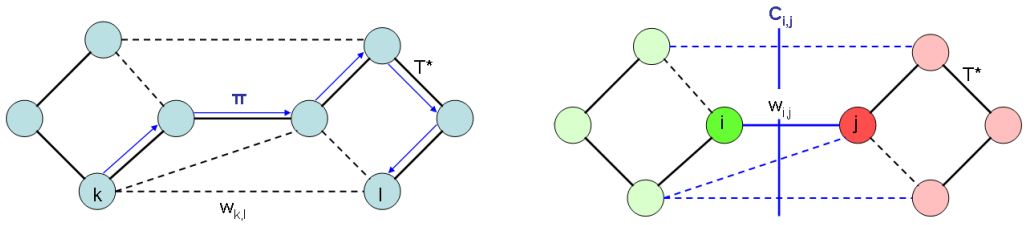
where π is the unique path from node k to node l in the tree T^*.

Since there is a unique path between two nodes of a spanning tree, deleting one edge $e_{i,j}$ of a spanning tree creates two distinct connected components. The set of edges of G that connects these two connected components in G is thus a graph cut separating nodes i and j. We say that $C_{(i,j)}$ is a cut induced in the graph G by deleting the edge $e_{i,j}$ in a spanning tree T^* of G. This notion of induced cut is illustrated in figure 1.10.

Theorem 1.4.7 (Cut Optimality [43]). *A spanning tree $T^* = (V, E_{T^*})$ of $G = (V, E, W)$ is a minimal spanning tree if and only if it satisfies the following cut optimality condition:*

$$\forall e_{i,j} \in E_{T^*}, \forall e_{k,l} \in C_{(i,j)}, \quad w_{i,j} \leq w_{k,l} \ , \tag{1.4.4}$$

where $C_{(i,j)}$ is the cut induced in the graph G by deleting the edge $e_{i,j}$ in T^*.



(a) (b)

Figure 1.10: Path Optimality and cut optimality of minimum spanning trees, theorem 1.4.6 and 1.4.7. (a) The unique path in the spanning tree from node k to node l is the set of blue edges. (b) A cut $C_{i,j}$ induced by deleting the arc $e_{i,j}$ in the spanning tree. The cut is the set of blue edges.

The path optimality of minimal spanning trees allows us to compute the so-called *minimax* path between two nodes of the graph, i.e. the path that minimizes the "height" of the path between the two nodes, i.e. that minimizes the maximum edge weight along the path.

Theorem 1.4.8 (Minimax Path [53, 43],). *A spanning tree $T^* = (V, E_{T^*})$ of $G = (V, E, W)$ is a minimal spanning tree if and only if for all distinct pairs of nodes (v_f, v_b) of G, the unique path π^* between v_f and v_b in T^* is a minimax path:*

$$\pi^* \in \underset{\pi \in \Pi_{(v_f, v_b)}}{\operatorname{argmin}} \ (L_\infty(\pi)) \ , \tag{1.4.5}$$

where $\Pi_{(v_f, v_b)}$ is the set of paths from node v_f to node v_b in G.

Alternatively, the cut optimality of minimal spanning trees allows us to compute the so-called *minimax* cut between two nodes of the graph, i.e. the set of edges that minimizes the maximum weight along the cut. Let us first introduce the notion of spanning forest cut.

Definition 1.4.9 (Spanning Forest Cut). *Let $G = (V, E, W)$ be an edge weighted graph. Let $F = (V, E_F)$ be a spanning forest of G. The set of edges of G that links the trees of F is called a spanning forest cut.*

Note that in general a spanning forest cut is a multi-terminal cut [25].

Theorem 1.4.10 (Minimax Cut [43]). *Let $F^* = (V, E_{F^*})$ be a maximal spanning forest of $G = (V, E, W)$ rooted on a set of nodes $\{t_1, ..., t_k\}$, then the spanning forest cut C^F separating the terminals $\{t_1, ..., t_k\}$ is a minimax cut:*

$$C^F \in \underset{C \in C_{\{t_1,..,t_k\}}}{\operatorname{argmin}} \left(L_\infty(C^F) \right) , \tag{1.4.6}$$

where $C_{\{t_1,..,t_k\}}$ is the set of all multi-terminal cuts separating the nodes $\{t_i\}$

Note also that the previous results can be derived in the case of maximal or minimal spanning trees. We will now describe an algorithm due to computer scientist Prim that computes an optimal spanning tree of a given graph.

1.4.3 Prim's Algorithm

A first version of this minimal spanning tree algorithm was first discovered in 1930 by the mathematician Vojtech Jarnik. Later on, this algorithm was independently rediscovered by the computer scientists Robert C. Prim in 1957 and Edsger Dijkstra in 1959. It is sometimes called the DJP algorithm, the Jarnik algorithm, or the Prim-Jarnik algorithm.

The algorithm is an iterative scheme that increases an initial arbitrary set composed of one node. At each step of the algorithm, an edge connected to the visited nodes is chosen such that its weight is minimal compared to all other edges connected to the visited nodes. This edge is then added to the tree, as well as the destination node. This process is repeated until each node of the graph has been visited. The edges choice ensures that the built tree is a minimal spanning tree. Similarly, a maximal spanning tree can be built by choosing edges of maximal weight at any step of the process.

Algorithm 4 Prim's Algorithm

Input: Connected Graph $G = [V, E, W]$
Output: Tree $T^* = [V^*, E^*, W^*]$
  $V^* = \{k\}$, where $\{k\}$ is an arbitrary node from V.
  while $V^* \neq V$ **do**
    Choose edge $e_{i,j} \in E$ with minimal weight such that $i \in V^*$ and $j \notin V^*$
    $V^* = V^* \cup \{j\}$
    $E^* = E^* \cup \{e_{i,j}\}$
  end while

A proof of correctness of the algorithm can be found in [43]. The complexity of the previous pseudocode is $O(|E|log(|V|))$. This complexity is meaningful if the graph is stored as an adjacency list. We also assume that a binary heap is used to sort the edges that can potentially be included in the spanning tree.

1.5 Links Between Optimal Structures

This section presents new results establishing links between the optimal graph structures that we have introduced so far. We give in section 1.5.1 a new proof of a result presented by Allène et al. [1], establishing a link between maximal spanning forest cuts and minimal cuts. Moreover we generalize their results to the case of multi-terminal cuts. Then, in section 1.5.3 we give a new result concerning minimal spanning trees and shortest path trees. The theoretical results presented in this section give an explicit edges weight transformation such that minimal cuts can give the same graph partitions as shortest path forests and maximal spanning forests.

1.5.1 Maximal Spanning Forests and Minimal Cuts

Let us first recall the result presented in [1] and give some different elements of proof of their results. Allène et al. have shown that minimal cuts and minimal spanning forest cuts coincide for a particular

weight map transformation. We extend their result to the case of multi-terminal cuts. This link can be highlighted using the general framework presented in [47].

Let $\{t_1, ..., t_k\}$ be a set of k distinct nodes, called terminals, of a graph $G = (V, E, W)$. A k-labeling X of the nodes V, where $k \in \mathbb{N}^*$, is a function from V into $\{1, ..., k\}$:

$$\begin{aligned} X \quad : \quad & V \to \{1, ..., k\} \\ & i \to X(i) = x_i \end{aligned}$$

We want to find a k-labeling X_n such that:

$$X_n^* \in \underset{X \in \mathcal{X}}{\operatorname{argmin}} \left(\sqrt[n]{\sum_{e_{i,j} \in E} (w_{i,j}.\delta(x_i \neq x_j))^n} \right) . \tag{1.5.1}$$

$$subject\ \ to \begin{cases} x_{t_1} = 1 \\ x_{t_2} = 2 \\ ... \\ x_{t_k} = k \end{cases} \tag{1.5.2}$$

Where $\mathcal{X}$ is the set of all possible k-labeling of the nodes, and δ is the indicator function : $\delta(x_i \neq x_j) = 1$ if and only if $x_i \neq x_j$ and $\delta(x_i \neq x_j) = 0$ otherwise.

This problem is equivalent to the search for a multi-terminal cut since the objective function does only take into account edges weights connecting nodes assigned with different labels. The last problem is thus equivalent to the search of an optimal multi-terminal cut C^* such that:

$$C^* = \underset{C \in C_{\{t_1,..,t_k\}}}{\operatorname{argmin}} \ (L_n(C)) \ . \tag{1.5.3}$$

Where $C_{\{t_1,..,t_k\}}$ is the set of all possible cuts separating each terminal t_i from all other terminals. We are thus interested in the minimal cut of a graph $G = (V, E, W)$ under the weight map W^n. The case $n = 1$ clearly corresponds to the classical minimal cut problem:

$$X_1^* \in \underset{X \in \mathcal{X}}{\operatorname{argmin}} \left(\sum_{e_{i,j} \in E} (w_{i,j} \delta(x_i \neq x_j)) \right) . \tag{1.5.4}$$

$$subject\ \ to \begin{cases} x_{t_1} = 1 \\ x_{t_2} = 2 \\ ... \\ x_{t_k} = k \end{cases} \tag{1.5.5}$$

The case $n = \infty$ has already been studied in [47] with $X_n^* : V \to [0, 1]$. Grady et al. have studied the problem of assigning a real value between 0 and 1 to each node of the graph. We consider here the problem with a labeling $X_n^* : V \to \{0, k\}$. This last constraint gives slightly different results than those presented in [47]; moreover our problem turns out to be NP-hard by equivalence to the multi-terminal cut. Given that:

$$\lim_{n \to \infty} \left(\sqrt[n]{\sum_{e_{i,j} \in E} (w_{i,j} \delta(x_i \neq x_j))^n} \right) = \max_{e_{i,j} \in E} (w_{i,j} \delta(x_i \neq x_j)) \tag{1.5.6}$$

the case $n = \infty$ corresponds thus to the minimax cut problem :

$$X_\infty^* \in \underset{X \in \mathcal{X}}{\operatorname{argmin}} \left(\max_{e_{i,j} \in E} (w_{i,j} \delta(x_i \neq x_j)) \right) . \tag{1.5.7}$$

$$subject\ \ to \begin{cases} x_{t_1} = 1 \\ x_{t_2} = 2 \\ ... \\ x_{t_k} = k \end{cases} \tag{1.5.8}$$

which is equivalent to the following formulation:

$$C^* \in \underset{C \in C_{\{t_1,..,t_k\}}}{\operatorname{argmin}} \ (L_\infty(C)) \ . \tag{1.5.9}$$

The minimax cut problem has already been studied. This problem is related to the maximal spanning forest problem, as expressed by theorem 1.4.10. The previous relation establishes that increasing the edges weights under a certain power n implies that the resulting minimal cut produces the same partition as a maximal spanning forest. Note that the inverse is in general not true. In other words, a maximal spanning forest cut is not guaranteed to be a minimal cut under the weight map W^n, unless the maximal spanning forest is unique. This is a generalization of the results presented in [1].

We detail now a convergence property of multi-terminal cuts that we use to build the proof of the results presented above. Let us first prove the following intermediate lemma:

Lemma 1.5.1 (Sequence Ordering)**.** *Let $\{a_1, a_2, ..., a_n\}$ and $\{b_1, b_2, ..., b_n\}$, be two finite non increasing sequences of n real positive numbers. Assume, without loss of generality that:*

$$\begin{cases} a_1 \geq a_2 \geq ... \geq a_n \\ b_1 \geq b_2 \geq ... \geq b_n \\ a_1 > b_1 \end{cases} \tag{1.5.10}$$

Then, the following property holds:

$$\exists m_0 \in \mathbb{R}, \forall m > m_0, \sum_{1 \leq i \leq n} (a_i)^m > \sum_{1 \leq i \leq n} (b_i)^m \tag{1.5.11}$$

Proof:
If $b_1 = 0$, the property is obvious. We suppose now that $b_1 > 0$:

$$\sum_{1 \leq i \leq n} (b_i)^m \leq \sum_{1 \leq i \leq n} (b_1)^m = n.(b_1)^m = (\sqrt[m]{n}.b_1)^m \tag{1.5.12}$$

$$\sum_{1 \leq i \leq n} (a_i)^m \geq a_1^m \tag{1.5.13}$$

It suffices thus to prove that:

$$a_1^m > (\sqrt[m]{n}.b_1)^m \Leftrightarrow \frac{a_1}{b_1} > \sqrt[m]{n} \tag{1.5.14}$$

Since $a_1 > b_1 \Rightarrow \left(\frac{a_1}{b_1} > 1\right)$ and

$$\lim_{m \to \infty} \sqrt[m]{n} = 1 \tag{1.5.15}$$

We can conclude our proof:

$$\exists m_0 \in \mathbb{R}, \forall m > m_0, \left(\frac{a_1}{b_1} > \sqrt[m]{n}\right) \Leftrightarrow \sum_{1 \leq i \leq n} (a_i)^m > \sum_{1 \leq i \leq n} (b_i)^m \tag{1.5.16}$$

Moreover if $\left(\frac{a_1}{b_1} = c\right)$:

$$\left(c > \sqrt[m_0]{n}\right) \Leftrightarrow m_0 > \frac{log(c)}{log(n)} \tag{1.5.17}$$

Theorem 1.5.2 (Convergence of Multi-terminal cuts)**.** *Given an edge weighted connected graph $G = (V, E, W)$, a set $T = \{t_1, ..., t_k\}$ of k specified nodes (called terminals), and a positive real weight $w_{i,j}$ for each $e_{i,j} \in E$, then there exists a real number m_0 such that for any $m \geq m_0$ a minimal multi-terminal cut C^* of $G^{m_0} = (V, E, W^{m_0})$ is also a a minimal multi-terminal cut of $G^m = (V, E, W^m)$.*

Proof: Without loss of generality, consider that there exist exactly p distinct multi-terminal cuts C_i relative to $\{t_1, ..., t_k\}$ in the graph G. It is also possible to order the edges weights of each multi-terminal cut such that the set of multi-terminal cuts of G can be written as:

$$\begin{cases} C_1 = \{a_1, a_2, ..., a_n\}, & with \quad w_{a_1} \geq w_{a_2} \geq ... \geq w_{a_n} \\ C_2 = \{b_1, b_2, ..., b_n\}, & with \quad w_{b_1} \geq w_{b_2} \geq ... \geq w_{b_n} \\ ... \\ C_p = \{e_1, e_2, ..., e_n\}, & with \quad w_{e_1} \geq w_{e_2} \geq ... \geq w_{e_n} \end{cases} \tag{1.5.18}$$

Note that each cut is written as a set of n edges. In the case where one of the cut C_i contains less than n edges, we simply add to the cut C_i some edges of null weights. Assume also that the cuts are ordered according to the lexicographical ordering of the edges weights:

$$\begin{cases} \exists i \in [1, ..., n], \forall j \in [1, ..., i], (w_{a_j} = w_{b_j}) \ \ and \ \ w_{a_i} > w_{b_i} \\ \exists i \in [1, ..., n], \forall j \in [1, ..., i], (w_{b_j} = w_{c_j}) \ \ and \ \ w_{b_i} > w_{c_i} \\ ... \end{cases} \tag{1.5.19}$$

From Lemma 1.5.1, we know that:

$$\begin{cases} \exists m_1 \in \mathbb{R}, \forall m > m_1, L_m(C_1) > L_m(C_2) \\ \exists m_2 \in \mathbb{R}, \forall m > m_2, L_m(C_2) > L_m(C_3) \\ ... \\ \exists m_{p-1} \in \mathbb{R}, \forall m > m_{p-1}, L_m(C_{p-1}) > L_m(C_p) \end{cases} \tag{1.5.20}$$

We can thus write that:

$$\exists m_0 \in \mathbb{R}, \forall m > m_0, \forall i \in [1, p-1], L_m(C_p) < L_m(C_i) \tag{1.5.21}$$

Moreover $\left(m_0 = \underset{i \in \{1,...,p-1\}}{max} (m_i) \right)$, which concludes the proof of theorem 1.5.2 since C_p is the minimal multi-terminal cut relative to the generalized length $L_m(.)$.

From equation 1.5.9 and theorem 1.5.2, we can finally give the main result establishing the link between multi-terminal cuts and maximal spanning forest cuts:

Theorem 1.5.3 (Multi-terminal Cuts)**.** *Given an edge weighted connected graph $G = (V, E, W)$, a set $T = \{t_1, ..., t_k\}$ of k specified nodes (called terminals), and a positive real weight $w_{i,j}$ for each $e_{i,j} \in E$, then there exists a real number m_0 such that for any $m \geq m_0$ a minimal multi-terminal cut C of $G^m = (V, E, W^m)$ is also a maximal spanning forest cut of $G = (V, E, W)$.*

There are two important points the theorem 1.5.3. First it extends a known result [1] to the case of multiple terminals and second, it shows that the minimum multi-terminal cut problem is solvable for a particular edges weights transformation since it is equivalent to the search of a maximal spanning forest cut. Theorem 1.5.3 is illustrated on an example in figure 1.11.

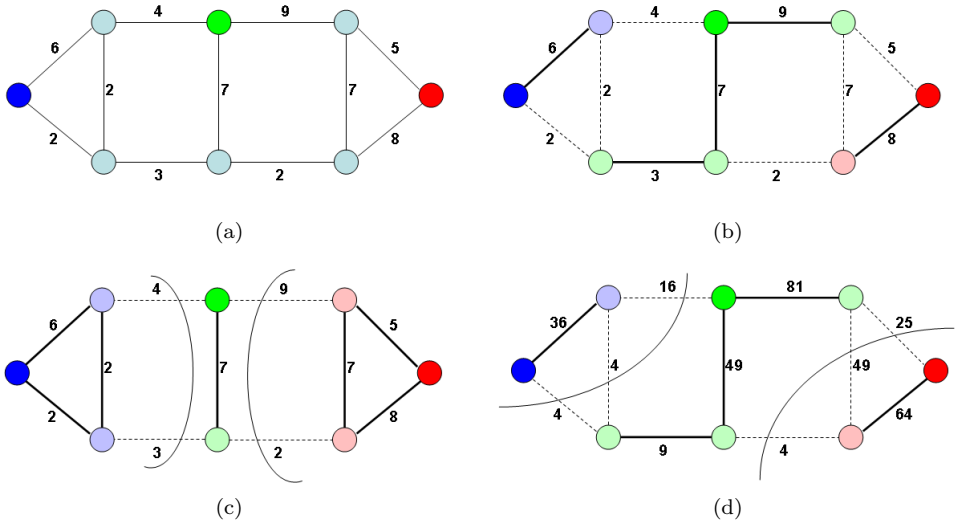


Figure 1.11: (a) A graph $G = (V, E, W)$ and three terminals represented by colored nodes. (b) A maximal spanning forest rooted on the terminals. (c) A minimal cut for the graph G. (d) A minimal cut for the graph G^2. The minimal cut for the graph G^2 is also a maximal spanning forest cut for the graph G.

1.5.2 A Note on Optimal Multi-Terminal Cuts

We have shown in the previous sections that an optimal multi-terminal cut for the graph $G^m = (V, E, W^m)$ is also a maximal spanning forest cut for the graph $G = (V, E, W)$. It is also well known that the multi-terminal cut problem is NP-hard in general. We want to precise here this link and give some possible techniques to obtain optimal multi-terminal cuts.

Let us study the example illustrated on figure 1.12. Despite the fact that an optimal multi-terminal cut converges to a maximal spanning forest cut, the isolation heuristic does not always permit to obtain this optimal multi-terminal cut as illustrated in figure 1.12. Since the isolating cuts are only composed of edges with a unique weight equal to 1, rising the weights to a certain power does not affect the result of the isolating heuristic. Therefore, the isolation heuristic does not always guarantee to produce an optimal multi-terminal cut. However the optimal multi-terminal cut corresponds to a possible maximal spanning forest cut as illustrated on figure 1.13.

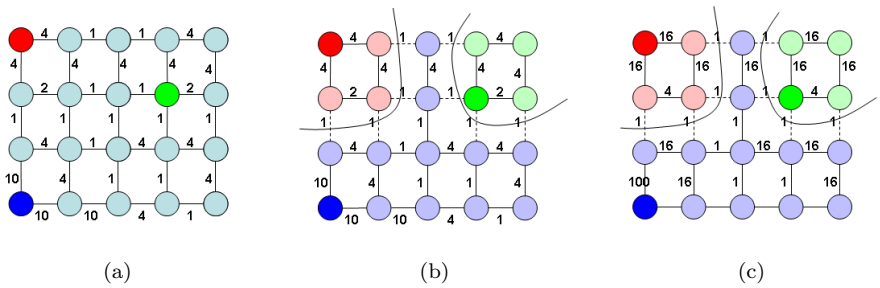
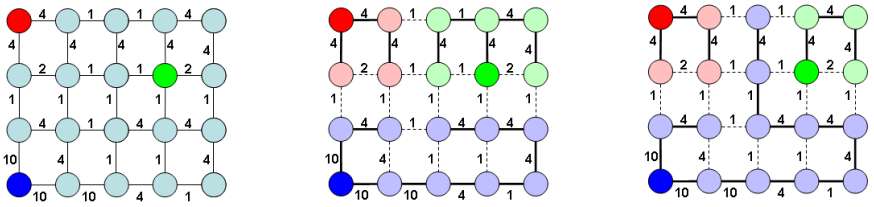


Figure 1.12: Optimal multi-terminal cuts. (a) A graph $G = (V, E, W)$ and three terminals represented by colored nodes. (b) A multi-terminal cut for the graph $G = (V, E, W)$ obtained with the isolation heuristic. (c) A multi-terminal cut for the graph $G^2 = (V, E, W^2)$ obtained with the isolation heuristic. The multi-terminal cut does not change when the power of the weight map increases because all the isolating cuts are composed of edges having a unique weight equal to 1.



(a) (b) (c)

Figure 1.13: Optimal multi-terminal cuts. (a) A graph $G = (V, E, W)$ and three terminals represented by colored nodes. (b) A maximal spanning forest rooted on the terminals. (c) Another possible maximal spanning forest rooted on the terminal. The forest on figure (b) also corresponds to an optimal multi-terminal cut, whereas the forest on figure (c) corresponds to the multi-terminal cut obtained with the isolation heuristic.

1.5.3 Minimal Spanning Trees and Shortest Path Trees

We present now a new result that connects minimal spanning trees with shortest path trees. Some related problems have already been studied, especially the link between shortest path trees and minimal spanning trees when the length of a path is taken as the maximum edge weight value along the path [1, 2] (the $L_\infty()$ length). However, these results were already established earlier and given by theorem 1.4.8. We now consider the link between these two structures for the generalized length of a path ($L_n()$ length). Let us consider the following path minimization problem. Given two nodes v_f and v_b of a graph $G = (V, E, W)$, we want to find a path π_n^* such that:

$$\pi_n^* \in \underset{\pi \in \Pi_{(v_f, v_b)}}{\operatorname{argmin}} \; L_n(\pi) \; . \tag{1.5.22}$$

We are thus interested in the shortest path from the source node v_f to v_b under the weight map W^n.

The case $n = 1$ corresponds to the classical shortest path problem introduced in section 1.2:

$$\pi_1^* \in \underset{\pi \in \Pi_{(v_f, v_b)}}{\operatorname{argmin}} \; L_1(\pi) \; . \tag{1.5.23}$$

The case $n = \infty$ corresponds to the following situation:

$$\pi_\infty^* \in \underset{\pi \in \Pi_{(v_f, v_b)}}{\operatorname{argmin}} \left(\max_{e_{i,j} \in \pi} (w_{i,j}) \right) = \underset{\pi \in \Pi_{(v_f, v_b)}}{\operatorname{argmin}} \; (L_\infty(\pi)) \; . \tag{1.5.24}$$

This last path property is in particular verified by the unique path between two nodes in a minimal spanning tree as expressed by theorem 1.4.8. On the other side, the set of shortest paths from one source node v_f can also be structured in a tree. Relation 1.5.24 highlights thus a link between shortest path trees and minimal spanning trees. In other words it establishes that elevating the edge weights to a certain power n provides a shortest path tree which is also a minimal spanning tree. Note also that the reverse is in general not true. A minimal spanning tree is not guaranteed to be a shortest path tree under the weight map W^n unless the minimal spanning tree is unique.

Theorem 1.5.4 (Minimal Spanning Tree and Shortest Path Tree)**.** *Let $G = (V, E, W)$ be an edge weighted graph with positive real weights; there exists a real number m such that for any $n \geq m$ a shortest path tree relative to $G^n = (V, E, W^n)$ is also a minimal spanning tree relative to $G = (V, E, W)$.*

We can also reformulate this theorem in terms of cut sets:

Theorem 1.5.5 (Minimal Spanning Forest Cut and Shortest Path Forest Cut)**.** *Let $G = (V, E, W)$ be an edge weighted graph with positive real weights. Let $\{t_1, .., t_k\}$ be a set of distinct nodes; then there exists a real number m such that for any $n \geq m$ a shortest path forest cut relative to $G^n = (V, E, W^n)$ separating the terminals $\{t_1, .., t_k\}$ is also a minimal spanning forest cut separating the terminals $\{t_1, .., t_k\}$ relative to $G = (V, E, W)$.*

Note that proofs of these theorems can easily be built following the proofs of theorems 1.5.3. The proof has to be built on the ordering of edges weights along a path of the graph instead of the ordering of edges weights along a cut.

The previous theorems are illustrated by an example in figure 1.14. The relation that we have established in theorems 1.5.4 and 1.5.5 illustrates the links between shortest path trees and minimal spanning trees from two different points of view.

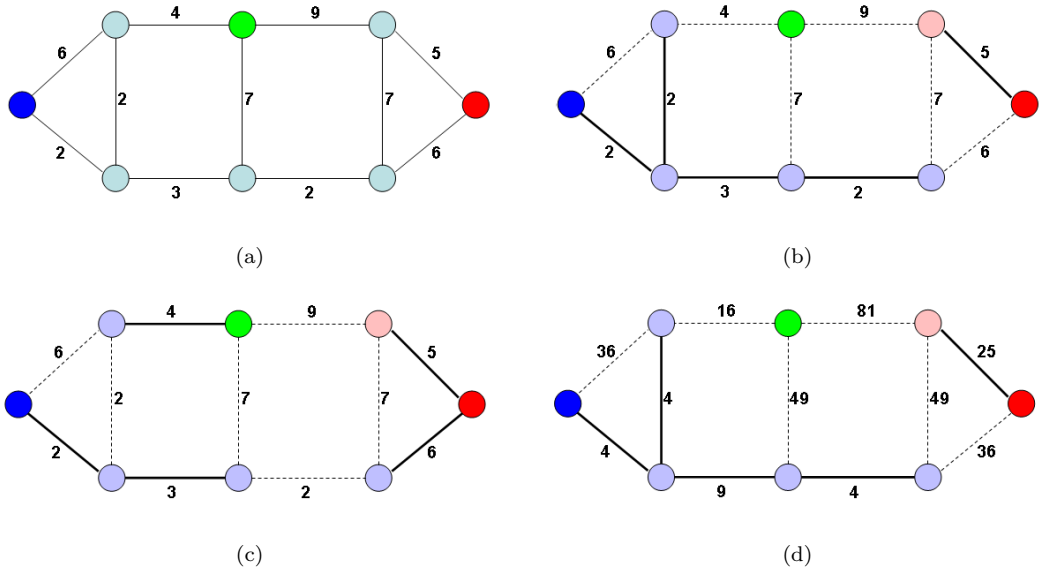


Figure 1.14: Minimal spanning trees and shortest path trees. (a) A graph $G = (V, E, W)$ and three terminals represented by colored nodes. (b) A minimal spanning forest rooted on the terminals. (c) A shortest path forest rooted on the terminals for the graph $G = (V, E, W)$. (d) A shortest path forest rooted on the terminals for the graph $G^2 = (V, E, W^2)$. The shortest path forest of the graph $G^2 = (V, E, W^2)$ is also a minimal spanning forest of the graph $G = (V, E, W)$.

1.5.4 From Minimal Cuts to Shortest Path Forest Cuts

The results established in the last two sections give concrete links between structures that seem very different at the first sight. The conclusion of these results can be illustrated by figure 1.15 that exhibits the sequence of graph partitions that can be obtained by varying the power of the edge weights of the graph. These properties are all derived from the characterization of maximal and minimal spanning forests. Moreover it will allow us to derive new schemes to provide balanced graph partitions that can vary from minimal cuts to shortest path forest cuts. Note also that these structures may not be unique for a given graph and that several optimal forests can exist, as well as multiple optimal cuts.

On the other hand, Grady et al. have shown a common framework unifying minimal cuts (from two terminals), random walkers and shortest paths forests [47]. Their approach of these problems was used in section 1.5.1 to link multi-terminal cuts and maximal spanning forests cuts. Grady et al. have formulated these problems as linear or quadratic problems (systems of linear or quadratic equations under linear or

quadratic constraints). They derived all these problems from a single and general system of equations, as shown below.

Figure 1.15: The sequence of optimal structures obtained by varying the power of the weight map.

Let $\{t_1, t_2\}$ be two distinct nodes of a graph $G = (V, E, W)$. We want to find a function $X_n^* : V \rightarrow [0, 1]$ such that:

$$X_n^* \in \underset{X \in \mathcal{X}}{\operatorname{argmin}} \left(\sqrt[n]{\sum_{e_{i,j} \in E} (w_{i,j}.|x_i - x_j|)^n} \right) . \tag{1.5.25}$$

$$subject\ \ to \begin{cases} x_{t_1} = 0 \\ x_{t_2} = 1 \\ \forall i \in V, x_i \in [0, 1] \end{cases} \tag{1.5.26}$$

From this general formulation, Grady et al. proved that the case $n = 1$ corresponds to the classical minimal cut problem. In this last case the total unimodular property of minimal graph cut guarantees that the solutions are binary vectors x [43].

The case $n = 2$ corresponds to the random walker problem studied by Grady in [45]. The random walker method simulates a random walk on a graph and partitions the graph according to the probabilities that nodes of the graph are attained by a random walker starting his walk at seed node t_1 or t_2. A node of the graph will be assigned the label 0 if the probability that a random walker attains the node from seed node t_1 is greater than the probability that a random walker attains the node starting from seed node t_2.

Finally the case $n = \infty$ is closely related to the shortest path problem. This last problem can be seen as the discrete counterpart of the minimal Lipschitz extension problem. In this problem, the aim is to find a function $X_n^* : V \rightarrow [0, 1]$ such that the maximum variation of the function is minimal. This problem can especially be solved using the shortest path from node t_1 to node t_2.

We can complete this framework since we also linked maximal spanning forests and minimal cuts. Our approach has simply consisted in constraining the linear problem to have integer solutions. This restriction allowed us to derive new links between several optimal graph cuts structures. It is thus also possible to derive minimal cuts, random walkers, shortest paths and optimal spanning trees from a single problem formulation, as illustrated in table 1.1 in the case of two terminals.

	Graph Cut	Random Walker	Grady [47]	Spanning Forest Cut
Obj. Function	$\sum_{e_{i,j} \in E} w_{i,j} \lvert x_i - x_j \rvert$	$\sum_{e_{i,j} \in E} w_{i,j}^2 \lvert x_i - x_j \rvert^2$	$\max_{e_{i,j} \in E} (w_{i,j} \lvert x_i - x_j \rvert)$	$\max_{e_{i,j} \in E} (w_{i,j} \lvert x_i - x_j \rvert)$
Domain	$x_i \in \{0, 1\}$	$x_i \in [0, 1]$	$x_i \in [0, 1]$	$x_i \in \{0, 1\}$
Optimization	Max Flow	Linear System	Shortest Paths	Spanning Trees

Table 1.1: A unifying framework for minimal cuts, random walkers, shortest paths and optimal spanning forests from two terminals.

1.6 Conclusion

In this chapter, we have introduced basic structures and algorithms that we use for graph based image segmentation. We have also highlighted some links between these structures. We have shown that minimal cuts coincide with maximal spanning forest cuts for a particular weight map transformation. Moreover we have proved these results in the case of multi-terminal cuts, which is known to be NP-hard in general. However we have shown that the problem can be solvable for a particular edge weights transformation. We have also shown that shortest path trees and minimal spanning trees of a graph are equivalent for a particular weight map modification. These links between optimal graph structures is a key point to have a better understanding of graph based image segmentation.

The new links that we have highlighted establish that minimal spanning trees are a central structure for graphs partitioning. The theoretical results presented give an explicit weight map transformation that provides optimal graph structures that vary from minimal cuts to shortest path forest cuts and maximal spanning forest cuts. We will study in the next chapter the direct use of these structures for image segmentation. We will also take advantage of the theoretical results that are presented in the last sections to design robust image segmentation tools.

2

Graph Based Segmentation

We address in this chapter the use of shortest path forests, minimal spanning forests and minimal cuts for interactive image segmentation and its associated tools. Indeed these structures take into account some user specified constraints (terminals for minimal cuts and roots for optimal spanning forests). These marker-based methods can also potentially be used for a large variety of image segmentation problems. These graph structures have already been used for image segmentation applications [71, 56, 49, 33, 47, 14]. Minimal spanning forests have mainly been used for hierarchical segmentation [10, 71, 69] and optimal cycles computation [56]. Shortest path forests have been extensively studied by Falcao et al. [33] through the so-called image foresting transform, generalizing a large class of morphological operators by the use of optimal paths forests. Finally, minimal cuts have been studied in connection with energy minimization problems such as Markov random fields and minimal surfaces [14, 17]. We give in this chapter some alternative solutions related to common problems of these optimal graph structures, namely the "shrinking" problem of minimal cuts and the "overflow" problem of shortest path forests. These problems are presented and possible solutions are proposed according to the theoretical results established in the previous chapter. We highlight the strengthes and the weaknesses of each of these methods and detail in which situation which method should be used. We also show that these methods can provide similar results under particular weight maps.

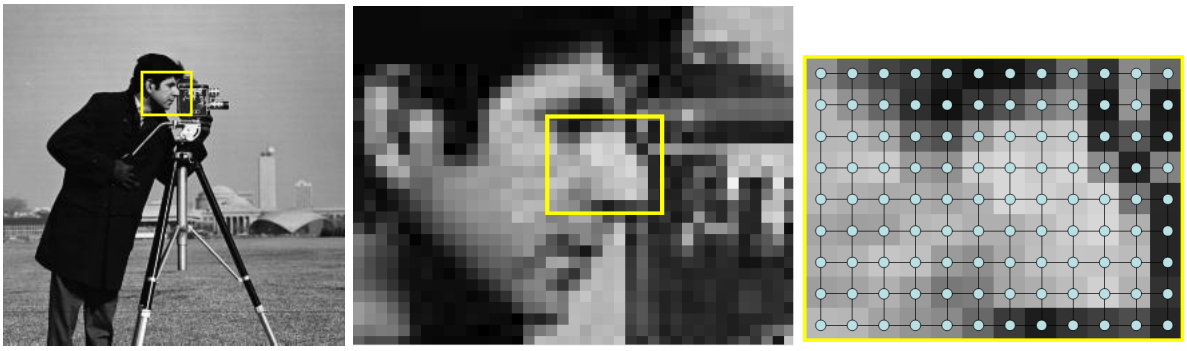
Section 2.1 introduces the graphs encountered in imaging applications. We define in this section the notions of pixel adjacency graphs and region adjacency graphs. These structures are the basic graphs that are used throughout the thesis to describe images. In this chapter we restrict ourselves to the use of the pixel adjacency graphs and the next chapter will be especially dedicated to the use of region adjacency graphs for the optimization and the extensions of graph cuts methods. As one of our major interest is interactive 3D image segmentation, we detail in section 2.2 how user specified constraints for graph based image segmentation can be incorporated, as markers indicating the objects of interest in the image. These markers roughly locate the objects to be segmented in the image and are included as constraints on the optimal graph structures to be computed.

2.1 Graphs Encountered in Image Segmentation

This section presents the graphs commonly encountered in imaging applications. We will consider 2D and 3D grey level images where each pixel takes a value between 0 (black) and 255 (white). A first level is the so-called "pixel adjacency graph" (see figure 2.1 for example). In a second level, an unsupervised low level segmentation of the image (i.e. a segmentation that provides much more regions than objects in the image) is used to build a "region adjacency graph".

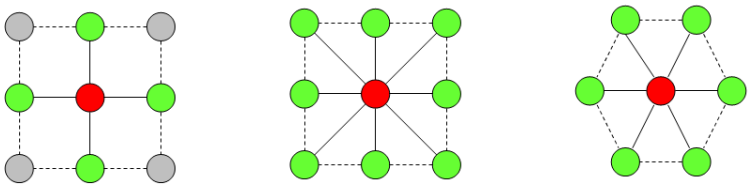
2.1.1 Pixel Adjacency Graphs

In the easiest case, the graph that we consider to represent an image is the "pixel adjacency graph", or informally the pixel graph, as illustrated in figure 2.1. In this case, the set of nodes of the graph is the set of pixels of the image, and the edges link neighboring pixels. We thus simply need an image and an adjacency system to build the corresponding pixel adjacency graph. Different common adjacency systems are illustrated in figure 2.2 and figure 2.3.



(a) (b) (c)

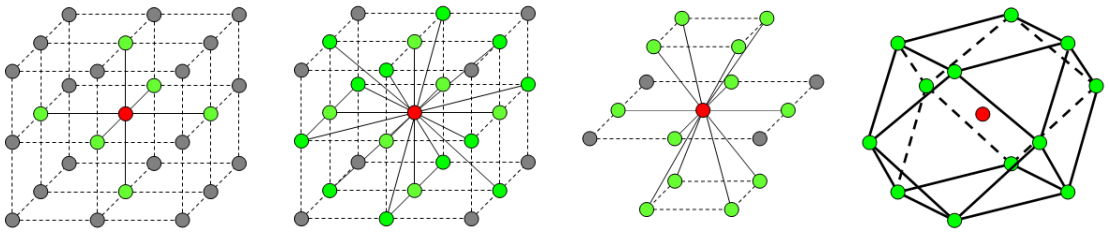
Figure 2.1: Pixel adjacency graph. (a) Original image. (b) Zoom on the outlined area of figure (a). (b) Zoom on the outlined area of figure (b), each pixel of the image is seen as a node of the graph, edges are built according to the 4-neighborhood adjacency system.



(a) (b) (c)

Figure 2.2: Adjacency systems and related pixel adjacency graphs in 2D. The figure illustrates some common adjacency systems used for 2D image processing. Figure (a) illustrates the 4 neighborhood adjacency system (noted V4) as well as the corresponding adjacency graph. Figure (b) illustrates the 8 neighborhood adjacency system (noted V8). Figure (c) illustrates the 6 neighborhood adjacency system on a hexagonal grid.

In our applications on 3D images segmentations, we will mainly consider the V6 or the V12 adjacency systems. For real applications it is always necessary to find a compromise between the algorithmic



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

Figure 2.3: Adjacency systems and related pixel adjacency graphs in 3D. The figure illustrates some common adjacency systems used for 3D image processing. Figure (a) illustrates the 6 neighborhood adjacency system (noted V6) as well as the corresponding adjacency graph. Figure (b) illustrates the 18 neighborhood adjacency system (noted V18). Figure (c) illustrates the 12 neighborhood adjacency system on a cuboctaedric grid (noted V12). Figure (d) illustrates a cuboctaedra which is the polyhedra associated with the 12 neighborhood adjacency system.

complexity of the method and the precision of the method. Most graph algorithms exhibit an algorithmic complexity dependent on the number of edges. The adjacency system that we use has thus an important impact on the speed of the algorithms used on the graph that represents the image. A possible way to bypass this problem is to consider a region adjacency graph instead of a pixel graph in order to reduce both the number of nodes and the number of edges of the considered graphs.

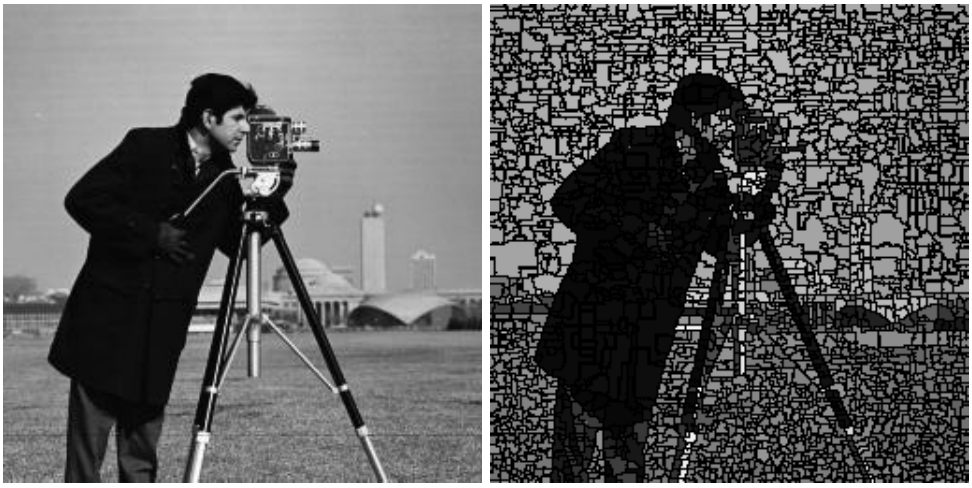
2.1.2 Region Adjacency Graphs

Another graph that we consider in our work is called the "region adjacency graph", or informally the region graph. In this case, the set of nodes represents some regions of the image, i.e. some clusters of pixels, and edges link neighboring regions. A region adjacency graph can be obtained by considering a low-level segmentation of the image. This low-level segmentation can be obtained from a first unsupervised clustering of the image, for instance one can apply the watershed transform [11, 75], λ-flat zones labeling [88, 24], or k-means clustering [70] to obtain such low-level segmentations. The idea behind this technique is to provide a simplification of the image and replace pixels by homogenous regions, which are in practice small and numerous.

An example of an unsupervised watershed segmentation of an image is illustrated in figure 2.4. Note that the important contours of the image are preserved during the segmentation and regions of the partition are mostly composed of homogenous pixels (pixels of similar grey values). In the context of the watershed transform, the images are seen as topographical surfaces. The watershed simulates a flooding of the image's gradient from its local minima to detect crest lines of the gradient, i.e. contours of high contrast in the image. Due to the numerous local minima of the gradient, an over-segmentation of the image is often obtained by this method when it is directly applied on an natural image. However the watershed transform performs a good detection of contours of the original images. We will detail in the next chapter the use of the watershed region adjacency graph in combination with graph cuts.

The quality of this first unsupervised low level segmentation is important to guarantee a minimal loss of information. An ideal situation would be that important information (major contours) of the original image appears on this first segmentation.

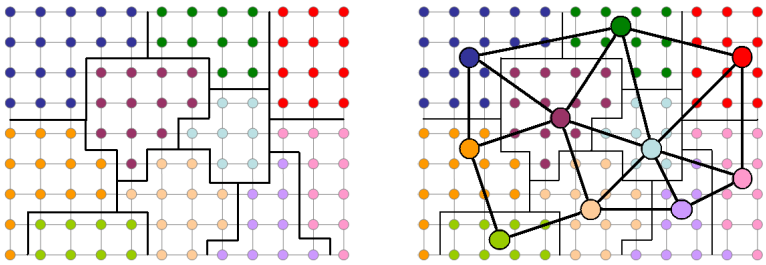
After the first unsupervised segmentation, each region is seen as a node of a graph, and neighbor regions are linked according to a given adjacency system. The constructed graph contains both less nodes and edges than a pixel graph. This property of the region graphs should permit to reduce the computation time of the applied graph algorithms. A region graph is illustrated in figure 2.5. We will detail the use of region graphs in the next chapter and highlight the possibilities and the optimizations



(a) (b)

Figure 2.4: Low-level segmentation. (a) Original image. (b) A low-level segmentation of the image obtained from a watershed transform. Each region of the watershed is assigned the mean grey level of the pixels of the first image.

that such a representation allows.



(a) (b)

Figure 2.5: Region adjacency graph. (a) A low-level segmentation of an image. (b) Region adjacency graph of the previous segmentation. The region adjacency graph provides a compact representation of the image.

2.2 Marker Based Image Segmentation

This section details the use of markers as constraints for graph based image segmentation. The user (or an automatic approach) specifies some zones that have to be included in the regions of interest of the image. These markers are given to indicate the objects and roughly locate the regions to be segmented in the image. This very simple approach is in fact a powerful tool to incorporate prior knowledge in the segmentation. The use of seeds or markers is a classical approach of many image segmentation techniques such as seeded region growing algorithms [3, 4, 34], watershed transform [75, 76] or constrained optimal forests and graph cuts [47, 13].

The classical use of graph based image segmentation consists in working on the pixel graph of an image. We consider the image segmentation from a set of markers and compute an optimal forest rooted on the

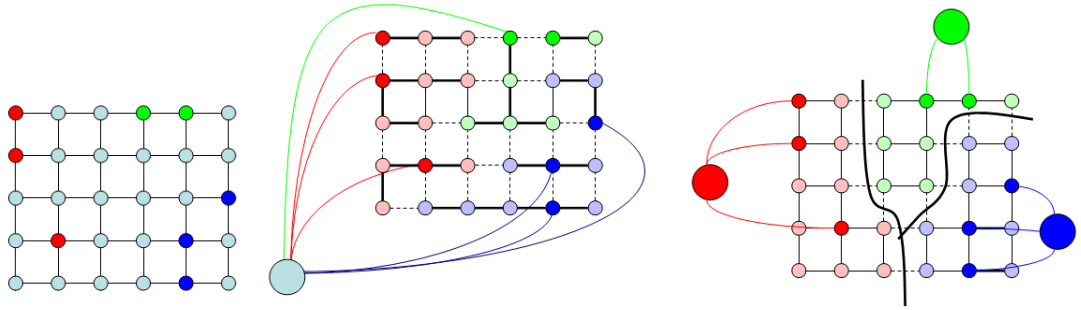
markers or a minimal cut separating these markers. The markers specify thus some hard constraints on the segmentation, each marker is forced to lie in a specific region of the final segmentation.

2.2.1 Constrained Optimal Forests

A way to build rooted forests has already been presented in the previous chapter. However we illustrate in figure 2.6(b) the procedure that is used in this chapter to compute the results presented later on. First an extra node is added to the graph and is only connected to the marked nodes. The extra edges have an infinite weight in the case of a maximal spanning forest or null weight in the case of a minimal spanning forest and shortest path forest. This choice of edges weights ensures that the marked nodes are roots of the forests and that each marker lies in a different region of the segmentation. Finally an optimal spanning tree is computed on this graph, and a spanning forest is obtained with the trees rooted on the markers. This procedure is always used for the computation of optimal forests. Note also that several nodes of the graph can be used as markers.

2.2.2 Constrained Minimal Cuts

For the computation of minimal cuts, the markers specify the terminals to be separated. AS described earlier, several extra nodes are added to the graph and connected to the marked nodes by edges of infinite weights. These edges ensure that the marked nodes are included in distinct region of the final partition. As many extra nodes as markers specified by the user are added to the graph to compute the multi-terminal cut. Finally the minimal cut induces a partition of the original graph that contains as many regions as markers. This procedure is illustrated in figure 2.6(c); it will always be used for the computation of minimal cuts between some sets of markers.



(a) (b) (c)

Figure 2.6: Segmentation with markers. (a) The colored nodes specify some object markers. (b) The spanning tree rooted at the extra node induces a spanning forest rooted at the markers. (c) Three nodes are added to the graph and are only connected to the marked nodes with infinite weights. A multi-terminal cut separating these extra nodes induces a separation of the markers in the original graph.

2.3 Weight map for image segmentation

It is now necessary to define some edge weights that can produce relevant segmentations of the images. We are going to define an edge weight mapping such that the optimal graph structures provide a good detection of boundaries of the objects present in the image, i.e. that separate the objects according to their grey level contrasts [33, 13, 14]. Given a pixel adjacency graph $G = (V, E, W)$, we consider the following edges weights mapping:

$$n \in \mathbb{N}, \forall e_{i,j} \in E, \quad w_{i,j}^n = \left(\frac{|p_i - p_j|}{d(i,j)} + 1 \right)^n . \tag{2.3.1}$$

where i and j are two nodes of the graph, p_i and p_j are the grey level values of neighbor pixels of the image and $d(i, j)$ is the Euclidean distance between the two neighbor pixels.

$w_{i,j}$ plays the role of a dissimilarity measure between neighbors pixels and can be seen as a local estimate of the image's gradient modulus. $w_{i,j}$ is a strictly positive increasing function of $|p_i - p_j|$. A small edge weight means that pixels i and j have similar values, whereas $w_{i,j}$ takes large values when pixels i and j have significant different values. Note that any kind of strictly positive increasing function of $|p_i - p_j|$ could be used to produce similar results. This mapping can also be easily extended to the case of vector valued images by considering a suitable dissimilarity measure. Note also that from the previous chapter results, we know that by varying the weight map power n, shortest path forests and minimal cuts both converge to the minimal spanning forest structure.

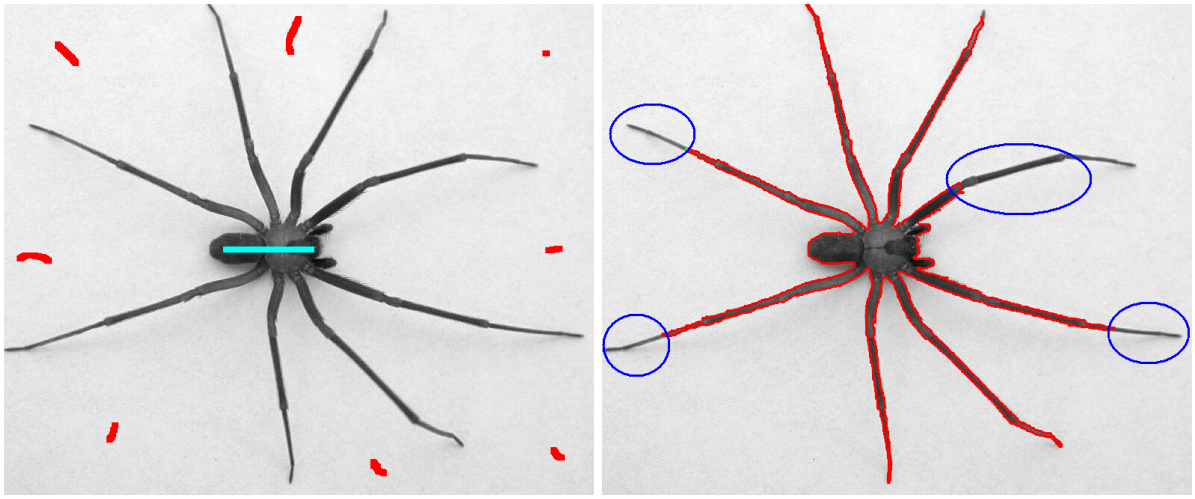
This mapping is used for image segmentation using minimal spanning forests and shortest path forests and $\left(\frac{1}{w_{i,j}}\right)$ is used for segmentation using graph cuts. According to this mapping, a minimal spanning forest and a shortest path forest should contain edges of low weights, which means that pixels of similar grey values are included in the same trees. As a consequence, edges between distinct trees of the forest should have a high weight, i.e. corresponding pixels have different grey levels. On the other side a minimal cut of the graph is composed of low weighted edges, which correspond to pixels whose values are different. As a consequence, the optimal graph structures that we are considering should provide a good detection of object boundaries.

2.4 Application to 2D Images Segmentation

We present in this section some segmentation results obtained on 2D grey level images using optimal forests and minimal cuts. The aim of this section is to illustrate the strength and the weakness of the presented methods through simple examples. In the following, a V4 adjacency system have been used to build the pixel graph of the image.

The first example is the segmentation of an object (a spider) composed of long and thin homogenous structures. Figure 2.7 illustrates the segmentation result obtained by using a minimal spanning forest of the pixel adjacency graph. The minimal spanning forest method produces a roughly good detection of the spider body. The images present an object that is well contrasted. However the tree corresponding to the object markers does not totally cover the spider and some of the spider's legs are not correctly detected, as illustrated on figure 2.7(b).

Figure 2.8 illustrates the segmentation results obtained with a shortest path forest of the pixel adjacency graph by varying the power of the weight map. For small power of the weight map, the shortest path forest does not provide good results and the segmentation method does not detect high contrast regions as illustrated in figure 2.8(a). In this case the shortest path forest is too sensitive to the markers location. We denote this problem of shortest path forests as the "overflow problem". However when the weight map power increases, the shortest path forest provides better segmentation results than the minimum spanning forest, as shown in figure 2.8(c). Finally, as theoretically described in the previous



(a) (b)

Figure 2.7: Minimal spanning forest segmentation. (a) Original image (789x656) surimposed with Object (blue pixels) and background (red pixels) markers. (c) Boundaries of the minimal spanning forest, bad segmentation areas are outlined with blue circles.

chapter, for values high enough of the weight map power, the shortest path forest provides the same result as the minimal spanning forest.

Figure 2.9 illustrates the segmentation results obtained with a minimal cut of the pixel adjacency graph by varying the power of the weight map. For small powers of the weight map, the minimal cut does not provide good results, and the object region is too small, as illustrated in figure 2.9(a). In this case the minimal cut does not take into account sufficiently the edges weights and the minimal cut simply minimizes the perimeter of the object region. We denote this problem of small cuts as the "shrinking problem". However when the weight map power increases, the minimal cut provides a relevant segmentation as shown in figure 2.9(c). Finally, as theoretically described in the previous chapter, for high values of the weight map power, the minimal cut provides similar results as a minimal spanning forest.

The next example illustrates a classical problem of image segmentation, which is the detection of objects with missing boundaries, as illustrated in figure 2.10(a). In this scenario the aim of the segmentation task is to recover the missing boundaries of some rectangles.

In this example, minimal spanning forests and shortest path forests do not provide good segmentation results. These forests structures cannot detect the missing object boundaries. Flat zones of the image (i.e. areas of equal grey level) cause some problems concerning optimal forests. First, concerning minimal spanning forests, flat zones of the image can be covered by numerous different minimal spanning forests. The optimal spanning forest is far from being unique on flat zones of the image. In practice, the forest computed by the Prim's algorithm does not provide a satisfying result, as illustrated on figure 2.10(c).

Concerning shortest path forests, the flat zones problem is less important since flat zones are segmented according to the influence zones of the markers. The pixels of the flat zones are partitioned according to their distances to the markers, as illustrated in figure 2.10(d). Shortest path forests provide thus a better segmentation than minimal spanning forests. Note also that the shortest path forest on a flat zone is also a possible minimal spanning forest of the image. Moreover increasing the weight map power does not affect the segmentation result since the edges of a flat zone have all the same weight.

The weakness of minimal cuts illustrated in the previous example (the problem of small cuts, the

shrinking problem) is in this example an advantage. Minimal cuts provide relevant results for the segmentation of objects with missing boundaries, as illustrated in figure 2.10(e). Firstly, the boundaries of the objects are correctly segmented, secondly flat zones are segmented so that the perimeter of the object is minimized. The minimal cut in a flat zone is composed of a set of edges of minimal cardinality. In other words, in the flat zones the missing object boundaries are composed of boundaries of minimal length. This property of minimal cuts permits to recover the missing boundaries of the rectangles.

2.5 Multi-Labels Segmentation

Finally, the previous techniques can also be used to provide multi-label segmentations. In this case the user must provide specific markers for each object as illustrated in figure 2.11. Then, we consider the segmentation as a minimal spanning tree rooted on the markers, see figure 2.11, as well as a shortest path forest rooted on the markers, see figure 2.12 and finally a minimal multi-terminal cut separating the markers, illustrated in 2.13. The remarks formulated in the previous section are still true in the case of a multi-label segmentation. The optimal graph structures suffer from the same limitations and an appropriate weight map power allows to obtain relevant results.

In the case of multi-terminal cuts, we also show the unlabeled nodes produced by the Dalhaus algorithm to illustrate the non-optimality of the computed multi-terminal cuts, see figure 2.13. This set of unlabeled nodes decreases when the weight map power increases, and finally the set of unlabeled nodes is empty for high values of the weight map power. This property guarantees that the computed multi-terminal cut is optimal.

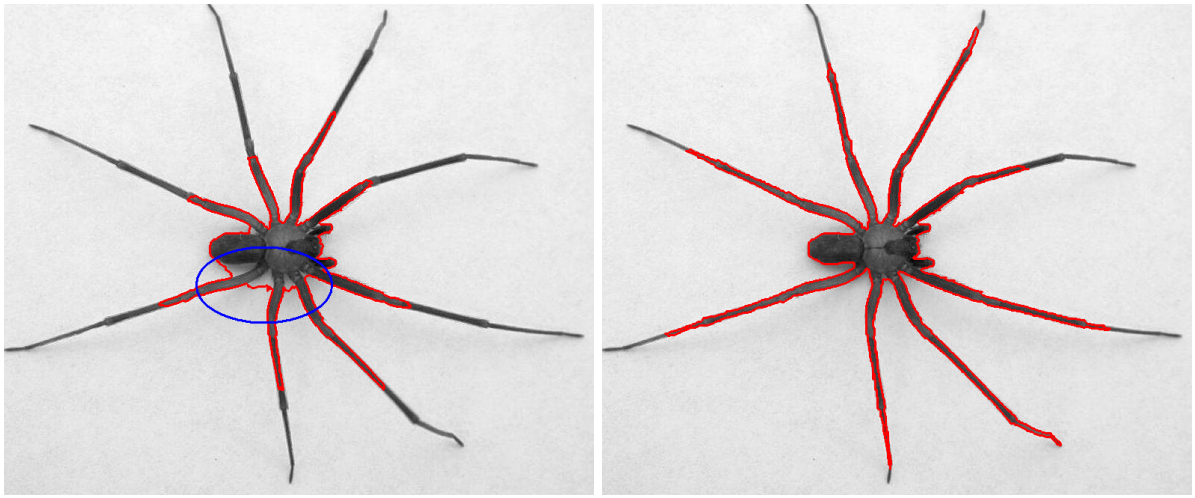
2.6 Summary

These first illustrations permit now to detail the weakness and the strength of the presented optimal graph structures:

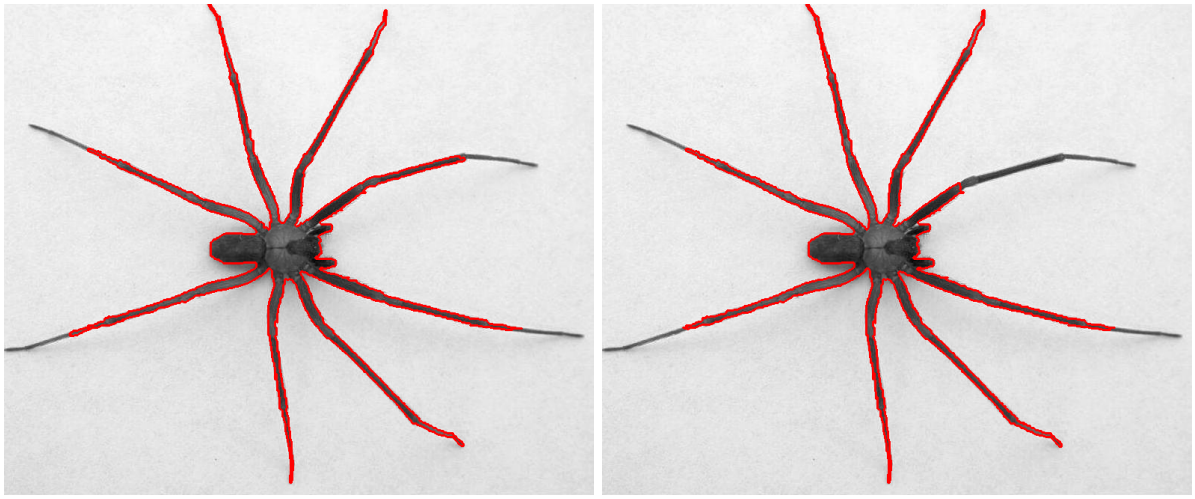
- Minimal spanning forests permit to detect homogenous and highly contrasted regions. Unfortunately, low contrasted structures are not correctly segmented by this structure. This fact is mainly due to the non-uniqueness of the minimal spanning forest in this kind of areas. Note also that minimal spanning forests do not take into account any kind of regularization in low contrasted areas. Minimal spanning forests can be computed in $O(|E|log(|V|))$ and is in practice the fastest method studied in this chapter.

- Shortest path forests also permit to detect homogenous and highly contrasted regions. Unfortunately, for low weight map power the shortest path forests suffer from the "overflow problem". Shortest path forests are especially sensitive to the markers location. Low contrasted structures are also not well segmented by this structure. However shortest path forests take into account a kind of regularization in low contrasted areas; the segmentation is, in these areas, composed of the zones of influence of the shortest path trees. Shortest path forests can be computed in $O(|E|log(|V|))$ and is in practice slower than minimal spanning forest.

- Obviously minimal cuts also permit to detect homogenous and high contrasted regions of an image. Unfortunately, for low weight map power the minimal cut suffers from the "shrinking problem". Minimal cuts sometimes provide segmentations of minimal perimeter. However this problem of graph cuts can sometimes be used advantageously since low contrasted areas and missing boundaries can be retrieved by segmentations of minimal perimeter. Minimal cut is the slowest method studied in this chapter in terms of computation time.

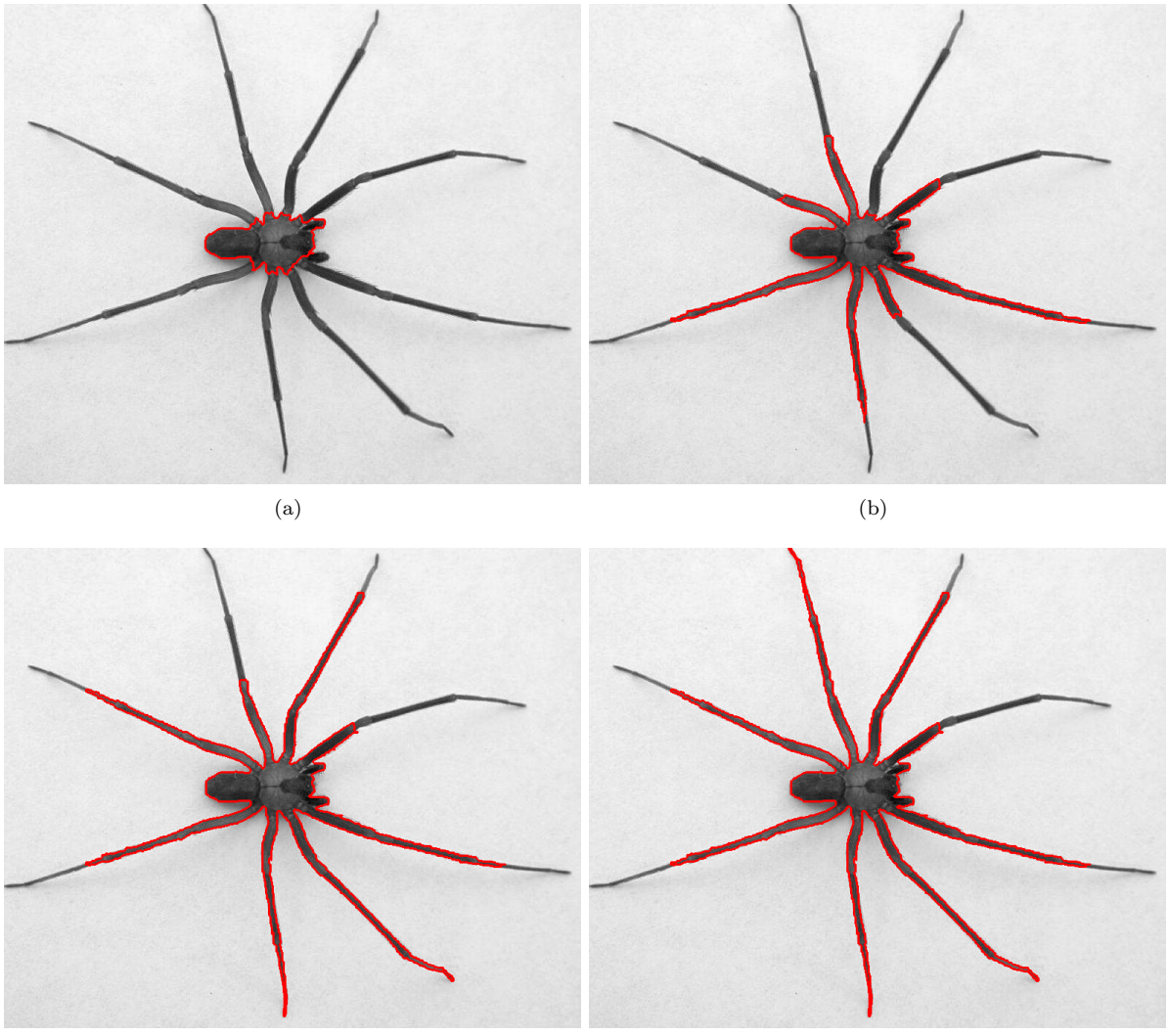


(a) (b)



(c) (d)

Figure 2.8: Shortest path forest segmentation for increasing weight map power. (a,b,c,d) Weight map power $n = 2, 4, 6, 16$. Bad detection area is outlined with a blue circle. Segmentation result for $n = 6$ is better than the one obtained with the minimum spanning forest, the legs of the spider are better detected. The shortest path forest is equivalent to a minimal spanning forest for $n = 16$.



(a) (b)

(c) (d)

Figure 2.9: Minimal cut segmentation for increasing weight map power. (a,b,c,d) Weight map power $n = 2, 4, 6, 16$. Object regions using minima cut do only cover the spider body. The minimal cut is equivalent to a minimal spanning forest cut for $n = 16$.

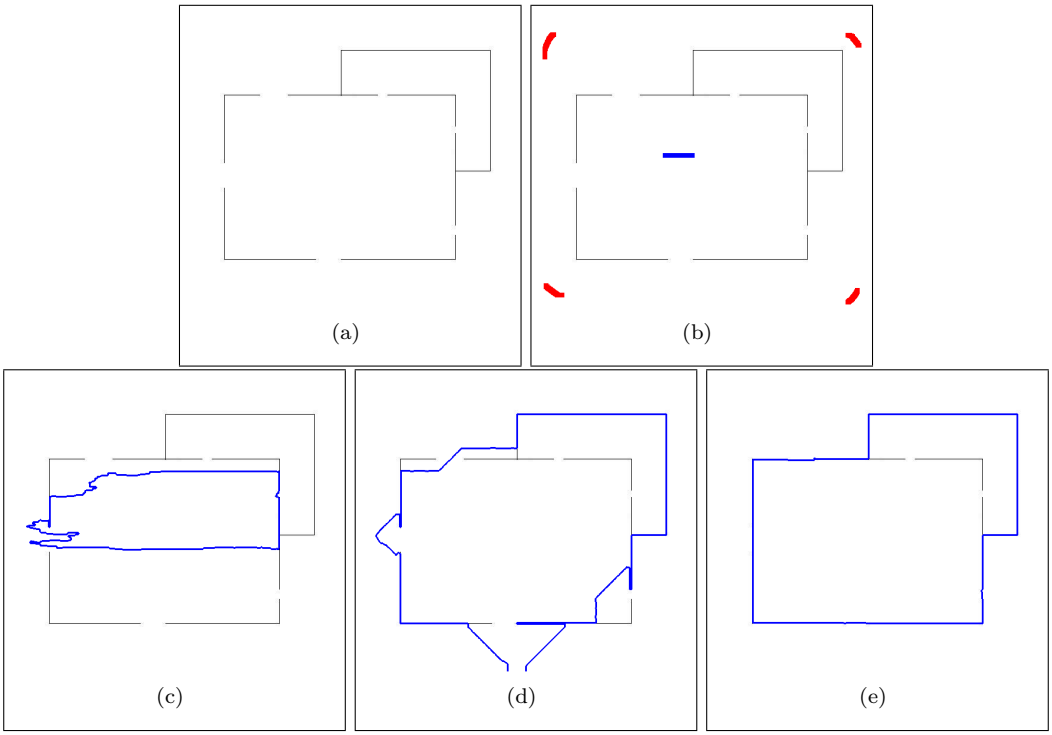
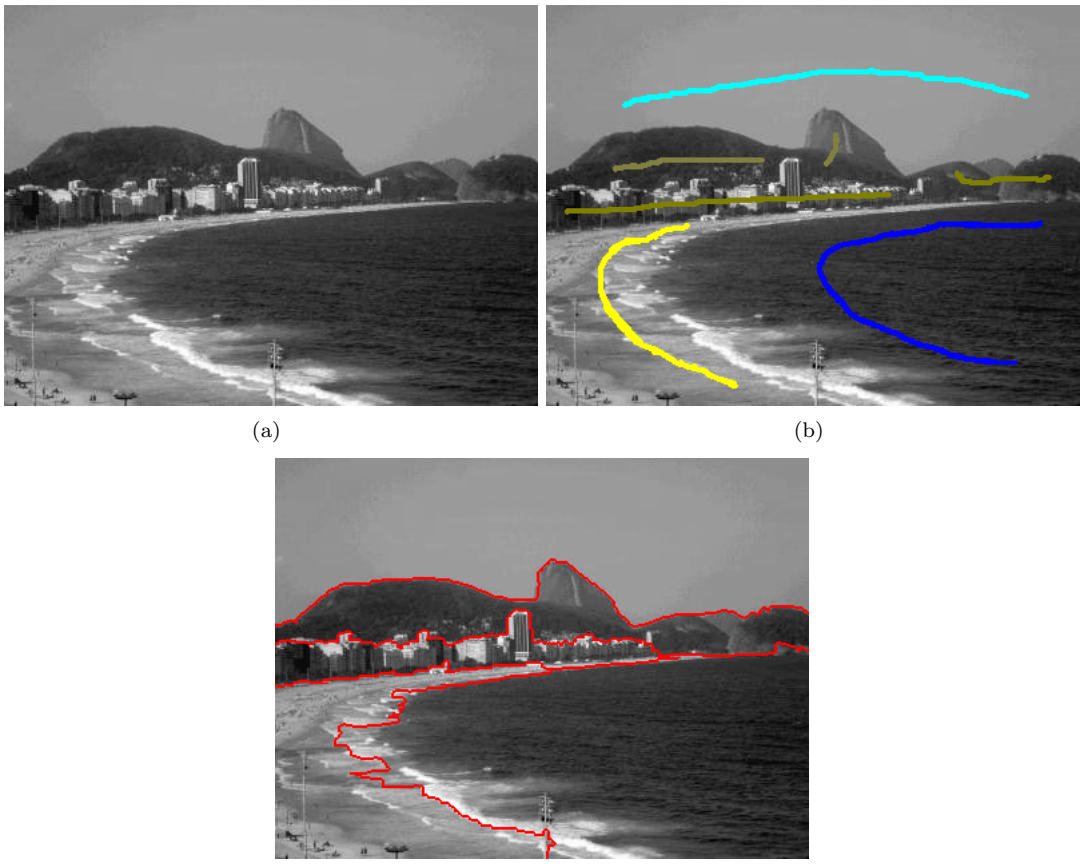


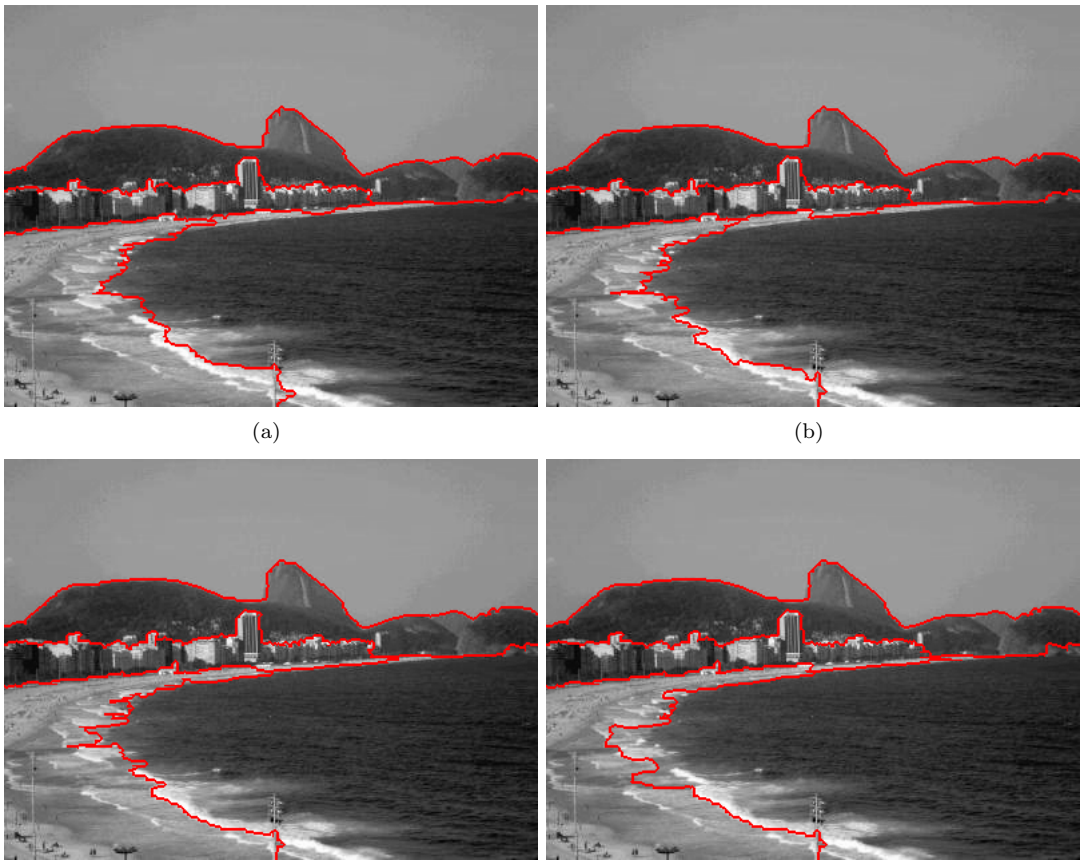
Figure 2.10: Objects with missing boundaries segmentation. (a) Original image (533x477). (b) Markers. (c) Minimal spanning forest segmentation. (d) Shortest path forest segmentation. (e) Minimal cut segmentation.



(a) (b)

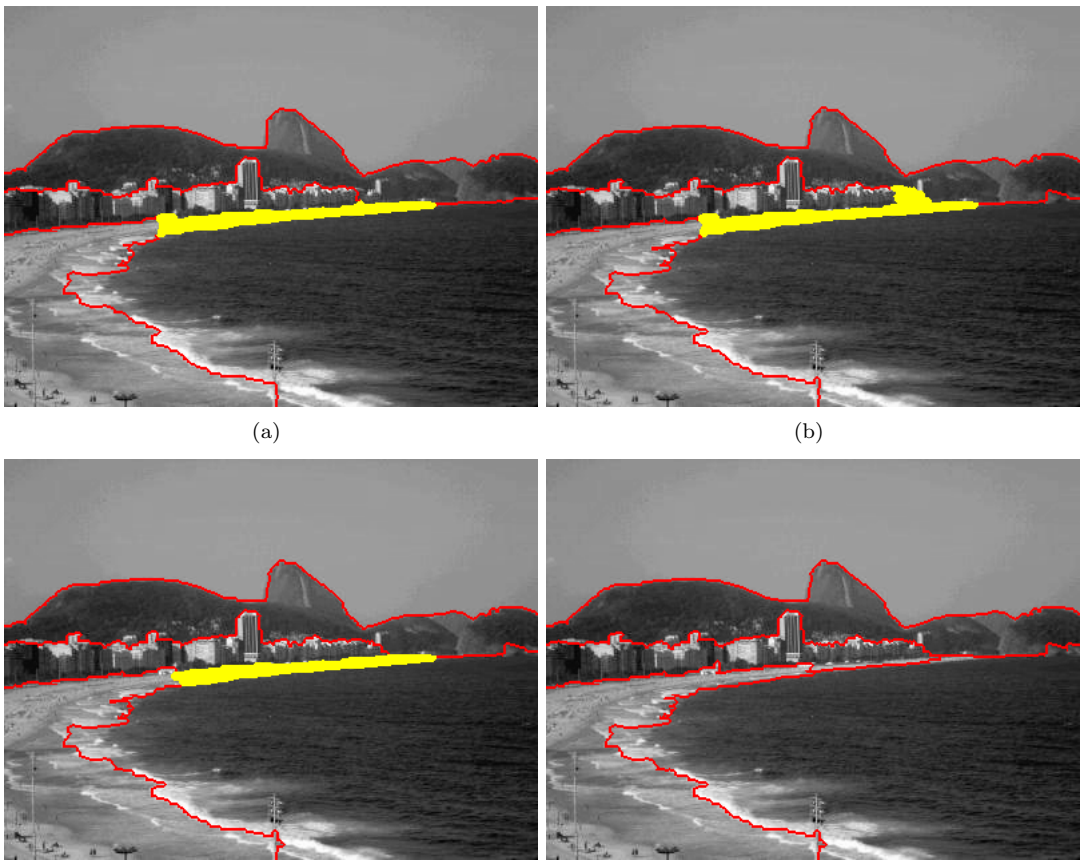
(c)

Figure 2.11: Minimal spanning forest segmentation. (a) Original image (400x300). (b) User provided markers. (c) Minimal spanning forest rooted on the markers.



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

Figure 2.12: Shortest path forest segmentation. (a-d) Segmentation results using the shortest path forest rooted on the markers with increasing weight map power, n=1,2,4,10.



(a) (b)

(c) (d)

Figure 2.13: Multi-terminal cut segmentation. (a-d) Segmentation results using the multi-terminal cut separating the markers with increasing weight map power, n=1,2,4,10. Unlabeled nodes are shown as red pixels. The set of unlabeled nodes, which gives an information on the optimality of the multi-terminal cut, usually decreases when the weight map power increases. For n=10, the computed multi-terminal cut is guaranteed to be optimal since all nodes are labeled.

2.7 Conclusion

Minimal cuts suffer from a shrinking problem, this technique often produces small cuts which are not relevant for image segmentation. The presented results offer an alternative way to compute minimal cuts which does not suffer from this problem. This procedure is faster than other methods proposed in the literature such as normalized cut [92], nested cut [112], spectral graph partitioning [97] and isoperimetric graph partitioning [46]. Indeed these methods are mainly based on the minimization of a ratio between the length of a cut and the area enclosed in it, which permit to obtain balanced partitions. However these methods require more complex and computational costly optimization tools, such as linear algebra tools for the computation of eigenvalues and vectors of large matrices.

On the other side, shortest path forests suffer from an overflow problem. This technique is very sensitive to the position and the distance of the pixels from the markers. Finally, minimal spanning forests seem to offer a good trade-off between these two last techniques and offer stable results. However we have seen that a particular weight map modification permits to use these methods and reduces their respective problems. However it seems that there is no particular method to compute automatically an ideal weight map power and very high weight map power can lead to numerical problems.

The presented methods are also adapted to different situations. The shortest path forests are particulary adapted to the segmentation of thin structures of homogenous grey level. On the other side minimal cuts are particulary adapted to the segmentation of compact objects with missing boundaries. Finally optimal spanning forests offers a good trade off between shortest paths forests and minimal cuts. It is also important to be aware of which method is potentially more adapted to a given problem since their computational complexities are different. Optimal spanning forests are the fastest methods but only permit to segment well contrasted structures. Shortest path forests exhibit a higher complexity but permit to detect thin and poor contrasted structures. Finally minimal cut is the slowest method but permit to detect low contrasted and compact structures.

3

Combining Morphological and Graph Cuts Segmentation

This chapter introduces two mathematical models widely used for image segmentation: geometric geodesics (2D images) and minimal surfaces (3D images), as well as probabilistic Markov random fields. We present optimization methods that we have developed to speed up the use of these mathematical models. Our approach consists in working on the "region adjacency graph" of a low-level morphological segmentation. These segmentation methods are based on the minimization of energy functions. We first detail how to design the objective functions and how to build the corresponding graph such that a minimal cut produces an optimal configuration for the studied models. We show that this approach offers a good trade-off between speed and precision compared to the classical use of graph cuts on the pixel graph. In both geometric and probabilistic models, we consider minimal graph cut on a region adjacency graph to minimize the objective functions.

We consider in this chapter the use of the region adjacency graph of an unsupervised watershed transform of the image. This low level segmentation consists in computing the watershed transform from all minima of the image's gradient modulus. The term "unsupervised watershed transform" denotes in this chapter the partition obtained by applying the watershed algorithm on the image's gradient modulus. This low level segmentation produces an output with numerous small regions. We make two assumptions to motivate our use of the region graph:

- first, this transform preserves all important boundaries of the image. The use of a region graph, instead of a pixel graph, should not penalize the detection of images boundaries. This property will be used to compute approximate geometric geodesics and minimal surfaces.
- this transform contains meaningful regions of homogenous intensities. The use of a region graph, instead of a pixel graph, should not penalize the detection of homogenous regions of an image. This property will be used to compute approximate maximum a posteriori estimates of probabilistic Markov random fields.

This chapter is subdivided as follows: section 3.1 introduces the general framework for minimizing an energy function with graph cuts. This framework is used to prove that the mathematical models we define later in the chapter can be optimized via graph cuts.

Section 3.2 presents the unsupervised watershed transform and its region adjacency graphs. In section 3.3, we detail how to compute approximate geodesics and minimal surfaces from a region adjacency graph. We especially consider curves of minimal perimeter and surfaces of minimal area in Riemmanian spaces. This approach is illustrated through interactive segmentation of 3D medical images [100, 107]. We also compare our approach, in terms of computational complexity and precision to the classical use of

this model at the pixel level. We show that the use of a region graph offers a good trade-off between speed and accuracy compared to the pixel graph, moreover it permits to introduce new geometric constraints in the segmentation which are impossible to consider at the pixel level.

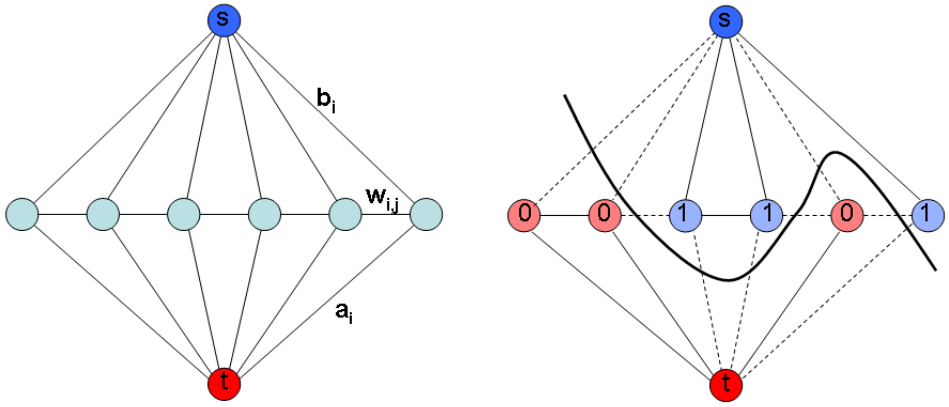
Finally, section 3.5 introduces the probabilistic Markov random field model for image segmentation and restoration. Markov random fields are powerful models for the restoration and the segmentation of images. Unfortunately using this method on the pixel graph is computationally costly and cannot always be used interactively on large datasets, such as 3D medical images. We present in this section a novel method to compute fast image segmentations using the Markov random field model and a morphological low-level segmentation. This technique allows us to use interactively this model on large datasets. This approach is used for the restoration of noisy data and the segmentation of 3D micro-tomography images of micro-structures [104].

3.1 Energy Minimization via Graph Cuts

Let us first introduce some generalities about energy minimization via graph cuts. In the last years, graph cuts have been widely used as an optimization process to minimize energy functions associated to common computer vision problems such as stereo vision, image segmentation or image restoration [17, 13, 14, 16, 65]. The key idea is that a graph cut implicitly defines an optimal labeling of the pixels of an image.

Let us consider a graph $G = (V, E, W)$. Assume that two extra terminal nodes s and t are added to the graph, these terminal nodes being connected to all nodes of the graph with the set of edges $E_s = \{e_{s,i}, i \in V\}$ and $E_t = \{e_{i,t}, i \in V\}$. We denote this new graph as $G_{s,t} = (V \cup \{s,t\}, E \cup E_s \cup E_t, W)$. Let $a_i \in \mathbb{R}^+$ be the weight of an edge $e_{s,i} \in E_s$, $b_i \in \mathbb{R}^+$ the weight of an edge $e_{i,t} \in E_t$ and $w_{i,j} \in \mathbb{R}^+$ the weight of an edge $e_{i,j} \in E$. Note that we restrict our attention to graphs in which it is possible to compute a minimal cut, i.e. a graph with positive edges weights.

We associate to a graph cut $C_{s,t}$, separating the nodes s and t, a labeling function x_i such that for $i \in V, x_i = 1$ if a path exists from node i to node s after the removal of edges of $C_{s,t}$ and $x_i = 0$ otherwise. We assign a distinct label of the set $\{0, 1\}$ to each connected components of the graph after the removal of the graph cut $C_{s,t}$ depending if this connected component contains the node s or t. This labeling procedure is illustrated in figure 3.1.



(a) (b)

Figure 3.1: (a) A graph with two additional nodes s and t. (b) A cut $C_{s,t}$ separating the terminal nodes s and t. The colors of the nodes indicate whether the nodes are assigned the label 0 or 1.

The weight of any cut $C_{s,t}$ in the graph $G_{s,t}$ can then be written as:

$$L_1(C_{s,t}) = \sum_{i \in V} a_i.x_i + \sum_{i \in V} b_i.(1 - x_i) + \sum_{e_{i,j} \in E} w_{i,j}.(1 - x_j).x_i \ . \tag{3.1.1}$$

We can then separate constant terms from the terms depending on up to two binary variables:

$$L_1(C_{s,t}) = \sum_{i \in V} b_i + \sum_{i \in V} (a_i - b_i).x_i + \sum_{e_{i,j} \in E} w_{i,j}.x_i - \sum_{e_{i,j} \in E} w_{i,j}.x_i.x_j \ . \tag{3.1.2}$$

$$L_1(C_{s,t}) = \sum_{i \in V} b_i + \sum_{i \in V} (a_i - b_i + \sum_{j \in V} w_{i,j}).x_i - \sum_{e_{i,j} \in E} w_{i,j}.x_i.x_j \ . \tag{3.1.3}$$

Finally the weight of a cut in the graph $G_{s,t}$ can be written as a function $E(x_1, ..., x_n)$ $(n = |V|)$ of the binary variable field x_i:

$$E(x_1, ..., x_n) = L_1(C_{s,t}) = A + \sum_{i \in V} \alpha_i x_i + \sum_{e_{i,j} \in E} \beta_{i,j}.x_i.x_j \ . \tag{3.1.4}$$

Where:

$$\begin{cases} A = \sum\limits_{i \in V} b_i \ , \\ \alpha_i = (a_i - b_i + \sum\limits_{j \in V} w_{i,j}) \ , \\ \beta_{i,j} = -w_{i,j} \ . \end{cases} \tag{3.1.5}$$

From this formula we can state the following characterization of energy functions of binary variables that can be minimized via graph cuts:

Theorem 3.1.1 (Energy Minimization via Graph Cuts)**.** *Any function which can be written as a sum depending on up to two binary variables $x_i \in \{0, 1\}$:*

$$E(x_1, ..., x_n) = A + \sum_{i \in V} \alpha_i x_i + \sum_{ei,j \in E} \beta_{i,j}.x_i.x_j$$

can be exactly minimized by graph cuts if and only if $A \in \mathbb{R}^+$, $\alpha_i \in \mathbb{R}$ and $\beta_{i,j} \in \mathbb{R}^-$.

A constructive proof of this theorem can easily be given by following the procedure that we described to obtain the graph $G_{s,t}$. Since a cut of minimal weight can be found in the graph $G_{s,t}$, it is straightforward to verify that the weight of the cut will also minimize the energy function $E(x_1, ..., x_n)$. Moreover the labeling process associated with the graph cut provides explicitly which configuration of binary variables is optimal for a given energy E. A proof of the theorem can be found in [61] or [37].

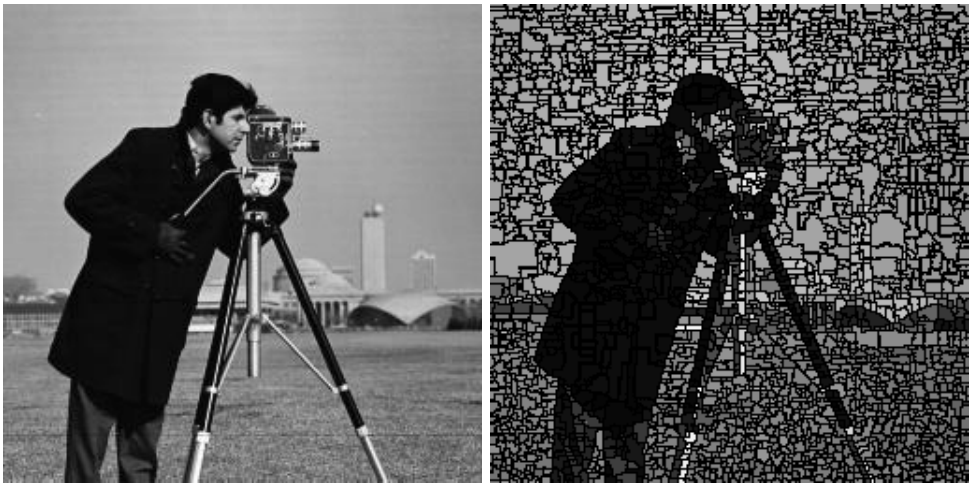
This theorem is the key result for the researchers working on energy minimization using graph cuts [61, 37]. Kolmogorov and Zabih [61] have characterized energy functions which can be written as a sum depending on up to three binary variables that can be minimized via graph cuts and Freedman et al. [37] have extended these results to the class of energy functions depending on n binary variables.

These results are very useful since graph cuts can potentially be an excellent optimization method as soon as the energy function to minimize fulfills the conditions of Theorem 3.1.1. We will see in the next sections that at least two usual mathematical models used for image segmentation can in fact be solved using graph cuts as an optimization tool. We will especially consider the computation of curves and surfaces minimizing a given Riemannian metric, as well as the maximization of the a posteriori estimation of a Markov Random Field.

3.2 Morphological Segmentation

A low-level morphological segmentation is used in this chapter to produce a simplification of the image, and afterwards only the region adjacency graph of the low-level segmentation is considered to build the final image segmentation. We consider here the low-level segmentation of the image produced by a watershed transform. An example of an unsupervised watershed segmentation of an image is illustrated in figure 3.2. The watershed transform [75], from mathematical morphology, allows to obtain a partition of an image composed of small and numerous homogenous regions. Moreover important contours of the image are preserved during the segmentation and regions of the partition are mostly composed of homogenous pixels (pixels of similar grey values). The quality of this first unsupervised segmentation is important to guarantee a minimal loss of information, an ideal situation would be that important information (contours and/or homogenous regions) about the original image is accessible from this first segmentation. These observations are not theoretically guaranteed but these assumptions are verified

when working on real life problems and natural images. Another important point is that the common watershed transform algorithm proposed by Meyer exhibits a linear complexity [72]. The time needed by the watershed transform is in practice negligible compared to the graph cuts algorithm.



(a) (b)

Figure 3.2: Low-level segmentation. (a) Original image. (b) A low-level segmentation of the image obtained from a watershed transform. Each region of the watershed is assigned the mean grey level of the pixels of the first image.

Section 3.3 is devoted to the extraction of an optimal curve or a surface composed of boundaries of the low-level segmentation (curve of minimal length in a specific Riemannian space, or a surface of minimal area). In this context we study the characteristics of the image on the boundaries of the low-level segmentation. We assume in the following section that a near optimal curve or surface can be extracted from a watershed low-level segmentation. Contrary to the classical use of the pixel graph, which takes into account all possible curves and surfaces of the lattice, we only consider the set of curves or surfaces composed by the union of contours of the watershed segmentation. The quality of the low-level segmentation is thus important since it must provide all important boundaries of the image (i.e. curves and surfaces that are representative of some objects of the image).

In section 3.5, image segmentation is considered from a probabilistic point of view. In this context we study the characteristics of the image inside regions of the low-level segmentation. We assume in this section that homogenous regions can be extracted from a watershed low-level segmentation. Contrary to the classical use of the pixel graph, which takes into account the grey level of each single pixel, we only consider the average grey level inside a region of the low-level segmentation.

3.3 Minimal Surfaces and Geodesics

Geodesics and minimal surfaces are widely used for medical image segmentation and are considered as a leading method for a wide range of applications. Up to now at least two distinct approaches of the problem are used to compute such segmentations. First, geodesic active contours [20] use a differential geometry framework to compute optimal contours minimizing a given Riemannian metric. Second, Boykov and Kolmogorov [14] have proposed a method based on integral geometry to compute similar contours using a graph representation of the image and graph cuts. In this section we present a technique to compute approximate geodesics and minimal surfaces using a low-level segmentation and graph cuts optimization. Our approach is designed to speed up the computation of minimal curves and surfaces by considering a

subset of interesting curves and surfaces composed of the contours of the watershed transform.

3.3.1 Differential Geometry Approach

One of the first applications of differential geometry in image segmentation has been introduced by Kass et al. in [59]. The method, called "snakes", as well as many variants of active contours models, has been widely used for image segmentation. An important development has been introduced via a new representation of active curves and surfaces [81], [91]. Parametric active contours have been replaced by an implicit representation of curves and surfaces via level-sets. This representation allows topological changes of curves and a better handling of numerical schemes to achieve the energy minimization. A further development of active contours has been introduced by Caselles et al. in [20, 21] with "Geodesic active contours". This method simplifies the energy function to be minimized. The problem is formalized as the minimization of the energy:

$$E(C) = \int_0^{|C|_\varepsilon} g(\|\nabla I(C(s))\|)ds \ , \tag{3.3.1}$$

where $|C|_\varepsilon$ is the Euclidean length of a curve C, s is the arc length on the curve. g is a positive and strictly decreasing function and $\|\nabla I(C(s))\|$ is the modulus of the gradient of the image I along the curve C.

The aim of the method is to find a curve such that the contrast (modulus of the gradient) of the image along the curve is maximal. This method is in fact equivalent to the minimization of the length of the curve C according to a Riemannian metric. The Riemannian metric depends here on the modulus of the gradient of the image I. The length of a geometric curve under a Riemannian metric can be written as:

$$|C|_R = \int_0^{|C|_\varepsilon} \sqrt{\tau_s^T D(C(s))\tau_s} ds \ , \tag{3.3.2}$$

where τ_s is a unit tangent vector to the curve C and D is a positive definite matrix, called the metric tensor, specifying the local metric. In the "Geodesic active contours" method [20, 21] the local Riemannian metric at a point p is given by the following positive and symmetric matrix:

$$D(p) = \begin{pmatrix} g(\|\nabla I(p)\|)^2 & 0 \\ 0 & g(\|\nabla I(p)\|)^2 \end{pmatrix} = g(\|\nabla I(p)\|)^2 . \mathrm{Id}_2 \ , \tag{3.3.3}$$

where Id_2 is the identity matrix. The length of a geometric curve under the Riemannian metric given by equation 3.3.3 is then equal to the energy function defined by equation 3.3.1. "Geodesic active contours" method aims to minimize the energy given by equation 3.3.1 via a gradient descent scheme and a level-sets representation of curves. Unfortunately, the method is sensitive to initialization and there is no guarantee that the global minimum of the energy is reached with this optimization process. However the method can also be extended to three dimensions [20, 21]. We will detail in the following sections how to compute a curve or a surface minimizing exactly this energy.

3.3.2 Integral Geometry Approach

Contrarily to Caselles et al. [20, 21], Boykov and Kolmogorov [14] have considered the computation of minimal surfaces and geodesics based on the Cauchy-Crofton formulas of integral geometry. French mathematician Augustin Louis Cauchy (1789-1857) established a formula which relates the length of a curve C to a specific measure of a set of lines intersecting it.

Let $L(\rho, \theta)$ be a straight line of $\mathbb{R}^2$ characterized in polar coordinates by the two parameters (ρ, θ). The Cauchy-Crofton formula expresses that the Euclidean length of a regular curve C in $\mathbb{R}^2$ under the

form:

$$|C|_\varepsilon = \frac{1}{2}\int_0^\pi \int_{-\infty}^{\infty} N(\rho,\theta)d\rho d\theta \ , \tag{3.3.4}$$

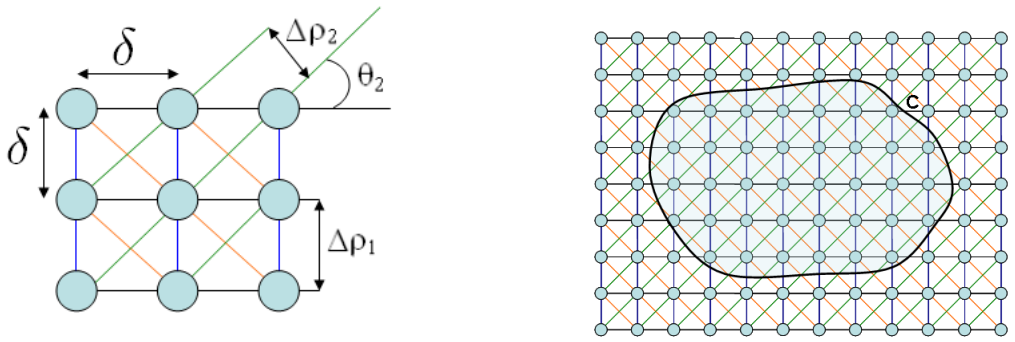
where $N(\rho,\theta)$ is the number of intersections of $L(\rho,\theta)$ with C.

This formula can also be extended to Riemannian spaces [14], the length of a curve C according to the metric tensor D is then given by:

$$|C|_R = \frac{1}{2}\int_0^\pi \int_{-\infty}^{\infty} \frac{detD}{2(u_L^T D u_L)^{3/2}} N(\rho,\theta)d\rho d\theta \ , \tag{3.3.5}$$

where u_L are the unit vectors in the direction of the line L. This formula is valid for any continuously differentiable and regular curve in $\mathbb{R}^2$.

These last formulas can be used in the case of a discrete space. Let N_n be an adjacency system defined on a discrete isotropic grid of size δ. N_n can be described as a finite set of vectors e_k, $N_n = \{e_k : 1 \leq k \leq n\}$, where n is the number of distinct vectors describing the adjacency system. Each vector e_k generates a family of lines as illustrated in figure 3.3 where each line is separated by a distance $\Delta\rho_k$ from the closest line of the family. Now let θ_k be an angular parameter. For a fixed θ_k, we obtain a family of parallel lines separated by a distance $\Delta\rho_k$ as illustrated in figure 3.3.



(a) (b)

Figure 3.3: (a) The 8-Neighborhood adjacency system. (b) The Cauchy-Crofton formula establishes a link between the Euclidean length of a curve C and the intersection of C with a finite set of lines.

Equation 3.3.4, considered in a discrete space, gives the following approximation of the Euclidean length of a curve C [14]:

$$|C|_\varepsilon \approx \frac{1}{2}\sum_{k=1}^{n}(\sum_i n_c(i,k)\Delta\rho_k)\Delta\theta_k = \sum_{k=1}^{n} n_c(k)\frac{\delta^2\Delta\theta_k}{2|e_k|} \ . \tag{3.3.6}$$

Where i indexes the k^{th} family of lines. $n_c(i,k)$ counts the number of intersections of line i of the k^{th} family of lines with curve C. $n_c(k) = \sum_i n_c(i,k)$ is the total number of intersections of the k^{th} family of lines with C. This approximation is used in the next section to compute the length of a curve defined on a discrete grid. Moreover this formula is linked to the cost of a graph cut.

3.3.3 Minimal Surfaces and Geodesics via Graph Cuts

Boykov and Kolmogorov [14] have considered the computation of minimal surfaces and geodesics separating two sets of pixels representing the "object" markers and the "background" markers. Two additional nodes s and t are respectively connected to "object" markers and "background" markers. "s-links" and "t-links", arcs connected to s or t, are given an infinite capacity to ensure that the the segmented regions respectively contain the "background" markers and the "foreground" markers as illustrated in figure 3.4. This method is the same as the method presented in the previous chapter for the computation of a minimal graph cut separating a set of marked nodes.

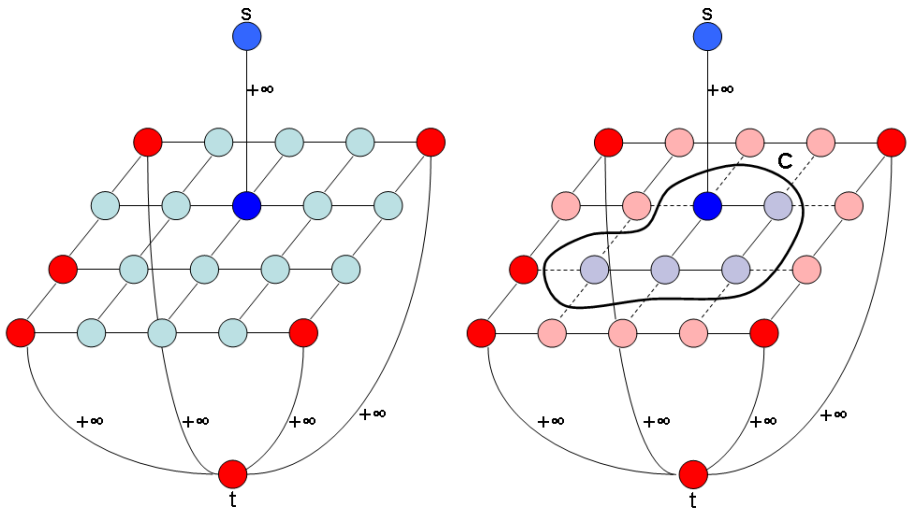


Figure 3.4: (a) Graph construction with two markers. (b) A Graph cut in the constructed graph.

Boykov and Kolmogorov [14] have connected the cost of a graph cut to the length of a curve crossing the edges of the cut as shown in 3.3. Let us consider an image embedded on a discrete grid and let N_n be an adjacency system defined on the image. As described in the previous section, the adjacency system defines a family of lines. The cost of a graph cut C in the constructed graph is equal to:

$$L_1(C) = \sum_{k=1}^{n} n_c(k) w_k \ . \tag{3.3.7}$$

Where $n_c(k)$ is the number of arcs of family k included in the graph cut, and w_k is the weight of the arcs of family k.

The Cauchy-Crofton formula given by 3.3.6 can be directly used to set arcs weights such that the cost of a graph cut approximates the Euclidean or Riemannian length of a curve that would cross the edges of the cut C:

$$L_1(C) = \sum_{k=1}^{n} n_c(k) w_k = \sum_{k=1}^{n} n_c(k) \frac{\delta^2 \Delta\theta_k}{2|e_k|} \ , \tag{3.3.8}$$

The previous relation can also be extended to Riemannian metrics by using the following edges weights [14]:

$$w_k(p) = \frac{\delta^2 . |e_k|^2 . \Delta\theta_k . det(D(p))}{2(e_k^T D(p) e_k)^{3/2}} \ . \tag{3.3.9}$$

Where $w_k(p)$ is the weight of the arcs leaving the node p, and $D(p)$ is the local Riemannian metric at node p. Note that the previous expressions can also be extended to 3D spaces [14]. These relations take

a very simple form, assuming the 4-neighborhood system, or the 6-neighborhood system (in 3D), and a discrete grid of size $\delta = 1$, we have:

$$w(p) \propto \frac{det(D(p))}{2(e_k^T D(p) e_k)^{3/2}} \ . \tag{3.3.10}$$

Considering the metric tensor defined by equation 3.3.3, we have:

$$w(p) \propto \frac{g(\|\nabla I(p)\|)^4}{2(g(\|\nabla I(p)\|))^3} = g(\|\nabla I(p)\|) \ . \tag{3.3.11}$$

These formulas show explicitly that the cost of a graph cut is related to the geometric length of a curve crossing a graph. Unfortunately, using this method on the pixel graph is computationally costly and cannot always be used interactively on large datasets such as 3D medical images. We present in the next section a novel method to compute fast approximate geodesics and minimal surfaces from an initial low level segmentation of the image.

3.3.4 Approximate Geodesics and Minimal Surfaces

The combination of graph cuts with a watershed low-level segmentation provides us an explicit way to compute geodesics and minimal surfaces. Our basic assumption is that the geodesic to be computed is embedded in the watershed low-level segmentation contours. This proposition is motivated by two observations. Firstly, the unsupervised watershed transform, without pre-processing or marker selection, produces an over-segmentation of real images. Secondly, the watershed lines contain all major boundaries of the image. We propose thus to solve the following combinatorial problem: finding a curve composed of a finite union of watershed contours such that the curve minimizes a given geometric functional as illustrated in figure 3.5. We will solve this problem by using graph cuts optimization on a region adjacency graph, as originally proposed by Li et al. [65] for interactive photo edition.

Following the method of Caselles et al. [20], we consider a curve C defined by a finite union of watershed contours and minimizing the following energy:

$$|C|_R = \int_0^{|C|_\varepsilon} \sqrt{\tau_s^T D(C(s)) \tau_s} ds \ . \tag{3.3.12}$$

Let us first consider the pixel graph $G = (V, E, W)$ of an image. We also define the region adjacency graph $G_R = (V_R, E_R, W_R)$ of the watershed transform where $V_R = \{r_k, k \in [1, ...n]\}$ is the set of nodes (i.e the regions of the watershed transform). Note that r_k is also a set of nodes of the pixel graph. E_R is the set of edges (i.e the neighborhood relation between regions) and W_R is the set of weights of the edges as illustrated in 3.6.

Let us define $F_{(r_i,r_j)}$ as the set of edges of the pixel graph between two regions r_i and r_j of the low-level watershed segmentation:

$$F_{(r_i,r_j)} = \{e_{m,n} \in E \mid m \in r_i, \ n \in r_j\} \ . \tag{3.3.13}$$

One should note that the set $F_{(r_i,r_j)}$ depends on the adjacency system of the pixel graph G. The set of edges of the pixel graph describes also implicitly a set of curves between the regions r_i and r_j as illustrated in figure 3.6. Let us now define $C_{(r_i,r_j)}$ as the set of curves that can cross the edges of $F_{(r_i,r_j)}$. Using Cauchy-Crofton formulas, it is possible to compute the length $|C_{(r_i,r_j)}|_R$ of the boundaries between two regions of the low-level segmentation. Note also that if regions r_i and r_j are not adjacent, $F_{(r_i,r_j)}$ and $C_{(r_i,r_j)}$ are simply empty sets.

We consider the following strictly positive and decreasing function g:

$$g(\|\nabla I(p)\|) = \left(\frac{1}{1 + \|\nabla I(p)\|} \right)^k \ . \tag{3.3.14}$$

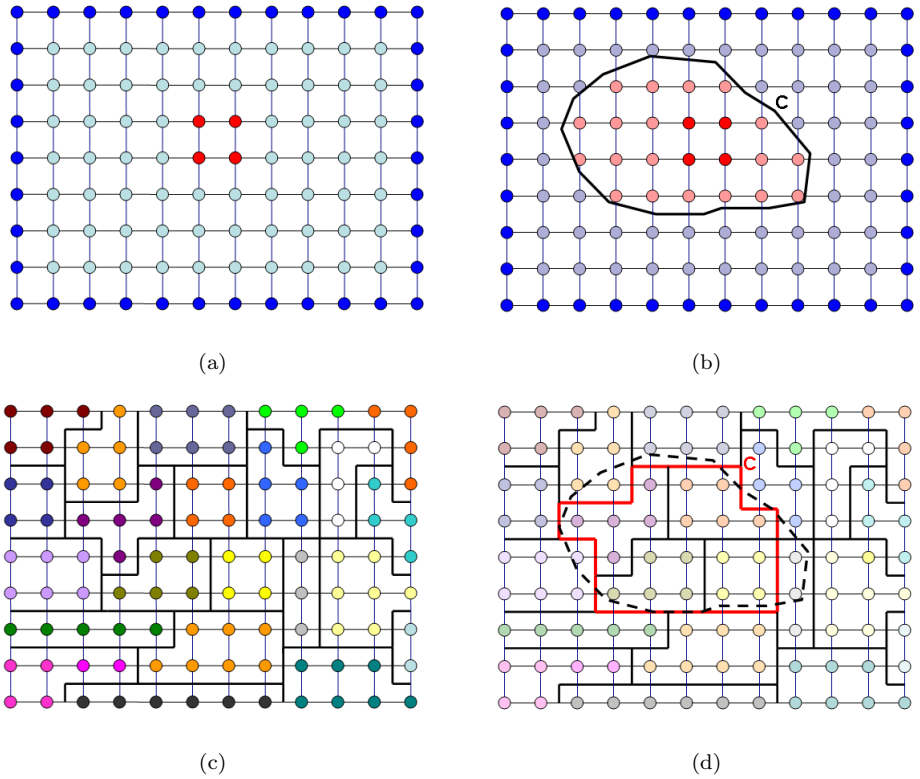


Figure 3.5: Comparison between approximate and exact geodesics. (a) Pixel graph and markers. (b) Geodesic computed on the pixel graph. (c) Region adjacency graph. (d) Geodesic computed on the region adjacency graph.

The parameter $k \in \mathbb{R}^+$ is a smoothing parameter as shown in chapter 1. In our application this parameter was set to $k = 2$.

Following Cauchy-Crofton formulas with the V_4 adjacency system, the Riemannian length of the contour $|C_{(r_i,r_j)}|_R$ is approximated by:

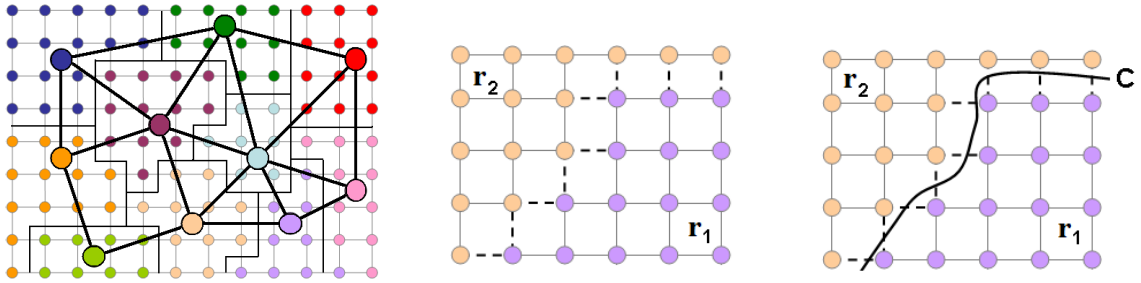
$$|C_{(r_i,r_j)}|_R \approx \sum_{(e_{m,n} \in F_{(r_i,r_j)})} g(\max(\|\nabla I(m)\|, \|\nabla I(n)\|)) \,, \tag{3.3.15}$$

where $(\|\nabla I(m)\|, \|\nabla I(n)\|)$ are the gradient magnitude of the end nodes of $e_{m,n}$.

The edges weights of the region adjacency graph are then set such that the weight of a graph cut equals the Riemannian length of the curve it implicitly defines:

$$w_{r_i,r_j} = |C_{(r_i,r_j)}|_R = \sum_{(e_{m,n} \in F_{(r_i,r_j)})} \left(\frac{1}{1 + \max(\|\nabla I(m)\|, \|\nabla I(n)\|)} \right)^k . \tag{3.3.16}$$

The Riemannian metric works as an edge indicator of the image I and takes a small value if neighbors pixels m and n take different grey values p_m and p_n. Note that the Riemannian length $|C_{(r_i,r_j)}|_R$ of the boundary between two regions is simply obtained by summing the local contrasts along the boundaries between these two regions. The weight w_{r_i,r_j} approximates the Riemannian length $|C_{(r_i,r_j)}|_R$ of a curve $C_{(r_i,r_j)}$ crossing the edges of $(F_{(r_i,r_j)})$ in the case of a 4-neighborhood adjacency system. According to the Cauchy-Crofton formula, the number of edges $(e_{m,n} \in F_{(r_i,r_j)})$ indicates the number of intersections



(a) (b) (c)

Figure 3.6: (a) Region adjacency graph of a low-level watershed segmentation. (b) The set of nodes of the pixel graph considered to compute boundary properties between regions, with a V_4 adjacency system. (c) A curve crossing the edges of the borders between regions r_1 and r_2.

of a curve with the horizontal and vertical lines describing the 4-neighborhood adjacency system. Alternatively, the Cauchy-Crofton formulas can also be used to compute the approximate Riemannian length of the curves $C_{(r_i,r_j)}$ between two adjacent regions in the case of different adjacency systems. However in our applications we only consider the 4-neighborhood (in 2D) and 6-neighborhood (in 3D) systems for simplicity.

Our minimization problem is reduced to the search of a curve among all curves implicitly described by the frontiers of the watershed regions instead of searching among all curves in the domain of the image. This approximation reduces drastically the search space and thus speeds up the computation of minimal curves. The quality of the low-level segmentation has also a direct impact on the quality of the approximation. Note also that the method can easily be extended to three dimensional spaces by considering sums along surfaces instead of curves. The aim of the segmentation task is now to find a surface S that minimizes the following energy:

$$|S|_R = \int\int_S g(||\nabla I(x,y)||)dxdy \ , \tag{3.3.17}$$

where S is a surface and g is a positive and strictly decreasing function.

We can use the same edges weights defined in equation 5.3.9, and apply them on the region adjacency graph in 3D ($w_{r_i,r_j} = |S_{(r_i,r_j)}|_R$). Note that in 3D $F_{(r_i,r_j)}$ describes implicitly a set of surfaces $S_{(r_i,r_j)}$ separating regions r_i and r_j. Our formulation is thus independent of the dimension of the underlying space.

3.3.5 Application of Minimal Surfaces to Medical Image Segmentation

This section illustrates the presented method by two studies concerning medical image segmentation.

The first example illustrates the method on the segmentation of the liver in a 3D CT image. The difficulty is that the liver presents low-contrasted boundaries. For such structures, the use of minimal surfaces remains a leading method. The graph cuts used on the watershed adjacency graph (see figure 3.7(a)) outperforms the results given by the classical marker-controlled watershed segmentation [75] (see figure 3.7(d)) and speeds up the graph cuts method proposed by Boykov et al. [14] (see figure 3.7(c)) without affecting the quality of the segmentation.

The last example illustrates the extension of the presented method in the case of multiple organs segmentation by multiple minimal surfaces. This extension uses multi-terminal cuts to compute each minimal surface. The example illustrates the segmentation of the liver, the lungs and the kidneys in

a 3D CT image. The liver and the kidney present low-contrasted boundaries and the segmentation of such organs remains a difficult task. For this application, minimal surface is a leading method that outperforms the classical marker-controlled watershed segmentation as shown in figure 3.8. Indeed, the marker-controlled watershed does not detect the contours of the liver due to the low contrast between this organ and the surrounding tissue.

Table 3.1 compares the graph sizes using our method and the graph cuts approach proposed in [14]. The nodes reduction factor and the edges reduction factor are respectively equal to $\frac{|V|}{|V_R|}$ and $\frac{|E|}{|E_R|}$. The results show that our method greatly reduces the graph sizes. This property has also a direct impact on the computation time needed for the segmentation, since graph cut algorithms exhibit a computational complexity proportional to both the graph nodes and edges number.

Table 3.2 illustrates the computation time needed by the three methods presented above: the marker-controlled watershed [75], our method, and the minimal surfaces by graph cuts proposed in [14]. The results show that our method reduces drastically the computation time. Moreover it does not seem to affect the results quality, as illustrated in figures 3.7 and 3.8.

Considering that the watershed transform contains all major boundaries in real images, our approximate segmentation is in practice quite efficient. However the graph cuts approach works slower than the classical marker-controlled watershed but offers more stable results. On the other hand our method is not as precise as the graph cuts method proposed by Boykov et al. [14], but it offers a good trade-off between speed and precision.

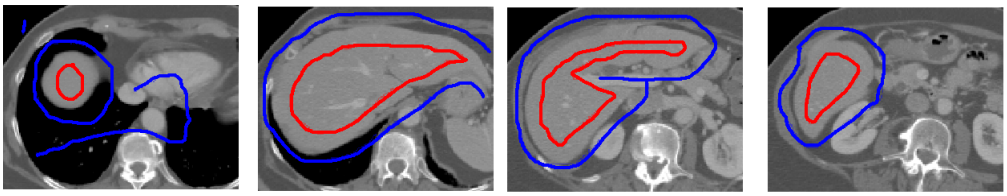
A Graph cuts approach cannot always be used on large images when the graph considered is the pixel adjacency graph because of the memory requirements and the computational complexity of the method. The developed method can efficiently be used on large images considering the region adjacency graph instead of the pixels graph. Our method does not seem to introduce large biases in the resulting segmentation of natural images. Limitations of our methods are quite clear since it can only be used when an over-segmentation can be obtained. The next section presents an interesting extension of our methodology by considering geometric properties that cannot be taken into account at the pixel level.

Image	Size	Nodes Reduction Factor	Edges Reduction Factor
Cardiac MRI	86x128x90	11.1	5.25
Liver CT	256x193x179.	24,3	14,2

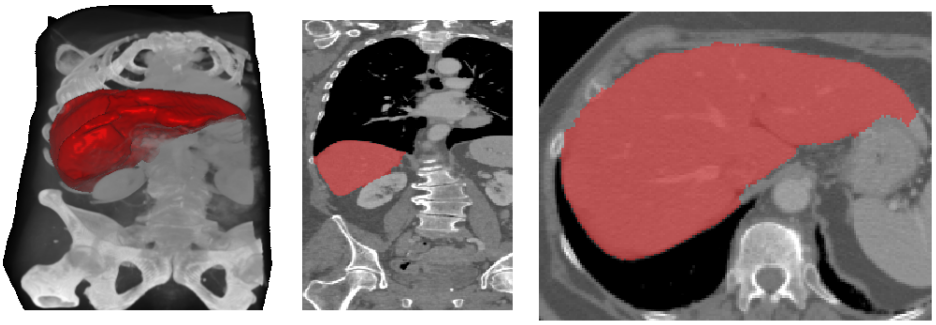
Table 3.1: Comparisons of Graphs Sizes.

Image	Watershed Transform [75]	Our method	Exact Min. Surfaces [14]
Cardiac MRI	1,5 sec.	4,2 sec.	15,2 sec.
Liver CT	9,7 sec.	41,6 sec.	1400,2 sec.

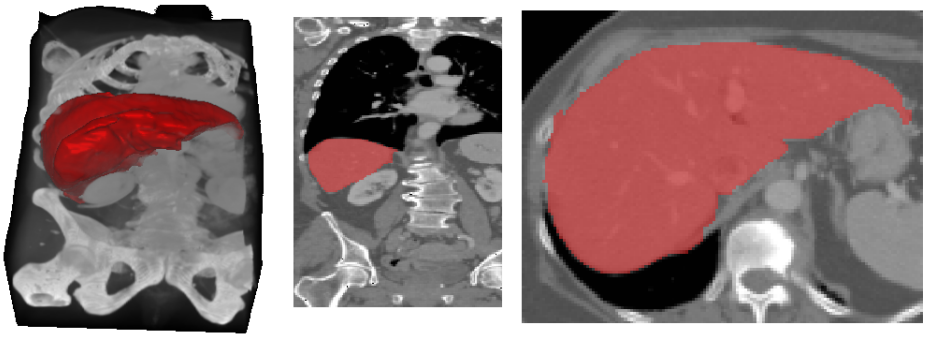
Table 3.2: Comparisons of computation time.



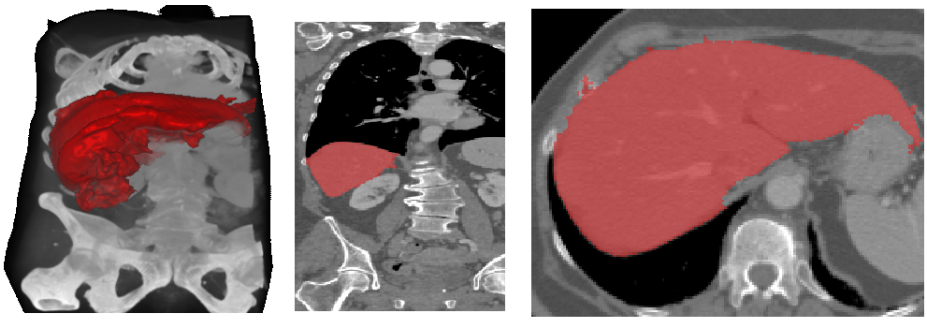
(a)



(b)

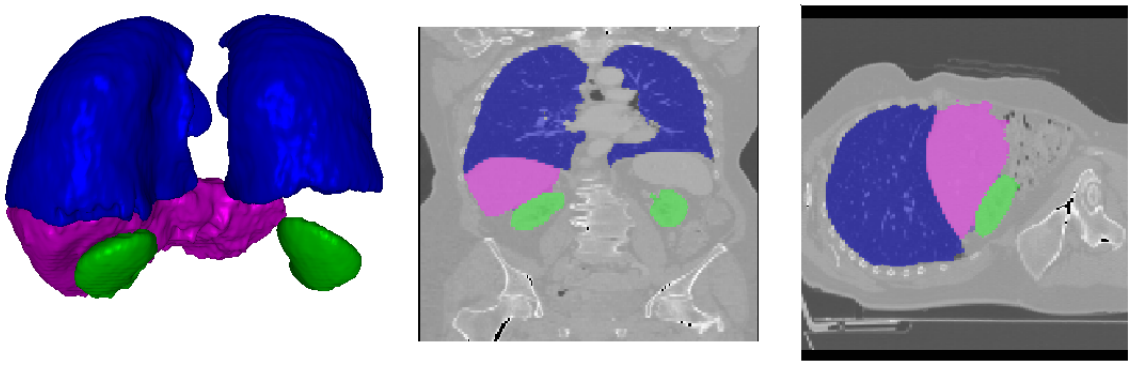


(c)

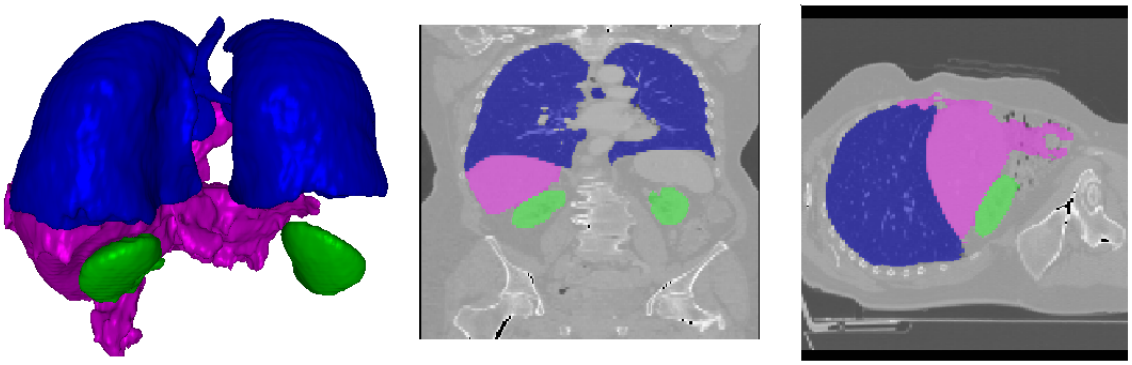


(d)

Figure 3.7: Liver Segmentation . (a) CT of the thorax and markers (256x256x128). (b) Approximate minimal surface computed by our method. (c) Exact minimal surface. (d) Marker-controlled watershed transform.



(a)



(b)

Figure 3.8: CT image of the thorax (256x256x128). (a) Surface Rendering and orthogonal cuts of the original data surimposed with the presented segmentation method. (b) Surface Rendering and orthogonal cuts of the original data surimposed with the marker controlled watershed segmentation.

3.4 Shape Constrained Curves and Surfaces

We highlighted, in the previous section, that the region graph offers a good trade-off between speed and precision for the computation of geodesics and minimal surfaces. We want to point out now that a region based approach is also richer than a pixel approach since a wide class of geometric functionals can be computed on each region of the watershed transform. We give now some precision about the constraints that may be added to the energy function given by equation 3.3.12.

A classical geometric constraint, widely used in the active contour community, is a smoothness constraint depending on the curvature of the curve or the surface to be computed. The graph cut method used on the pixel graph can unfortunately not handle such a constraint because the curvature or any smoothness term cannot be measured locally at the pixel level. Obviously, such a constraint can be added in our methodology since smoothness of each region contours can be easily computed. For instance curvature, or any measure of smoothness of the border between two adjacent regions, can be computed and used to add a shape constraint to the energy to be minimized. Let us consider the following edges weights :

$$w_{r_i,r_j} = |C_{(r_i,r_j)}|_R + \alpha.\kappa_{(r_i,r_j)} \ , \tag{3.4.1}$$

where $\kappa_{(r_i,r_j)}$ is a function of (r_i, r_j) measuring the smoothness of the contour $C_{(r_i,r_j)}$. α is a free parameter used to balance the effect of the smoothness term. For $\alpha = 0$, the edges weights correspond to the classical weights used to compute geodesics. A simple and robust measure of this smoothness can be for instance the morphological tortuosity of the contours. This factor is given by the following formula, for 2D contours:

$$\kappa_{(r_i,r_j)} = 1.0 - \left(\frac{\|p_1 - p_n\|}{|C_{(r_i,r_j)}|_\varepsilon} \right) \ , \tag{3.4.2}$$

where p_1 and p_n are the endpoints of the contour $C_{(r_i,r_j)}$, $\|p_1 - p_n\|$ is the Euclidean length between these end points, and $|C_{(r_i,r_j)}|_\varepsilon$ is the Euclidean length of the contour. $\kappa_{(r_i,r_j)}$ measures if the contour $C_{(r_i,r_j)}$ is close to a segment. In the case where $C_{(r_i,r_j)}$ is a segment, $\kappa_{(r_i,r_j)} = 0$, in any other case $0 < \kappa_{(r_i,r_j)} \leq 1$.

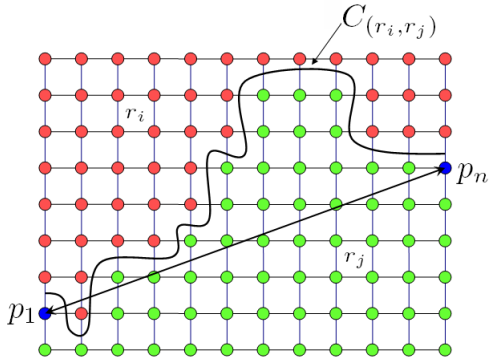
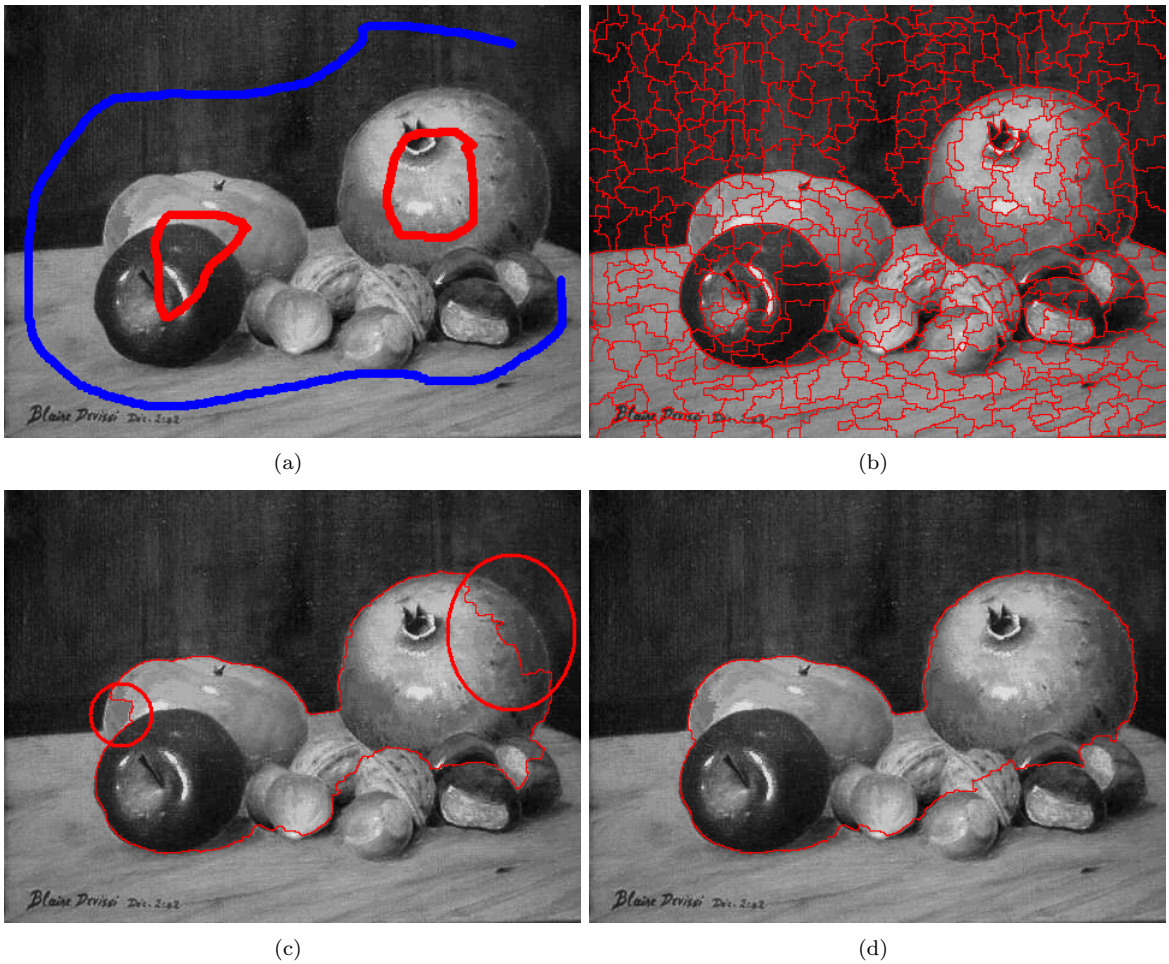


Figure 3.9: Morphological tortuosity for 2D curves. The smoothness of a curve is expressed as the ratio between the Euclidean length of the segment of its end points and the Euclidean length of the curve.

The defined edges weights penalize regions that do not have smooth boundaries. A minimal graph cut of the region graph should thus be composed of edges corresponding to regions of the low level segmentation having high contrast and smooth boundaries. Figures 3.11 and 3.10 compare some results using the classical geodesics and the regularized geodesics computed on a region graph, i.e. computed with the weights given by equation 3.4.1. The examples illustrate that the use of a smoothing parameter provides better perceptual results. Moreover only a region graph can bring this information into the classical graph cut framework. Note also that the extension of the method is also possible in 3D, by considering

curvature of region boundaries or a distance between the regions boundaries and the best fitting planes.

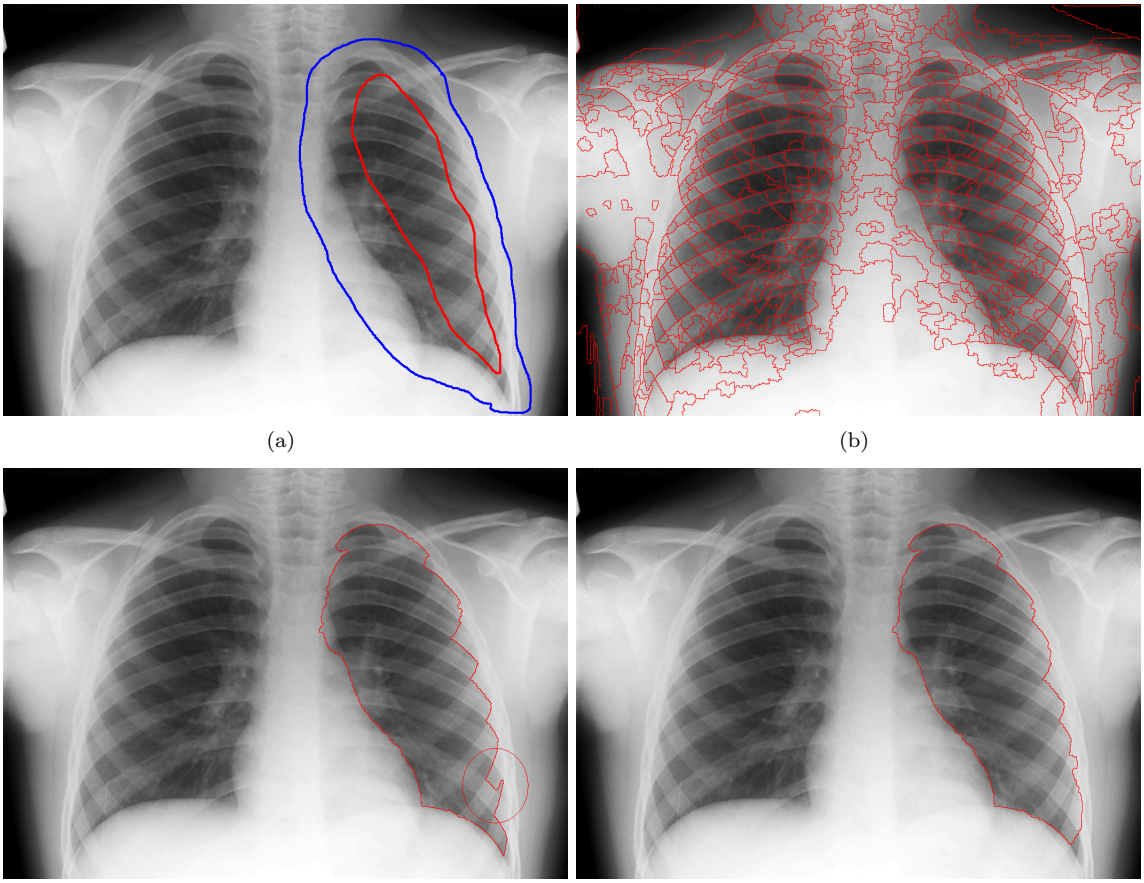
The use of a region adjacency graph is thus potentially richer than a pixel approach since it can take into account a larger class of geometric functionals. We have presented a method that considers the smoothness of the low level segmentation. First, the use of a region graph offers a good trade-off between speed and accuracy compared to the pixel graph, moreover it permits to introduce new geometric constraints in the segmentation which are impossible to consider on the pixel graph.



(a) (b)

(c) (d)

Figure 3.10: Natural image segmentation. (a) Original Image (480x360). (b) Markers. (c) Unsupervised watershed segmentation. (d) Geodesic computed on the region graph. (e) Geodesic regularized with the morphological tortuosity of the contours, $\alpha = 0.5$.



(a) (b)

(c) (d)

Figure 3.11: Chest X-ray, right lung segmentation. (a) Original Image (1058x769). (b) Markers. (c) Unsupervised watershed segmentation. (d) Geodesic computed on the region graph. (e) Geodesic regularized with the morphological tortuosity of the contours with, $\alpha = 2.0$.

3.5 Markov Random Fields

This section presents the image segmentation problem as a maximum a posteriori (MAP) estimation of a Markov random field (MRF). This model has been introduced by Besag [8] and later on by Geman and Geman in [40] for the optimal restoration of binary images. In this context, an image segmentation is seen as an optimal configuration of a Markov random field which has to be estimated from a given noisy image. This chapter recalls some definitions and useful properties of Markov random fields as well as the simplest model used for segmentation. This model is based on the well known Ising model issued from ferromagnetism [55]. Later on, this model has been extended to the multiple label case [54, 17].

We will also present some extensions that allow to use Markov random fields in combination with a morphological low level segmentation. This combination permits to speed up the use of Markov random fields without affecting practically the quality of the segmentation. As in the previous section, the optimal random field configuration is computed by a minimal graph cut [17, 48]. We will also present some extensions of the Ising model with contrast preserving models.

3.5.1 Definitions

Let us consider a graph $G = (V, E)$ composed of n nodes $V = \{v_1, ..., v_n\}$. The set of neighbor nodes of a given node v_i is denoted as N_i. The considered adjacency system leads to some specific configuration of connected nodes. These configurations are called the cliques of the graph, see figure 3.12.

Definition 3.5.1 (Clique)**.** *A clique in an undirected graph G is a set of nodes C such that for every couple of nodes in C, there exists an edge connecting the two nodes. The size of a clique is the number of nodes it contains.*

In this section, we study the optimal labeling of the graph nodes, such that each node v_i of the graph is assigned a label x_i in $L = \{0, 1\}$. Let us consider a field of random variables $(X_i)_{i \in V}$ where each random variable X_i takes a value x_i in the set L.

In this probabilistic context, a possible labeling is denoted as the event $\{X_1 = x_1, .., X_m = x_m\}$. $\{X_1 = x_1, .., X_m = x_m\}$ is abbreviated as $X = x$ where $x = (x_i)_{i \in V}$ is a specific configuration of the random field X. $Pr(X = x) = Pr(x)$ denotes the probability that the random field X takes the specific configuration x. $Pr(X_i = x_i) = Pr(x_i)$ denotes the probability that a given node i is assigned the label x_i. We denote by Ω the set of all possible configurations x.

Definition 3.5.2 (Markov Random Fields)**.** *X is a Markov Random Field if and only if it satisfies the following conditions:*

$$\begin{cases} \forall x \in \Omega, Pr(x) > 0, (positivity) \\ \forall p \in V, Pr(x_i | x_{V \setminus \{i\}}) = Pr(x_i | x_{N_i}), (markovianity) \ . \end{cases} \tag{3.5.1}$$

Each random variable X_i depends on other variables in the field X only through its neighborhood, $x_{N_i} = (x_i)_{i \in N_i}$.

Markov random fields are related to Gibbs random fields, which is a well known probabilistic model used in statistical physics to describe particles systems.

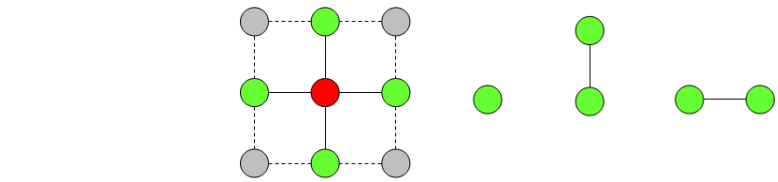
Definition 3.5.3 (Gibbs Distribution)**.** *The probability distribution $Pr(x)$ is a Gibbs distribution if and only if it can be written as:*

$$Pr(x) = \frac{1}{Z} exp(-U(x)) \ , \tag{3.5.2}$$

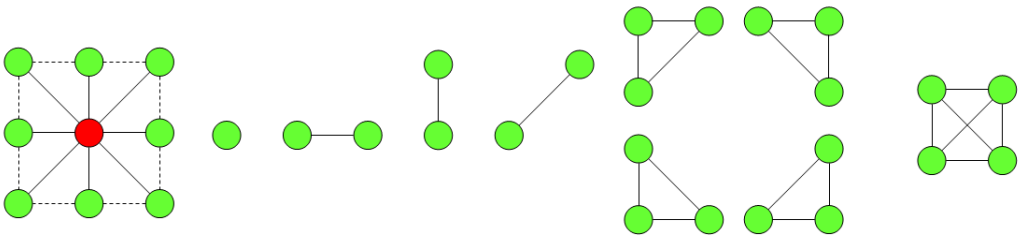
where $U(x) = \sum_{Cl \in N} V_{Cl}(x)$ is an energy function. Cl is the set of cliques associated to an adjacency system N. V_{Cl} is called the clique potential. $Z = \sum_{x \in \Omega} exp(-U(x))$ is called the partition function.

A Markov random field is characterized by its local property (the markovianity) whereas a Gibbs random field is characterized by its global property (the Gibbs distribution). However Hammersley and Clifford [52] have established the equivalence of these two types of random fields. A proof of this theorem can be found in [9] and [60].

Theorem 3.5.4 (Gibbs Markov Equivalence Theorem [52, 9, 60])**.** *X is a Markov random field on V with respect to N if and only if $P(X = x)$ is a Gibbs distribution.*



(a)



(b)

Figure 3.12: Cliques. (a) 4 neighbors adjacency system and associated cliques. (b) 8 neighbors adjacency system and associated cliques.

3.5.2 Maximum A Posteriori Estimation

Let $x = (x_i)_{i \in V}$, $x_i \in \{0, 1\}$, be a binary field. Assume we are given a noisy version $p = (p_i)_{i \in V}, p_i \in [0, 1]$, of this binary field. We can use the Bayesian inference framework to estimate the original vector x by maximizing the posterior probability:

$$Pr(x|p) = \frac{Pr(p|x).Pr(x)}{Pr(p)} \ . \tag{3.5.3}$$

The Bayesian model has become increasingly popular in the computer vision community. There are at least two reasons for that: first, the conditional probability $Pr(p|x)$ is easy to model; it represents the likelihood of the observation p given a state of the model x. This term is called the data term. Secondly, the Bayesian inference allows to integrate prior knowledge about the configuration x through the term $P(x)$. In our study we model the field x to be estimated as a Markov random field.

Problem Statement

Given an image I, we wish to obtain the configuration of labels x that maximizes the posterior probability $Pr(x|p)$. From equation 3.5.3:

$$Pr(x|p) \propto Pr(p|x).Pr(x) \ , \tag{3.5.4}$$

since $Pr(p)$ is constant for a given realization p.

It follows that the maximum a posteriori estimate x must maximize the following energy function:

$$\hat{x} = \underset{x \in \Omega}{\operatorname{argmax}} \left(Pr(p|x).Pr(x) \right) \ . \tag{3.5.5}$$

Prior Model $Pr(x)$

We first consider that the field of labels x is a Markov random field (MRF). According to the Hammerseley Clifford theorem, Markov random field x follows a Gibbs distribution:

$$Pr(x) \propto exp(-\sum_{Cl \in N} V_{Cl}(x)) \ . \tag{3.5.6}$$

For simplicity we restrict our attention to MRF's whose clique potentials involve only pairwise interaction:

$$Pr(x) \propto exp(-\sum_{i \in V} \sum_{j \in N_i} V_{i,j} \ , (\{x_i, x_j\})) \tag{3.5.7}$$

i.e. to adjacency systems whose cliques are composed of two nodes. In the following, the considered adjacency systems will thus be the 4-neighborhood in 2D and the 6-neighborhood in 3D. Let us also assume that the clique potential is not equal to zero only if neighbor nodes are assigned different labels:

$$V_{\{i,j\}}(\{x_i, x_j\}) = u_{i,j}.\delta(x_i \neq x_j) \ , \tag{3.5.8}$$

where $u_{i,j}$ is a function and δ is the indicator function: $\delta(x_i \neq x_j) = 1$ if and only if $x_i \neq x_j$ and $\delta(x_i \neq x_j) = 0$ otherwise.

Likelihood function $Pr(p|x)$**:**

We consider now the specific case where the conditional probabilities $Pr(p_i|x_i)$ are mutually independent and that moreover they only depend on the value x_i at the node i. Under these assumptions, the data term can be written as: $Pr(p|x) = \prod_i P(p_i|x_i)$. This assumption is verified if one assumes that the observed data p are corrupted by with independent and identically distributed noise components:

$$Pr(p|x) = \prod_i Pr(p_i|x_i) = exp(-\sum_{i \in V} -ln(Pr(p_i|x_i))) \ . \tag{3.5.9}$$

Energy Function:

The maximum a posteriori estimation of a Markov random field is finally stated as the following maximization problem:

$$\hat{x} = \underset{x \in \Omega}{\operatorname{argmax}} \left(exp \left(-\sum_{i \in V} -ln(Pr(p_i|x_i)) - \sum_{i \in V} \sum_{j \in N_i} u_{i,j}.\delta(x_i \neq x_j) \right) \right) \ . \tag{3.5.10}$$

This problem is also equivalent to the following minimization problem:

$$\hat{x} = \underset{x \in \Omega}{\operatorname{argmin}} \left(\sum_{i \in V} -ln(Pr(p_i|x_i)) + \sum_{i \in V} \sum_{j \in N_i} u_{i,j}.\delta(x_i \neq x_j) \right) \ . \tag{3.5.11}$$

From theorem 3.1.1, we know that this energy function can be minimized by using graph cuts if and only if $u_{i,j} \in \mathbb{R}^+$ and $-ln(Pr(p_i|x_i)) \in \mathbb{R}$, conditions that are clearly fulfilled. We use thus a minimal graph cut to compute the exact maximum a posteriori estimation $\hat{x}$. The next section describes how a graph can be built such that a minimal graph cut corresponds to the maximum a posteriori estimate $\hat{x}$.

Graph Cuts

Let us consider the following undirected graph $G_{s,t} = (V \cup \{s,t\}, E \cup E_s \cup E_t, W)$, composed of:

- $V = \{1, ..., n\}$, the set of nodes.
- $E = \{e_{i,j} | i \in V, j \in N_i\}$, the set of edges.
- two additional nodes, s and t.
- $E_s = \{e_{s,i} | i \in V\}$, the set of edges connecting the node s to all other nodes of the graph.
- $E_t = \{e_{i,t} | i \in V\}$, the set of edges connecting the node t to all other nodes of the graph.

A graph cut C separating s and t in the constructed graph implicitly describes two non connected sets of nodes by removing the sets of edges of the cut. One set S is connected to the source s and one set T is connected to the sink t. A graph cut C separating s and t in the constructed graph has the following weight:

$$L_1(C) = \sum_{i \in S} w_{i,t} + \sum_{i \in T} w_{s,i} + \sum_{i \in S, j \in T} w_{i,j} \ . \tag{3.5.12}$$

The edges weights of the graph G are finally given by table 3.3.

Edge	Weight	for
$w_{s,i}$	$-ln(Pr(p_i \| x_i = 0))$	$i \in V$
$w_{i,t}$	$-ln(Pr(p_i \| x_i = 1))$	$i \in V$
$w_{i,j}$	$u_{i,j}$	$i \in V, j \in N_i$

Table 3.3: Edges Weights for maximum a posteriori estimation.

Thus we have:

$$L_1(C) = \sum_{i \in T} -ln(Pr(p_i | x_i = 0)) + \sum_{i \in S} -ln(Pr(p_i | x_i = 1)) + \sum_{i \in S, q \in T} u_{i,j} \ . \tag{3.5.13}$$

$$L_1(C) = \sum_{i \in T} -ln(Pr(p_i | x_i = 0)) + \sum_{i \in S} -ln(Pr(p_i | x_i = 1)) + \sum_{e_{i,j} \in E} u_{i,j}.\delta(x_i \neq x_j) \ . \tag{3.5.14}$$

$$L_1(C) = \sum_{i \in V} -ln(Pr(p_i | x_i)) + \sum_{e_{i,j} \in E} u_{i,j}.\delta(x_i \neq x_j) \ . \tag{3.5.15}$$

Finally the weight $L_1(C)$ of a graph cut separating s and t is identical to the value of the a posteriori estimate of a Markov random field. Moreover there is a one to one correspondence between a graph cut in the graph G and a configuration f of labels. Thus the energy defined in equation 3.5.11 is exactly minimized by computing a minimal graph cut separating s and t.

3.5.3 Prior Models for Markov Random Fields

We now consider an image $I = (p_i)_{i \in V}, p_i \in [0,1]$ as an image corrupted with a white gaussian noise. The aim of this section is to restore the original binary data x.

The Ising model

The Ising model, named after the physicist Ernst Ising [55], is a model used in statistical mechanics to describe ferromagnetism. It can be represented on a graph where its configuration space is the set of all possible assignments of labels $L \in \{0, 1\}$ to each node of the graph. The Ising model uses the following clique potential:

$$V_{i,j}(\{x_i, x_j\}) = \beta.\delta(x_i \neq x_j) \ , \tag{3.5.16}$$

where δ is the indicator function and β is a positive real value. The Ising clique potential is an isotropic function since:

$$V_{i,j}(\{x_i, x_j\}) = V_{j,i}.(\{x_j, x_i\}) \ , \tag{3.5.17}$$

and is in general non-homogeneous since it can depend on the relative position of i and j on the lattice. The value β describes the energetic preference that neighbor nodes take the same value. It can also be seen as a regularization parameter since it penalizes the length of the sets of nodes labeled with different values simultaneously.

We also assume that the data are corrupted with a white gaussian noise such that the likelihood function can be written as:

$$Pr(p_i|x_i) = exp(-\frac{(p_i - \mu_{x_i})^2}{2\sigma^2}) \tag{3.5.18}$$

where μ_{x_i} is the mean value of the pixels expected to take the value x_i, and σ^2 is the variance of the gaussian distribution. Note that we consider that the classes x_i have the same variance but different means.

For the binary image restoration problem, we have to minimize the following energy function:

$$E(x) = \sum_{i \in V} \frac{(p_i - \mu_{x_i})^2}{2\sigma^2} + \sum_{i \in V} \sum_{j \in N_i} \beta.\delta(x_i \neq x_j) \ , \tag{3.5.19}$$

where the label x_i can take the values 0 or 1. We can solve this problem by computing a minimal graph cut as described in the previous section. The arcs capacities $w_{i,j}$ between two nodes of the graph G are set such that:

$$\forall i \in V, \ w_{s,i} = -ln(Pr(p_i|x_i = 0)) = \frac{(p_i - \mu_{(x_i=0)})^2}{2\sigma^2} \ , \tag{3.5.20}$$

$$\forall i \in V, \ w_{i,t} = -ln(Pr(p_i|x_i = 1)) = \frac{(p_i - \mu_{(x_i=1)})^2}{2\sigma^2}, \ \ i \in V \ , \tag{3.5.21}$$

$$\forall i \in V \ , \forall j \in N_i, \ w_{i,j} = \beta \ . \tag{3.5.22}$$

The graph cuts in the graph G are thus in one to one correspondence with the possible configurations of labels, and the value of the cut equals the energy function given by equation 3.5.19.

Example

This section illustrates the use of Markov random fields for the restoration of corrupted binary images. Figure 3.13 illustrates the restoration of a binary image corrupted with a gaussian noise with mean value 0 and of variance 1.0. The image values are scaled such that each pixel p_i takes a value in $[0, 1]$. The figures illustrate the role of the regularizing parameter β. The restored image gets smoother when the parameter β increases. For this example, the mean value in the region expected to take the label 0 is $\mu_{(x_i=0)} = 0.3$ and the mean value in the region expected to take the label 1 is $\mu_{(x_i=1)} = 0.7$.

The next section presents another prior function used to solve the problem of transition detection. The standard Ising model fails to detect high contrast transition in the field x when the smoothing parameter β is too high. This problem can be solved by specifying a data dependant prior function. The next section presents a contrast sensitive prior model for the pairwise interaction between neighbor pixels.

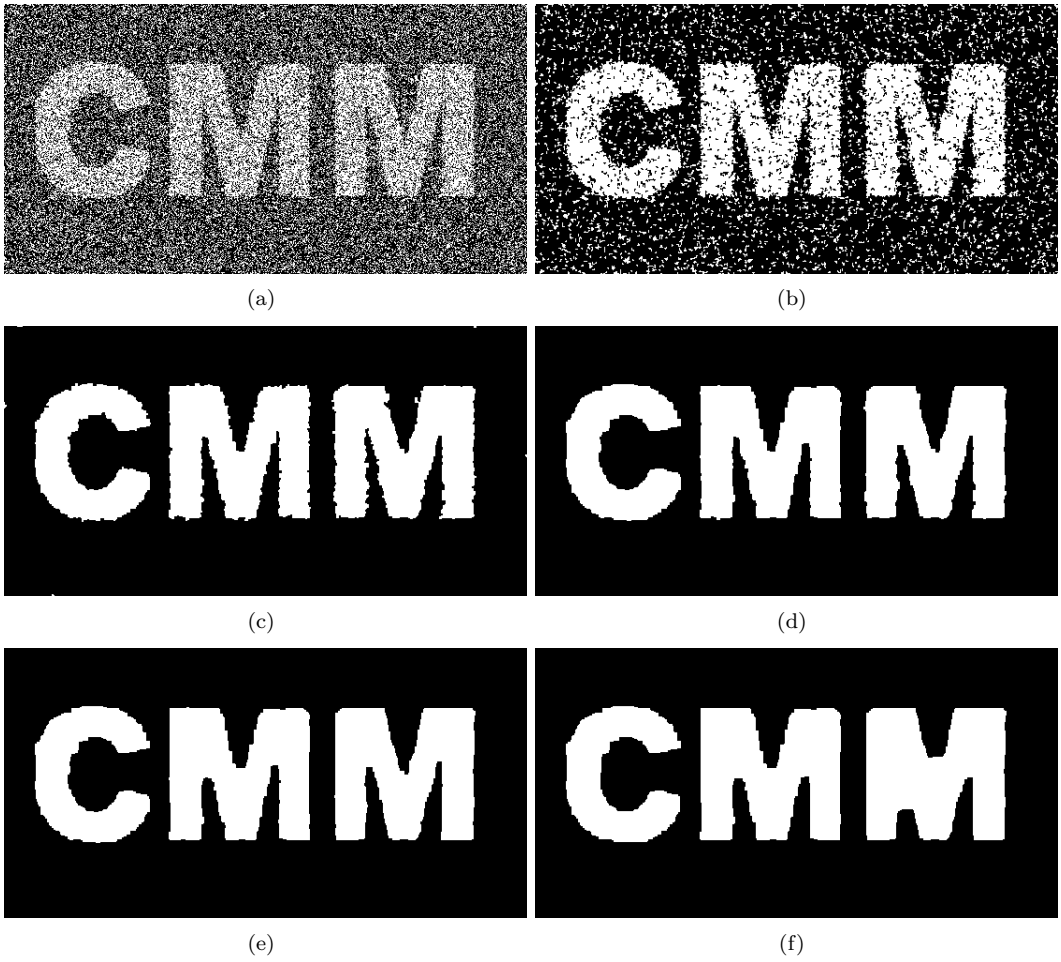


Figure 3.13: Binary image restoration. (a) Original Image (559x288). (b) Restored image with $\beta = 0.25$ (c) Restored image with $\beta = 0.5$ (d) Restored image with $\beta = 0.75$ (e) Restored image with $\beta = 1.0$ (f) Restored image with $\beta = 1.25$.

Contrast Sensitive Prior Model

The Ising model does not take into account values in the corrupted field p but it does only penalize the assignment of different labels to neighbor nodes in the labeling x. We now consider a model that preserves discontinuities in the field p when the contrast is sufficiently important and is a normal regularizer for low contrasted zones.

We define the following contrast sensitive prior:

$$V_{i,j}(\{x_i, x_j\}) = (\beta - \beta.(p_i - p_j)^n) \, .\delta(x_i \neq x_j) \tag{3.5.23}$$

The smoothness term is composed of a contrast sensitive term which will penalize the high gradients zones in the field p, and a regularization parameter β coming from the standard Ising model. The parameter n allows to vary the threshold for which the contrast sensitive term will dominate on the smoothing parameter. When n increases, the contrast sensitive term dominates the smoothing parameter only for very high gradient zones. This model allows us to detect high gradient zones in the field p and does not over-smooth the image in highly contrasted zones.

Example

This section illustrates the use of Markov random fields for the segmentation of images with a contrast sensitive prior function. Figure 3.14 illustrates the presented model for the segmentation of an image of blood cells. The image values are scaled such that each pixel p_i takes a value in $[0, 1]$. The figures illustrate the role of the regularizing parameter β. For this example, the parameter n was set to 0.5. The figure illustrates the comparison between the classical Ising model and the contrast sensitive model. The example shows the advantage of using contrast sensitive models for image segmentation. This model permits to take advantage of the gradient of the image and the model does not over-smooth the resulting segmentation. For the presented application, the mean value in the region expected to take the label 0 is $\mu_{(x_i=0)} = 0.3$ and the mean value in the region expected to take the label 1 is $\mu_{(x_i=1)} = 0.7$.

Markov random fields are powerful models for the restoration and the segmentation of images. Unfortunately using this method on the pixel graph is computationally costly and cannot always be used interactively on large datasets, such as 3D medical images. We present in the next section a novel method to compute fast image segmentations using the Markov random field model and a morphological low-level segmentation. This technique allows us to use interactively this model on large datasets.

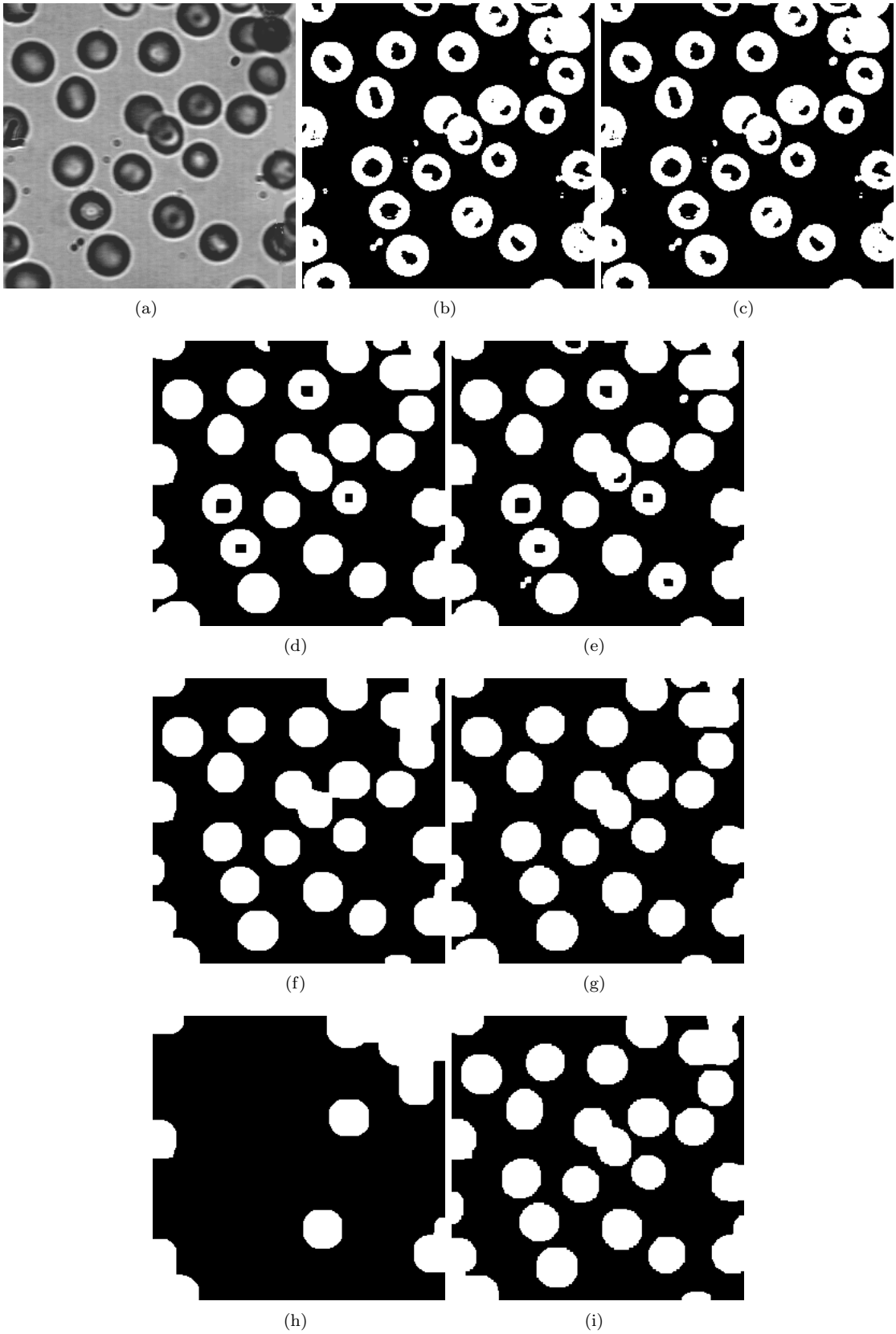


Figure 3.14: Image Segmentation with contrast sensitive (CS) model. (a) Original Image (272x265). (b) Ising prior, restored image with $\beta = 0.0$. (c) CS prior, restored image with $\beta = 0.0$. (d) Ising prior, $\beta = 0.5$ (e) CS prior, $\beta = 0.5$. (f) Ising prior, $\beta = 1.0$. (g) CS prior, $\beta = 1.0$. (h) Ising prior, $\beta = 1.5$. (i) CS prior, $\beta = 1.5$.

3.5.4 Approximate Maximum a Posteriori Estimation

We are now going to take into account our second assumption about the watershed transform: the unsupervised watershed transform of a natural image is composed of regions of homogenous grey level intensities. The use of a region graph instead of a pixel graph can thus be used without affecting the classification of images grey levels. This assumption permits us to model the image to be segmented as a Markov Random Field where each random variable corresponds to the mean value inside a region of the watershed transform.

Let us consider the region adjacency graph $G_R = (V_R, E_R, W_R)$ of a low level watershed transform. Instead of taking into account the grey level of each single pixel of the image, we consider the mean grey level inside each region of the watershed transform. For the binary image restoration problem, we have considered in the previous section the minimization of :

$$E(x) = \sum_{i \in V} -ln(Pr(p_i|x_i)) + \sum_{i \in V} \sum_{j \in N_i} u_{i,j}.\delta(x_i \neq x_j) \ . \tag{3.5.24}$$

We detail in the following sections how each term of this energy function can be efficiently approximated by using the region adjacency graph of the watershed transform.

Likelihood Function

Since we assume that each region of the watershed transform is composed of regions having homogenous intensities, we can approximate the likelihood term by:

$$\sum_{i \in V} -ln(Pr(p_i|x_i)) \approx \sum_{i \in V_R} -|r_i|.ln(Pr(\mu_{r_i}|x_i)) \ , \tag{3.5.25}$$

$$\mu_{r_i} = \frac{1}{|r_i|} \sum_{p_i \in r_i} p_i \ , \tag{3.5.26}$$

where $|r_i|$ is the number of pixels inside the region r_i.

The previous approximation becomes an equality if each pixel inside a region has exactly the same grey level. Alternatively we could have considered the region adjacency of a flat zones labeling of the image [88]. In this last case, regions are composed of connected pixels having the same grey levels. The previous approximation becomes thus a strict equality. However we detail in this chapter the use of the watershed transform, keeping in mind that its extension to other unsupervised low level segmentation is straightforward.

The previous approximation is motivated by the fact that the watershed transform of a real image contains regions of homogenous intensities. In other words, since the pixels inside a region of a watershed transform have all more or less the same grey levels, there is no need to take into account the variability of the grey levels inside a region. However, the grey levels inside a region could also be filtered to eliminate outliers and to smooth the grey level values.

Assuming that the data are corrupted with a white gaussian noise, the likelihood function can finally be written as:

$$Pr(\mu_{r_i}|x_i) = exp(-\frac{(\mu_{r_i} - \mu_{x_i})^2}{2\sigma^2}) \tag{3.5.27}$$

where μ_{r_i} is the mean grey level of the region r_i, and μ_{x_i} is the mean value of the pixels expected to take the value x_i.

Prior Function

The prior function is dependent on the boundaries properties of the labeling. We thus need to consider the boundaries between two regions to compute the regularizing term of the energy function $\left(\sum_{i\in V}\sum_{j\in N_i}\beta.\delta(x_i \neq x_j)\right)$. Let us recall that the boundary between two regions r_i and r_j of the watershed transform is defined by the following set of edges:

$$F_{(r_i,r_j)} = \{e_{m,n} \in E \mid m \in r_i,\ n \in r_j\} \ . \tag{3.5.28}$$

We assume now that all pixels inside a region of the watershed transform are assigned the same label x_i, thus the prior function only depends of the pixels lying in the boundaries between two regions:

$$\sum_{i\in V}\sum_{j\in N_i} u_{i,j}.\delta(x_i \neq x_j) \approx \sum_{i\in V_R}\sum_{j\in N_{r_i}} |F_{(r_i,r_j)}|.u_{i,j}.\delta(x_i \neq x_j) \ . \tag{3.5.29}$$

where N_{r_i} is the set of neighbor regions of the region r_i and $|F_{(r_i,r_j)}|$ is the number of edges of $F_{(r_i,r_j)}$.

Moreover, assuming that the prior function is a contrast sensitive model, the regularizing function can be written as:

$$u_{i,j} = \left(\beta - \beta * (\mu_{r_i} - \mu_{r_j})^n\right) \ . \tag{3.5.30}$$

Graph Cuts

Finally, we have to minimize the following energy function:

$$E(x) = \sum_{i\in V_R} |r_i| \frac{(\mu_{r_i} - \mu_{x_i})^2}{2\sigma^2} + \sum_{i\in V_R}\sum_{j\in N_{r_i}} |F_{(r_i,r_j)}|.u_{i,j}.\delta(x_i \neq x_j) \ . \tag{3.5.31}$$

As shown previously, we can minimize the energy function by computing a minimal graph cut. The arcs capacities w_{r_i,r_j} between two nodes of the region adjacency graph G are set according to the weights given in table 3.4.

Edge	Weight	for
$w_{s,i}$	$-ln(Pr(\mu_{r_i}\|x_i = 0)) = \frac{(\mu_{r_i} - \mu_{(x_i=0)})^2}{2\sigma^2}$	$r_i \in V_R$
$w_{i,t}$	$-ln(Pr(\mu_{r_i}\|x_i = 1)) = \frac{(\mu_{r_i} - \mu_{(x_i=1)})^2}{2\sigma^2}$	$r_i \in V_R$
$w_{i,j}$	$\|F_{(r_i,r_j)}\|.u_{i,j}$	$r_i \in V_R, j \in N_{r_i}$

Table 3.4: Edges Weights for approximate maximum a posteriori estimation.

3.5.5 Application of Markov Random Fields to Microstructure Segmentation.

The aimed application of the Markov random field models is the segmentation of 3D micro-tomography images of granular material. The images are provided by CEG Gramat (Délégation Générale pour l Armement) using a Skyscan X-ray microtomograph. These microtomography images have an 10 μm isotropic resolution. The aim of the application is to extract the two phases of the material. Figure 3.15s compares the segmentation results of the Markov random field model solved on the pixel graph and the region graph. A contrast sensitive prior model was used to compute the presented results with $\beta = 0.01$, $\mu_{(x_i=0)} = 0.25$, $\mu_{(x_i=1)} = 0.75$, and $\sigma = 0.2$.

Table 3.5 compares the graphs sizes using our method and the graph cuts approach at the pixel level. The Nodes reduction factor and the edges reduction factor are respectively equal to $\frac{|V|}{|V_R|}$ and $\frac{|E|}{|E_R|}$. The

results show that our method largely reduces the graphs sizes.

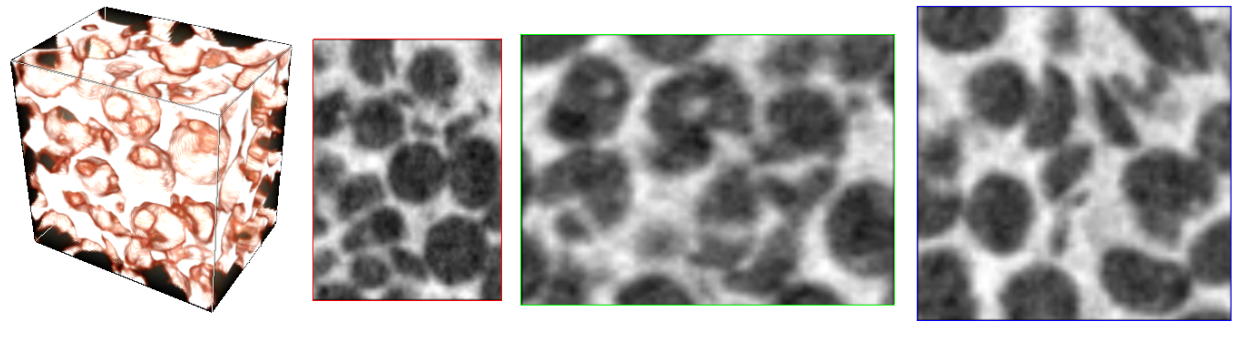
Table 3.6 also illustrates the computation time needed by the methods presented above: Markov random field solved on the pixel graph and the region graph. The results shows that our method reduces drastically the computation time and allows to segment images that are too large to fit in the memory, using a pixel approach. Moreover our method does not seem to affect the results quality, as illustrated in figure 3.15.

Image	Size	Nodes Reduction Factor	Edges Reduction Factor
Image 1	69x69x100	16.6	7.9
Image 2	166x171x104	40.3	18.5
Image 3	250x250x77	8.9	4.6

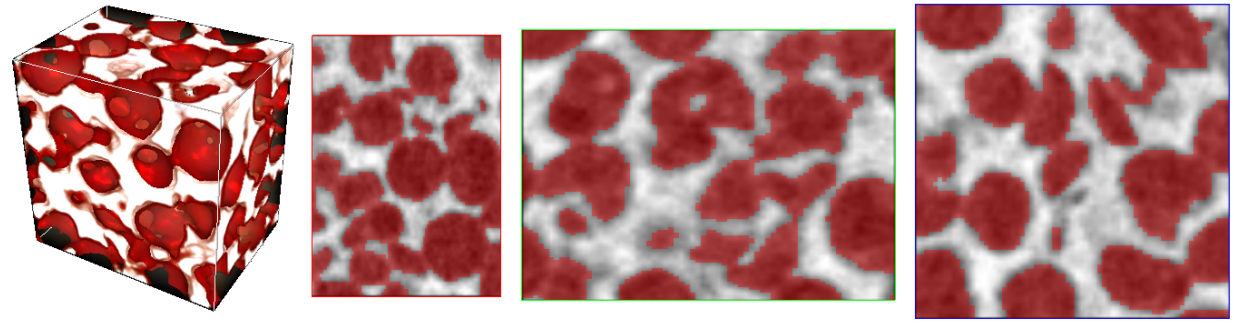
Table 3.5: Comparisons of Graph Sizes.

Image	Size	Our Method	Exact Markov Random Fields
Image 1	69x69x100	1.36 sec.	8.01 sec.
Image 2	166x171x104	34,9 sec.	not enough memory
Image 3	250x250x77	25.8 sec.	not enough memory

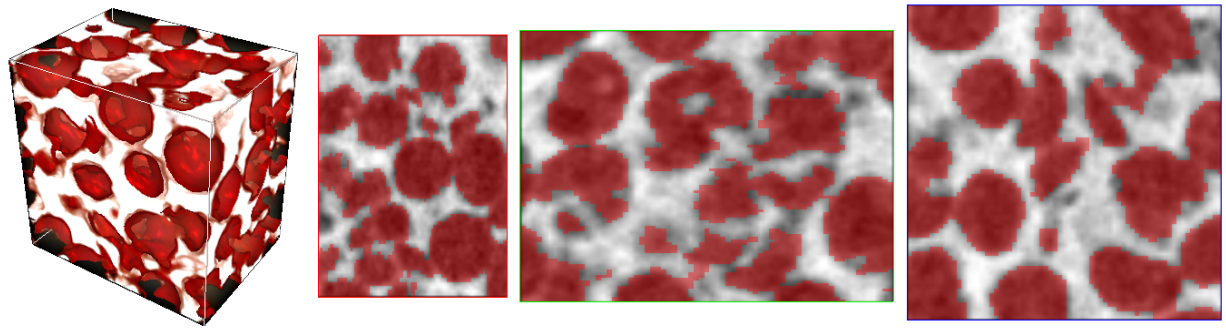
Table 3.6: Comparisons of computation time.



(a)



(b)



(c)

Figure 3.15: Microstructure Segmentation. (a) Original image (69x69x100). (b) Segmentation of the pixel graph. (c) Segmentation of the region graph.

3.5.6 Adding Statistical Properties

Using the region adjacency graph instead of the pixel adjacency graph can be advantageous in some situations. A wide class of statistical properties can be computed on each region of the watershed transform such that additional constraints can be added to the energy function to be minimized. For instance, it is possible to take into account the distribution of the grey levels inside a region of the watershed. It is thus possible to define the foreground and background properties through grey levels histograms and use distances between histograms instead of simple grey level differences. The use of the region adjacency graph permits also to filter possible outliers that could lie inside a region. The use of a region adjacency graph is thus potentially richer, since it can take into account more properties than simply the grey level of a single pixel.

3.6 Conclusion

This chapter described the use of two common mathematical models for image segmentation: the Markov random field model and the minimal curves and surfaces model. These two models are fundamental tools for various image segmentation problems, and especially for medical images. We have detailed a technique to speed up these methods without affecting the quality of the results, by using a region adjacency graph of a watershed transform instead of the pixel graph. We have experimentally shown that this technique considerably reduces the computational cost of graph cuts methods and that the provided results do not suffer from our approximations. These methods are illustrated on real applications in the next chapter.

We have also shown that this methodology can handle a larger class of energy functions, compared to energy functions defined by properties at the pixel level. The energy functions defined on regions are richer than properties defined on a single pixel. This interesting property of region graphs should be further investigated in the future to design energy functions that can take into account high level and prior information such as explicit shape constraints models. It would also be possible to combine geometric and statistical properties of regions to design energy functions. The extension of classical mathematical models on region graphs is still under-exploited and should lead to powerful image segmentation methods in the future.

4

Application to Medical Image Segmentation

In the last years, the use of graph based techniques have shown a great success for a large field of computer vision problems such as image restoration [40, 54, 17], stereo vision [14], image segmentation [13] and energy minimization [16]. Researchers have also been focused on the development of automatic and semi automatic graph based image segmentation techniques [5, 116, 10, 92, 117, 118, 68, 57, 97, 112]. These different graph based frameworks have permitted to establish new techniques for hierarchical segmentation, the extraction of perceptual properties of images, geometric based features detection, etc. Unfortunately medical imaging applications can be seldom solved automatically and require the supervision of a specialist such as a radiologist. The difficulties linked to medical imaging applications are numerous and mainly due to:

- the noise due to the imaging technologies,
- the low contrasted structures resulting from the imaging technologies,
- the variability of shapes, sizes, and characteristics of anatomical structures,
- the large inter-patient shape variability,
- the large intra-patient shape variability,
- the large deformations due to pathologies.

Image segmentation is thus one of the most challenging problems of medical image analysis. Moreover, image segmentation is only a first step in the work flow of a clinician. Relevant segmentations are needed for quantitative measurements, diagnosis, treatment planning, and modeling of anatomical structures. In these conditions, a fully automatic approach is not always suitable for the radiologists since the provided results must be checked and may be corrected. That's why we are mainly interested in interactive image segmentation since it can potentially provide robust results in various cases by using some prior information provided by a specialist. We present now some medical applications of the methods developed in the previous chapter, which combines morphological and graph cuts segmentation. Note also that other studies have been undertaken to estimate the quality of the watershed transform for contours detection [111], and interactive segmentation [99, 98]. Our methods have all been implemented within a 3D visualization software, which is presented in section 4.1.

Section 4.2 illustrates qualitatively a first application of graph based image segmentation to medical images. We consider in this section the segmentation of 3D MRI of the heart for surgery planning and for the generation of models for virtual surgery. Computer aided surgery is one of the great topics of interest for the medical imaging community. This new research field proposes challenging and promising

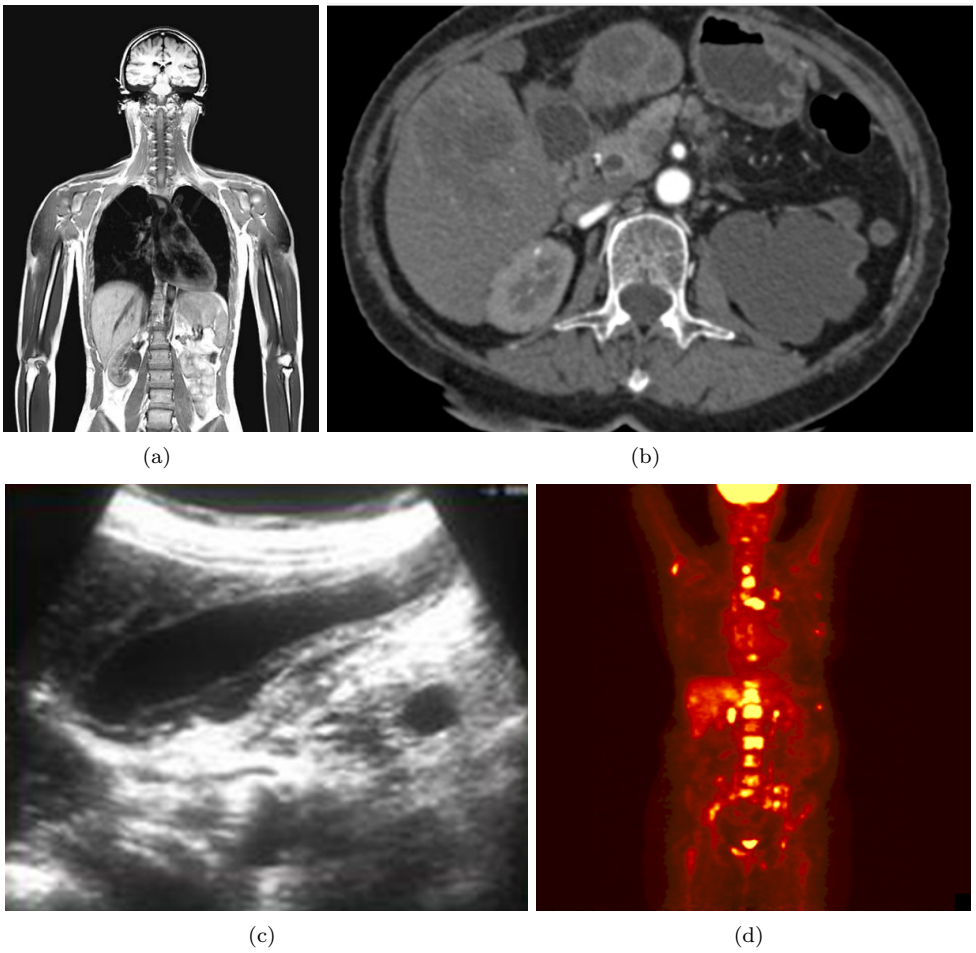


Figure 4.1: Imaging technologies used in clinical routine. (a) Magnetic Resonance Imaging (MRI). (b) Computed Tomography (CT). (c) Echography. (d) Positron Emission Tomography (PET).

problems. We detail in this section our collaboration with University of Aarhus, that aims at creating patient specific models for surgery planning and simulation. We detail in this section the use of minimal surfaces for the creation of these models.

Section 4.3 illustrates another application of minimal surfaces to medical images segmentation. We consider in this section the segmentation of 3D computed tomography (CT) of lung tumors for radiotherapy planning. This project is developed in collaboration with "Institut Gustave Roussy". The imaging department of this institute provided the data and helped us creating the software presented in the first section. We detail, for this application, the use of minimal surfaces. We also present some recent results for the tracking of lung tumors in time series of 3D CT images.

Finally Section 4.4 presents a quantitative study of our segmentation tools for the segmentation of liver tumors. We illustrate in this section an application of minimal surfaces and Markov random fields for the interactive segmentation of liver tumors in CT images. We detail how Markov random fields can be used to delineate tumors boundaries in 3D CT images and compare the segmentation results with hand made segmentation done by experienced radiologists. This comparisons prove that the proposed approach is particularly adapted to tumor segmentation problems. We present in this section the results of a quantitative study undertaken to prove the reliability of both our software and methods for medical imaging applications. We also present a comparative study of our method with some recent algorithms.

4.1 Graphical User Interface

We have developed a graphical user interface dedicated to 3D medical image segmentation and visualization. The software is entirely developed in Python and C++ based on the VTK library [1] and the image analysis library developed at the Center for Mathematical Morphology [2]. Our software allows the user to explore a highly detailed view of the data-set for easy interpretation. Visualization is particularly important for segmentation validation purposes. Data sets can be explored through 2D orthogonal cuts of the 3D volume and 3D rendering of the whole image, or of some user specified sub-volume, as illustrated in figure 4.2.

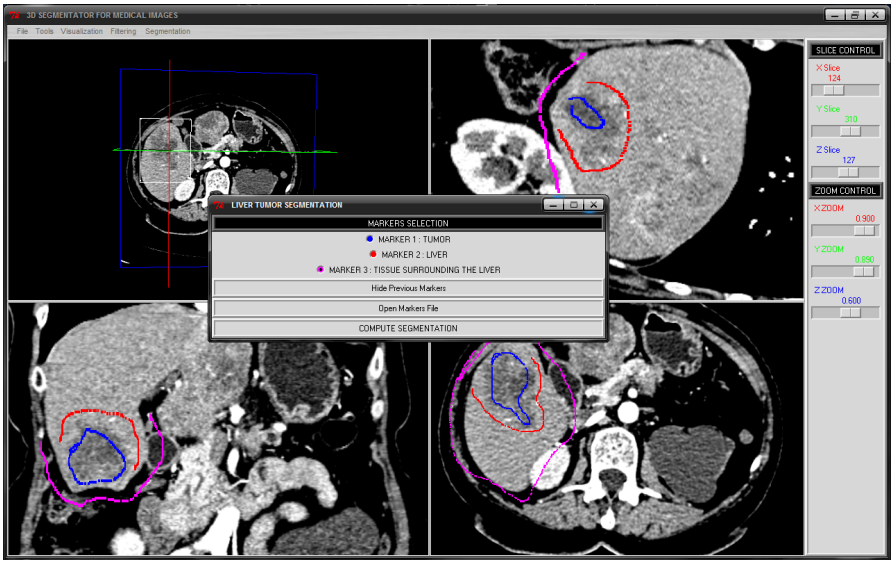
(a)

Figure 4.2: Snapshot of the graphical user interface of our software. Visualization of a 3D CT of the liver.

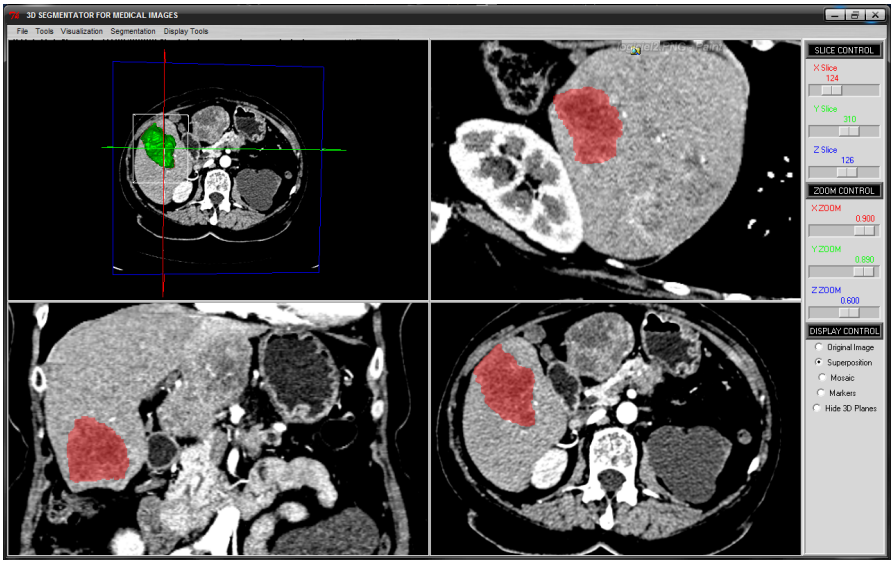
The user can interactively provide the markers needed for the segmentation algorithms by drawing on slices of the 3D image, as illustrated in figure 4.3. He can visualize the segmented image as a set of surfaces, one surface for each object. Detailed surfaces are extracted by using the marching cubes algorithm [114]. Users can also overlay the surfaces on the original image. It is possible to superpose 2D slices of both segmented and original image as well as a 3D surface rendering of the segmented data set. Combination of all this visualization methods allows an easy and fast interpretation of the segmentation result. Complex morphologies can therefore be explored easily and interactively.

[1] www.vtk.org, The Visualization ToolKit (VTK) is an open source, freely available software system for 3D computer graphics, image processing, and visualization.

[2] http://cmm.ensmp.fr/Morph-M/, Morph-M is the result of the work of several researchers at the Centre for Mathematical Morphology. Morph-M provides a rich environment for the development of image processing algorithms. Most of current research projects at the CMM are based on Morph-M.



(a)



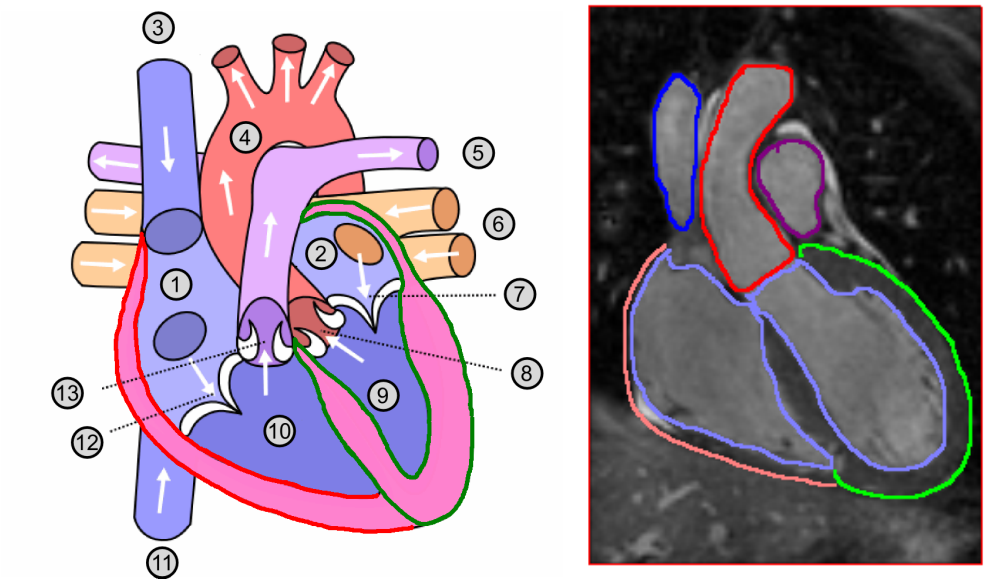
(b)

Figure 4.3: Snapshot of the graphical user interface of our software. (a) Interactive marker drawing. (b) Visualization of a segmented data set.

4.2 Cardiac MRI Segmentation

4.2.1 General Description

One application of our segmentation model is the required preprocessing for open heart surgery simulation. This specific simulator is currently being evaluated for preoperative planning and teaching in congenital heart disease [95]. The aim of surgery simulation is to offer patient specific models so that the surgeons can define the best approach for a given patient. In this context, a patient specific model has to be built from a MRI image. The model should reflect patient morphology as well as defects of his heart. The chosen strategy is here to use the image segmentation result to build a volumetric model of the heart. In a first step, our aim is to obtain a volumetric segmentation of the heart muscle (the myocardium). A diagram of the heart, as well as a slice of a 3D MRI are shown in figure 4.4.



(a) (b)

Figure 4.4: (a) Heart Diagram. The heart muscle is represented by the areas outlined in red and green. (1) Right atrium. (2) Left atrium. (3) Superior Vena Cave. (4) Aorta. (5) Pulmonary Artery. (6) Pulmonary Vein. (7) Mitral valve. (8) Aortic valve. (9) Left ventricle. (10) Right ventricle. (11) Inferior Vena Cave. (12) Tricuspid Valve. (13) Pulmonary Valve. (b) A slice of a 3D Heart MRI superposed with a hand made segmentation. The segmentation highlights the aorta (in red), the superior vena cave (in dark blue), the pulmonary artery, the right and left chamber (light blue) and the heart muscle (green and light red).

The heart wall consists of three layers: the epicardium, the myocardium, and the endocardium. The epicardium is a thin outer layer which gives the surface of the heart a smooth texture. The endocardium is the smooth inner lining of the heart and is continuous with the large blood vessels to which the heart connects. The myocardium makes up the bulk of the heart and is responsible for its pumping action. The myocardium is made up of strong cardiac muscle fibres which are connected by electrical synapses. The myocardium is clearly visible in 3D MRI whereas epicardium and endocardium are generally too thin to be clearly detected.

4.2.2 Data

The imaging modality in this application is an isotropic 3D MRI as illustrated in figure 4.5. This imaging technology provides 50 to 120 slices covering the whole heart with a voxel resolution ranging from $1.5mm^3$

to $2.0mm^3$ depending on the heart rate and size of the patient.

The images resolution does not permit to detect all the structures of the heart as illustrated in figure 4.4. The whole myocardium is not always visible on 3D MRI datasets, especially along the right chamber, because the myocardium is too thin in this area. In our application, we will thus segment the left part of the myocardium which is outlined in green in figure 4.4. The other parts of the myocardium have been obtained with manual segmentations.

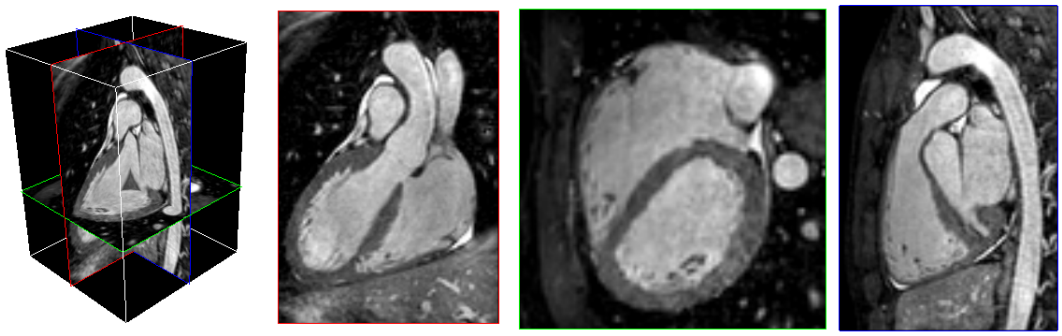


Figure 4.5: Heart MRI. Original 3D MRI image of the whole heart.

4.2.3 Segmentation Strategy

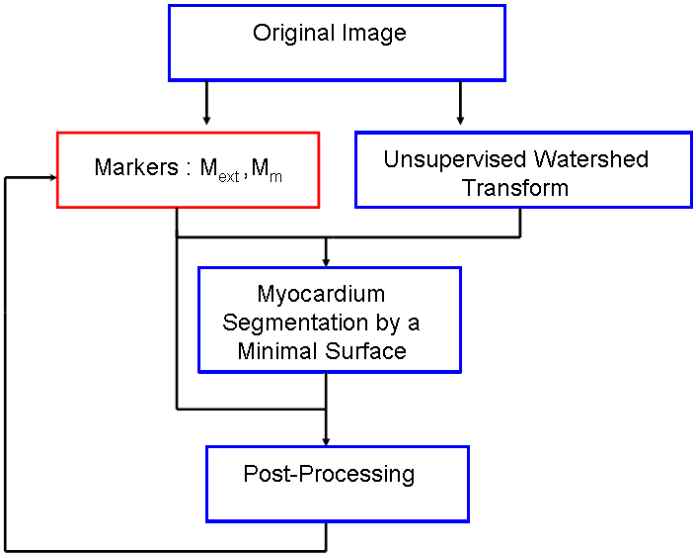


Figure 4.6: Myocardium Segmentation by Minimal Surfaces. The information provided by the user is outlined in red. Automatic steps are outlined in blue.

The myocardium segmentation is realized through the computation of a minimal graph cut of the region adjacency graph, because this technique should permit to delineate the low contrasted boundaries of the lower part of the myocardium. In this scenario, the user has to interactively specify the location of the myocardium as well as a marker to specify the background, i.e. the objects that have not to be segmented. The segmentation protocol is illustrated in figure 4.6. Additionally the user can add or delete markers to refine the segmentation results. A post-processing step is also proposed to smooth the segmentation result. This post-processing step consists in a morphological opening of the segmentation.

4.2.4 Implementation Details

The region adjacency graph is obtained from the watershed transform computed from all minima of the morphological gradient of the original MRI image using Meyer's algorithm based on hierarchical queues [74]. Both minima of the gradient and the watershed transform are computed using the 6-neighborhood adjacency system. From this first low-level segmentation, a region adjacency graph is extracted and used for the next optimization steps.

The user specified markers are used to compute a minimal graph cut separating the markers specifying the myocardium and the external tissues. We denote the markers that specify the myocardium as the set of regions M_m. The markers specifying the tissues surrounding the myocardium are denoted by M_{ext}.

We refer the reader to chapter 3 for details about the segmentation method using minimal surfaces. Let us recall that $F_{(r_i,r_j)}$ is defined as the set of edges of the pixel graph connecting two regions r_i and r_j of the low-level watershed segmentation:

$$F_{(r_i,r_j)} = \{e_{m,n} \in E \mid m \in r_i,\ n \in r_j\} \ . \tag{4.2.1}$$

In the following, we consider the strictly positive and decreasing function g used as an edge indicator for the a minimal surface:

$$g(\|\nabla I(p)\|) = \left(\frac{1}{1+\|\nabla I(p)\|}\right)^k \ . \tag{4.2.2}$$

In our application to the myocardium segmentation the parameter was set to $k = 2$.

The edges weights of the region adjacency graph $G_R = (V_R, E_R, W_R)$ are then set such that the weight of a graph cut equals the energy function of a surface (the integral of the image gradient along the surface that it implicitly defines:

$$w_{r_i,r_j} = \sum_{(e_{m,n} \in F_{(r_i,r_j)})} (g(\|\nabla I(p)\|)) \ . \tag{4.2.3}$$

The myocardium boundaries are finally extracted by computing a minimal graph cut of the region adjacency graph with weights given by equation 4.2.3. The minimal cut is obtained on the region adjacency graph with two additional nodes s and t, respectively connected to the markers of the myocardium and the markers of the external tissues. The edges weights of the graph are summarized in table 4.1.

Edge	Weight	for
w_{s,r_i}	$+\infty$	$r_i \in M_m$
$w_{r_i,t}$	$+\infty$	$r_i \in M_{ext}$
w_{r_i,r_j}	$\sum_{(e_{m,n} \in F_{(r_i,r_j)})} (g(\|\nabla I(p)\|))$	$r_i \in V_R, r_j \in N_{r_i}$

Table 4.1: Edges Weights for Approximate Minimal Surfaces.

4.2.5 Results and Comparisons

Figure 4.8 illustrates the presented segmentation method (Figure 4.8(b)) and compares it to the classical marker-controlled watershed segmentation (see figure 4.8(d)) and the minimal surface computed with the technique proposed by Boykov et al. in [14] (see figure 4.8(c)). Our method outperforms the marker-controlled watershed segmentation [75] and produces approximately the same segmentation results as the graph cut method used on the pixel graph and proposed by Boykov et al. in [14]. Our strategy permits to extract the boundaries of the myocardium precisely. Moreover, the use of minimal surfaces is not sensitive to the low contrasted areas present in the lower part of the myocardium. This low contrasted area is especially problematic for the marker controlled watershed transform whose results show a large leak at this location.

4.2.6 Multi-Labels Segmentation

Finally, the proposed technique is used to provide multi-label segmentations. In this scenario, it is possible to separate the myocardium from the blood pools, i.e. the right and left chambers. In this case the user must provide specific markers for each of these structures, as illustrated in figure 4.9.

The user specified markers are used to compute a minimal multi terminal cut separating the markers specifying the myocardium, the left chamber, the right chamber and the external tissues. We denote the markers that specify the myocardium as the set of regions M_m, the markers that specify the left and right chamber respectively as the set of regions M_l and M_r. The markers specifying the tissues surrounding the heart are denoted by M_{ext}. The edge weights of the graph are finally summarized in table 4.2.

Edge	Weight	for
w_{t_1,r_i}	$+\infty$	$r_i \in M_m$
w_{t_2,r_i}	$+\infty$	$r_i \in M_l$
w_{t_3,r_i}	$+\infty$	$r_i \in M_r$
w_{t_4,r_i}	$+\infty$	$r_i \in M_{ext}$
w_{r_i,r_j}	$\sum_{(e_{m,n} \in F_{(r_i,r_j)})} (g(\|\nabla I(p)\|))$	$r_i \in V_R, r_j \in N_{r_i}$

Table 4.2: Edges Weights for Approximate Multi Minimal Surfaces.

Then, we consider the segmentation as a minimal multi-terminal cut separating the markers, as illustrated in 4.9. The proposed strategy outperforms the marker controlled segmentation and provides slightly similar results than the minimal surfaces described by Boykov et al. in [14].

4.2.7 Conclusion

The presented methods provide stable and relevant segmentation results in the case of 3D MRI segmentation of the heart. These segmentation techniques have been studied in the context of surgery planning and simulation in collaboration with the Center for Advanced Visualization and Interaction of the University of Aarhus, Denmark. Joint work with J. Mosegaard and T.S. Sørensen have permitted to study the possibility of using graph based techniques for the creation of suitable models for virtual surgery [96, 106, 107].

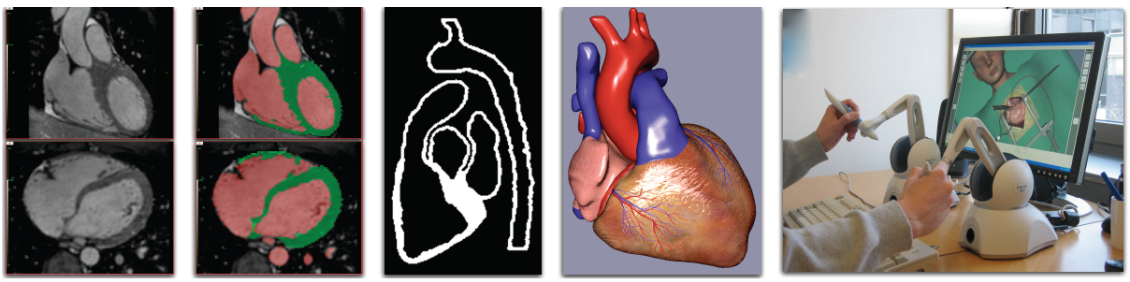
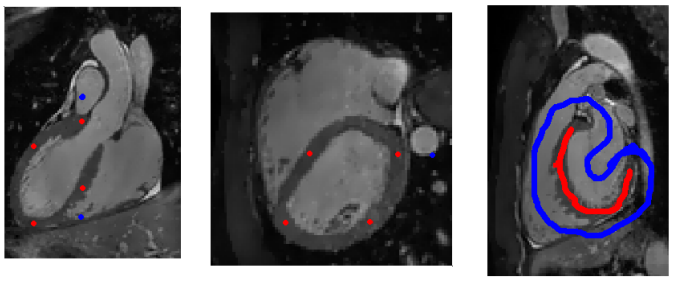
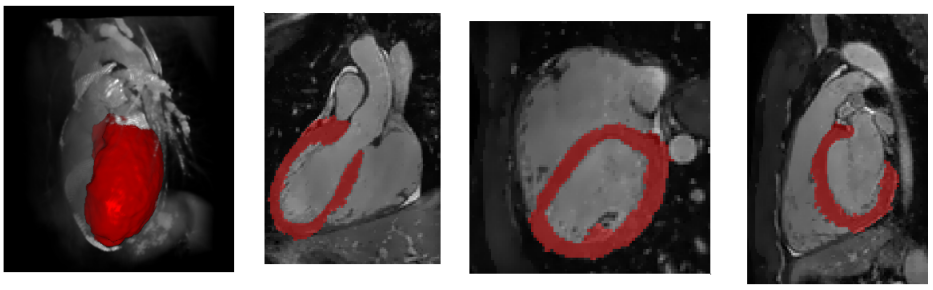


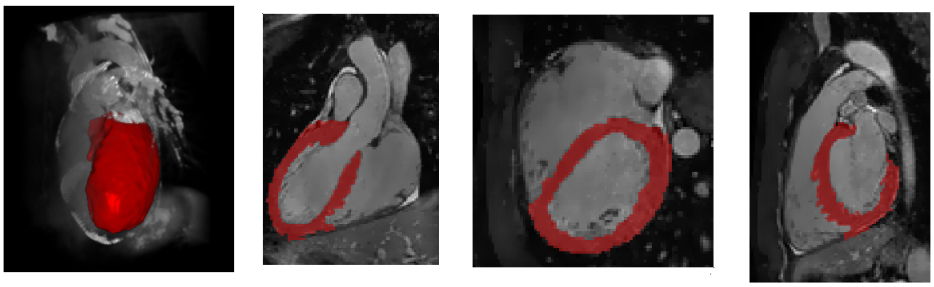
Figure 4.7: Obtaining suitable patient specific models for surgery simulation and planning [96].



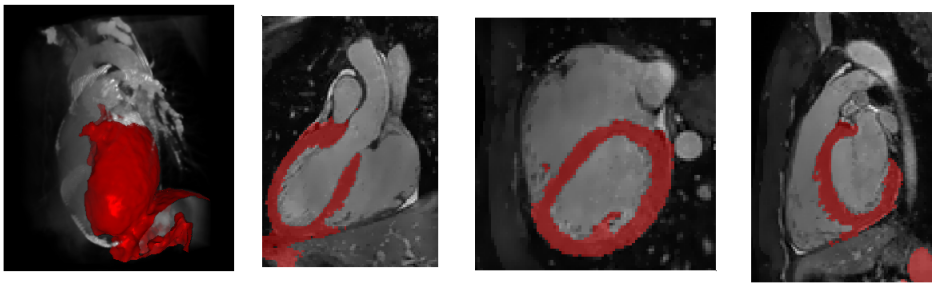
(a)



(b)



(c)



(d)

Figure 4.8: Myocardium Segmentation. (a) Heart MRI superposed with markers (86x128x90). (b) Approximate minimal surface computed by our method. (c) Exact minimal surface. (d) Marker-controlled watershed transform.

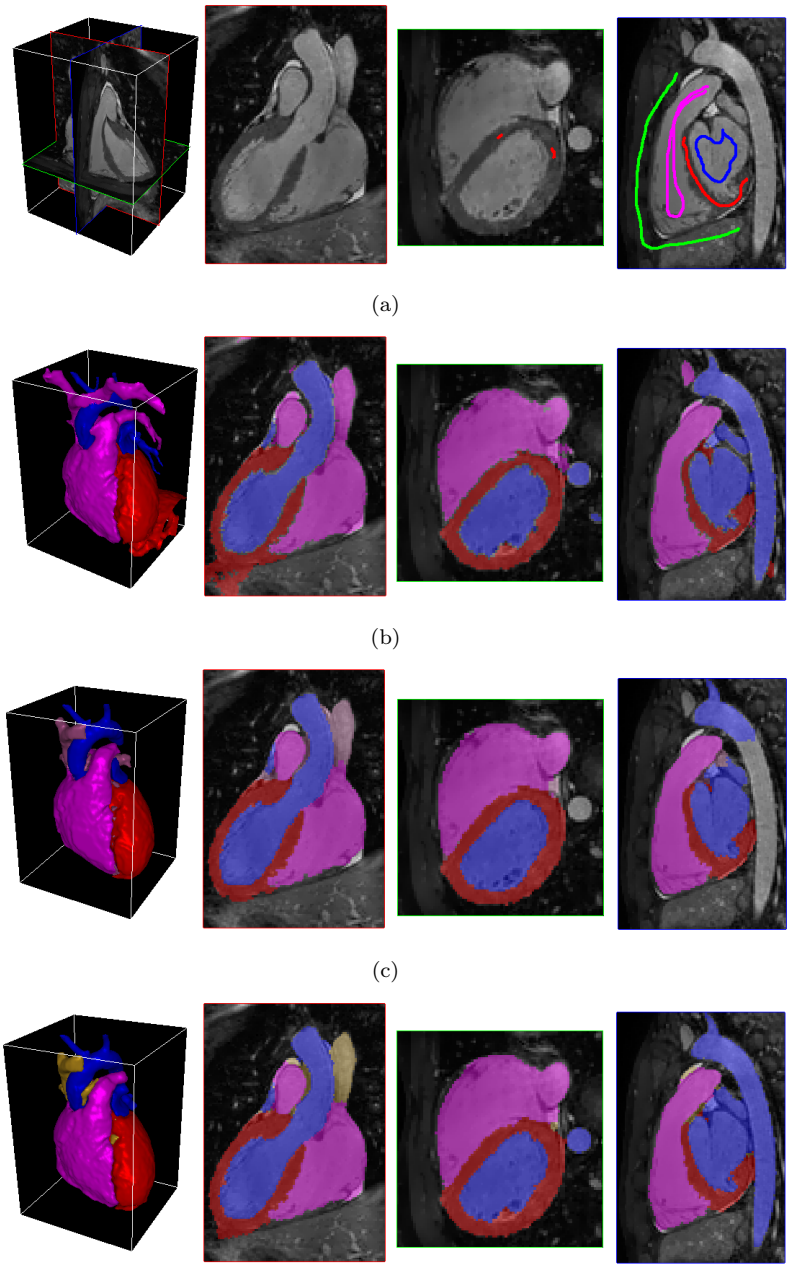


Figure 4.9: Multi-terminal cut segmentation. (a) Markers. (b) Marker controlled watershed. (c) Multi terminal cut on the pixel graph. (d) Multi terminal cuts on the region adjacency graph. The graph cut computed on the region adjacency graph detects the whole aorta whereas it only detects a part of it by using the pixel graph. This difference is due to a large region that covers the aorta in the low-level segmentation.

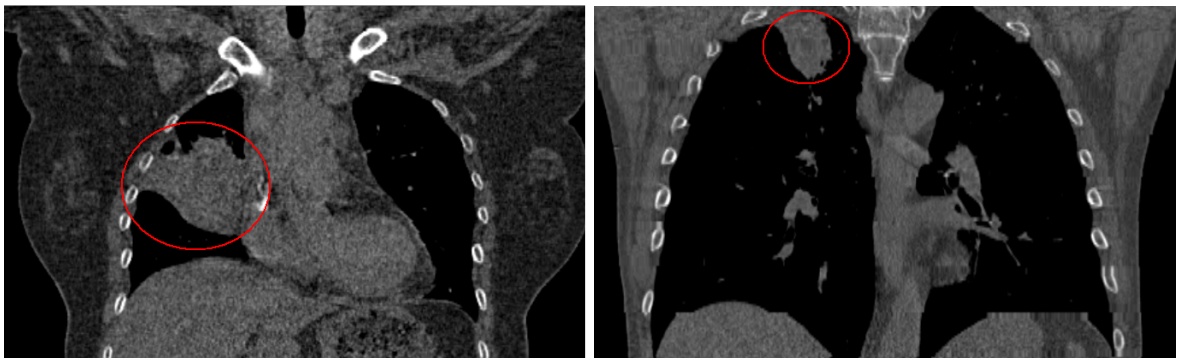
4.3 Lung Tumors Segmentation

Lung cancer is a disease of uncontrolled cell growth in tissues of the lung. This growth may lead to metastasis, invasion of adjacent tissue and infiltration beyond the lungs. Lung cancer is the most common cause of cancer-related death in men and the second most common in women. This cancer is especially responsible for 1.3 million deaths worldwide annually[3]. Early lung tumors detection remains thus a challenging task of medical imaging.

Lung cancer is probably one of the most dangerous cancer and its radiotherapy or surgery treatment is a difficult task. The detection of the exact location of the tumor is difficult, due to the breathing motion of the patient. Tumors can also be closed or even attached to the heart, which makes the treatments very risky. Finally the presence of numerous blood vessels and arteria present a large risk of dissemination of tumoral cells in the whole body of the patient. This dissemination of tumoral cells leads to the apparition of multiple tumors in various location of the body. An early and accurate treatment of the lung cancer is thus absolutely necessary to give a possible chance of remission to the patient. In this scenario, the segmentation and the tracking of the lung tumors can potentially help to optimize the radiotherapy treatment. We give in this section some experimental results on the segmentation of 3D and 4D (3D + time) CT images to detect and track the tumors location during the breathing motion of the patient.

4.3.1 General Description

We present here another application of minimal surfaces to the segmentation of lung tumors in 3D CT images of the thorax. The main application related to lung tumor segmentation is especially radiotherapy and surgery planning. This work summarizes a long cooperation with the "Institut Gustave Roussy", which provided the 3D CT images and helped us interpreting the datasets.



(a) (b)

Figure 4.10: Lung 3D CT images. (a-b) The tumors are indicated by red circles. The tumors present low contrasted boundaries with the surrounding tissues of the thorax.

4.3.2 Data

The lung tumors CT images are quasi-isotropic. The images present a slice thickness of 2mm and an in-plane resolution of 1.7mm. A typical dataset covers the whole thorax of the patient, leading to images with size 512x512x256.

[3]World Health Organization 2006.

4.3.3 Segmentation Strategy

The lung tumors segmentation is obtained through the computation of a minimal graph cut of the region adjacency graph. In this scenario, the user has to interactively specify the location of the tumors as well as a marker to specify the background, i.e. the objects that have not to be segmented. The segmentation protocol is illustrated in figure 4.11. Additionally, the user can add or delete markers to refine the segmentation results. A post-processing step is also proposed to smooth the segmentation result. This post-processing step consists in a morphological opening of the segmentation.

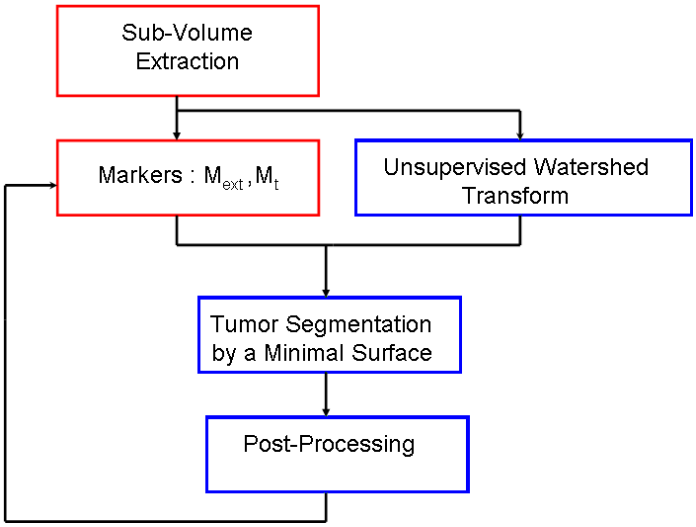


Figure 4.11: Lung Tumors Segmentation by Minimal Surfaces. The user provided information are outlined in red. Automatic steps are outlined in blue.

4.3.4 Implementation Details

The lung tumors are detected by the same method used to segment the myocardium. We refer thus the reader to the previous section for the implementation details. The tumors boundaries are extracted by computing a minimal graph cut of the region adjacency graph with weights given by equation 4.2.3. The minimal cut is computed on the region adjacency graph with two additional nodes s and t, respectively connected to the markers of the tumors and the markers of the external tissues. The edges weights of the graph are summarized in table 4.3.

Edge	Weight	for
w_{s,r_i}	$+\infty$	$r_i \in M_t$
$w_{r_i,t}$	$+\infty$	$r_i \in M_{ext}$
w_{r_i,r_j}	$\sum_{(e_{m,n} \in F_{(r_i,r_j)})} (g(\|\nabla I(p)\|))$	$r_i \in V_R, r_j \in N_{r_i}$

Table 4.3: Edges Weights for Approximate Minimal Surfaces.

4.3.5 Results

Figures 4.12 and 4.13 illustrate some segmentation results obtained by a minimal surface computed on the region adjacency graph of the watershed transform. The method permits to detect the tumor boundaries precisely even in low contrasted areas, where the tumors are in contact with tissues surrounding the

lungs, such as the heart.

Minimal surface is a leading method for the segmentation of lung tumors. The tumors are often surrounded by numerous blood vessels and tissues which present a low contrast with the tumors. The numerous blood vessels surrounding a lung tumor is also a major cause of the high mortality provoked by the lung cancer. Tumoral cells can easily be transported by these vessels in the whole body and secondary tumors can then grow in different locations.

Unfortunately we did not have the opportunity to quantify the quality of our segmentation method by comparing the results with hand made segmentations. This procedure has to be done in the future in order to prove the efficiency of the proposed method. A quantitative study will be presented in the next section for liver tumors segmentation.

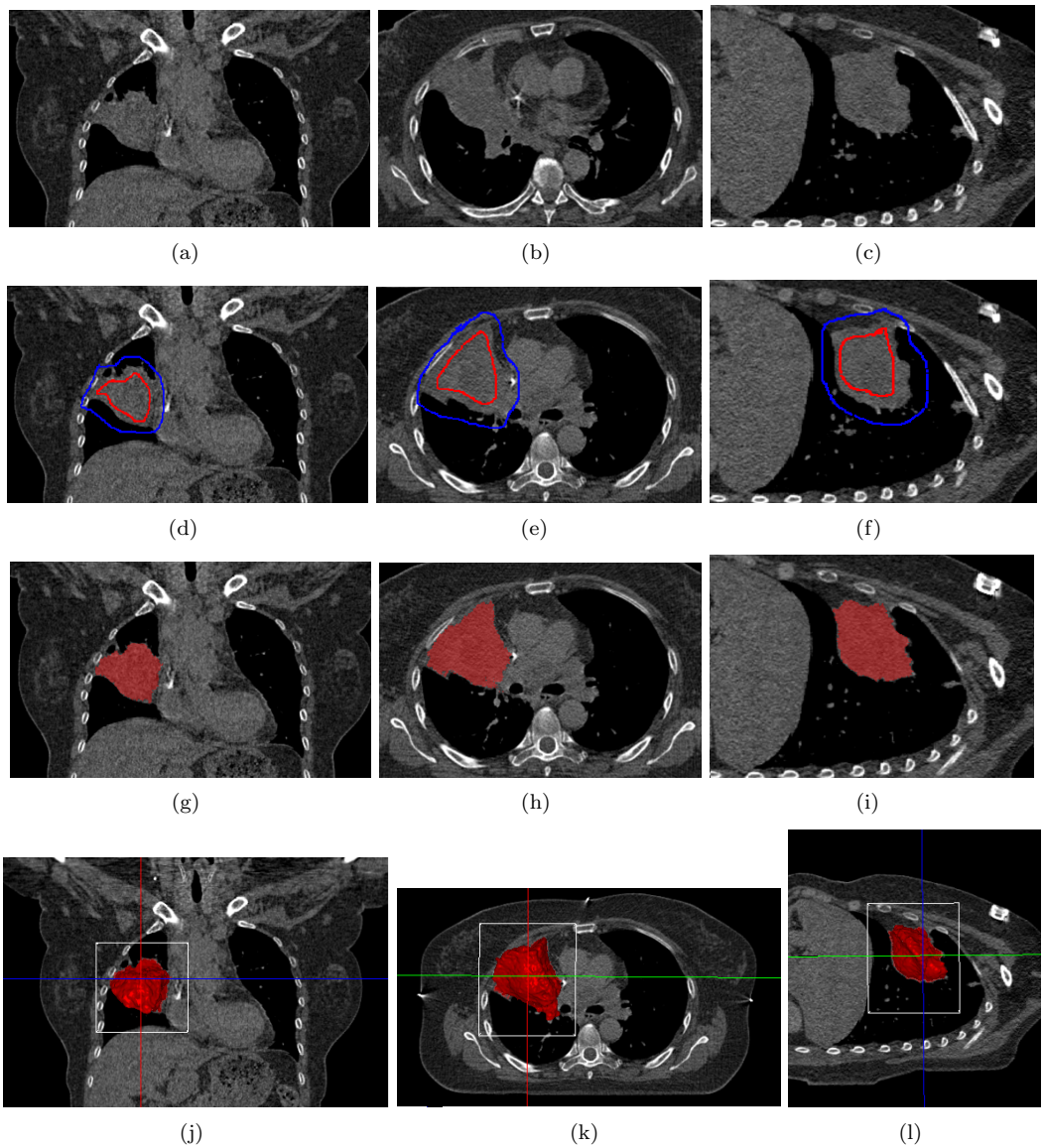


Figure 4.12: Lung tumors Segmentation. (a-c) Original image (512x512x256). (d-f) Markers. (g-i) Tumors boundaries. (j-l) Tumors boundaries, surface rendering.

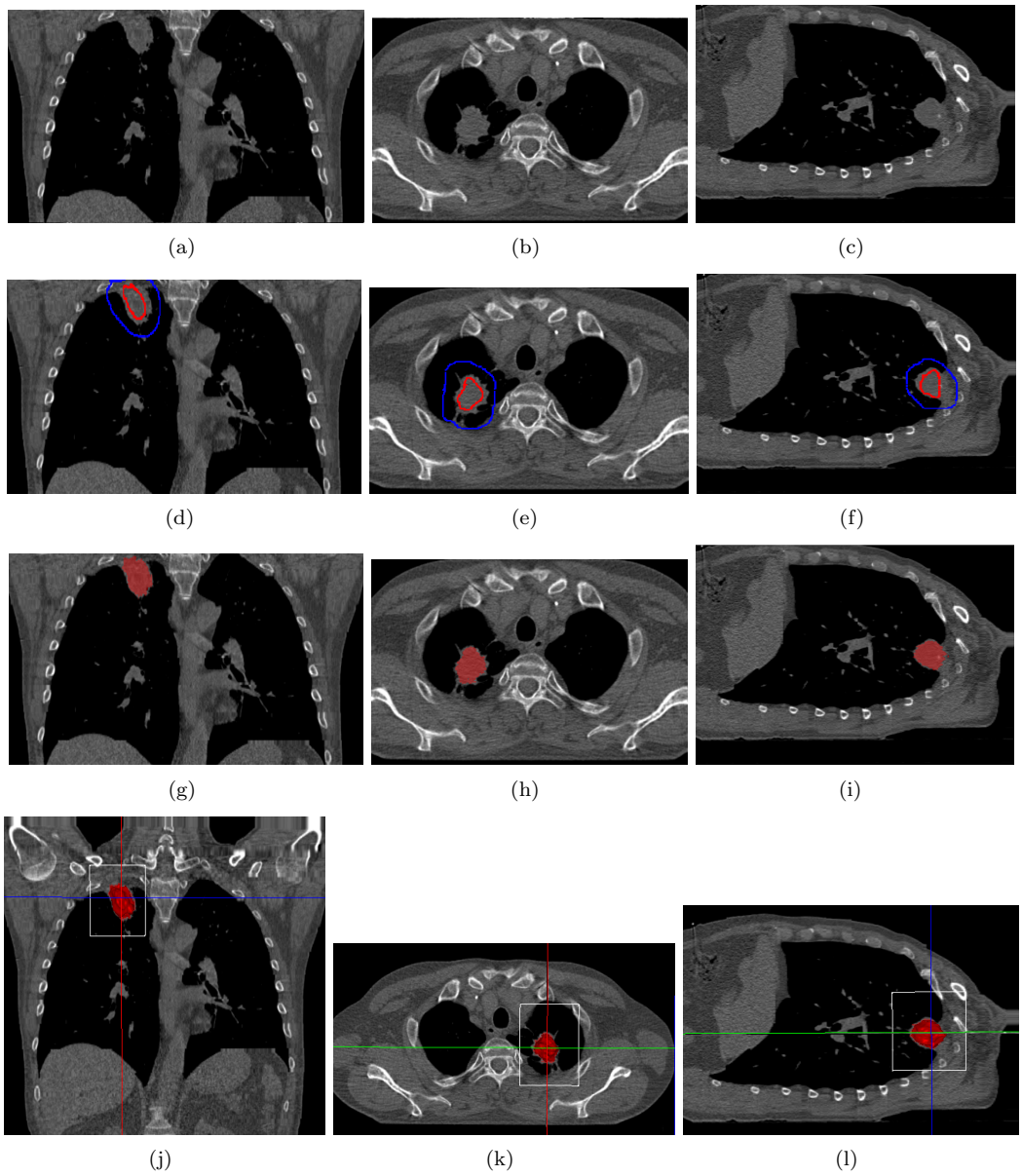


Figure 4.13: Lung tumors Segmentation. (a-c) Original image (512x512x256). (d-f) Markers. (g-i) Tumors boundaries. (j-l) Tumors boundaries, surface rendering.

4.3.6 Tumor Tracking in 4D CT images

We have extended the minimal surface method for the segmentation of 4D CT images in order to track the tumor boundaries during the breathing motion of the patient. In this case the images are acquired in combination with a measurement of the breathing of the patient. Several images are acquired at different steps of the breathing cycle as illustrated in figure 4.14. These images can then be used to track the boundaries of the different organs of the patient and follow their location during the breathing motion. This procedure aims at optimizing the radiotherapy treatment. In a ideal situation the irradiation of the tumors could be guided by the segmentation results of the 4D CT images. This procedure would allow the patient to breathe normally during the radiotherapy treatment. Up to now, the patients have to stop breathing during the irradiation, which is not very comfortable for patient having breathing problems caused by a cancer.

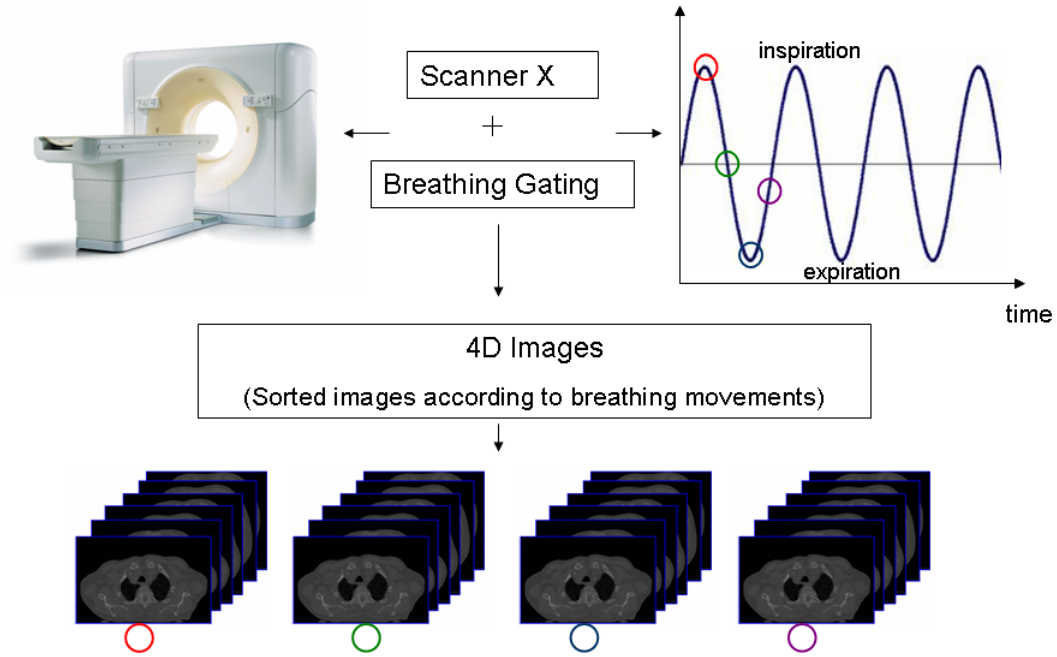


Figure 4.14: 4D CT images of the lungs acquisition protocol. The breathing motion of the patient is measured and a 4D image is reconstructed from the breathing gating.

4D CT Images Segmentation Protocols

We propose two protocols for the segmentation of tumors in 4D volumes (3D+t). The first one is a direct extension of the protocol used to segment 3D volume. In this first protocol we propose to build a 4D volume by concatenating the 3D images. Since watershed segmentation can be computed on any gradient image, the extension of the previous method requires thus only the computation of a spatio-temporal gradient image. The user has the to specify some markers on a 3D volume of the time-series. A watershed low-level segmentation is then computed according to a spatio-temporal gradient and finally a minimal surface is extracted from the region graph of the 4D watershed transform. This procedure is illustrated in figure 4.15.

The main advantage of this first protocol is that the whole 4D volume is segmented in a single step and large motions are allowed by this procedure. However this method requires a huge amount of memory since the whole 4D volume has to be stored to compute the watershed transform and a large graph has to be stored to compute the minimal cut. This first method is thus in practice very slow when used on

a classical personal computer.

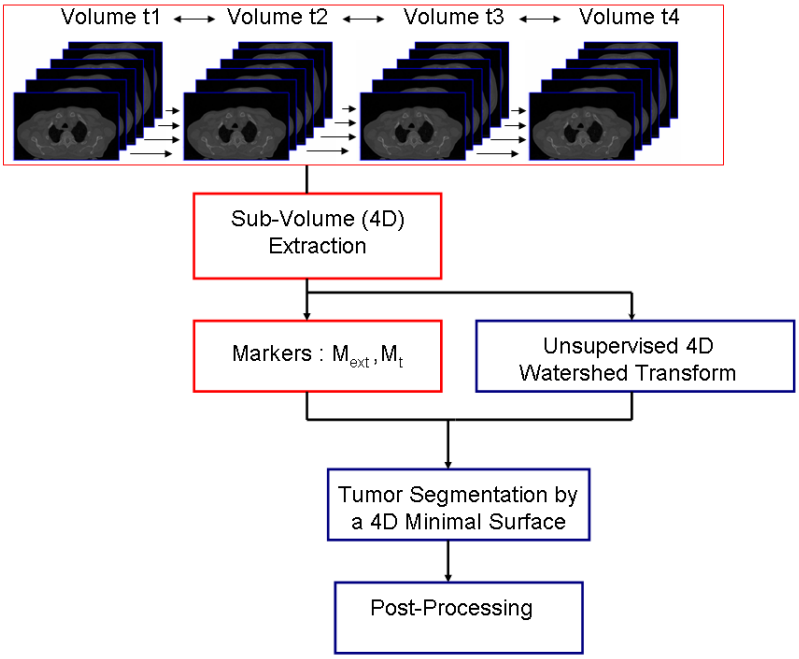


Figure 4.15: 4D CT images of the lungs segmentation protocol 1.

The second protocol is slightly different and aims at providing a fast and accurate 4D segmentation tool. We consider in this scenario a sequential segmentation of 3D images. The user has to provide markers of the tumors and the surrounding tissues on the first image of the time-series. The result of the segmentation at time t is then used to produce markers for the segmentation of the tumors at time $t = t + 1$. Since the motion of the thorax is relatively small, the new markers can be easily obtained from the previous segmentation. In our approach, we eroded each region of the segmentation to ensure that the resulting eroded image can be used as markers for the next image. This procedure works on these images because the motion is small and the tumors are large and compact objects. For thin objects segmentation, this procedure would fail because the erosion step will delete the thin structures and no markers could be extracted from the first segmentation. This procedure is illustrated in figure 4.16.

The main advantage of this second protocol is that the method does not need more memory than a classical 3D segmentation technique since it is based on sequential segmentation of 3D images. However this method does not allow large motion and does not allow to segment thin structures. This second method have finally been chosen for our experiments since the thorax motion is small and tumors are compact structures.

Results

Figure 4.17 illustrates a segmentation result on a 4D CT image. The second segmentation protocol was used to obtain the presented results. The segmentation was obtained by using a multi-terminal cut algorithm. The user has provided markers for the lungs, the tumor, as well as the surrounding tissues. The tumor and the lungs have been correctly delineated by using this strategy since the motion of the lungs is especially small on the upper part of the lungs, where the tumor is located.

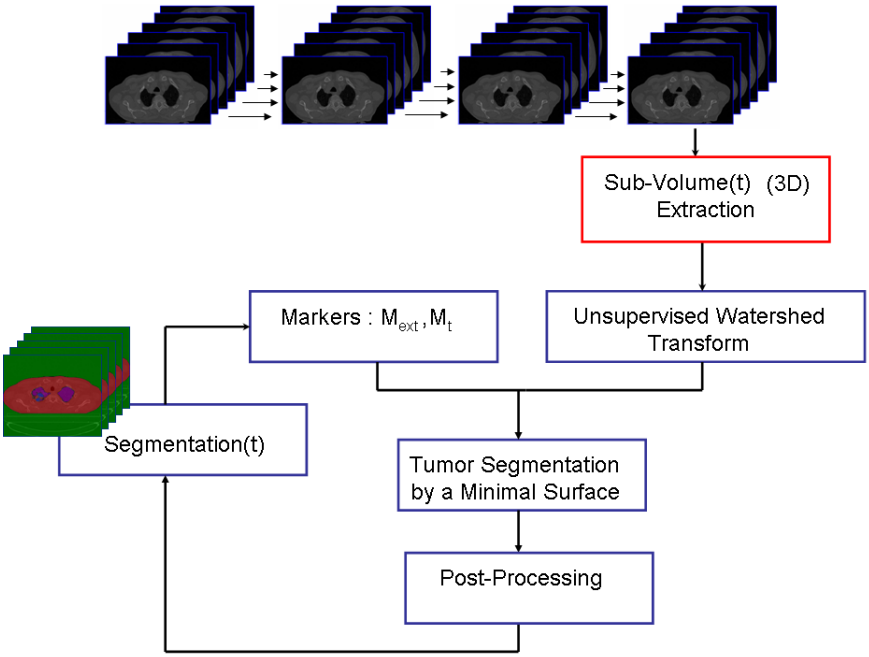


Figure 4.16: 4D CT images of the lungs segmentation protocol 2.

4.3.7 Conclusion

Minimal surfaces computed on the region adjacency graph provide again stable and robust segmentation results for the aimed application, the detection of lung tumors. Moreover, the method can be easily extended to track the tumors during the breathing motion of the patient. The method is sufficiently fast and precise to be used on large datasets such as 4D images. In such conditions the analysis of all datasets by a radiologist cannot be realized by hand made segmentation. The size of the datasets is too large to be manually segmented in a reasonable time. The proposed method is thus a first solution for the tracking of lung tumor. With the exponential growth of medical image data, it is clear that such interactive methods are good alternatives to fully manual segmentations. On the other side, fully automatic methods fail to achieve relevant segmentation results in all the cases and are not well suited for clinical applications which need always the supervision of a radiologist.

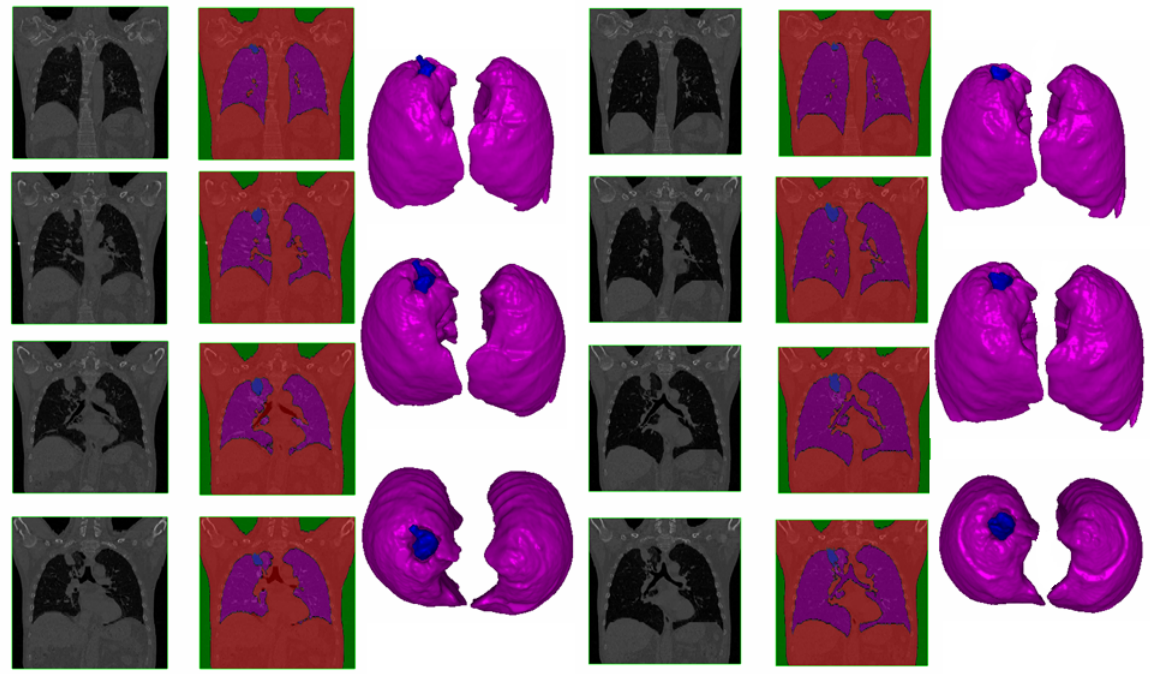


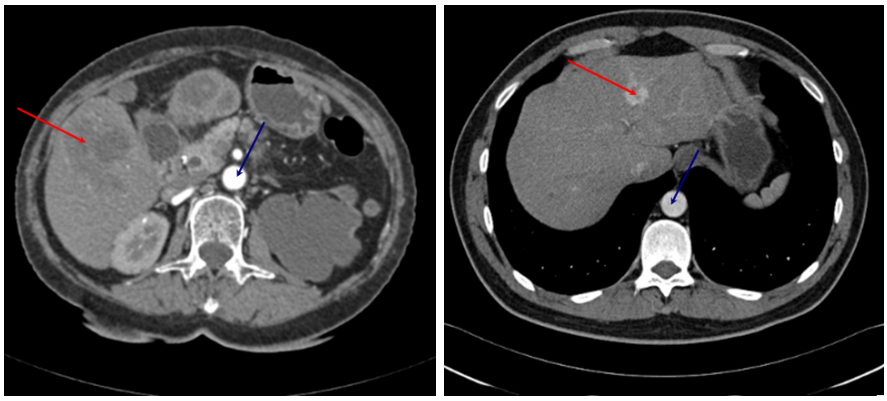
Figure 4.17: 4D CT images segmentation results at two different steps of a breathing cycle. The segmentation was obtained with multi-terminal cuts.

4.4 Liver Tumors Segmentation

We present in this section an application of minimal surfaces and Markov random fields for the segmentation of liver tumors. This strategy leads to an interactive method that we use to delineate tumors in 3D CT images. We detail our strategy to achieve relevant segmentations of these structures and compare quantitatively our results to hand made segmentations done by experienced radiologists. This section summarizes our participation to the MICCAI 2008 [4] workshop called: "3D segmentation in the clinic : A Grand Challenge II". This work aims at proving that the developed methods are well suited to be used in clinical routine. The quantitative and comparative analysis of the results show that our interactive segmentation strategy outperforms semi-automatic and automatic methods, and provides stable and robust segmentations.

Possible applications related to liver tumor segmentation are mainly radiotherapy and surgery planning. In these cases, the knowledge of the exact location and volume of the tumors is important to ensure that the chosen therapy is adapted to the patient. The robust extraction of liver tumor boundaries from CT images remains an open problem of medical imaging. Liver tumors present low contrasted boundaries and exhibit a large variability of shapes, sizes and locations in the liver. Due to these multiple difficulties, automatic or model-based approaches seem to be inadequate for this application. However, a few semi-automatic approaches have already been proposed in the literature. Recent approaches are mainly based on active contours [67], level-sets [83], as well as machine learning [64]. These methods have the interesting property to allow an interaction with its user through landmarks positioning or interactive refinement of the segmentation. We propose in this paper an interactive segmentation method based on user defined markers to extract liver tumors boundaries in 3D CT images.

4.4.1 Data



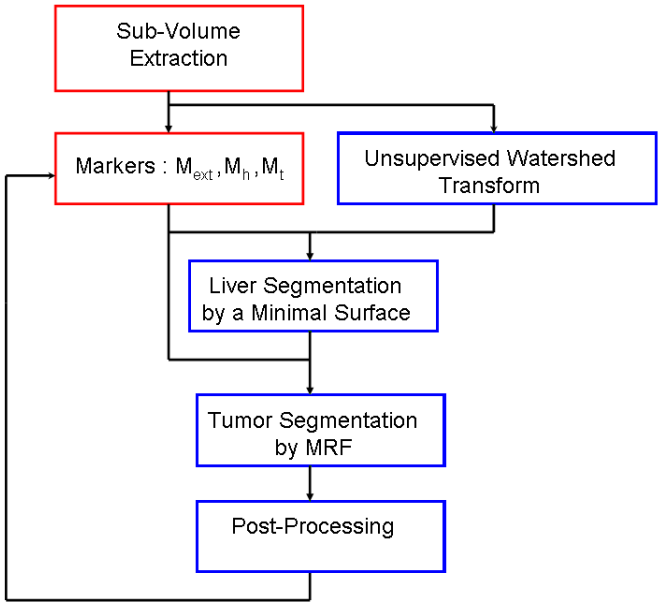
(a) (b)

Figure 4.18: Liver 3D CT images. (a) The tumor, indicated by a red arrow, presents a negative contrast with the liver. The contrast enhancing liquid is actually going through the aorta, indicated by a blue arrow, and the liver blood vessels. (b) The tumor, indicated by a red arrow, presents a positive contrast with the liver. The contrast enhancing liquid has already been transported in the liver vessels. The aorta, indicated by a blue arrow, is not very bright, which means that the contrast enhancing liquid is not present in it.

[4] MICCAI 2008 is the 11th International Conference on Medical Image Computing and Computer Assisted Intervention, September 6 - 10, New York, USA.

The liver tumors CT images, provided by the workshop's organizers [5], were acquired on one 64-slice and two 40-slice CT scanners using a standard four-phase contrast enhanced imaging protocol. The resulting images have a slice thickness of 1mm or 1.5mm and an in-plane resolution of 0.6-0.9mm. The resulting images are typically of size $521 \times 512 \times 256$. The imaging protocol consists in injecting a phase contrast liquid to the patient such that the contrast between tumors and the surrounding tissues is improved. The images are then acquired approximatively 30 seconds to one minute after the injection, when the contrast enhancing liquid is attaining the liver. Depending on several parameters such as the patient size or the patient cardiac rhythm, the images will present different enhanced contrasts in the liver. Several cases are illustrated in figure 4.18. The liver segmentation presents some difficulties that have to be taken into account to design a relevant segmentation protocol. First, tumors can be in positive or negative contrast with the liver, depending on the acquisition time of the images, as illustrated in figure 4.18. Secondly, the tumors boundaries are not well defined and perceptual properties have to be used to define the exact contours of the tumors.

4.4.2 Segmentation Strategy



(a)

Figure 4.19: Liver tumors 3D CT images strategy. The user provided information is outlined in red. Automatic steps are outlined in blue. An additional post processing step is also proposed to the user, this step consists in smoothing the segmentation by using morphological operators such as openings.

Our segmentation strategy is entirely based on the use of the region adjacency graph of an unsupervised watershed segmentation [75]. This low-level segmentation is used in combination with graph cuts to compute approximate minimal surfaces and approximate maximum a posteriori estimates of a Markov random field. Note also that we do not process the whole 3D CT image. The first step of our methodology is the manual definition of a sub-volume containing one or more tumors that need to be segmented in order to reduce the computation time of the segmentation algorithm.

The region adjacency graph is obtained from the watershed transform computed from all minima of the morphological gradient of the original CT image using Meyer's algorithm based on hierarchical queues

[5] The organizers of the workshop are Wiro Niessen, Martin Styner, Simon K. Warfield and Xiang Deng

[74]. Both minima of the gradient and the watershed transform are computed using the 6-neighborhood adjacency system. From this first low-level segmentation, a region adjacency graph is extracted and used for the following optimization steps.

Our segmentation protocol is motivated by this simple observation: the liver presents two kinds of tissues: tumoral tissues and healthy tissues. The classification of liver pixels in one of these two classes permits to extract the tumors. In other words, the tumors can be extracted from the liver by the mean of their grey levels. To achieve this classification we model the liver pixels as a Markov random field and the classification is performed through the maximum a posteriori estimation. One class corresponds to the normal liver tissues and the other class corresponds to the tumoral tissues. The classification is supervised by user defined markers that specify the tumors and the normal tissues. The markers are used to locate the tumors and to estimate the grey levels characteristics of these structures.

However, the liver pixels classification needs also that the liver boundaries are extracted. This task is realized by computing a minimal surface based on user defined markers. Note that the liver boundaries extraction is only necessary if the sub-volume considered for the segmentation contains non-liver tissues. In any other cases, the liver boundaries are not computed, and only the classification step is performed. The user has finally to specify normal liver tissues, tumoral tissues, and external tissues surrounding the liver. Based on these markers the liver is first extracted and secondly the pixels of the liver are classified and the tumors are finally extracted. The different energy minimization strategies (minimal surfaces and Markov random field) are based on the computation of a minimal graph cut as described in the previous chapter.

4.4.3 Implementation Details

Liver Segmentation

We denote the markers that specify the liver tissues as the sets of regions M_t and M_h, respectively for tumoral and healthy tissues. The markers specifying the tissues surrounding the liver are denoted by M_{ext}.

We refer the reader to the chapter 3 section 3, for details about the segmentation method using minimal surfaces. Let us recall that $F_{(r_i,r_j)}$ is defined as the set of edges of the pixel graph connecting two regions r_i and r_j of the low-level watershed segmentation:

$$F_{(r_i,r_j)} = \{e_{m,n} \in E \mid m \in r_i,\ n \in r_j\} \ . \tag{4.4.1}$$

In the following, we consider the strictly positive and decreasing function g used as an edge indicator for minimal surfaces:

$$g(\|\nabla I(p)\|) = \left(\frac{1}{1+\|\nabla I(p)\|}\right)^k \ . \tag{4.4.2}$$

In our application to tumor segmentation the parameter was set to $k = 2$.

The edges weights of the region adjacency graph $G_R = (V_R, E_R, W_R)$ are then set such that the weight of a graph cut equals the energy function of a surface (the integral of the image gradient along the surface) that it implicitly defines:

$$w_{r_i,r_j} = \sum_{(e_{m,n} \in F_{(r_i,r_j)})} g(max\,(\|\nabla I(m)\|, \|\nabla I(n)\|)) \ , \tag{4.4.3}$$

where $\|\nabla I(m)\|$ and $\|\nabla I(n)\|$ are the gradient magnitudes of the end points of $e_{m,n}$.

The liver boundaries are extracted by computing a minimal graph cut of the region adjacency graph with weights given by equation 4.4.3. The minimal cut is computed on the region adjacency graph with

two additional nodes s and t, respectively connected to the markers of the liver and the markers of the external tissues. The edges weights of the graph are given in table 4.4.

Edge	Weight	for
w_{s,r_i}	$+\infty$	$r_i \in M_t$
w_{s,r_i}	$+\infty$	$r_i \in M_h$
$w_{r_i,t}$	$+\infty$	$r_i \in M_{ext}$
w_{r_i,r_j}	w_{r_i,r_j}	$r_i \in V_R, r_j \in N_{r_i}$

Table 4.4: Edges Weights for Approximate Minimal Surface.

Tumor Segmentation

We recall here that the Markov random field model leads to the minimization of (see chapter 3, section 5):

$$E(x) = \sum_{i \in V_R} |r_i| \frac{(\mu_{r_i} - \mu_{x_i})^2}{2\sigma^2} + \sum_{i \in V_R} \sum_{j \in N_{r_i}} |F_{(r_i,r_j)}|.u_{i,j}.\delta(x_i \neq x_j) \ . \tag{4.4.4}$$

Assuming that the data are corrupted with a white gaussian noise, the likelihood function can finally be written as:

$$Pr(\mu_{r_i}|(x_i = 0)) = exp(-\frac{(\mu_{r_i} - \mu_{(x_i=0)})^2}{2\sigma_0^2}) \tag{4.4.5}$$

$$Pr(\mu_{r_i}|(x_i = 1)) = exp(-\frac{(\mu_{r_i} - \mu_{(x_i=1)})^2}{2\sigma_1^2}) \tag{4.4.6}$$

where μ_{r_i} is the mean gray level of the region r_i, and μ_{x_i} is the mean value of the pixels expected to take the value x_i. In our experiments the values of σ_0 and σ_1 were set to $\sigma_0 = 0.3$ and $\sigma_1 = 0.2$. In our model, σ_0 represents the grey level variance of the tumoral tissues and σ_1 represents the grey level variance of the healthy tissue. We have experimentally found that the variance of tumoral tissues is slightly higher than the grey level variance of the healthy tissues. On the other side, the values of μ_{x_i} are estimated from user's markers as:

$$\mu_{(x_i=0)} = \frac{1}{|M_t|} \sum_{r_i \in M_t} \mu_{r_i} \tag{4.4.7}$$

$$\mu_{(x_i=1)} = \frac{1}{|M_h|} \sum_{r_i \in M_h} \mu_{r_i} \tag{4.4.8}$$

where $|M_h|$ and $|M_t|$ are respectively the number of regions marked as healthy or tumoral.

The prior function $u_{i,j}$ that we use in our application is a contrast sensitive function:

$$u_{i,j} = \left(\beta - \beta * (\mu_{r_i} - \mu_{r_j})^n\right) \ . \tag{4.4.9}$$

where n is a free parameter describing the strength of the term $\beta * (\mu_{r_i} - \mu_{r_j})$. Since the mean grey levels μ_{r_j} are real values between 0 and 1, the parameter n balances the effect of the local contrast. In the following we fixed the parameter to $n = 4$.

The previous function takes into account the contrast between two regions and is equal to a constant in the areas where the contrast is low. This function allows to detect correctly the boundaries of high contrasted tumors, whereas low contrasted boundaries are smoothed such that the perimeter of the object is minimized.

The minimal cut is finally computed on the region adjacency graph with two additional nodes s and t, respectively connected to the markers of the liver and the markers of the external tissues. The edges

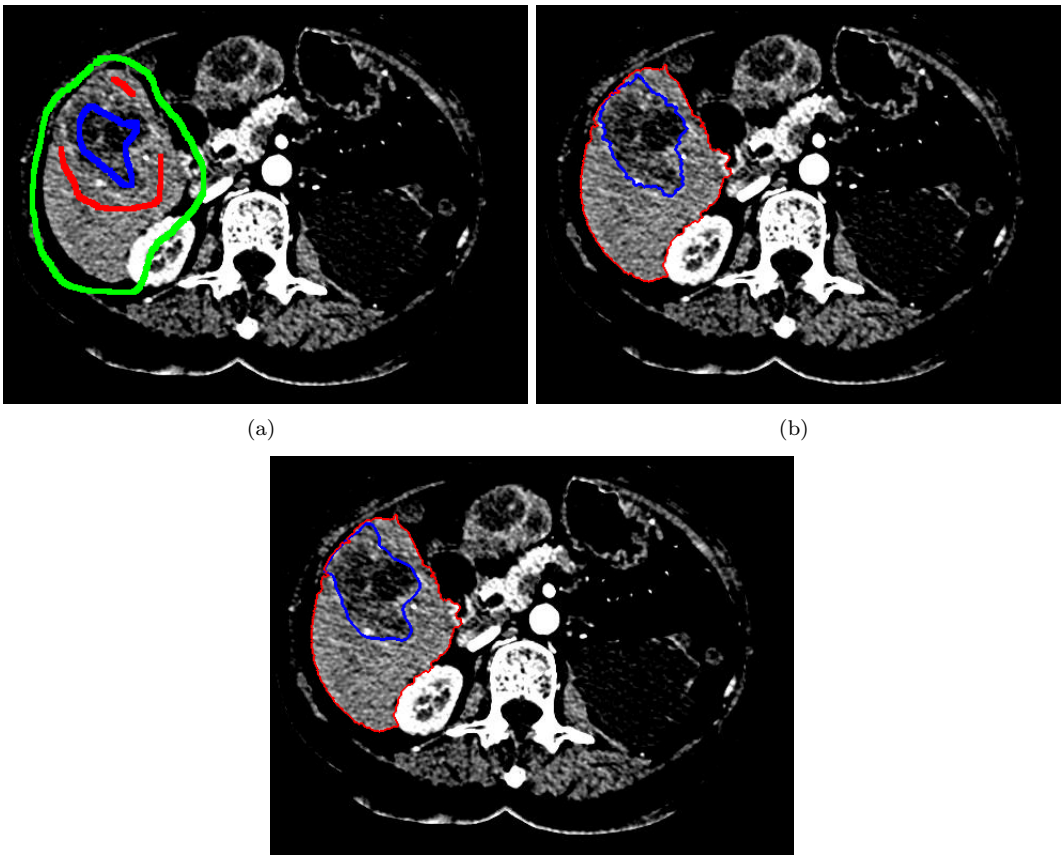
weights of the graph are given in the table 4.5. The results obtained using our methodology reduces drastically the computation time compared to a pixel graph approach. Moreover it does not seem to affect the resulting segmentations. This method was previously presented for the segmentation of micro-tomography images of granular materials [104].

Edge	Weight	for
w_{s,r_i}	$+\infty$	$r_i \in M_t$
$w_{r_i,t}$	$+\infty$	$r_i \in M_h$
w_{s,r_i}	$\frac{(\mu_{r_i}-\mu_{(x_i=0)})^2}{2\sigma^2}$	$r_i \in V_R \setminus \{M_t \cup M_h\}$
$w_{r_i,t}$	$\frac{(\mu_{r_i}-\mu_{(x_i=1)})^2}{2\sigma^2}$	$r_i \in V_R \setminus \{M_t \cup M_h\}$
w_{r_i,r_j}	$\lvert F_{(r_i,r_j)}\rvert.u_{i,j}$	$r_i \in V_R, r_j \in N_{r_i}$

Table 4.5: Edges Weights for Approximate maximum a posteriori estimation.

Example

Figure 4.20 illustrates our segmentation strategy on a single slice of a 3D CT image. The obtained results are in good concordance with the expected results obtained by a hand made segmentation. Alternatively, the user can add or delete markers if he is not satisfied with the computed segmentation.



(a) (b)

(c)

Figure 4.20: Liver tumors segmentation. (a) User specified markers. In blue, the tumor markers, in red, the liver markers and in green, the external markers. (b) Results of our segmentation strategy. (c) Radiologist hand made segmentation of the tumor.

4.4.4 Evaluations

We recall here the evaluation scores description given by the organizers of the workshop. For each CT image, a reference segmentation done by an experienced radiologist is available. In order to evaluate the accuracy of the developed method, five scores are computed from five different evaluation measures:

- Volumetric overlap. This score is computed as the number of voxels in the intersection of segmentation and reference, divided by the number of voxels in the union of segmentation and reference. This value is equal to 1 for a perfect segmentation and is equal to 0 as the lowest possible value, when there is no overlap at all between the segmentation and the reference. The volumetric overlap error is then given as 1 minus the volumetric overlap score. Note also that two different hand made segmentations typically present a volumetric overlap error of 10-15 % depending on the structures sizes.

- Relative absolute volume difference, in percent. The total volume of the segmentation is divided by the total volume of the reference. From this number 1 is subtracted, the absolute value is taken and the result is multiplied by 100. This value is 0 for a perfect segmentation and larger than zero otherwise. Note that the perfect value of 0 can also be obtained for a non-perfect segmentation, as long as the volume of that segmentation is equal to the volume of the reference.

- Average symmetric absolute surface distance, in millimeters. The boundaries voxels of segmentation and reference are determined. These are defined as those voxels in the object that have at least one neighbor (from the 26 nearest neighbors) that does not belong to the object. For each voxel in these sets, the closest voxel in the other set is determined (using Euclidean distance and real world distances, so taking into account the generally different resolutions in the different scan directions). All these distances are stored, for boundary voxels from both reference and segmentation. The average of all these distances gives the averages symmetric absolute surface distance. This value is 0 for a perfect segmentation.

- Symmetric RMS surface distance, in millimeters. This measures is similar to the previous measure, but stores the squared distances between the two sets of border voxels. After averaging the squared values, the root is extracted and gives the symmetric RMS surface distance. This value is 0 for a perfect segmentation.

- Maximum symmetric absolute surface distance, in millimeters. This measure is similar to the previous two, but only the maximum of all voxel distances is taken instead of the average. This value is 0 for a perfect segmentation.

Finally, for each metric a score between 0 and 100 is assigned to the segmentation results based on the typical variability of manual segmentations. A score of 100 is assigned to the perfect segmentation which matches exactly the reference segmentation. For each metric, a reference value from segmentation performed by independent users is assigned with a score of 90, which is shown below:

- Volumetric overlap error: 12.94%

- Relative absolute volume difference: 9.64%

- Average symmetric surface distance: $0.40mm$

- RMS symmetric surface distance: $0.72mm$

- Maximum symmetric surface distance: 4.0

A Score of each metric for a given segmentation can be obtained, using a linear interpolation or extrapolation between the two points specified above. Note that 0 is the minimum score that one segmentation will get. A score of 90 means that the segmentation results is as good as a manual segmentation. More details about the evaluations can be found on the web site of the organizers of the challenge : $http://lts08.bigr.nl/$.

4.4.5 Training Results

Table 4.6 summarizes the evaluation scores of our method on a set of 4 CT images presenting 10 tumors with known hand made segmentations. The evaluation scores compare our results with the radiologists segmentations. The important point is that these results have been obtained with the knowledge of the hand made segmentations. The results have been obtained such that the similarity between the two segmentations is visually satisfactory.

The mean surface distance between our segmentations and the references is less than a millimeter, which is the typical resolution of a voxel of the studied 3D CT images. The volumetric overlap error shows that approximatively 83 % of our segmentation volume is in perfect match with a hand made

segmentation. The mean total score obtained on the training data set is 88, which is not far from the score obtained from independent hand made segmentations performed by experienced radiologists. Figure 4.23 illustrates some comparisons between the hand-made segmentation and the results obtained with our methodology.

	Overlap Error		Volume Difference		Average Surface Distance		RMS Surface Distance		Maximum Surface Distance		Total Score
Tumor	(%)	Score	(%)	Score	(mm)	Score	(mm)	Score	(mm)	Score	
Tumor 1	19,11	85	4,94	95	1,17	71	1,53	79	5,7	86	83
Tumor 2	17,84	86	11,04	89	0,53	87	0,79	89	3,69	91	89
Tumor 3	21,29	84	5,08	95	0,79	80	0,97	87	3,05	92	88
Tumor 4	17,71	86	13,74	86	0,62	85	0,93	87	4,04	90	87
Tumor 5	19,2	85	6,07	94	0,57	86	0,85	88	3,19	92	89
Tumor 6	29,47	77	15,95	84	0,56	86	0,85	88	2,5	94	86
Tumor 7	13,47	90	3,68	96	0,92	77	1,28	82	6,41	84	86
Tumor 8	11,24	91	8,57	91	0,45	82	0,73	90	3,37	92	89
Tumor 9	10,63	92	7,53	92	0,74	82	1,08	85	5,93	85	87
Tumor 10	10,02	92	1,17	99	0,46	89	0,77	89	4,37	89	92
Average	17,0	87	7,78	92	0,68	83	0,98	86	4,23	90	88

Table 4.6: Results of comparison metrics and scores for all ten training tumors.

4.4.6 Testing Results

Table 4.7 summarizes the evaluation scores of our method on a set of 5 CT images presenting 10 tumors with unknown hand made segmentations. The evaluation scores compare our results with the radiologists segmentations. The important point is that these results have been obtained without the knowledge of the hand made segmentations. A comparison between the training scores and the testing scores is illustrated in figure 4.21. The mean surface distance between our segmentations and the references is approximatively one and a half millimeter, which represents 2 or 3 voxels of the studied 3D CT images. The volumetric overlap error shows that approximatively 71 % of our segmentation volume is in perfect match with a hand made segmentation. Figure 4.24 illustrates some results obtained with our methodology. Some evaluation results show that we have misunderstood some structures that have to be extracted. This problem leads to very low scores that could have been avoided with the help of a radiologist (see for instance tumor number 5). However our results are promising, considering the low quality of some images of the dataset.

The computation time of our method depends on of the image size and the number of refinements of the segmentation. The first step, which consists in extracting a sub-volume, is typically done in one minute. The markers placement requires basically one to two minutes. In our experiments we have drawn markers in three orthogonal slices centered on the tumor in about one minute. The computation time of the segmentation algorithms is then relatively fast: about 1 seconds for the watershed transform, 2 seconds for the liver extraction and 2 other seconds for the tumor extraction (for a typical sub-volume of size $100 \times 100 \times 100$, computed on a classical personal computer). The time needed for the refinement steps is then mainly due to the markers placement. In our experiments, two to five refinement steps were needed to provide the presented results. The typical time spent for these refinements varied from one to five minutes. The total time needed for the segmentation of a tumor is then approximatively five to seven minutes.

	Overlap Error		Volume Difference		Average Surface Distance		RMS Surface Distance		Maximum Surface Distance		Total Score
Tumor	(%)	Score	(%)	Score	(mm)	Score	(mm)	Score	(mm)	Score	
Tumor 1	27,16	79	21,68	77	2,01	49	2,90	60	10,50	74	68
Tumor 2	36,41	72	24,98	74	1,36	66	1,83	74	6,15	85	74
Tumor 3	31,99	75	16,93	82	1,18	70	1,59	78	5,64	86	78
Tumor 4	33,19	74	1,86	98	0,84	79	1,09	85	4,00	90	85
Tumor 5	61,24	53	119,82	0	2,29	42	2,95	59	9,30	77	46
Tumor 6	21,83	83	10,68	89	2,65	33	3,58	50	18,31	54	62
Tumor 7	21,44	83	4,85	95	0,93	77	1,38	81	6,89	83	84
Tumor 8	16,02	88	14,47	85	1,84	54	2,70	62	9,73	76	73
Tumor 9	22,87	82	2,14	98	0,65	83	0,97	87	5,10	87	87
Tumor 10	22,78	82	21,27	78	1,21	69	1,73	76	7,33	82	77
Average	29,49	77	23,87	78	1,50	62	2,07	71	8,29	79	73

Table 4.7: Results of comparison metrics and scores for all ten test tumors.

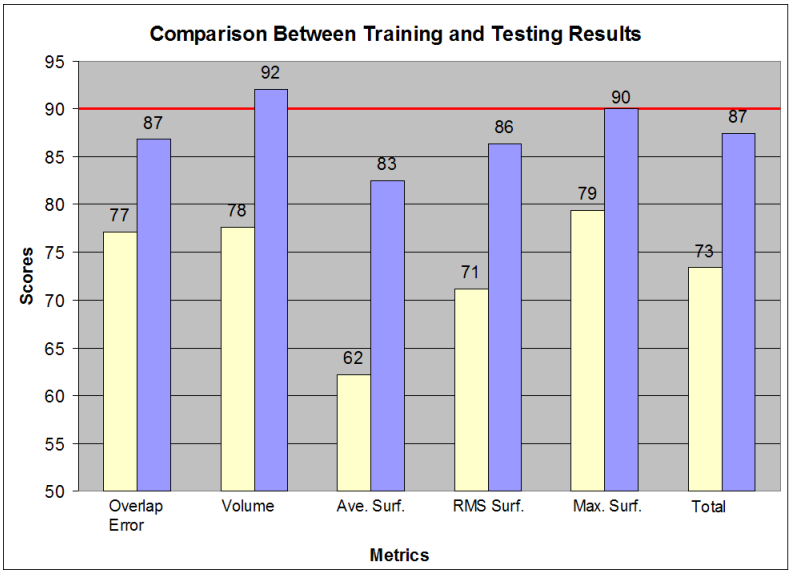


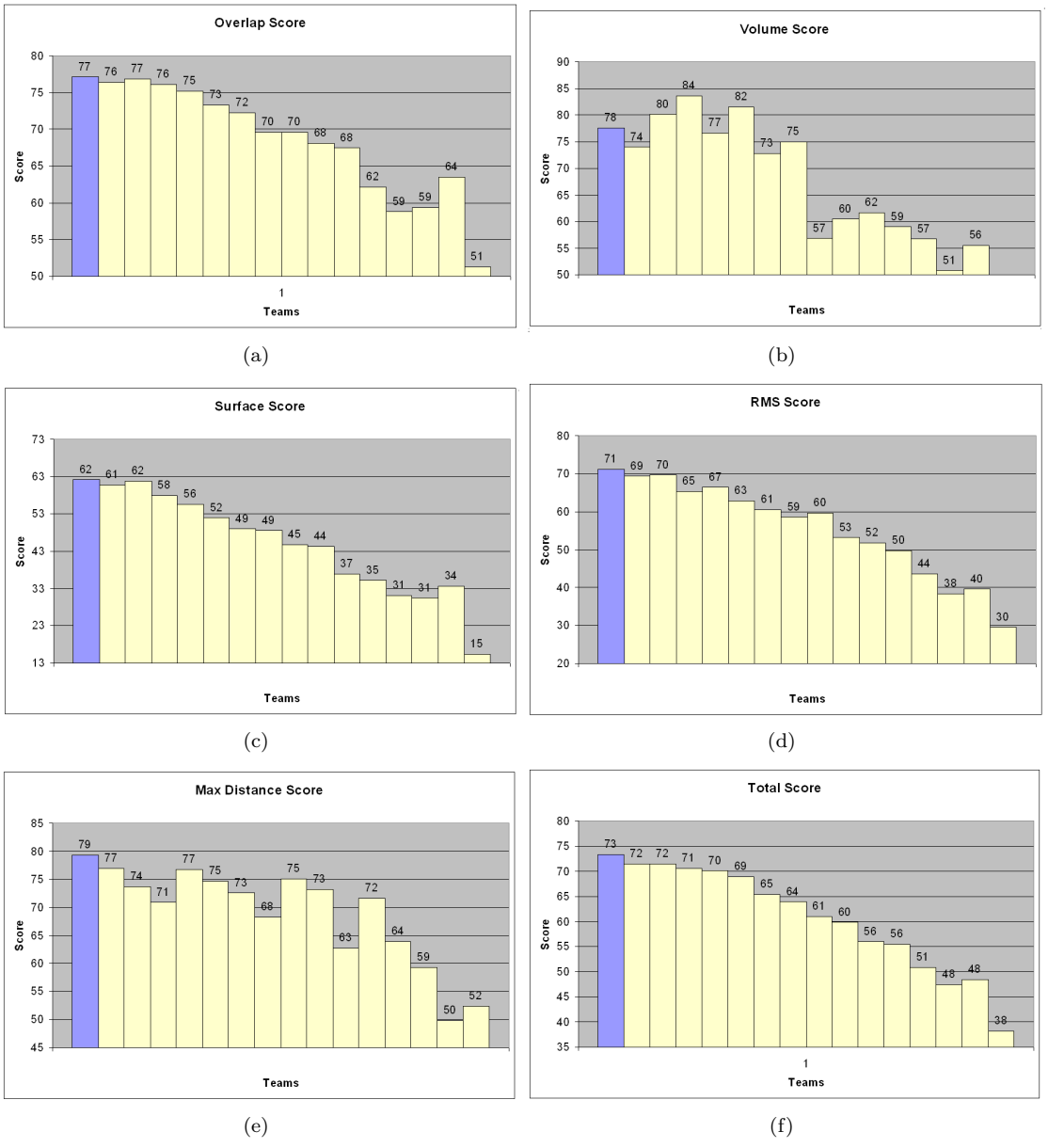
Figure 4.21: Comparison Between Training and Testing Results. The scores obtained on the training data set are shown in blue, whereas the scores obtained on the testing data set are shown in light yellow. The red line indicates the reference value obtained from the segmentation results performed by independent and experienced radiologists.

4.4.7 Total Scores

Our segmentation method has provided the best results over 15 other interactive, semi-automatic and fully automatic techniques after the first competition of the challenge organized for MICCAI 2008. The final competition will be help during the conference. A large variety of methods were proposed for the segmentation of liver tumors including surface evolution [94, 50], region growing [86, 115], machine learning [94, 119, 93], cognition network [89], thresholding and morphological operators [22, 78], and finally probabilistic methods [109, 6]. Our interactive segmentation method has provided stable and accurate segmentation results in all the different cases presented by the data. Noisy data, low contrasted tumors with positive or negative contrast, large tumors, and very small tumors were segmented efficiently. A summary of the evaluation scores are illustrated in figure 4.22. The scores (between 0 and 100) represent the quality of the provided segmentations compared to hand made segmentations as detailed earlier. Perfect segmentations should reach the score 100. Our method has obtained the higher mean score of 73.

4.4.8 Conclusion

Our method exhibits promising results for the aimed application. However some open problems still remain. First, the segmentation of multiple tumors in the same liver often requires additional user markers to correctly separate the tumors. The developed method merges the tumors in a single object when different tumors are too close. This problem requires that the user adds markers between the merged tumors. This additional interaction speeds down the segmentation protocol. However the used methods (minimal surfaces and Markov random fields on a region adjacency graph) are fast and can be used interactively. Secondly we did not develop any preprocessing step such as filtering of the images. There is thus still some possible improvements of our methodology. Future work will be concentrated on the development of adapted filters to simplify the segmentation and the classification step of our methodology.

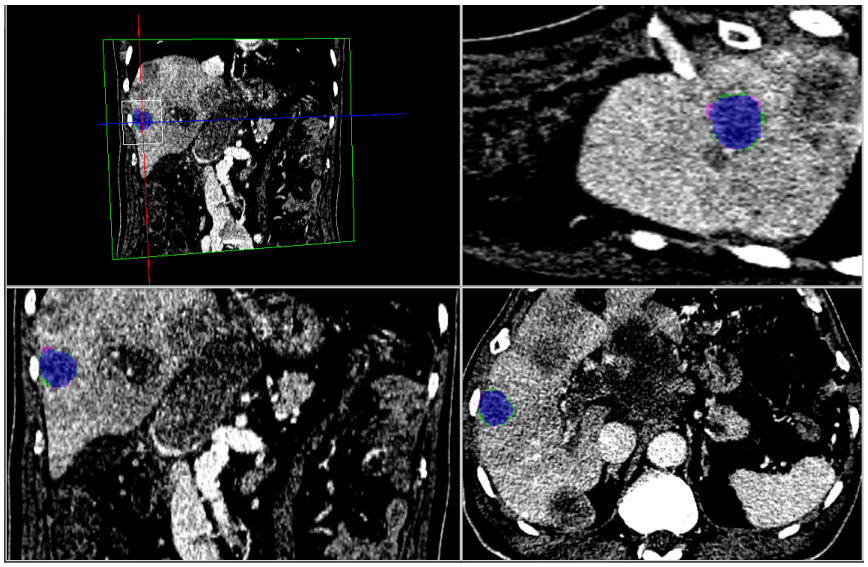


(a) (b)

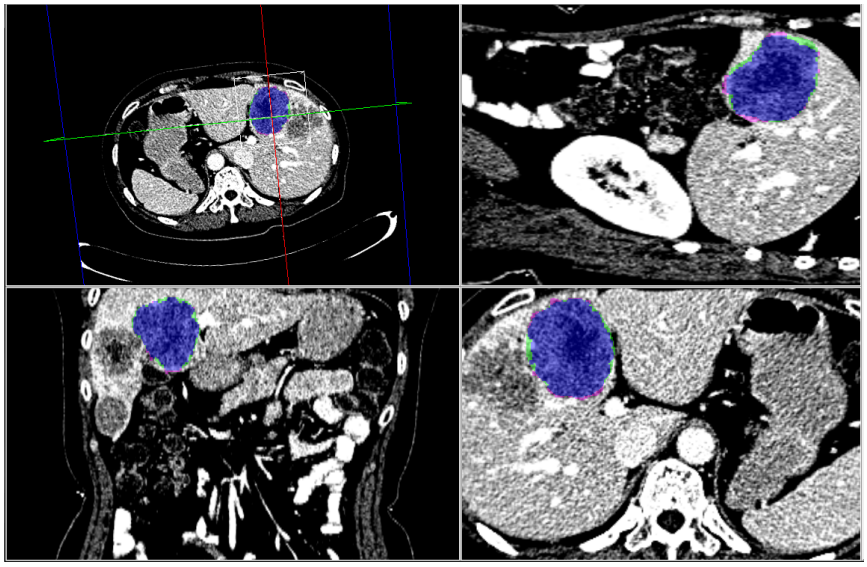
(c) (d)

(e) (f)

Figure 4.22: MICCAI 2008 Liver Tumor Segmentation Challenge. The scores expresses the quality of the segmentation compared to a hand made segmentation. Our results, highlighted in blue, have the best mean score and our segmentation strategy provides the best results according to four different quantitative measures.

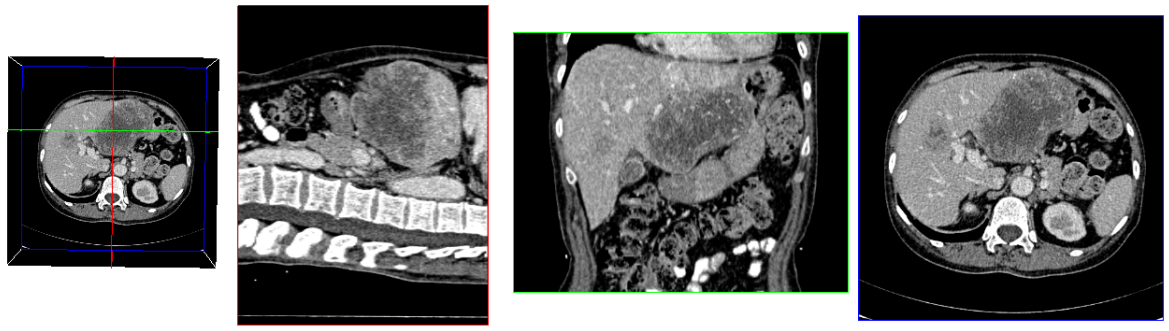


(a)

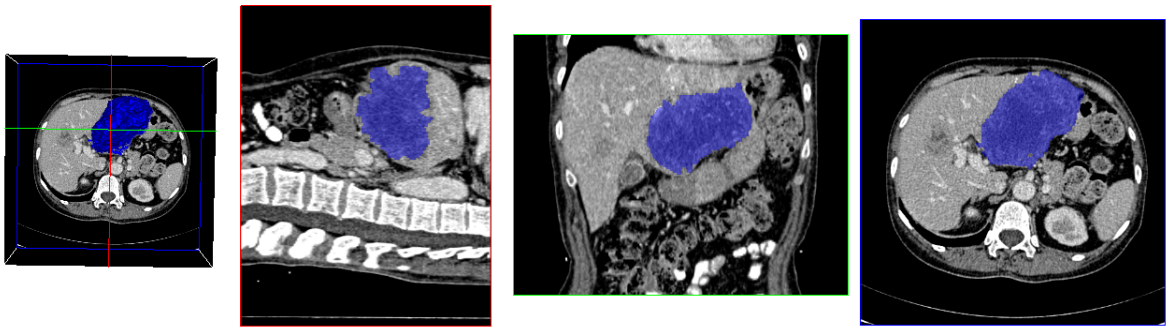


(b)

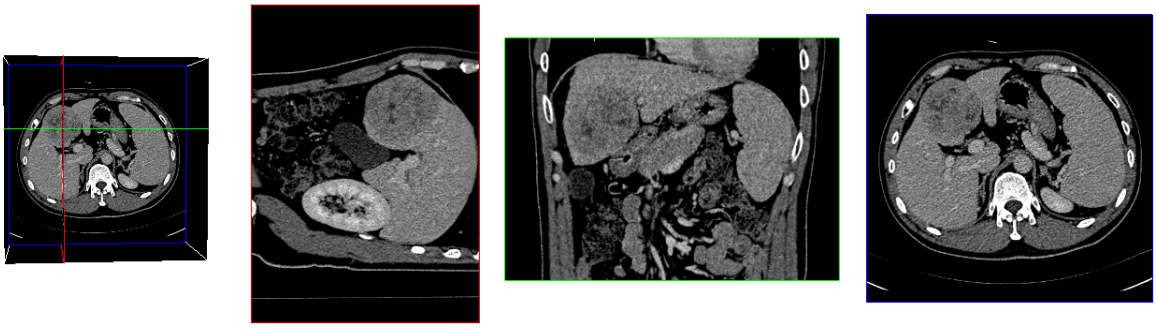
Figure 4.23: Training data results. Comparison between hand made and the proposed segmentation method. Blue pixels indicate a perfect match, green pixels indicate zones that are not detected by our method and pink pixels indicate zones not detected by the radiologist. (a) Tumor 5 results. (b) Tumor 7 results.



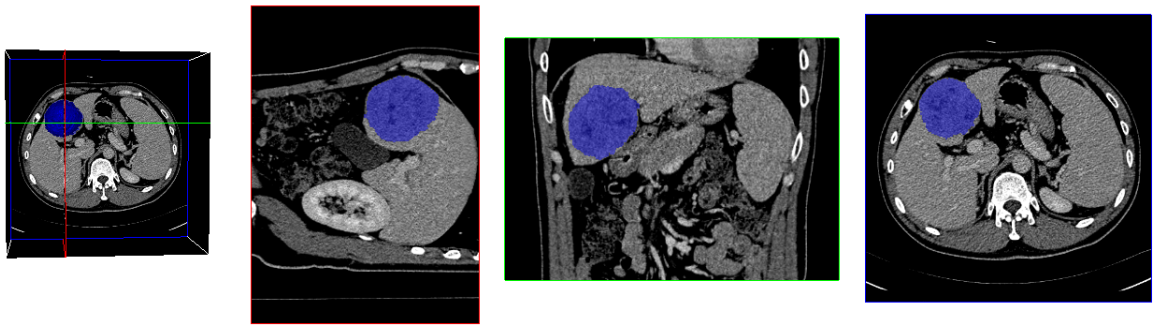
(a)



(b)



(c)



(d)

Figure 4.24: Testing data results. (a) Original image. (b) Tumor 6 results. (c) Original image. (d) Tumor 8 results.

4.5 Conclusion

Our segmentation tools have been integrated in a complete software that permits to visualize and segment images interactively. This software has been successfully used in many different medical applications, including model creation for surgery planning and simulation, lung tumor segmentation and tracking for radiotherapy planning. Finally our tools have been studied quantitatively and comparatively to other manual, interactive, semi-automatic and automatic segmentation methods. This study has proved that our methods are leading techniques for the segmentation of liver tumors. Our software and our segmentation methods are thus well suited to be used in a clinical routine. The developed methods provide almost the same quality than hand-made segmentations which require much more time than our interactive methods. Future work will be concentrated on the improvement of our software in order to match all the clinical requirements.

5

The Watershed Transform on Graphs

This chapter presents an algebraic point of view of some path optimization problems introduced in the first chapter: minimum spanning forests and shortest path forests. A large variety of graph problems are amenable to certain optimal paths computation, as described in chapter 1. Instances of paths problems are numerous: checking path existence, finding shortest or most reliable paths, listing all paths, etc. Dedicated algorithms have been developed to solve efficiently each of these problems; for instance, Dijkstra algorithm for the single source shortest paths problem. However, the development of a general framework for the whole family of path optimization problems is desirable to provide a better understanding of the underlying structure of these problems.

We propose such a unifying framework by introducing some suitable algebraic structures. A few variants of the algebraic approach for solving path problems have already been proposed [43, 19, 77, 12]. Our favorite approach issued from [43] uses a structure called "path algebras". This approach is based on some analogies of graphs with ordinary linear algebra.

The benefit of this algebraic point of view is that it allows us to define all these techniques from a general, abstract, path optimization problem. It is then possible to define a common algorithm to solve all instances of path optimization problems: the generalized Dijkstra algorithm [28, 23]. We have implemented this generic algorithm; it constitutes the basis of a $C/C++$ library that we have developed and used to present the results of this thesis.

An important contribution of this work is a redefinition of the watershed transform [75, 72, 73]as an instance of a particular optimal paths problem [72, 73, 113, 80]. This property leads to clear definitions of the watershed transform. Finally we propose some extensions of the watershed transform. First, we extend the watershed transform in the case of dissimilarity measures, in connection with minimal spanning forests and shortest paths forests methods presented in the second chapter. We also propose two watershed transform definitions in the case of vector-valued images, which necessitate the use of graphs whose edges weights are also vectors. This last extension is particulary interesting to segment images with the knowledge of multiple imaging modalities. Multi-modal medical images or color images are particular cases of vector-valued images to which our extension of the watershed transform can be applied to.

5.1 Introduction

The path algebra point of view has already been used in the context of mathematical morphology through the watershed transform, especially explored by Meyer [73]. In his paper, Meyer has shown that the watershed transform of mathematical morphology can be seen as an instance of a path optimization problem. He has introduced a specific path algebra based on a lexicographical distance to provide a suitable framework for the watershed transform from markers.

On the other side, a generalized Dijkstra algorithm has also been used by Falcao et al. in [33]. They derived the so called image foresting transform generalizing the problem of minimal weighted forests. However Falcao did not link the structure of minimal weight forest problems with an algebraic structure like path algebras. We think that the path algebra point of view is suitable to obtain an efficient algorithms for solving the problem of minimal weighted forest, and besides provide solutions to a large class of connected problems. We will present in this chapter the path algebra point of view that has been developed in the context of graph theory by Gondran and Minoux in [43]. We will not add any new theoretical result about path algebra; our aim is to show that this structure is well suited to generalize already known graph based image segmentation methods.

We also study the watershed transform on images and its counterpart on graphs. We recall the different definitions of this segmentation technique and show that each of these definitions correspond to a particular shortest path problem. We detail each path optimization problem and give its corresponding path algebra. Finally we extend the technique for dissimilarity based paths optimization and vector valued images.

This chapter is subdivided as follows: section 5.2 introduces notations and definitions of path algebras; in section 5.3 the problem of shortest path forest and the generalized Dijkstra algorithm are formalized. This algorithm allows us to solve a large variety of path optimization problems. Finally sections 5.4 and 5.5 give precise definitions of the watershed transform, as well as some extensions of the watershed transform for vector-valued images.

5.2 Path Algebras

Path algebras provide a very powerful theoretical framework for image segmentation based on paths optimization and particulary for watershed segmentation. The main idea of this approach is to associate an algebraic structure to graph problems. A specific graph problem is then represented by a matrix equation in a particular algebraic structure. The matrix representation can then be manipulated such that paths computation, enumeration and optimization are seen as linear algebra problems. Most paths optimization problems can be written as linear equations in particular spaces. However, each type of problem will generally require a different algebraic structure. We consider especially the paths optimization problems and we will not detail path enumeration and listing problems.

5.2.1 Definitions

We first start this chapter by some definitions and notations about algebraic spaces.

Definition 5.2.1 (Monoid)**.** *Let M be a set and $\oplus$ a binary operation on M. $(M, \oplus, \epsilon)$ is a monoid, if $\oplus$ is associative and ϵ is its identity element: $\forall a \in M, \epsilon \oplus a = a \oplus \epsilon = a$.*

Alternatively, one can define a monoid as a semigroup with an identity element. Note that a monoid satisfies the axioms of a group with the exception of having inverses.

Definition 5.2.2 (Commutative Monoid)**.** *A monoid M, is a commutative monoid if $\oplus$ is commutative.*

Definition 5.2.3 (Dioid)**.** *A dioid D, written $(D, \oplus, \otimes, \epsilon, \gamma)$, is an algebraic structure following the properties:*

1. *$(D, \oplus, \epsilon)$ is a commutative monoid. The identity element ϵ of $\oplus$ is called the zero of the dioid.*
2. *$(D, \otimes, \gamma)$ is a monoid. The element γ is called the unit element of the dioid.*
3. *The operation $\otimes$ is distributive with respect to the operation $\oplus$, and ϵ is an absorbing element for $\otimes$:*

$$\forall (a, b, c) \in D^3, \quad a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \; . \tag{5.2.1}$$

$$\forall (a, b, c) \in D^3, \quad (b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a) \; . \tag{5.2.2}$$

$$\forall a \in D, \quad a \otimes \epsilon = \epsilon \otimes a = \epsilon \; . \tag{5.2.3}$$

Some authors consider all dioids as path algebras [43]; in our presentation we define a path algebra as a dioid that admits an order relation $\prec$, see proposition 5.2.5. Alternatively, some authors call the dioids semirings. Indeed, if every element of D would have an opposite, D would have a ring structure.

Definition 5.2.4 (Path Algebra)**.** *A path algebra P, written $(P, \oplus, \otimes, \epsilon, \gamma)$, is an algebraic structure with the properties:*

1. *$(P, \oplus, \otimes, \epsilon, \gamma)$ is a dioid.*
2. *The operation $\oplus$ is idempotent:*

$$\forall a \in P, \quad a \oplus a = a \; . \tag{5.2.4}$$

5.2.2 The Natural Order

Proposition 5.2.5 (Order Relation)**.** *The relation $a \prec b \equiv (a \oplus b = a)$ is an order in the monoid $(P, \oplus, \epsilon)$ if the law $\oplus$ is idempotent.*

Proof.

1. the relation $\prec$ is reflexive, $(a \prec a)$ since $\oplus$ is idempotent.
2. the relation $\prec$ is transitive: Let us show that $(a \prec b) \wedge (b \prec c) \Rightarrow (a \prec c)$:

$$(a \prec b) \wedge (b \prec c) \equiv (a \oplus b = a) \wedge (b \oplus c = b) \; . \tag{5.2.5}$$

$$\Rightarrow (a \oplus (b \oplus c)) = a \; . \tag{5.2.6}$$

$$\Rightarrow ((a \oplus b) \oplus c)) = a \; . \tag{5.2.7}$$

$$\Rightarrow (a \oplus c) = a \equiv (a \prec c) \; . \tag{5.2.8}$$

3. the relation $\prec$ is antisymmetric $[(a \prec b) \wedge (b \prec a) \Rightarrow a = b]$:

$$(a \prec b) \Rightarrow \exists d \in P, a = b \oplus d \tag{5.2.9}$$

$$(b \prec a) \Rightarrow \exists c \in P, b = a \oplus c \tag{5.2.10}$$

$$[(a \prec b) \wedge (b \prec a)] \equiv [b = a \oplus c \wedge a = b \oplus d] \; . \tag{5.2.11}$$

$$a \oplus b = a \oplus (a \oplus c) = a \oplus c = b \; . \tag{5.2.12}$$

$$a \oplus b = (b \oplus d) \oplus b = b \oplus d = a \; . \tag{5.2.13}$$

Since we consider only path optimization problems, we restrict ourselves to the study of the path algebra with an idempotent law $\oplus$. However path enumeration problems and path existence problems can be solved in a similar algebraic structure having a law $\oplus$ which is not necessarily idempotent. For instance path existence problems can be solved using the classical Boolean algebra and path enumeration problems can be solved with a law $\oplus$ that corresponds to the union of sets.

The path algebra contains a partial order relation induced by the idempotent law $\oplus$ that will allow us to define a notion of distance over the set P. This order relation is often referred to as the natural order. Similar algebraic structures have already been treated in [43].

Notation	Algebra	Name	Applications
$\mathbb{R}_{max}$	$(\mathbb{R}^+ \cup \{-\infty\}, \max, +, -\infty, 0)$	max algebra	Longest Paths
$\mathbb{R}_{min}$	$(\mathbb{R}^+ \cup \{+\infty\}, \min, +, +\infty, 0)$	min algebra	Shortest Paths
$\mathbb{R}_{maxmin}$	$(\mathbb{R}^+ \cup \{+\infty\}, \max, \min, 0, +\infty)$	bottleneck algebra	Maximum Capacity Path
$\mathbb{R}_{minmax}$	$(\mathbb{R}^+ \cup \{+\infty\}, \min, \max, +\infty, 0)$	bottleneck algebra	Minimum Capacity Path
$\mathbb{P}$	$([0,1], \max, \times, 0, 1)$	Probability algebra	Markov Chains
$\mathbb{B}$	$(\{0,1\}, \max, \min, 0, 1)$	Boolean algebra	Path Existence

Table 5.1: Different path algebras with idempotent law $\oplus$.

Table 5.1 lists different instances of path algebras with their specific range of applications. However we are mainly focused on the class of optimization problems, i.e. the minimization or the maximization of some objective function related to a measure on paths. Optimization problems require an order relation of the elements of the path algebra. Otherwise the notions of minimization and maximization are meaningless. We will now detail how graphs are represented in path algebras and then give the general formulation of finding an optimal path in a graph.

5.2.3 Graphs Representation

We consider in this section the manipulation of graphs in a path algebra. It is thus necessary to define a suitable graph representation. As in ordinary linear algebra, the basic structures that we are considering are matrices. We introduce here the so called generalized incidence matrix of a graph. The generalized incidence matrix permits to encode a weighted graph by a matrix.

Definition 5.2.6 (Generalized Incidence Matrix)**.** *Given a graph $G = (V, E, W)$ and a path algebra $(P, \oplus, \otimes, \epsilon, \gamma)$ such that: $\forall w_{i,j} \in W, w_{i,j} \in P$; one can define a generalized incidence matrix $A = [a_{ij}]$ of G where each line of the matrix corresponds to a node of G, and each column corresponds to a node of G and such that:*

$$\begin{cases} a_{ij} = w_{i,j} \ \ if \ \ (i,j) \in E \ , \\ a_{ij} = \epsilon \ \ if \ \ (i,j) \notin E \ , \\ a_{ii} = \gamma. \end{cases} \tag{5.2.14}$$

Furthermore it is possible to define additions and multiplication as in an ordinary linear algebra. For $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ over $(P, \oplus, \otimes, \epsilon, \gamma)$, we define the following operations:

$$\begin{cases} A \oplus B = [a_{ij} \oplus b_{ij}] \ (addition). \\ A \otimes B = [\bigoplus_{k=1}^{n} a_{ik} \otimes b_{kj}] \ (multiplication). \end{cases} \tag{5.2.15}$$

Matrices operations in path algebras highlight some particular convergence properties, especially for the sequence of matrices products $A^n = A \otimes A ... \otimes A$. The path optimization problems will also often be reduced to the computation of a particular power of the generalized incidence matrix A of a given graph. However we are not going to detail here these convergence properties; we refer instead the readers to the textbook [43] for more details on matrix manipulations in dioids. Some of these properties are illustrated in the next sections through simple examples.

5.2.4 Example

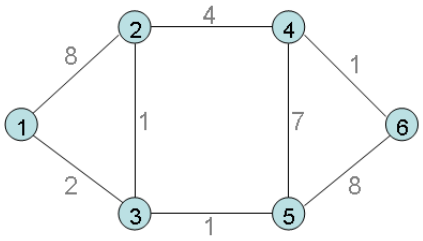


Figure 5.1: Example Graph.

Let us consider the graph displayed on figure 5.1. We are going to show on this graph what are the generalized incidence matrices operations in paths algebras. The graph is composed of 6 nodes $\{1, 2, 3, 4, 5, 6\}$. Given that we consider non oriented graphs, we are not making any differences between the arc (i, j) and (j, i) even if the generalized incidence matrix of a graph can take into account the difference. As we have mentioned earlier, in any other case, we will speak of directed or oriented edges. The generalized incidence matrix of the graph G can be written in the path algebra $(\mathbb{R}^+ \cup \{-\infty\}, \max, +, -\infty, 0)$ as:

$$L = [l_{ij}] = \begin{pmatrix} 0 & 8 & 2 & -\infty & -\infty & -\infty \\ 8 & 0 & 1 & 4 & -\infty & -\infty \\ 2 & 1 & 0 & -\infty & 1 & -\infty \\ -\infty & 4 & -\infty & 0 & 7 & 1 \\ -\infty & -\infty & 1 & 7 & 0 & 8 \\ -\infty & -\infty & -\infty & 1 & 8 & 0 \end{pmatrix}$$

This symmetric incidence matrix can be manipulated as any common matrix in an algebraic space. For instance, the multiplication of the matrix L by itself in $\mathbb{R}_{max}$, i.e. its square, is equal to:

$$L \otimes L = L^2 = [\max_{k \in \{1,..,n\}} (a_{ik} + a_{kj})] = \begin{pmatrix} 16 & 8 & 9 & 12 & 3 & -\infty \\ 8 & 16 & 10 & 4 & 11 & 5 \\ 9 & 10 & 4 & 8 & 1 & 9 \\ 12 & 4 & 8 & 14 & 9 & 15 \\ 3 & 11 & 1 & 9 & 16 & 8 \\ -\infty & 5 & 9 & 15 & 8 & 16 \end{pmatrix}$$

The algebra $\mathbb{R}_{max}$ can be used to compute the longest paths between two nodes containing a given number of edges. In our example, elements of the matrix L^2 indicate the length of the longest path between two nodes of the graph containing at most two edges. We detail in the next section a general formalization of the shortest path problem. We also show that minimal capacity paths or classical shortest paths are only some particular cases of this general linear problem.

Matrices manipulation in a path algebra is thus similar as in ordinary linear algebra. This particular path algebra has been extensively studied in the last years in connection with system theory and opti-

mal control. Some important properties of this algebraic structure have been highlighted by the group "Max Plus", which is a collective name for a working group on the so called max-plus algebra at INRIA Rocquencourt [39, 38]. The path algebra structure provides a suitable framework for a large variety of problems. However we are not going to detail here all the applications of path algebras and other related algebraic structures. The field of application of this theory is too large to be enclosed in a single thesis.

5.3 Generalized Shortest Path

This section presents the generalized shortest path problem. The aim of this section is to derive several common path optimization problems from a single system of linear equations. We will see how the shortest path problem, the maximal capacity path problem and many others path optimization problems can be obtained as a particular instance of the same abstract problem.

5.3.1 Definition

Let us consider a path algebra $(P, \oplus, \otimes, \epsilon, \gamma)$, and a weighted graph $G = (V, E, W)$. The generalized shortest path problem consists in finding a path $\pi_{(v_f, v_b)}$, between two nodes (v_f, v_b) such that the generalized length of the path $L_n(\pi_{(v_f,v_b)})$ is minimal. The problem of the generalized shortest path can be written as the following minimization problem:

$$\begin{cases} L_n(\pi_{(v_f,v_f)}) = \gamma \ , \\ L_n(\pi_{(v_f,j)}) = \bigoplus\limits_{j \in \Gamma^{-1}(i)} (L_n(\pi_{(v_f,i)}) \otimes w_{i,j}), \ \ if \ \ i \neq v_f \ . \end{cases} \tag{5.3.1}$$

where $\Gamma^{-1}(i)$ is the set of adjacent nodes of the node i. We can consider that non-existent arcs (i, j) have an "infinite" weight $w_{i,j} = \epsilon$.

We can finally write the shortest path problem as the following linear equations:

$$\begin{cases} L_n(\pi_{(v_f,v_f)}) = (\bigoplus\limits_{j \in V} L_n(\pi_{(v_f,j)}) \otimes w_{j,v_f}) \oplus \gamma \ , \\ L_n(\pi_{(v_f,i)}) = (\bigoplus\limits_{j \in V} L_n(\pi_{(v_f,i)}) \otimes w_{i,j}) \oplus \epsilon, \ \ if \ \ i \neq v_f \ . \end{cases} \tag{5.3.2}$$

The previous formulation can be seen dynamically as a recursive formulation of the shortest path problem. The formulation expresses that a shortest paths from a source node to a destination node can be found iteratively by traveling backwards from the destination node through the closest nodes of the source. We can reformulate the shortest path problem in the defined path algebra to highlight the linearity of the shortest path equations:

$$\Pi = \Pi \otimes L \oplus B \ , \tag{5.3.3}$$

where L is the generalized incidence matrix of G in the path algebra $(P, \oplus, \otimes, \epsilon, \gamma)$, Π is the vector of shortest paths lengths from v_f, and B is the vector:

$$B = \begin{vmatrix} \gamma \\ \epsilon \\ . \\ . \\ . \\ \epsilon \end{vmatrix} .$$

Node indices have been reordered in such a way that the node v_f corresponds to the first line and first column of the generalized incidence matrix. Under this formulation, solving the shortest path problem is thus equivalent to solving a system of linear equations. This formulation of the problem is very interesting since we can use classical tools of linear algebra to solve the shortest path problem.

5.3.2 Algebraic Tools

Methods for solving linear systems of equations are numerous. However there are two famous methods: the Jacobi method and the Gauss-Seidel method. The Jacobi method is an iterative algorithm for determining the solutions of a system of linear equations with diagonally dominant matrices, i.e. $\forall i, \|a_{i,i}\| > \sum_{j \neq i} \|a_{i,j}\|$. The Gauss-Seidel method can be seen as an improved version of the Jacobi method. It is defined on matrices with non-zero diagonals; convergence is guaranteed if the matrix is either diagonally dominant or symmetric and positive definite. For instance, one can apply the Jacobi method to solve the shortest paths linear system of equations. The Jacobi method in a path algebra can be described by the following iterative scheme:

- First $\Pi^0 = B$,
- Then at the k^{th} iteration, $\Pi^k = \Pi^{k-1} \otimes L \oplus B$,
- repeat the previous step until $\Pi^k = \Pi^{k-1}$,

Π^k represents the vector whose elements are the shortest paths lengths containing up to k edges. The iterative scheme converges to Π^n which is the vector of shortest path lengths. The previous method can thus be used to solve the single source shortest paths problem. The same scheme can also be used to solve the all pairs shortest paths problem, i.e the shortest paths between each pair of nodes. It is sufficient to notice that the single source shortest paths takes into account only a single row of the generalized incidence matrix (the row of the source node) to compute the shortest paths. Replacing the vector Π by the whole incidence matrix, we obtain the following iterative scheme:

- First $L^0 = L$,
- Then at the k^{th} iteration, $L^k = L^{k-1} \otimes L$,
- repeat the previous step until $L^k = L^{k-1}$,

At the end, the elements $[l_{ij}]$of the matrix L^k indicates the shortest path length from node i to node j. As mentioned previously the all pairs shortest paths problem is here reduced to the computation of a sequence of products $L^n = L \otimes L ... \otimes L$.

Algebraic tools can thus be exploited to solve common graph problems. However the use of the matrix representation of a graph is not optimal to handle graphs that contain numerous nodes and edges. 3D images often contain millions of nodes and a matrix representation of a graph becomes inefficient or simply impossible to use in practice. It is thus necessary to define another solver that does not need a matrix representation of the graph. We detail in the next section the so-called generalized Dijkstra algorithm.

5.3.3 Generalized Dijkstra Algorithm

We present here the so-called generalized Dijkstra algorithm that takes advantage of the path algebra point of view of many graph path optimization problems. This algorithm is a direct extension of the classical Dijkstra algorithm, described in the path algebra framework [28, 23]. The recursive formalization of paths optimization problems highlights the fact that an optimal path can be found by recursively analyzing the graph, and updating the set of shortest paths. The pseudo code of the algorithm is given bellow.

This algorithm can be applied on any graphs represented in the path algebra $(P, \oplus, \otimes, \epsilon, \gamma)$ as soon as there is an order relation at our disposal. For instance, if the law $\oplus$ is idempotent, one can use the natural order relation $\prec$. The Dijkstra algorithm can thus solve the shortest path problem as already mentioned in chapter 1. However this algorithm can be used for a fairly general class of optimization problem. Finally another interesting property is that the generalized Dijsktra algorithm outputs a shortest path tree.

Algorithm 5 Generalized Dijkstra's algorithm

Input: Graph G, Root Vertex v_f, compare $\prec$, combine $\otimes$, infinite distance ϵ, zero distance γ
Output: Predecessor Map $PredMap$, Distance Map $Distmap$

```
for all vertex v in G do
  Distmap[v] := ε
  PredMap[v] := v
  color[v] := WHITE
end for
Distmap[v_f] := γ
INSERT(Queue,v_f,Distmap[v_f])
color[s] = GRAY
while Queue is not empty do
  u := EXTRACT-MIN(Queue)
  for all vertex v adjacent to u do
    if (w_{u,v} ⊗ Distmap[u]) ≺ Distmap[v] then
      Distmap[v] := (w_{u,v} ⊗ Distmap[u])
      PredMap[v] := u
      if color[v] = WHITE then
        color[v] := GRAY
        INSERT(Queue, v, Distmap[v])
      else
        if color[v] = GRAY then
          DECREASEPRIORITY(Queue, v)
        end if
      end if
    end if
  end for
end while
```

5.3.4 Examples

The Dijkstra algorithm can thus solve the shortest path problem in a path algebra having an order relation. Obviously many common path optimization problems can be formalized in such ordered path algebras.

Shortest L_1 path

The classical shortest path problem (with the $L_1()$ norm) can be formalized in the path algebra as the following minimization problem:

$$\begin{cases} L_1(\pi_{(v_f,v_f)}) = 0 \ , \\ L_1(\pi_{(v_f,j)}) = \min\limits_{j\in\Gamma^{-1}(i)}(L_1(\pi_{(v_f,i)}) + w_{i,j}), \ \ if \ \ i \neq v_f \ . \end{cases} \tag{5.3.4}$$

Non-existent arcs (i,j) are given an infinite weight $w_{i,j} = \infty$. Let us now consider the algebraic structure $\mathbb{R}_{min}$ with the following operations:

$$\begin{cases} a \oplus b = \min(a,b) \ . \\ a \otimes b = a + b \ . \end{cases} \tag{5.3.5}$$

The shortest path problem is thus equivalent to :

$$\begin{cases} L_1(\pi_{(v_f,v_f)}) = \min[\min\limits_{j\in V}(L_1(\pi_{(v_f,j)}) + w_{j,v_f}), 0] = (\bigoplus\limits_{j\in V} L_1(\pi_{(v_f,j)}) \otimes w_{j,v_f}) \oplus \{0\} \ , \\ L_1(\pi_{(v_f,j)}) = \min[\min\limits_{j\in V}(L_1(\pi_{(v_f,i)}) + w_{i,j}), +\infty] = (\bigoplus\limits_{j\in V} L_1(\pi_{(v_f,i)}) \otimes w_{i,j}) \oplus \{+\infty\}, \ \ if \ \ i \neq v_f \ . \end{cases} \tag{5.3.6}$$

The shortest (L_1) path problem is a particular case of the generalized shortest path problem in $(R^+ \cup \{+\infty\}, \min, +, +\infty, 0)$. Moreover the natural order derived from the law min is the usual order relation $\leq$.

Minimal Capacity Path (L_∞)

The problem of the minimal capacity path can be written as a shortest path with the L_∞ norm:

$$\begin{cases} L_\infty(\pi_{(v_f,v_f)}) = 0 \ , \\ L_\infty(\pi_{(v_f,j)}) = \min\limits_{j\in\Gamma^{-1}(i)}(\max(L_\infty(\pi_{(v_f,i)}), w_{i,j})), \ \ if \ \ i \neq v_f \ . \end{cases} \tag{5.3.7}$$

By convention, non-existent arcs (i,j) have an infinite weight $w_{i,j} = \infty$. Considering now the following operations:

$$\begin{cases} a \oplus b = \min(a,b) \ . \\ a \otimes b = \max(a,b) \ , \end{cases} \tag{5.3.8}$$

we can reformulate the minimal capacity path problem as the following minimization problem:

$$\begin{cases} L_\infty(\pi_{(v_f,v_f)}) = \min[\min\limits_{j\in V}(\max(L_\infty(\pi_{(v_f,j)}), w_{j,s})), 0] = (\bigoplus\limits_{j\in V} L_\infty(\pi_{(v_f,j)}) \otimes w_{j,s}) \oplus \{0\} \ , \\ L_\infty(\pi_{(v_f,j)}) = \min[\min\limits_{j\in V}(\max(L_\infty(\pi_{(v_f,i)}), w_{i,j})), +\infty] = (\bigoplus\limits_{j\in V} L_\infty(\pi_{(v_f,i)}) \otimes w_{i,j}) \oplus \{+\infty\}, \ \ if \ \ i \neq v_f \ . \end{cases} \tag{5.3.9}$$

The minimal capacity path is also a particular case of the generalized shortest path problem and can thus be solved with the Dijkstra algorithm. Moreover a minimal capacity paths tree is also a minimal spanning tree as described in chapter 1.

Examples

The generalized Dijkstra algorithm can be used to solve a fairly general class of path optimization problems. This single algorithm can be especially used to solve the following problems:

- Shortest Path : DIJKSTRA(G, s, $\leq$, +, $+\infty$, 0). Edges weights represent the length between two nodes.
- Minimal capacity path : DIJKSTRA(G, s, $\leq$, max, $+\infty$, 0). Edges weights represent the capacity of some commodity that can flow through the edges.
- Maximal capacity path : DIJKSTRA(G, s, $\geq$, min, 0, $+\infty$). Edges weights represent the capacity of some commodity that can flow through the edges.
- Most reliable path : DIJKSTRA(G, s, $\leq$, $\times$, 1, 0). Edges weights represent the probability to travel from one node to the other.
- Generalized shortest path : DIJKSTRA(G, s, $\preceq$, $\otimes$, ϵ, γ). Edges weights represent any values or set of values that can be ordered using the law $\preceq$.

The results based on optimal path presented in this thesis have been obtained with this algorithm. The next section is devoted to the Watershed transform of mathematical morphology. We first recall the different definitions of the watershed transform based on optimal paths and present the related path algebras. We detail how the watershed transform can be formalized in the path algebra framework. We also show how the watershed on graphs can be extended to the case of vector valued images using lexicographical distances.

5.4 The Watershed Transform on Graphs

The watershed transform is one of the most powerful tools for image segmentation. The method was originally proposed by Beucher and Lantuéjoul in 1979 [11]. Since its creation, the algorithm and its related theory have always been in active development. We detail in the following sections the different methods related to the watershed transform based on graph paths optimization. We also detail these methods in the path algebra framework and extend it to vector valued images.

5.4.1 Basics

The intuitive idea underlying the watershed transform comes from geography: the image is seen as a topographic relief, where the pixel grey levels are related to the height of the points. The relief is then flooded by rain and watersheds delimit the different regions where a drop of water will fall [90]. An alternative view is to imagine the topographic surface being immersed by lakes coming from the local minima. The lakes or catchment basins fill up the relief with water starting at these local minima, and, at points where water coming from different basins meet, a watershed line is built. When the water level has reached the highest level of the topographic surface, the process is stopped. As a result, the image is partitioned into regions. The name "watershed transform" denotes especially a labeling of the image, such that all points of a given lake have a same and unique label. Note also that in practice one applies the watershed transform to the gradient of an image [74]. This method produces watersheds at the points of high contrast as it is commonly desired for image segmentation purposes. A simple flooding process is illustrated in figure 5.2.

One of the difficulties with the watershed concept is that it leads to different formalizations and definitions. However in this section we consider the watershed transform of a graph as the influence zones of the local minima relative to some specific distances, as proposed by Meyer in [72, 73]. We refer the reader to [87] for a detailed survey about algorithms and definitions of the watershed transform.

5.4.2 Topographic Distances

The first watershed definition is based on the topographic distances of a continuous function. We restrict ourselves here to the definition given in [72] and [80], but other choices may be considered [85]. Assume

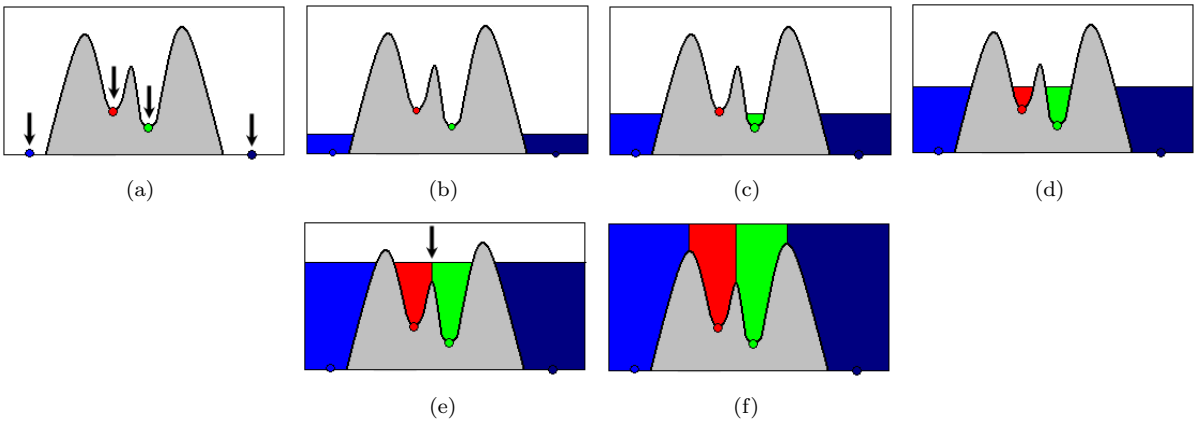


Figure 5.2: Watershed segmentation by flooding. A topographic surface is flooded from its local minima, the result is a partition of the surface.

that the image I is a real twice continuously differentiable function. The topographical distance between points p and q lying in the domain of definition of I is given by:

$$T_f(p,q) = \min_{\pi \in \Pi_{(p,q)}} \int_{\pi} \|\nabla I(s)\| ds \tag{5.4.1}$$

where the minimum is taken over all paths $\Pi_{(p,q)}$ from point p to point q. This equation leads to the following precise definition of the watershed transform.

Definition 5.4.1 (Continuous Watershed Transform)**.** *Let I be a real twice continuously differentiable function with distinct local minima $\{m_i\}$. The catchment basin $CB(m_i)$ of a minimum m_i is defined as the set of points which are topographically closer to m_i than to any other regional minimum m_j. The watershed transform is then defined by the labeling of the catchment basins.*

This definition can also be extended to discrete spaces and digital images described by a graph. Assume that a digital image I is given by its pixel graph $G = (V, E, W)$, where each node of the graph is a pixel of the image. We detail now an alternate definition of the watershed transform for digital images through the notion of lower slope. The lower slope $LS(p)$ of I at a pixel p, is defined as the maximal slope linking p to any of its neighbors of lower altitude:

$$LS(p) = \max_{q \in N_p \cup p} (I(p) - I(q)) \tag{5.4.2}$$

where N_p is the set of neighbors of pixel p in the graph $G = (V, E, W)$. Note that pixels whose neighbors are all of higher grey values have a null slope. We define then the cost $c_{p,q}$ of traveling from pixel p to a neighbor pixel q as:

$$c_{p,q} = \begin{cases} d(p,q).LS(p), & if \;\; I(p) > I(q) \;, \\ d(p,q).LS(q), & if \;\; I(q) > I(p) \;, \\ d(p,q).\frac{(LS(p)+LS(q))}{2}, & if \;\; I(p) = I(p) \;. \end{cases} \tag{5.4.3}$$

where $d(p,q)$ is the geometric distance between pixel p and q. In the following, we assume for simplicity that $d(p,q) = 1$. From these notions, we define the topographic distance between two pixels p and q as:

$$T(p,q) = \min_{\pi \in \Pi_{(p,q)}} \left(\sum_{e_{i,j} \in \pi} c_{i,j} \right) = \min_{\pi \in \Pi_{(p,q)}} (L_1(\pi)) \tag{5.4.4}$$

Proposition 5.4.2 (Geodesics of the topographical distance)**.** *Let $I(p) > I(q)$, a path $\pi_{p,q}$ is a path of steepest descent if and only if $T(p,q) = (I(p) - I(q))$. The paths of steepest descent are thus the geodesics (shortest paths) of the topographic distance.*

According to the topographic distance, we can state that a pixel p will be assigned the label of a minima $m_i \in M$, where M is the set of minima of I, if and only if:

$$\forall m_k \in M - \{m_i\}, \min_{\pi \in \Pi_{(m_i,p)}} (I(m_i) + L_1(\pi)) < \min_{\pi \in \Pi_{(m_k,p)}} (I(m_k) + L_1(\pi)) \tag{5.4.5}$$

This proposition leads to the following definition of the edges weights: the cost $w_{p,q}$ of traveling from pixel p to a neighbor pixel q is defined as :

$$w_{p,q} = \begin{cases} LS(p) + I(q), & if \;\; I(p) > I(q) \;, \\ LS(q) + I(p), & if \;\; I(q) > I(p) \;, \\ \max(LS(p), LS(q)) + I(p), & if \;\; I(p) = I(p) \;. \end{cases} \tag{5.4.6}$$

The watershed transform of a digital image is finally restated in the following way:

Definition 5.4.3 (Topographic Watershed Transform)**.** *Assume that a digital image I is given by a graph $G = (V, E, W)$ where the edges E are weighted according to equation 5.4.6. Let $\{m_i\}$ be distinct nodes of G representing the minima of I. The watershed transform is defined by the labeling of the trees of a shortest path forest of the topographical distance rooted on the minima m_i.*

The last definition holds for any "lower complete" digital image, which means that each pixel which is not in a minimum has a neighbor of lower grey value [74]. This definition can thus not be used for the watershed transform from arbitrary sources of flooding. Note also that we can link the previous definition in terms of a shortest path in the path algebra $(R^+ \cup \{+\infty\}, \min, +, +\infty, 0)$, since the watershed transform is simply a shortest path forest rooted on the minima.

We will now extend the definition of the watershed transform to deal with arbitrary images (not only lower complete) as well as arbitrary sources of flooding.

5.4.3 Ultrametric Flooding Distances

An interesting use of the watershed transform consist in specifying the sources of the flooding. Instead of simulating a flooding from all the minima, the flooding is constrained to start from some specified points. This technique allows to obtain a segmentation of the image containing as much regions as markers specified for the flooding process. This method is illustrated in figure 5.3.

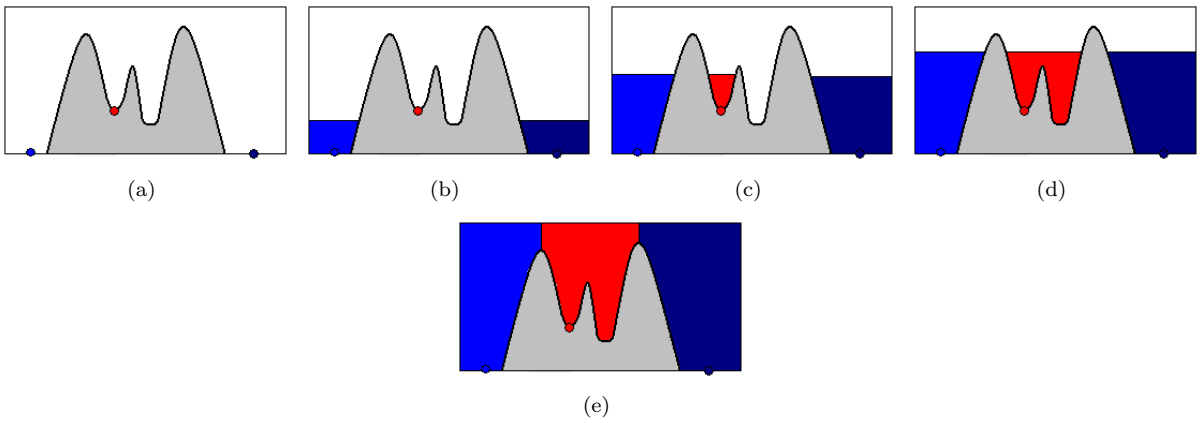


Figure 5.3: Watershed segmentation from markers. A topographic surface is being flooded from selected markers, the result is a partition of the image.

An interesting property of the flooding is the lowest level of flooding for which two points p and q of the image belong to the same lake. This lowest level is given by:

$$d_\infty(p,q) = \min_{\pi \in \Pi_{(p,q)}} \left(\max_{\pi}(I(p), I(k), I(k+1), ..., I(q)) \right) . \tag{5.4.7}$$

One can especially note that the lowest level $d_\infty(p,q)$ is also an ultrametric distance between two points of the image:

$$\begin{cases} d_\infty(p,p) = 0 , \\ d_\infty(p,q) = d_\infty(q,p) , \\ \forall p,q,r, \;\; d_\infty(p,r) \leq \max(d_\infty(p,q), d_\infty(q,r)) . \end{cases} \tag{5.4.8}$$

We define then the cost $w_{p,q}$ of traveling from pixel p to a neighbor pixel q as :

$$w_{p,q} = \max(I(p), I(q)) . \tag{5.4.9}$$

From the flooding ultrametric distance, we define the cost or the length of a path $\pi_{p,q}$ as the L_∞ distance of a path:

$$L_\infty(\pi_{(p,q)}) = \left(\max_{e_{i,j} \in \pi_{(p,q)}} (w_{i,j}) \right) \tag{5.4.10}$$

The L_∞ distance of a path gives explicitly the lowest level of flooding for which two points belong to the same lake. The L_∞ distance of pathes yield an ultrametric distance too. According to the flooding simulation from a set of specified markers M, we can state that a pixel p will be assigned the label of a marker $m_i \in M$ if and only if:

$$\forall m_k \in M - \{m_i\}, \min_{\pi \in \Pi_{(m_i,p)}} (L_\infty(\pi)) < \min_{\pi \in \Pi_{(m_k,p)}} (L_\infty(\pi)) \tag{5.4.11}$$

Proposition 5.4.4 (Geodesics of the flooding distance)**.** *The geodesics of the flooding distance are minimax paths of the graph weighted according to equation 5.4.9. In other words, any node p of a graph $G = (V, E, W)$ is assigned the label of a marker m_i after a flooding if and only if :*

$$\pi_{(p,m_i)} \in \operatorname*{argmin}_{\pi \in \Pi_{(p,m_i)}} (L_\infty(\pi)) , \tag{5.4.12}$$

The last property is exactly the paths characterization of minimal spanning trees, as detailed in chapter 1. We can thus propose the following definition of the digital watershed transform from markers:

Definition 5.4.5 (Ultrametric Watershed Transform)**.** *Assume that a digital image I is given by a graph $G = (V, E, W)$ where the edges E are weighted according to equation 5.4.9. Let $\{m_i\}$ be k distinct nodes of G representing some sources for the flooding process. The watershed transform is defined by the labeling of the trees of a minimal spanning forest rooted on the markers m_i.*

The last definition can also be formulated in terms of a shortest path in the bottleneck path algebra $\mathbb{R}_{minmax}$, since the watershed transform is in this case a minimal spanning forest rooted on the markers. Moreover the paths of a minimal spanning forest are minimax paths introduced in chapter 1.

5.4.4 Bi-criteria Distances

We address now the problem of flat zones segmentation for the watershed transform. The previous definitions of the watershed transform only hold for lower complete images. We relax now this restriction and propose a definition allowing the segmentation of flat zones in an image according to influence zones of the lakes, as illustrated in figure 5.4. We introduce a solution to the flat zones problem through lexicographical orders as proposed by Lotufo et al. [66].

The simple knowledge of the height of a node or the distance of this node to a marker is not sufficient to express completely the flooding process. We propose here a combination of these two notions. We define the cost $w_{p,q}$ of traveling from a pixel p to a neighbor pixel q as the vector:

$$w_{p,q} = (\max(I(p), I(q)), d(p,q)) . \tag{5.4.13}$$

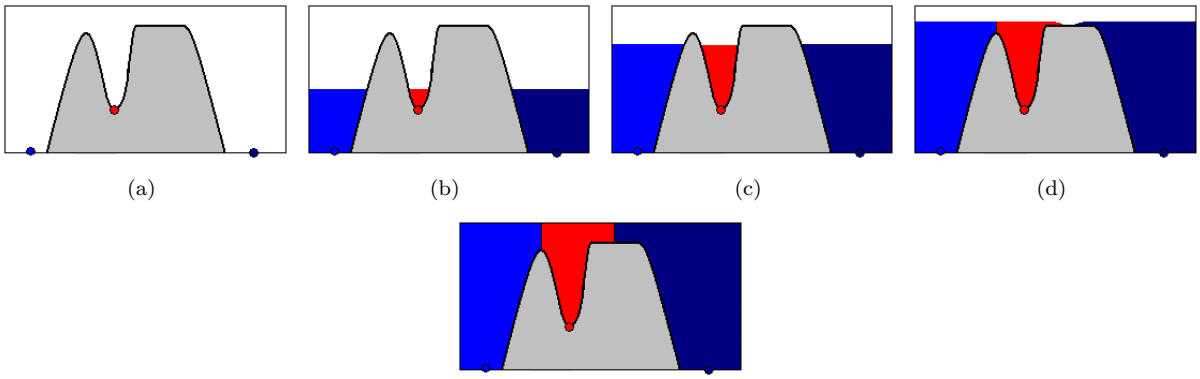


Figure 5.4: Watershed Segmentation from markers : Plateau Problems. A topographic surface is being flooded from selected markers. When two different lakes are about to flood a single flat zone, the resulting partition is obtained according to the distance between a point of the flat zone and the closest lake. The partition of the flat zones is thus obtained according to the zones of influences of the lakes.

where $d(p,q)$ is the geometric distance between pixels p and q. In the following, we assume for simplicity that $d(p,q) = 1$. We also define the following lexicographical order on the set of edges weights:

$$(a_1, a_2) \prec (b_1, b_2) \ . \tag{5.4.14}$$

$$\Leftrightarrow a_1 < b_1 \ , \tag{5.4.15}$$

$$or a_2 < b_2 \ . \tag{5.4.16}$$

The corresponding natural order relation is the usual lexicographical order. We create now the path algebra $((R^+ \cup \{+\infty\})^2, \oplus, \otimes, (\infty,\infty), (0,0))$ where the two operators $\oplus$ and $\otimes$ are defined as follows:

- the operation $\oplus$, called the addition, is defined by:

$$\forall (a,b) \in ((R^+ \cup \{+\infty\})^2, a \oplus b = \left\{ \begin{array}{l} a \ if \ a \preceq b \ . \\ b \ if \ b \preceq a \ . \end{array} \right. \tag{5.4.17}$$

  Its neutral element, or zero element, is (∞,∞) since $a \oplus (\infty,\infty) = a$. Note also that $\oplus$ is idempotent. Remark: this operation $\oplus$ is the equivalent of a "min" operation.

- the operation $\otimes$, called the multiplication, is defined by:

$$\forall (a,b) \in ((R^+ \cup \{+\infty\})^2, \ \ a \otimes b = [\max(a_1, b_1), a_2 + b_2] \ , \tag{5.4.18}$$

  The operation $\otimes$ has a neutral element, or identity element $(0,0)$ since $a \otimes (0,0) = (0,0) \otimes a = a$. Note also that (∞,∞) is an absorbing element for $\otimes$ since $a \otimes (\infty,\infty) = (\infty,\infty) \otimes a = (\infty,\infty)$.

As a consequence, the algebraic structure $((R^+ \cup \{+\infty\})^2, \oplus, \otimes, (\infty,\infty), (0,0))$ is a path algebra. Note that the operation $\otimes$ is designed so that the length (according to equation 5.3.2) of a path takes into account the distance to the markers on the flat zones. Besides, the length of a path takes only into account the maximum height (the ultrametric flooding distance) of a point in non-flat zones. Note also that the length of a path is not a scalar but a vector, and distances are sorted lexicographically. According to this lexicographical distance, we can state that a pixel p will be assigned the label of a marker $m_i \in M$ if and only if:

$$\forall m_k \in M - \{m_i\}, \ \min_{\pi \in \Pi_{(m_i,p)}}{}_{\text{lex2}} \ (L_{lex2}(\pi)) \prec \min_{\pi \in \Pi_{(m_k,p)}}{}_{\text{lex2}} \ (L_{lex2}(\pi)) \tag{5.4.19}$$

where min_{lex2} is the minimum according to the lexicographical order $\prec$ and $L_{lex2}(\pi)$ is the lexicographical length of the path π defined as:

$$L_{lex2}(\pi) = \bigoplus_{e_{(i,i+1)} \in \pi} \left(w_{(i,i+1)} \otimes w_{(i+1,i+2)} \right) \tag{5.4.20}$$

where π is a path written $\pi = \{e_{(i,i+1)}, e_{(i+1,i+2)}, ..., e_{(i+n-1,i+n)}\}$. This path algebra gives rise to the following definition of the watershed transform.

Definition 5.4.6 (Bi-Criteria Watershed Transform)**.** *Let a digital image I be given by a graph $G = (V, E, W)$ where the edges E are weighted according to equation 5.4.13. Let $\{m_i\}$ be distinct nodes of G representing the sources for the flooding process. The watershed transform is defined by the labeling of the trees of a shortest path forest rooted on the markers m_i in the path algebra $((R^+ \cup \{+\infty\})^2, \oplus, \otimes, (\infty, \infty), (0, 0))$.*

5.4.5 Lexicographical Distances

We discuss now the definition proposed by Meyer in [73], using another lexicographical order to manage competing flooding from different sources. This situation appears when the watershed is computed from a set of arbitrary markers. This order appears naturally with Meyer's flooding algorithm based on hierarchical queues [71]. The idea behind this lexicographical order is to rank the different lakes according to the paths that the flooding has taken. In order to understand this ranking, let us consider the following example, illustrated in figure 5.5.

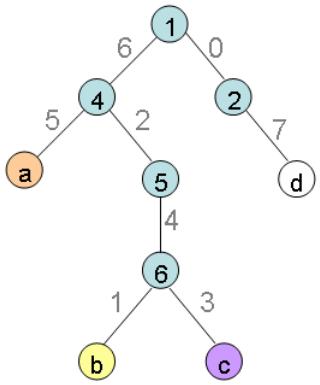


Figure 5.5: The figure illustrates the graph used for our example. The flooding markers are represented by colored nodes.

The markers for the flooding are the nodes a, b, c, d. The questions that we address are the following: first, what label will the node 1 be assigned to? and second, is there any distance such that the nodes are assigned a label according to a flooding from these markers ?

By simulating a flooding of this graph it is obvious that the nodes will be labeled as illustrated in figure 5.6. First the lake coming from marker b reaches the node 6 before the lake coming from marker c, because the altitude between b and 6 is lower than the altitude between c and 6. Later on, since the lake coming from source b also reaches the first node 4, node 1 will be finally assigned the label of the marker b leading to the illustrated labeling in figure 5.6. Let us check that the previously defined distances cannot manage the presented situation:

- The ultrametric flooding distances from node 1 to nodes a, b, c are all equal to 6. The flooding distance is thus myopic and cannot discriminate the different lakes arising from node a, b and c. Only the lake coming from source d is discriminated by the ultrametric flooding distance.

- The bi-criteria distance ranks the lakes coming from sources a, b and c by counting the number of edges (all edges having a unit length) along the paths from markers to node 1. The bi-criteria distance considers thus that node 1 will be reached first by the lake coming from a and does not make any difference between the lakes coming from b and c. This ordering is not the one obtained by simulating the flooding.

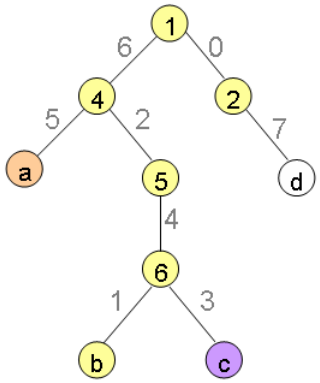


Figure 5.6: The figure illustrates the graph used for our example. The markers for the flooding are represented by colored nodes.

Meyer noticed that the distance which is minimal along the winning paths is in fact a lexicographic distance. The lexicographic length of a path is defined as a sequence $(\lambda_1, \lambda_2, ..., \lambda_n)$ of decreasing weights $(\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n)$. The largest weight λ_1 is the maximum edge weight along the path, λ_2 is the second maximum edge weight along the path and so on. The lexicographical order between paths permits finally to discriminate the flooding coming from different markers.

Definition 5.4.7 (Lexicographical Order Relation)**.** *The order relation* $\prec$ *defined by:*

$$a = (a_1, a_2, ..., a_n) \prec b = (b_1, b_2, ..., b_n) \ . \tag{5.4.21}$$

$$\Leftrightarrow a_1 < b_1 \ , \tag{5.4.22}$$

$$or \ \ \exists s \in \{1, ..., n-1\}, \ \forall i \in \{1, ..., s\}, \ a_i = b_i, \ a_{s+1} < b_{s+1} \ . \tag{5.4.23}$$

is called the lexicographical order.

From this lexicographical order we create the path algebra $((R^+ \cup \{+\infty\})^n, \boxplus, \boxtimes, +\infty, 0)$, where the two operators $\boxplus$ and $\boxtimes$ are defined by:

- the operation $\boxplus$, called the addition, is defined by:

$$\forall (a, b) \in ((R^+ \cup \{+\infty\})^n, a \boxplus b = \begin{cases} a \ if \ a \preceq b \ . \\ b \ if \ b \preceq a \ . \end{cases} \tag{5.4.24}$$

  its neutral element, or the zero element is ∞ since $a \boxplus \infty = a$. Note also that $\boxplus$ is idempotent and that the lexicographical $\prec$ corresponds to the natural order for the law $\boxplus$.

- the operation $\boxtimes$ is called the multiplication and is defined by:
  - if $b_1 > a_1$, $a \boxtimes b = b$.
  - if $a_n > b_1$, $a \boxtimes b = (a_1, a_2, ..., a_n, b_1, b_2, ..., b_n)$.
  - else let j be the highest index verifying $b_{j-1} \geq b_1 \geq a_j$: $a \boxtimes b = (a_1, a_2, ..., a_{j-1}, b_1, b_2, ..., b_n)$.

The operation $\boxtimes$ has a neutral element, or identity element 0 since $a \boxtimes 0 = 0 \boxtimes a = a$. Note also that ∞ is an absorbing element for $\boxtimes$ since $a \boxtimes \infty = \infty \boxtimes a = \infty$.

As a consequence the algebraic structure $((R^+ \cup \{+\infty\})^n, \boxplus, \boxtimes, +\infty, 0)$ is a path algebra and the shortest path according to the lexicographical length can be solved by the generalized Dijkstra algorithm. According to this lexicographical distance, we can state that a pixel p we will be assigned the label of a marker $m_i \in M$ if and only if:

$$\forall m_k \in M - \{m_i\}, \min_{\pi \in \Pi_{(m_i,p)}}{}_{\text{lex}} \left(L_{lex}(\pi)\right) \prec \min_{\pi \in \Pi_{(m_k,p)}}{}_{\text{lex}} \left(L_{lex}(\pi)\right) \tag{5.4.25}$$

where min_{lex} is the minimum according to the lexicographical order $\prec$ and $L_{lex}(\pi)$ is the lexicographical length of the path π defined as:

$$L_{lex2}(\pi) = \boxplus_{(e_{(i,i+1)} \in \pi)} \left(w_{(i,i+1)} \boxtimes w_{(i+1,i+2)}\right) \tag{5.4.26}$$

where π is a path written $\pi = \{e_{(i,i+1)}, e_{(i+1,i+2)}, ..., e_{(i+n-1,i+n)}\}$. This path algebra leads to the following definition of the watershed transform.

Definition 5.4.8 (Lexicographic Watershed Transform)**.** *Assume that a digital image I is given by a graph $G = (V, E, W)$ where the edges E are weighted according to equation 5.4.13. Let $\{m_i\}$ be distinct nodes of G representing the sources for the flooding process. The watershed transform is defined by the labeling of the trees of a shortest path forest rooted on the markers m_i in the path algebra $((R^+ \cup \{+\infty\})^n, \boxplus, \boxtimes, +\infty, 0)$.*

5.4.6 Comparison of Different Watershed Transforms

This section illustrates the results obtained with the different definitions of the watershed transform. The first example, in figure 5.7, illustrates the different watershed definitions on a simple image. Two different markers for the watershed transform lie in the same flat zone. The example make it evident that only the bi-criteria and lexicographical distances produce the same segmentation as a flooding simulation using hierarchical queues [71]. The segmentation in the flat zones is composed of the influence zones of the markers. The minimal spanning forest computed with the ultrametric flooding distance does not permit to obtain this desired segmentation in the flat zones. Obviously all the given definitions allow the detection of the main objects in the image.

The next example, displayed in figure 5.8, illustrates the watershed transform on a natural image. The example highlights that the bi-criteria, the lexicographical and the ultrametric flooding distances produce the same segmentations as a flooding simulation using hierarchical queues [71]. However, the difference between the different watershed definitions are not noticeable for natural images. All the different strategies lead to the same partition of the image.

The different watershed strategies presented in this section are based on different path based criteria. However, when working on natural images, the different strategies produce slightly or exactly similar results. We will see in the next sections how these definitions can be extended to produce segmentations based on dissimilarity measures. We will also see that the definitions based on path algebra can be extended to vector valued images.

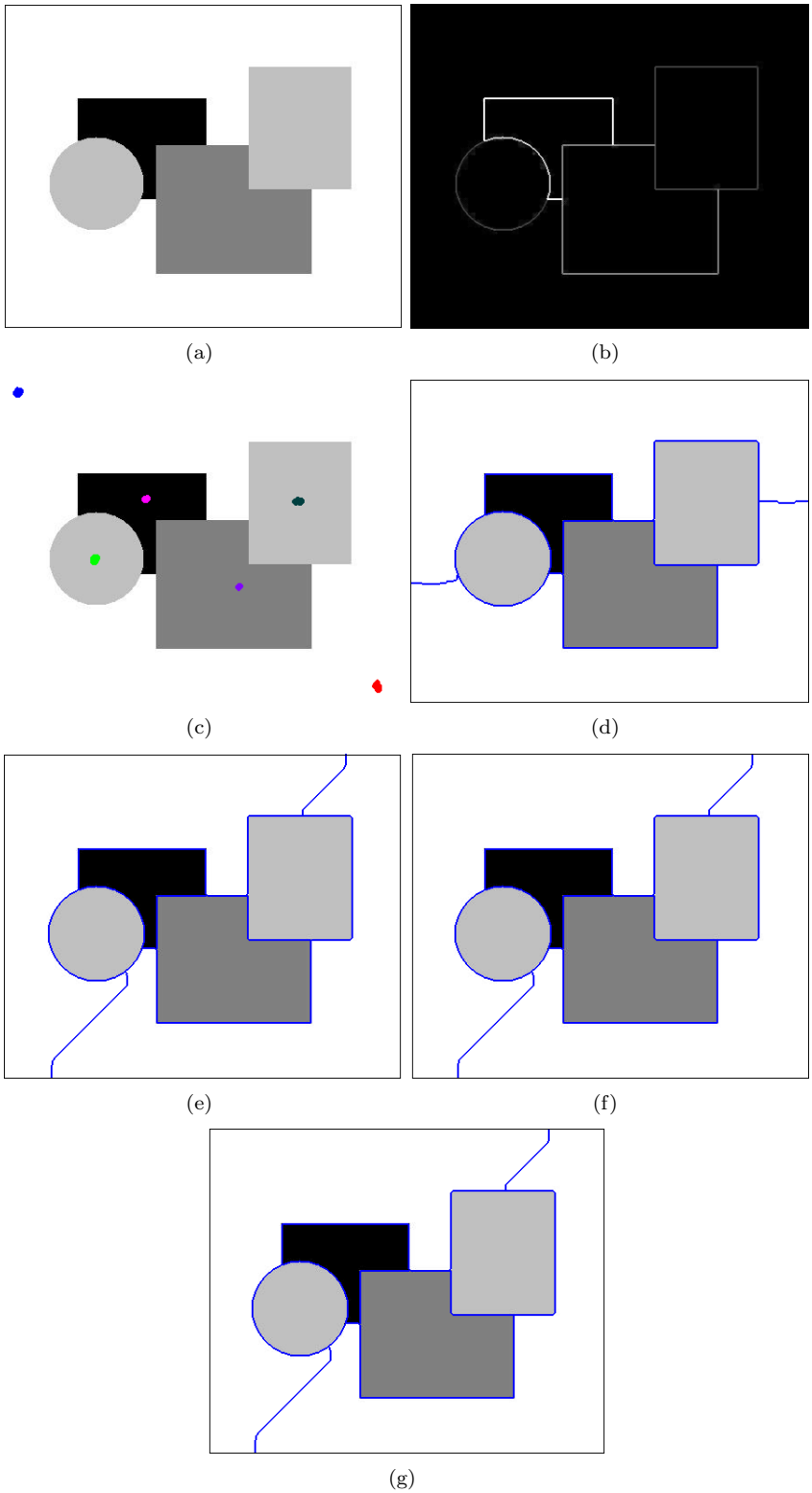


Figure 5.7: Watershed Segmentation. (a) Original image (443x361). (b) Morphological gradient. (c) Markers. (d) Minimal spanning forest of the flooding distance. (e) Bi-criteria shortest path forest. (f) Lexicographical shortest path forest. (g) Watershed using Meyer's algorithm based on hierarchical queues.

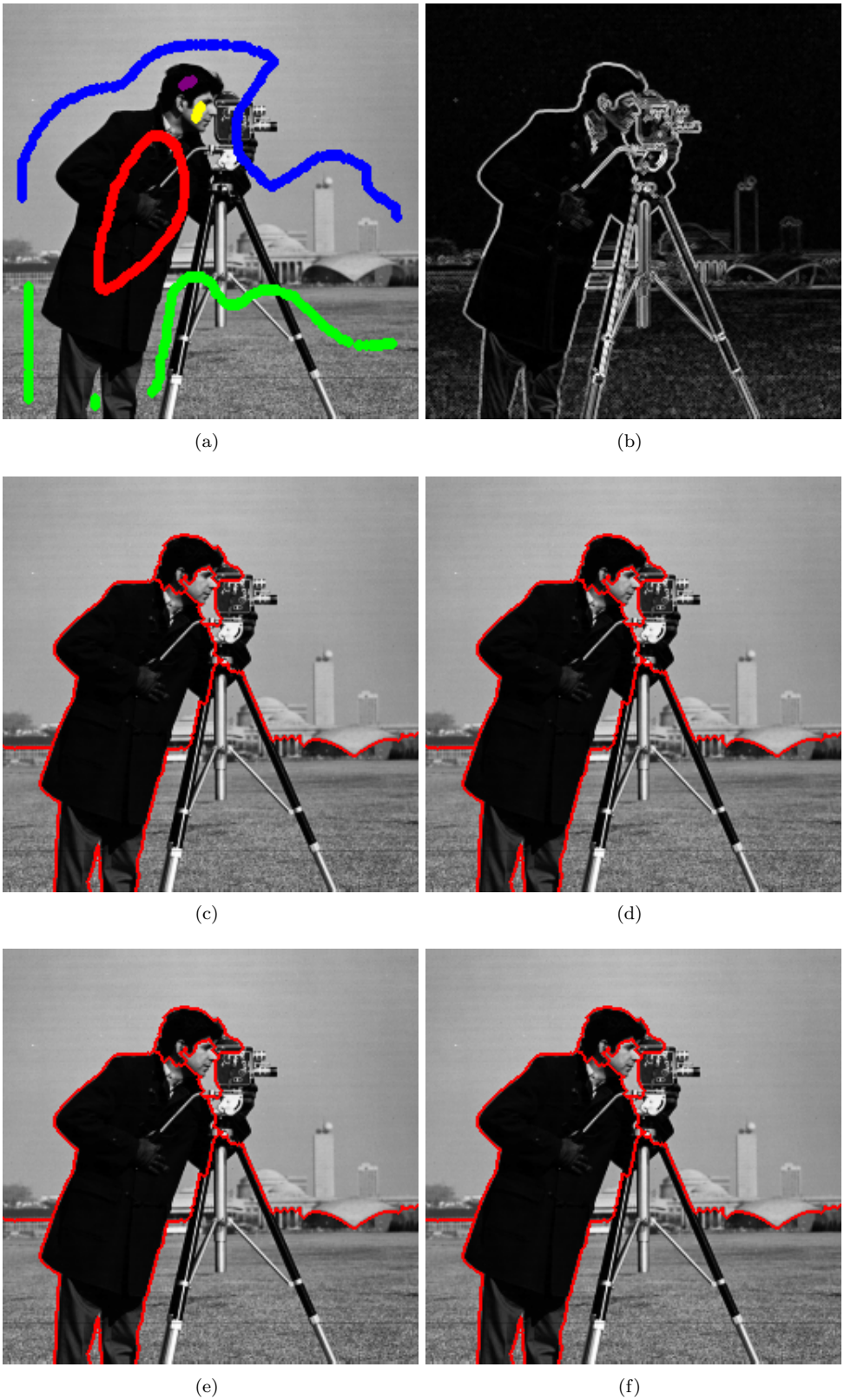


Figure 5.8: Watershed Segmentation. (a) Original image (256x256) surimposed with the markers. (b) Morphological gradient. (c) Minimal spanning forest of the flooding distance. (d) Bi-criteria shortest path forest. (e) Lexicographical shortest path forest. (f) Watershed using Meyer's algorithm based on hierarchical queues.

5.5 On Variations and Extensions of the Watershed Transform

In the following sections we detail some variations of the watershed transform. We now consider exclusively the watershed transform of the image's gradient modulus since it is the one used for segmentation purposes. The following sections detail some alternate watershed definitions that hold for the watershed transform of the images's gradient modulus. We detail also some possible extensions of the watershed for vector-valued images and give some examples of segmentations of vector valued images.

5.5.1 Watershed Based on Dissimilarity Measures

The watershed transform definitions can be slightly modified in such a way that it produces a segmentation according to different path based criteria. For image segmentation purposes the watershed transform is computed on the image's gradient modulus, usually the morphological gradient. The estimation of the gradient has thus also an importance for the quality of the segmentation. A first extension of the classical watershed transform of the gradient image is based on the local dissimilarity between neighbor pixels. This approach, called watershed by dissimilarity was originally proposed by Pard{'as [82] and Lotufo et al. in [66]. The edges weights are identical to those proposed in chapter 2; given a pixel adjacency graph $G = (V, E, W)$, we consider the following edges weights mapping:

$$\forall e_{i,j} \in E, \quad w_{i,j} = \left(\frac{1}{d(i,j)} . |p_i - p_j| + 1 \right) , \tag{5.5.1}$$

where i and j are two nodes of the graph, p_i and p_j are the grey level values of neighbor pixels of the image and $d(i, j)$ is the distance between the two pixels.

These edges weights represent the local estimation of the image's gradient modulus. Instead of computing the morphological gradient of the image at each pixel, the gradient is here estimated "between" the pixels. This weight map especially provides a better detection of thin contrasted objects. The watershed transform using the ultrametric flooding distance is then defined as the minimal spanning forest rooted on the minima. Alternatively, the watershed transform can also be defined as the bi-criteria shortest path forest rooted on the minima (or the markers) to deal with plateaus of the image as proposed in [66]. Finally, the watershed transform can also be defined as the lexicographical shortest path forest rooted on the minima (or the markers) as defined in the previous section to deal with competition between floodings.

This weight map has the main advantage to achieve a better detection of small details compared to the classical morphological gradient. However we want also to point out that this weight map can only be used for the computation of the watershed transform of the image's gradient. This weight map cannot be obtained if one wants to compute the watershed of a given image. In this last case, the previous definitions that we have detailed should be used.

Examples and Comparisons

Figure 5.9 illustrates the watershed based on dissimilarity measure using the ultra-metric and the lexicographical distances which lead to the computation of a minimal spanning forest and a shortest path forest respectively. Figure 5.9 also compares these approaches with the classical watershed transform of the morphological gradient using Meyer's algorithm based on hierarchical queues. This example illustrates that the watershed based on dissimilarity performs a better detection of small details but it is also more sensitive to possible leaks, which correspond to small zones where the contrast is low. These differences are highlighted in figure 5.10.

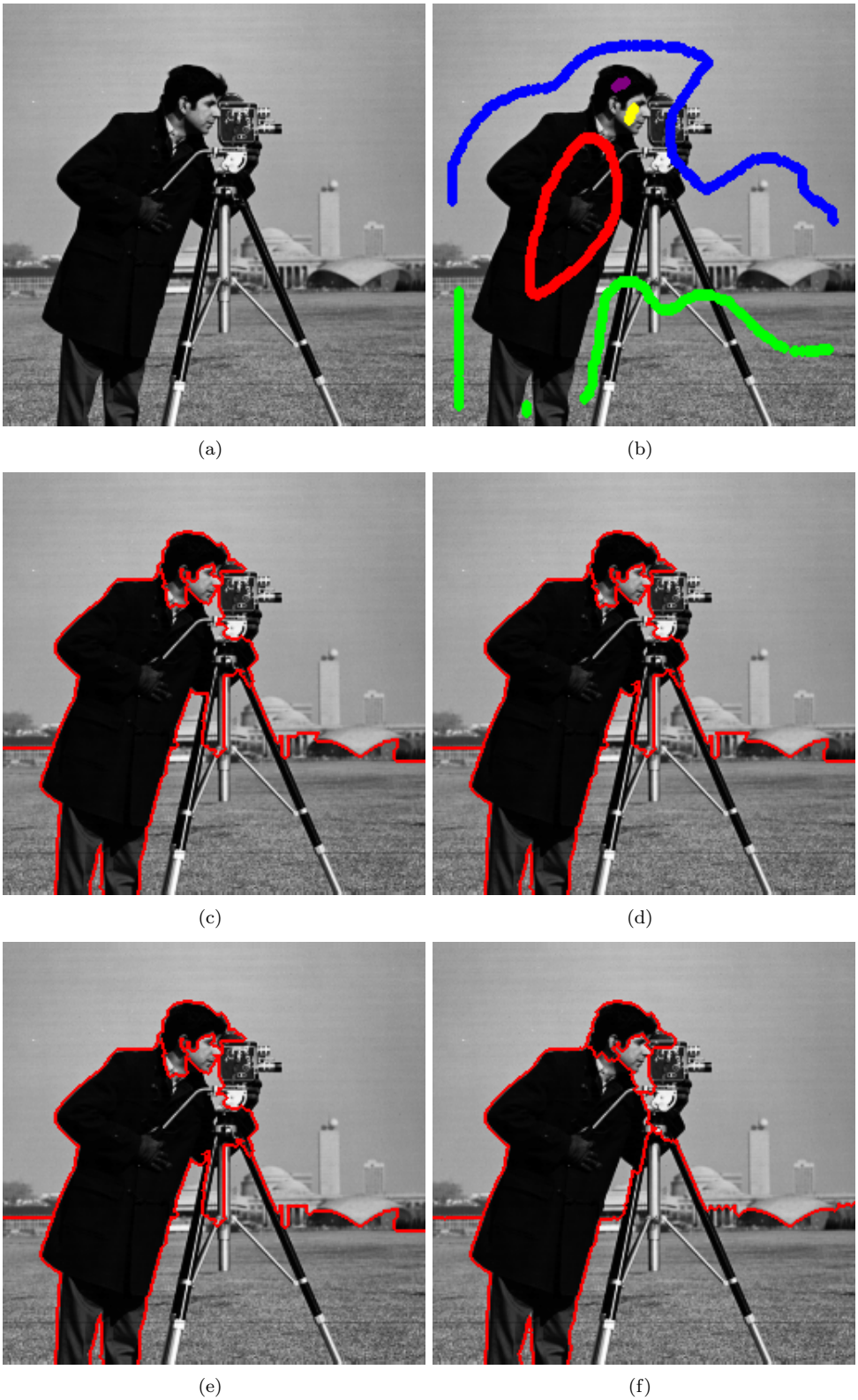


Figure 5.9: Watershed Segmentation based on local dissimilarity. (a) Original image (256x256). (b) Markers. (c) Minimal spanning forest. (d) Bi-criteria shortest path forest. (e) Lexicographical shortest path forest. (f) Watershed using Meyer's algorithm based on hierarchical queues computed on the morphological gradient.

(a) (b) (c)

Figure 5.10: Watershed Segmentation based on local dissimilarity. (a) Original image. (b) Lexicographical shortest path forest. (c) Watershed using Meyer's algorithm based on hierarchical queues computed on the morphological gradient.

5.5.2 Minimal Spanning Forests for Vector Valued Images

The path algebra point of view of shortest path forest has an interesting feature: the length of a path and the corresponding order are not limited to scalar values. Note also that the generalized Dijkstra algorithm is not limited to scalar valued edges weights. Using lexicographical orders, it is quite straightforward to extend the watershed definitions to vector valued images such as color images. Let us consider now that an image is given by a vector field. Each pixel of the image is given by a vector $p_i = (p_{i,1}, p_{i,2}, ..., p_{i,n})$. Let us now consider the pixel adjacency graph $G = (V, E, W)$ of this image, and the following set of vectors:

$$\forall e_{i,j} \in E, \;\; r_{i,j} = \left((d(i,j).|p_{i,1} - p_{j,1}| + 1), (d(i,j).|p_{i,2} - p_{j,2}| + 1), ..., (d(i,j).|p_{i,n} - p_{j,n}| + 1) \right) . \tag{5.5.2}$$

The vector $r_{i,j}$ measures the local contrast between neighbor pixels in each dimension. This measure is the direct extension of the dissimilarity measure proposed in the previous section. Finally, we also propose to order lexicographically this vector such that the largest contrast is the first element of the vector. We use this ordered vector as the edges weights of the pixel adjacency graph:

$$\forall e_{i,j} \in E, \;\; w_{i,j} = \left((d(i,j).|p_{i,k} - p_{j,k}| + 1), (d(i,j).|p_{i,l} - p_{j,l}| + 1), ..., (d(i,j).|p_{i,m} - p_{j,m}| + 1) \right) . \tag{5.5.3}$$

such that:

$$(d(i,j).|p_{i,k} - p_{j,k}| + 1) \geq (d(i,j).|p_{i,l} - p_{j,l}| + 1) \geq ... \geq (d(i,j).|p_{i,m} - p_{j,m}| + 1) \; . \tag{5.5.4}$$

From this edges weights, we can use the classical lexicographical order $\prec$ to create a new path algebra $((R^+ \cup \{+\infty\})^n, \boxplus, \star, (+\infty)^n, (0)^n)$. The element $(+\infty)^n$ is the vector $(+\infty, +\infty, ..., +\infty)$ and the element $(0)^n$ is the vector $(0, 0, ..., 0)$. The two operators $\boxplus$ and $\star$ are defined as follows:

- the operation $\boxplus$, called the addition, is defined by:

$$\forall (a,b) \in ((R^+ \cup \{+\infty\})^n, a \boxplus b = \left\{ \begin{array}{l} a \; if \; a \preceq b \; . \\ b \; if \; b \preceq a \; . \end{array} \right. \tag{5.5.5}$$

  its neutral element, or zero element is $(+\infty)^n$ since $a \boxplus (+\infty)^n = a$. Note also that $\boxplus$ is idempotent and the corresponding natural order is the usual lexicographical order $\prec$.

- the operation $\star$ is called the max-multiplication and is defined by:

$$\forall (a,b) \in ((R^+ \cup \{+\infty\})^n, a \star b = (\max(a_1, b_1), \max(a_2, b_2), ..., \max(a_n, b_n)) . \tag{5.5.6}$$

  The operation $\star$ has a neutral element, or identity element $(0)^n$ since $a \star (0)^n = (0)^n \star a = a$. Note also that $(+\infty)^n$ is an absorbing element for $\star$ since $a \star (+\infty)^n = (+\infty)^n \star a = (+\infty)^n$.

It follows that we can define shortest path forests on this new path algebra since $\boxplus$ is idempotent. The shortest path forest in the path algebra $((R^+ \cup \{+\infty\})^n, \boxplus, \star, (+\infty)^n, (0)^n)$ can be seen as the counterpart for vector valued edges weights of a minimal spanning forest for scalar valued edges weights. This structure can thus be used for vector valued image segmentation and the corresponding shortest path forests correspond to the direct extension of the watershed transform by dissimilarity measures using the ultrametric flooding distance.

5.5.3 Shortest Path Forests for Vector Valued Images

From the previously defined edges weights, we can also use the usual lexicographical order $\prec$ to create a new path algebra $((R^+ \cup \{+\infty\})^{n \times n}, \boxtimes, *, (+\infty)^n, (0)^n)$. The operators $\boxtimes$ and $\star$ are then defined as follows:

- the operation $\boxtimes$, called the addition, is defined as in the previous section.

- $\forall (a,b) \in ((R^+ \cup \{+\infty\})^{n \times n}$, the operation $*$ is called the lex-multiplication and is defined by:

– if $b_1 \succeq a_1$, $a \star b = b$.
– if $a_n \succeq b_0$, $a \star b = (a_1, a_2, ..., a_n, b_1, b_2, ..., b_n)$.
– else let j be the highest index verifying $b_{j-1} \succeq b_1 \succeq a_j$: $a * b = (a_1, a_2, ..., a_{j-1}, b_1, b_2, ..., b_n)$.

The operation $*$ has a neutral element, or identity element $(0)^n$ since $a * (0)^n = (0)^n \star a = a$. Note also that $(+\infty)^n$ is an absorbing element for $*$ since $a * (+\infty)^n = (+\infty)^n * a = (+\infty)^n$. Note that the elements a and b are vectors whose elements are also vectors.

It follows that we can define shortest path forests on this new path algebra. The shortest path forest in the path algebra $((R^+ \cup \{+\infty\})^{n\times n}, \boxplus, *, (+\infty)^n, (0)^n)$ is the counterpart for vector valued edges weights of a shortest lexicographical path forest for scalar valued edges weights. This structure can thus also be used for vector valued image segmentation and the corresponding shortest path forests corresponds to the direct extension of the watershed transform by dissimilarity measures using the lexicographical distance. We recall also that the lexicographical distance can handle competitive flooding as well as color flat zones, unlike the ultra-metric distance.

5.5.4 Color Images Segmentation

We detail here the use of the previously defined path algebras for color images segmentation. Let us consider that a color image is given by a vector field. To each pixel of the image is associated a vector $p_i = (p_{i,1}, p_{i,2}, p_{i,3})$ representing the red, green and blue channels . Let us now consider the pixel adjacency graph $G = (V, E, W)$ of this image, and the following set of vectors:

$$\forall e_{i,j} \in E, \quad r_{i,j} = \left((d(i,j).|p_{i,1} - p_{j,1}| + 1), (d(i,j).|p_{i,2} - p_{j,2}| + 1), (d(i,j).|p_{i,3} - p_{j,3}| + 1)\right) . \quad (5.5.7)$$

The vector $r_{i,j}$ measure the local contrast between neighbor pixels in each channel of the color image. This measure is the direct extension of the dissimilarity measure for grey level images. Finally, we order lexicographically this vector so that the largest contrast is the first element of the vector. We use this ordered vector as the edges weights of the pixel adjacency graph:

$$\forall e_{i,j} \in E, \quad w_{i,j} = \left((d(i,j).|p_{i,k} - p_{j,k}| + 1), (d(i,j).|p_{i,l} - p_{j,l}| + 1), (d(i,j).|p_{i,m} - p_{j,m}| + 1)\right) . \quad (5.5.8)$$

such that:

$$(d(i,j).|p_{i,k} - p_{j,k}| + 1) \geq (d(i,j).|p_{i,l} - p_{j,l}| + 1) \geq (d(i,j).|p_{i,m} - p_{j,m}| + 1) . \quad (5.5.9)$$

The watershed segmentation is then realized by computing a shortest path forest in the path algebras $((R^+ \cup \{+\infty\})^n, \boxplus, \star, (+\infty)^n, (0)^n)$ and $((R^+ \cup \{+\infty\})^{n\times n}, \boxplus, *, (+\infty)^n, (0)^n)$, which respectively correspond to a minimal spanning forest and a shortest lexicographical forest for grey level images.

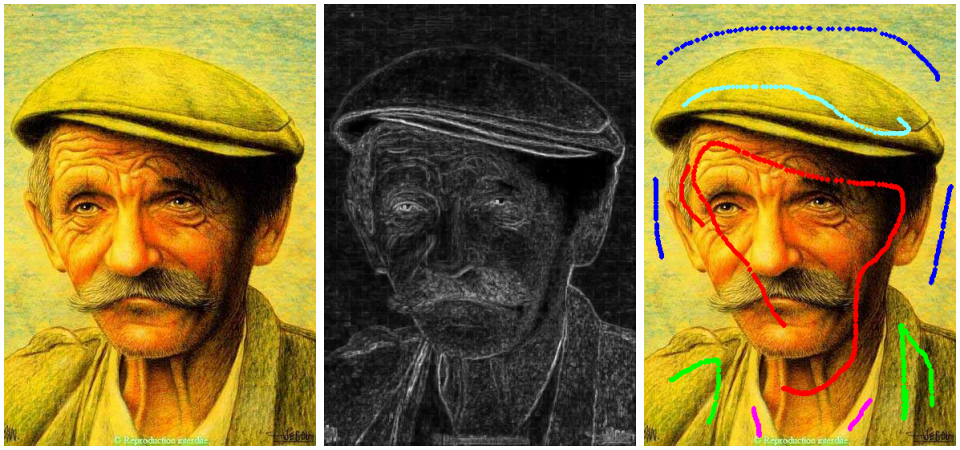
Examples

Figure 5.11 illustrates the proposed method for the segmentation of a color image. This example illustrates that the use of the color information gives slightly better results than the watershed transform applied on the color gradient [31]. The proposed methods perform a good detection of the objects boundaries. The example illustrates also that lexicographical and ultra-metric distances provide similar results on natural color images.

Discussion

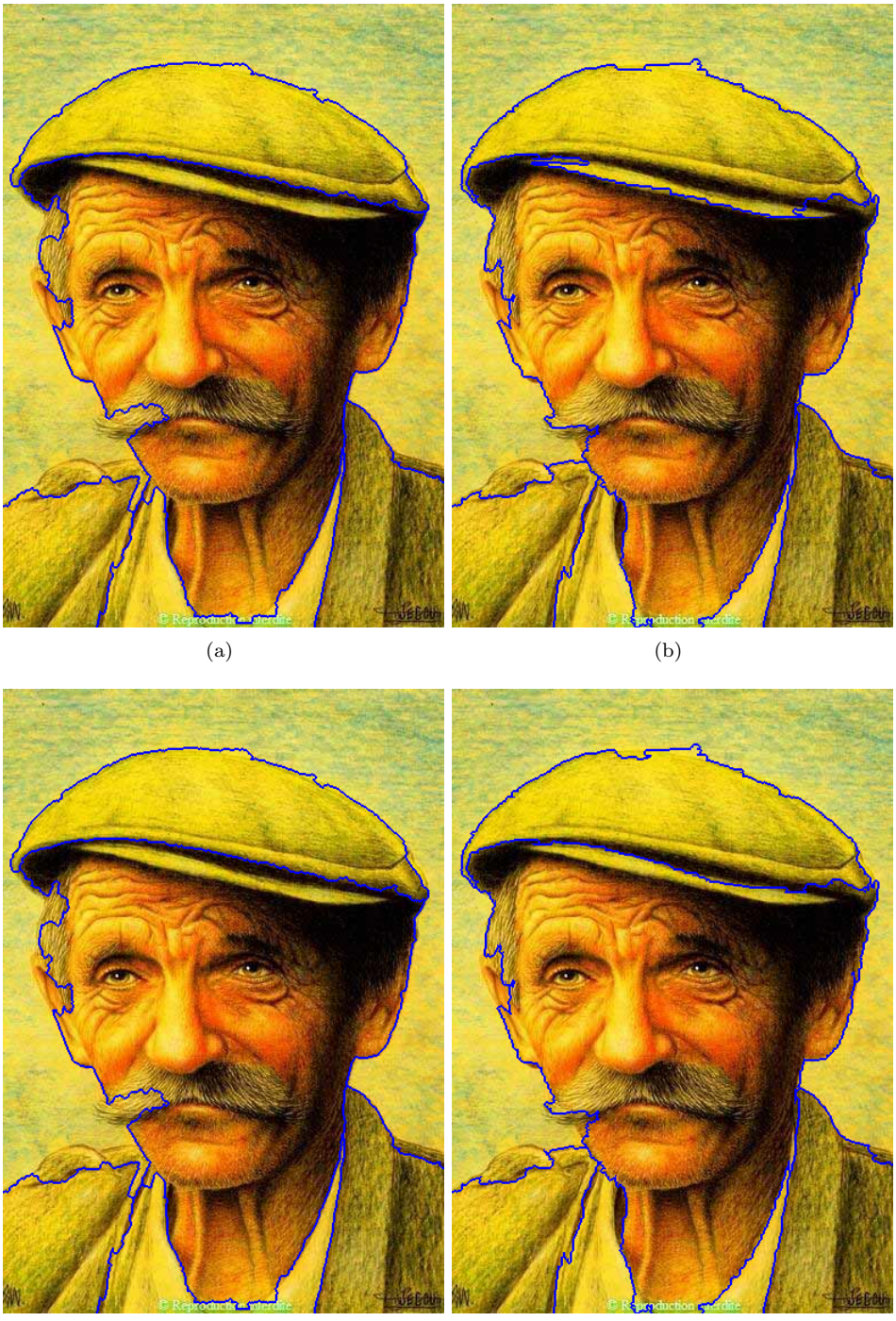
The path algebra framework offers numerous possibilities for path based image segmentation. The presented method for color images is only the direct extension of the classical method used on grey level images. But it is possible to design more complex measures to design alternate methods. In our example we have used an ordered vector reflecting the contrast in each dimension as the edges weights for the pixel adjacency graph. One should notice that other orders could be used. It is for instance possible to

choose a pre-defined order on the dimension of a vector-valued image. The image segmentation is then based on some priority given to each dimension. In our example, the priority is given, for each edge, to the dimension in which the contrast is the largest. Different priorities could be computed based on statistical, spectral or geometrical properties. These criteria should be designed depending on the aimed application.



(a) (b) (c)

Figure 5.11: Watershed Segmentation based on local dissimilarity for color images. (a) Original color image (389x550). (b) Color gradient. (c) Markers.



(a) (b)

(c) (d)

Figure 5.12: (a) Minimal spanning forest of the color image gradient. (b) Minimal spanning forest of the color image. (c) Lexicographical shortest path forest of the color image gradient. (d) Lexicographical shortest path forest of the color image.

5.6 Conclusion

We have revisited the watershed transform in the path algebra framework. We detailed the optimal properties of the watershed transform on graphs and gave some clear definitions of the watershed of vector valued images. We have seen that the watershed transform can be formalized as an instance of shortest path forests under some particular distances. The paths optimization criteria for image segmentation has to be refined in order to design objective functions that are not only based on the contrast in the image. This special point should be investigated in a deeper way for the design of efficient algorithms for color and multi-spectral image segmentation. These extensions could be a stable basis to create new paradigms for the segmentation of multi-spectral images.

The path algebra theory offers an unifying framework for path based image segmentation. This point of view permits to solve various path optimization problems with a single algorithm, the generalized Dijkstra algorithm. The watershed transform can especially be computed for both grey level and color images by using the same algorithm.

Finally, this work also motivates further research about mathematical morphology on graphs, in order to define new morphological operators based on the rich properties of graphs or trees structures. This field have been investigated during the thesis but is not exactly in the scope of this work. However it is clear that the graphs offers a slightly richer structure than a classical lattice.

6

Perspectives

This chapter presents essentially some perspectives and experiments concerning the development of the minimal cut problem for solving constrained segmentation issues. These constraints bring the classical minimal cut problem in the linear and quadratic programming framework. We focus on the use of some additional geometric constraints that can be used with graph cuts. We consider in this chapter the minimal cut problem on a pixel adjacency graph, keeping in mind that its extension to region adjacency graphs is possible and could be advantageously used to speed up and extend the presented techniques.

Linear and quadratic programs represent an instance of linear systems of equations that have to be solved under linear constraints. These types of problems have already been considered for imaging applications. The most famous one is probably the random walker segmentation method [45] which is an instance of a quadratic program. Linear and quadratic programs have now a great appeal in the imaging community and new applications of these methods are developed constantly. Linear and quadratic programs have been so far efficiently applied to image registration, reconstruction and segmentation problems [41, 44, 45, 110]. We present in this chapter how these frameworks can be used to introduce new types of constraints for optimal cuts. We especially detail the computation of optimal cuts under geometric constraints. Using the linear programming framework, it becomes possible to compute optimal cuts with volume, area and perimeter constraints. We also consider the computation of optimal segmentations with other types of markers, constraining the segmentation to have some specific boundary properties. This method was first proposed by Grady et al. [44] to extend shortest path curves to 3D; our contribution is the use these types of constraints for optimal cuts.

Graph based image segmentation algorithms have been extensively studied the last years. Minimum spanning trees, shortest paths trees and minimum cuts have been used to segment images according to some optimal properties [1, 2, 33, 47, 73]. Unfortunately none of these problems produce a unique solution. As a consequence, results presented by these methods for image segmentation are highly dependent upon the implementations of the solver. Our work is mainly motivated by the need to propose reproducible image segmentation algorithms. Bringing additional constraints or regularizers to optimal graph structures reduces the number of solutions of the underlying optimization problems and attenuate the impact of the solver implementation.

6.1 Definitions

Let us first introduce the theoretical foundations of the methods that we use in this chapter, namely linear programming (LP), integer programming (IP) and quadratic programming (QP). Many combinatorial problems, such as graph partitioning problems used for image segmentation, can be written as linear, integer or quadratic programs. Linear programming refers to the following general problem:

Definition 6.1.1 (Linear Program)**.** *A Linear Program is defined as the following minimization problem:*

$$\min_{x \in \mathcal{X}} (f^T x), \quad subject \ to \begin{cases} A_{ineq}.x \leq b_{ineq} \\ A_{eq}.x = b_{eq} \\ lb \leq x \leq ub \end{cases} \tag{6.1.1}$$

where x represents a vector of real or integer variables to be determined. x takes its values in the set $\mathcal{X}$. f, b_{eq} , b_{ineq} are vectors with real coefficients. A_{ineq}, A_{eq} are matrices with real coefficients. lb and ub are respectively upper and lower bounds for the solution x.

A LP can be seen as the following optimization problem: given a polytope (defined by the equalities and inequalities constraints), and a real-valued affine function f defined on this polytope, the goal is to find a point in the polytope where the function f takes its smallest value. Note that such points may not exist, but if they do, searching through the polytope vertices is guaranteed to be successful at finding at least one of them. When the linear program is constrained to have an integer solution, $x \in \mathbb{N}$, the corresponding problem is called as an integer program. The geometry of integer and linear programming is illustrated in figure 6.1.

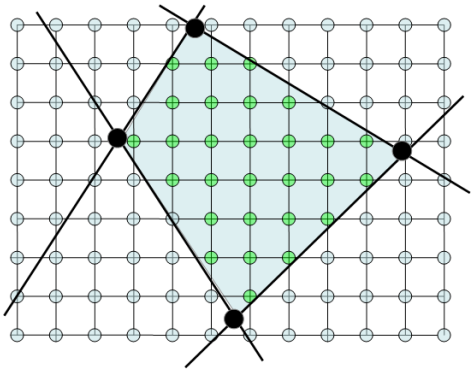


Figure 6.1: The geometry of linear and integer programming. The inequalities constraints, represented by dark lines, specify a region of feasible solution colored in light blue. The optimal solution of the linear program lies on an vertex, drawn in large black disks, of the polytope. The feasible solution of the corresponding integer programming are represented by green disks.

The readers unfamiliar with linear programming are referred to the following textbooks [43, 62]. A linear program solver aims at finding a feasible solution x, i.e. that satisfies the constraints defined in equation 6.1.1, and that minimizes the objective function $f^T x$. The best known algorithms for solving this problem are Dantzig's simplex method [26] and interior point methods [84]. On the other side, integer programs require more complex solvers such as the one based on the branch and bound method [63].

The other type of classical problems met in imaging applications is the minimization of a quadratic objective function subject to linear constraints. This kind of optimization problem is called a quadratic program (QP).

Definition 6.1.2 (Quadratic Program)**.** *A quadratic program is defined as the following minimization*

problem:

$$\min_{x \in \mathcal{X}}(x^T.A.x + f^T x), \quad subject \ \ to \begin{cases} A_{ineq}.x \leq b_{ineq} \\ A_{eq}.x = b_{eq} \\ lb \leq x \leq ub \end{cases} \tag{6.1.2}$$

There are also many algorithms that solve exactly or approximatively quadratic programs. For instance if there are only equality constraints, then the QP can be reduced to a linear system of equations [45]. For instance the well known random walker on a graph is a QP that can be reduced to a linear system of equations. Otherwise, a variety of methods for solving QPs are commonly used, including interior point methods, active sets and conjugate gradient methods [43, 62]. We refer the reader to the cited textbooks for more details and theoretical background on these optimization problems.

6.2 Minimal Cut as a Linear Program

Linear programming starts to show interesting applications in imaging applications and especially image segmentation, registration and reconstruction [41, 44, 110]. Recently Grady presented an extension in 3D of the shortest path problem to compute exact discrete minimal surfaces [44]. This method is based on linear programming and highlights the possibilities of such techniques for image segmentation. The classical way to solve the minimal cut problem is based on network flow algorithms such as variants of the Ford-Fulkerson algorithm as described in the first chapter. We present in this section the minimal cut problem as a linear program and give some interesting extensions allowed by this formulation.

6.2.1 Minimal Cut

Given an edge weighted graph $G = (V, E, W)$ and two nodes $s \subset V$ and $t \subset V$, $s \neq t = \emptyset$, the minimum cut problem is to find a set of edges $C_{s,t}$ between s and t such that the weight of the cut $L_1(C_{s,t}) = \sum_{e_{i,j} \in C_{s,t}} w_{i,j}$ is minimum and such that the removal of the edges $C_{s,t}$ disconnects the node s and t (i.e. there is no paths between the sets s and t after the removal of edges of $C_{s,t}$). A graph cut also produces two connected components S and T such that $s \in S$ and $t \in T$.

6.2.2 Linear Programming

We detail now how the minimal cut problem can be formulated as a linear program. We first compute the objective function to be minimized and then write the set of constraints such that the solution produces a minimal graph cut. Let us first define the indicator vectors z and y as:

- $\forall e_i \in C_{s,t} \subseteq E, z_i = 1$,
- $\forall e_i \in E \setminus C_{s,t}, z_i = 0$,
- $\forall i \in S, y_i = 1$,
- $\forall i \in T, y_i = 0$.

The elements z_i indicate if the edges e_i are in the minimal cut, whereas the elements y_i indicate if the nodes i are in the set S or T. Let us now define the vector x as the concatenation of the vectors z and y. Let us also define the vector f as the concatenation of a vector of size $m = |E|$ representing the edges

weights and a null vector of size $n = |V|$:

$$x = \begin{pmatrix} z_1 \\ . \\ . \\ z_m \\ y_1 \\ . \\ . \\ y_n \end{pmatrix}, f = \begin{pmatrix} w_1 \\ . \\ . \\ w_m \\ 0 \\ . \\ . \\ 0 \end{pmatrix} \tag{6.2.1}$$

Vector z indicates the edges of the graph cut $C_{s,t}$. $z_i = 1$ means that the edge e_i is in the cut. Similarly the vector y indicates the corresponding labeling of the nodes. y_i takes the value 1 if it remains connected to the set of nodes S and takes the value 0 otherwise. If we consider that the set $C_{s,t}$ is a graph cut, then clearly the objective function to minimize (i.e. the weight of the cut) can be written as the following product:

$$L_1(C_{s,t}) = f^T x = \begin{pmatrix} w_1 \\ . \\ . \\ w_m \\ 0 \\ . \\ . \\ 0 \end{pmatrix}^T . \begin{pmatrix} z_1 \\ . \\ . \\ z_m \\ y_1 \\ . \\ . \\ y_n \end{pmatrix} = \sum_{1 \leq i \leq m} z_i . w_i \tag{6.2.2}$$

In order to ensure that the minimization of $f^T x$ produces a graph cut, one has to specify some constraints on the solution x. In any other case, the minimization of $f^T x$ has a trivial solution which is a vector with only zeros as elements. The first constraints concern the nodes labeling in order to ensure this labeling is coherent with a graph cut. First, nodes s and t have to take respectively the values 1 and 0. These first constraints express the equalities constraints of our linear program:

$$\begin{cases} y_s = 1 \\ y_t = 0 \end{cases} \tag{6.2.3}$$

The second type of constraints, the inequalities constraints, ensure that the cut separates the terminals s and t and that the corresponding labeling of the nodes is coherent. These constraints are expressed by the following inequalities:

$$\forall (e_i = e_{k,l}) \in E, z_i - y_k + y_l \geq 0 \tag{6.2.4}$$

This inequality ensures that at least one edge e_i lies in the graph cut on each path from node s to node t. These set of inequalities are thus constraining the labeling of the nodes to be coherent with a graph cut labeling. Finally the minimum cut can be found by solving the following linear program:

$$\min_{x \in \mathcal{X}} \ (f^T x) \tag{6.2.5}$$

$$subject \ \ to \begin{cases} \forall (e_i = e_{k,l}) \in E, z_i - y_k + y_l \geq 0 \\ y_s = 1 \\ y_t = 0 \\ 0 \leq x \leq 1 \end{cases} \tag{6.2.6}$$

Like many other graph methods, linear programming takes into account the orientation of the graph edges. The algebraic formulation of the minimal cut takes into account the difference between the edge $e_{i,j}$ and $e_{j,i}$. As a consequence, in case of undirected graphs, it is necessary to consider both direct and reverse edges. It is also necessary to add the corresponding inequalities and equalities constraints that ensure that both direct and reverse edges, $e_{i,j}$ and $e_{j,i}$, take the same values in the solution x.

For comparison, we recall here the formulation of the random walker problem:

$$\min_{x \in \mathcal{X}} \left(\sum_{e_{i,j} \in E} w_{i,j}^2 . |x_i - x_j|^2 \right) \tag{6.2.7}$$

$$subjectto \begin{cases} y_s = 1 \\ y_t = 0 \\ 0 \leq x \leq 1 \end{cases} \tag{6.2.8}$$

The random walker leads to the minimization of a quadratic objective function. Although this last problem looks quite similar to the minimal cut problem (which has the following objective function: $\sum_{e_{i,j} \in E} w_{i,j} . |x_i - x_j|$); its resolution requires different optimization strategy and is defined as a quadratic problem (QP). This problem, as well as few variants, has been extensively studied by Grady for image segmentation purposes [47, 45, 46].

6.2.3 Example

Let us illustrate the coherence of the proposed formulation and the way it produces the expected results on an example. Consider the directed weighted graph illustrated in figure 6.2. We would like to find the minimal cut that separates the nodes s and t as shown in figure 6.2. Our first example illustrates the method on a directed graph.

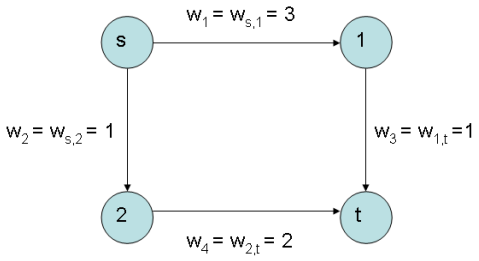


Figure 6.2: Example of a graph composed of 4 nodes {s,1,2,t}. Edges weights appear in gray.

According to our formulation, the minimum cut can be found by solving the following linear program:

$$\min_{x \in \mathbb{R}^8} \ (f^T x) = \min_{x \in \mathbb{R}^8} \begin{pmatrix} 3 \\ 1 \\ 1 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T . \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \tag{6.2.9}$$

$$subject \ \ to \ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 \end{pmatrix} . x \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{6.2.10}$$

$$and \ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} . x = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{6.2.11}$$

$$and \ 0 \leq x \leq 1 \tag{6.2.12}$$

This system of equations can be solved using a common linear program solver such as the simplex algorithm [26, 35]:

$$x = \begin{pmatrix} z_1 = 0 \\ z_2 = 1 \\ z_3 = 1 \\ z_4 = 0 \\ y_1 = 1 \\ y_2 = 1 \\ y_3 = 0 \\ y_4 = 0 \end{pmatrix} \tag{6.2.13}$$

This solution gives the weight of the minimum cut: $(L_1(C_{s,t}) = f^T.x = 2)$. Moreover the solution x gives also the corresponding labeling of the nodes, which is indicated by the vector y, as shown in figure 6.3.

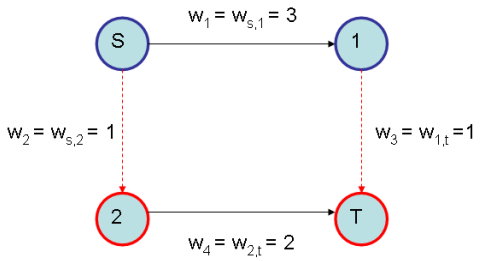


Figure 6.3: Representation of the edges of the minimum cut, in red dotted segments, as well as the corresponding labeling of the nodes.

6.2.4 Comments and precisions

We give now some details about our formulation. First, our formulation of minimal graph cut is not exactly right since we relaxed the solution x to be in the real interval $[0, 1]$ and the solution should instead exactly be contained in the set $\{0, 1\}$. In fact, due to the total unimodular property of the minimal cut problem [62], the solution is always guaranteed to be contained in the set $\{0, 1\}$ even if we allow the solver to search for solutions in the real interval $[0, 1]$.

Readers should also wonder what kind of improvements the use of linear program can give since purely graph based algorithms are more efficient and faster than a linear program solver for the minimal cut problem. In fact, linear programming offers a common framework to a large class of connected problems. For instance we could decide to solve a slightly different problem than the minimal graph cut between two sets of nodes. We could need to find the minimal cut between two sets of nodes that contains some specified set of edges. This slightly different problem can not be solved by a common graph algorithm such as Ford-Fulkerson algorithm, but it can be solved by a linear program solver by adding some other constraints on the solution. This slightly different problem can be useful for image segmentation since edge constraints can define a set of curves, or surfaces that should be contained in the segmentation boundaries. This specific example is illustrated in the next section.

A second issue deals with the uniqueness of the solution provided by the optimal cut problem. Unfortunately common graph problems, such as minimal cuts, do not always have a unique solution. To our knowledge, the regularization of the minimal cut problem has not been solved yet. This point will be partially solved by using additional geometric constraints for the minimal cut. We discuss in the next sections the computation of minimal cuts with a maximum area and perimeter constraints. One should also be aware that polynomially solvable problems, like the classical minimal cut problem, can become NP-hard when adding additional constraints. For instance the minimal cut problem containing

a given number of edges is NP-hard. We consider thus in our study the linear programming relaxation of these problems. In practice, solutions of these problems provided by common solvers are not integer. A classical way to obtain an integer solution consists in rounding the real solutions. In other words, we will only consider only approximate solutions.

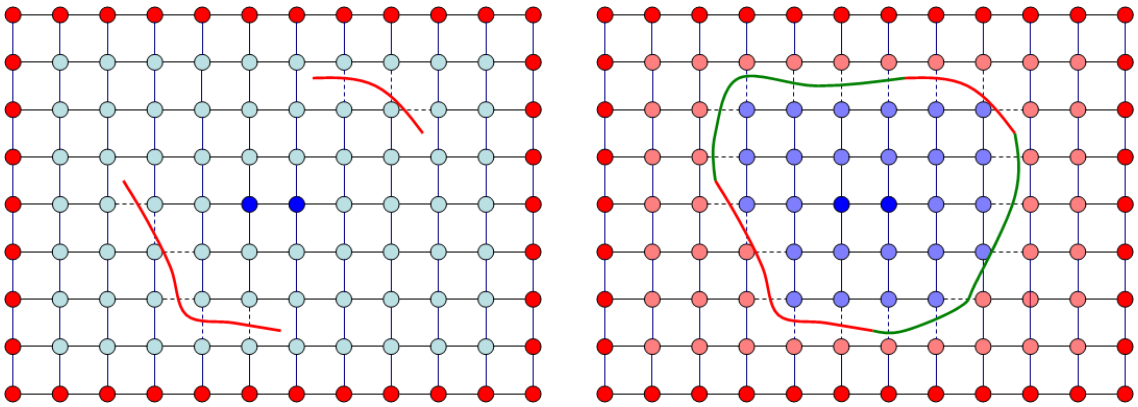
Finally, in all our examples and illustrations for image segmentation, a V4 adjacency system has been considered to compute the graphs and we used the following ad-hoc edges weights mapping [45]:

$$\forall e_{i,j} \in E, \quad w_{i,j} = e^{-\left(\sqrt{2500.(p_i - p_j)^2 + 1}\right)} . \tag{6.2.14}$$

where i and j are two nodes of the graph, p_i and p_j are the grey level values of neighbor pixels of the image, $p_i \in [0, 1]$.

6.3 Boundary Constrained Minimal Cuts

An interesting application of constrained minimal cuts consists in adding boundary constraints. These constraints are given as a set of edges that have to be included in the graph cut. These kind of markers can be advantageously used for image segmentation purposes [32, 79, 44]. These boundary constraints can model pieces of curves (2D) or surfaces (3D) that have to be included in the resulting segmentation boundaries, as illustrated in figure 6.4.



(a) (b)

Figure 6.4: Boundaries constrained minimal cuts. The set of constraints is given by a set of edges, that implicitly represent a set of curves drawn in red. The aim is to find a minimal cut separating the markers, drawn in blue and red, that includes the dotted edges. (b) The obtained minimal cut is then constrained to include these edges.

6.3.1 Definition

The boundary constrained minimal cut can be first defined in terms of a classical optimal graph cut problem.

Definition 6.3.1 (Boundary Constrained Minimal Cut)**.** *Let $G = (V, E, W)$ be an edge weighted graph and $T = \{t_1, t_2\}$ be two distinct nodes. Let $E_b \subset E$ be a set of distinct edges of G. The boundary constrained minimal cut problem consists in finding a set of edges C^* of E separating the terminals t_1 and t_2 such that:*

$$C^* \in \underset{C \in C_{\{t_1,t_2\}}}{\operatorname{argmin}} \ (L_1(C)) \ . \tag{6.3.1}$$

$$subject \ \ to : \forall e_{i,j} \in E_b, \ e_{i,j} \in C \tag{6.3.2}$$

where $C_{\{t_1,t_2\}}$ is the set of all possible cuts separating terminals t_1 and t_2.

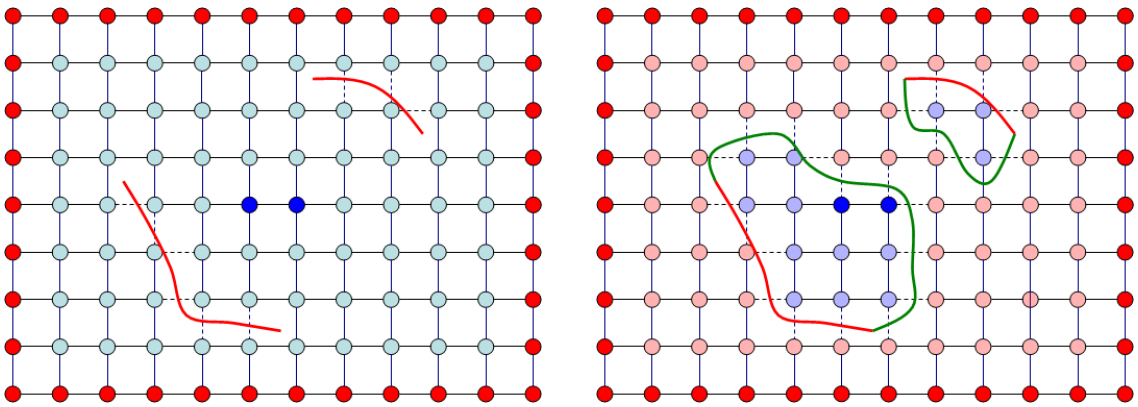
This specific problem cannot be solved by a usual graph algorithm such as Ford-Fulkerson algorithm. The additional constraints cannot be handled by this algorithm. It is thus necessary to use a linear program solver to find the corresponding optimal cut. We define thus the boundaries constrained minimal cut as the following linear program, using notations introduced earlier.

Definition 6.3.2 (Boundary Constrained Minimal Cut)**.** *The boundary constrained minimal cut is an instance of the following linear program:*

$$\min_{x \in \mathcal{X}} \ (f^T x) \tag{6.3.3}$$

$$subject \ \ to \begin{cases} \forall (e_i = e_{k,l}) \in E, z_i - y_k + y_l \geq 0 \\ \forall (e_i = e_{k,l}) \in E_b, z_i = 1 \\ \forall (e_{k,l}) \in E_b, y_k + y_l = 1 \\ y_s = 1 \\ y_t = 0 \\ x \in \{0, 1\} \end{cases} \tag{6.3.4}$$

The boundary constraints can be easily added to the linear programming framework. These constraints are linear equalities depending on the variable z_i and the corresponding end nodes y_k and y_l. Constraints on the nodes around the cut ($y_k + y_l = 1$) are added to ensure that the labeling remains coherent along the boundary constraints. Note also that these equalities do not permit to know the topology (i.e. the number of connected components) produced by the graph cut. A possible optimal configuration of this linear programm is illustrated in figure 6.5.



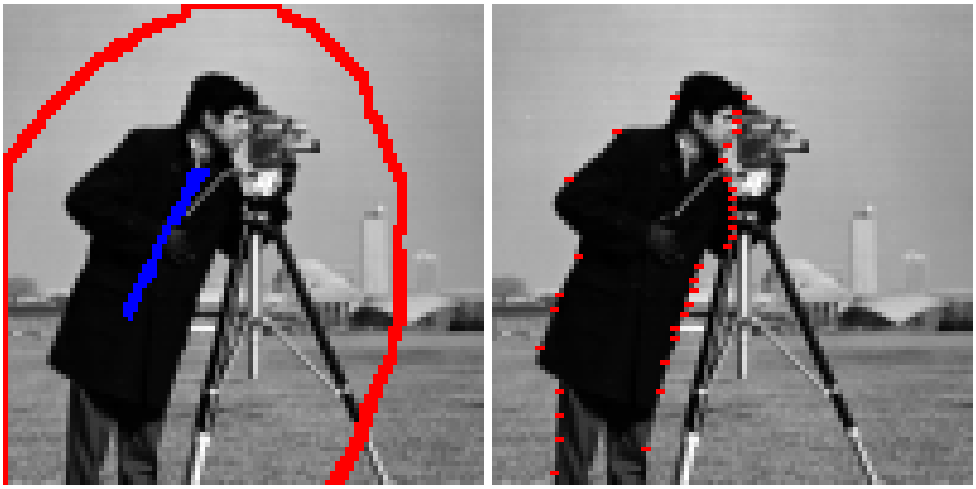
(a) (b)

Figure 6.5: Boundary constrained minimal cuts. The set of constraints is given by a set of edges, that implicitly represent a set of curves drawn in red. The aim is to find a minimal cut separating the markers, drawn in blue and red, that includes the dotted edges. (b) The obtained minimal cut is then constrained to include these edges, but no constraint guarantees that the number of connected components created by the graph cut is equal to the numbers of markers.

This problem about the number of connected components can unfortunately not be easily solved. Adding constraints on the number of connected components would lead to a linear program with an exponential number of constraints. In many situations the minimal cut solution acts like an optimal interpolation of the contours constraints. Note also that this method can be directly used in 3D with curves and/or surface boundary constraints.

6.3.2 Example

This section illustrates the use of boundary constraints in addition to classical marker constraints for image segmentation purposes based on minimal cuts. The example, illustrated in figure 6.6, compares the segmentation results obtained by a classical minimal cut separating two sets of markers and the minimal cut with boundary constraints. The boundaries constraints, which correspond to a set of edges that are constrained to be in the minimal cut, can improve the quality of the segmentation results. In our example the constrained minimal cut delineates the object correctly, whereas the classical minimal cut detects boundaries of higher contrast but that do not correspond to the searched object.



(a) (b)

(c) (d)

(e)

Figure 6.6: Boundary constrained minimal cut. (a) Original image (100x100) surimposed with the markers. (b) Edge constraints, i.e. edges that have to be in the computed cut. (c) Classical minimal cut separating the markers. (d) Random walker segmentation. (e) Minimal cut separating the markers and including the edges constraints.

6.4 Perimeter Constrained Minimal Cut

Another interesting application of constrained minimal cuts consists in adding geometric constraints. As described earlier, a graph cut implicitly describes a curve in 2D or a surface in 3D. It is then possible to specify some properties on these curves or surfaces and set the corresponding constraints in the linear programming formulation. These constraints can for instance be given as a maximal length (2D) or surface (3D) that the minimal cut cannot exceed. This kind of properties can also be advantageously used for image segmentation purposes to avoid non-smooth segmentation results.

In graph theory, the perimeter constrained minimal cut can be seen as a variant of the cardinality constrained cut problem [18]. In this last problem, the aim is to find a minimal cut of given cardinality, i.e. a cut with a predefined number of edges. This problem is proved to be NP-hard in the general case [18]. However, the linear programming relaxation of this problem offers a simple and efficient way to obtain near optimal solutions. The results computed by a generic linear program solver is thus in most cases a real solution $x \in [0,1]$ and not an integer solution $x \in \{0,1\}$. The integer solution $x \in \{0,1\}$ is then simply obtained by rounding the solution vector x. This scheme is far from being optimal for all integer programming problems but it has given satisfactory results in our experiments.

6.4.1 Definition

The perimeter constrained minimal cut is defined as the following multi-criteria problem:

Definition 6.4.1 (Perimeter Constrained Minimal Cut)**.** *Let $G = (V, E, W)$ be an edge weighted graph and $T = \{s, t\}$ be two distinct nodes. The perimeter constrained minimal cut problem consists in finding a set of edges C^* of E separating the terminals s and t such that:*

$$C^* \in \underset{C \in C_{\{s,t\}}}{\operatorname{argmin}} \left(L_1(C)\right) \ . \tag{6.4.1}$$

$$subject\ \ to : L_0(C) \leq \beta \tag{6.4.2}$$

where $C_{\{t_1,t_2\}}$ is the set of all possible cuts separating the terminal s and t. β is a real positive parameter that constrains the number of edges (the cardinality) of the cut C.

We model here the perimeter of the cut as the number of edges of the cut. Alternatively, geometric properties of the graph edges could be used to compute a real Euclidean length of the cut. Once more, this specific problem can not be solved by a usual graph algorithm such as Ford-Fulkerson algorithm. The previous definition is an instance of a minimal multi-criteria cut [18]. To our knowledge there are no graph algorithms other than the linear program solvers that can solve this optimization problem efficiently. The formulation in terms of linear programming is given below.

Definition 6.4.2 (Perimeter Constrained Minimal Cut)**.**

$$\underset{x \in \mathcal{X}}{min}\ \ (f^T x) \tag{6.4.3}$$

$$subject\ \ to \begin{cases} \forall (e_i = e_{k,l}) \in E, z_i - y_k + y_l \geq 0 \\ \left(\sum\limits_{1 \leq i \leq m} z_i \right) \leq \beta \\ y_s = 1 \\ y_t = 0 \\ 0 \leq x \leq 1 \end{cases} \tag{6.4.4}$$

Perimeter constraints are included in the linear programming framework by adding inequalities constraints on the number of edges included in the cut $\left(\sum\limits_{1 \leq i \leq n} z_i \right) \leq \beta$. Since this is a set of simple linear inequalities, the problem can be solved by any generic linear program solver. We recall here that $z_i = 1$ if the edge belongs to the cut and $z_i = 0$ otherwise.

6.4.2 Example

This section illustrates the use of perimeter constraints in addition with classical markers constraints for image segmentation purposes based on minimal cuts. The example, illustrated in figure 6.7, compares the segmentation results obtained by a classical minimal cut separating two sets of markers and the minimal cut with a maximal perimeter constraint. The classical minimal cut has a cardinality of 314 edges. The minimal cut is progressively constrained in the following examples to have a maximal cardinality of 250, 200 and 150 edges. The segmentation results correspond to our expectations. However one should note that the solutions of the corresponding linear programs are not integer. The final segmentation is obtained by rounding the solution at the value 0.5. The results do not completely fulfill the cardinality constraints, but are near optimal results and are in good concordance with the expected segmentations.

6.5 Area Constrained Minimal Cut

Let us finally present an interesting application of constrained minimal cuts with added area constraints. As described earlier, a minimal graph cut separating two terminals s and t produces two connected components S and T, one connected to the node s and the other connected to the node t. It is then possible to specify some properties on these two sets and define corresponding constraints. These constraints can for instance be given as an interval of areas (2D) or volume (3D) in which one of the connected component S or T should be situated. This kind of properties can also be advantageously used for image segmentation purposes to avoid too large segmentation results.

6.5.1 Definition

The area constrained minimal cut is defined as the following multi-criteria problem:

Definition 6.5.1 (Area Constrained Minimal Cut)**.** *Let $G = (V, E, W)$ be an edge weighted graph and $T = \{s, t\}$ be two distinct nodes. The perimeter constrained minimal cut problem consists in finding a set of edges C^* of E separating the terminals s and t such that:*

$$C^* \in \underset{C \in C_{\{s,t\}}}{\operatorname{argmin}} \left(L_1(C)\right) \ . \tag{6.5.1}$$

$$subject\ \ to : |S| \leq \beta \tag{6.5.2}$$

where $C_{\{s,t\}}$ is the set of all possible cuts separating the terminal s and t. $|S|$ is the number of nodes such that $x_i = 1$, i.e. the number of nodes of the set S. β is a real positive parameter that constraints the area of the set of nodes S.

We model here the area of a set as the number of nodes included in the set. Alternatively geometric properties of the graph could be used to compute a real geometric area of the set S. This specific problem can neither be solved by a common graph algorithm such as the Ford-Fulkerson algorithm. The previous definition is an instance of minimal multi-criteria cut. To our knowledge there are no graph algorithms other than the linear program solvers that can solve this optimization problem. The formulation in terms of linear programming is given below.

Definition 6.5.2 (Area Constrained Minimal Cut)**.**

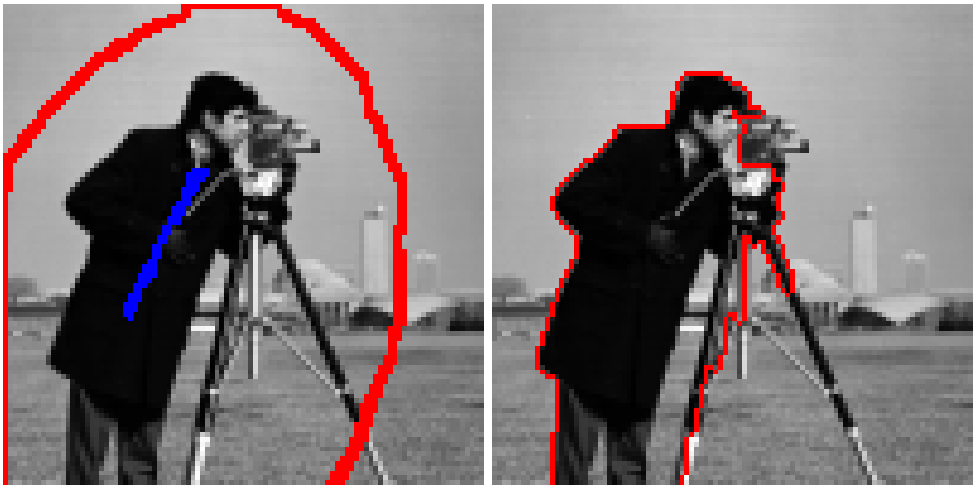
$$\min_{x \in \mathcal{X}} \ (f^T x) \tag{6.5.3}$$

$$subject\ \ to \begin{cases} \forall (e_i = e_{k,l}) \in E, z_i - y_k + y_l \geq 0 \\ \left(\sum_{1 \leq i \leq n} y_i \right) \leq \beta \\ y_s = 1 \\ y_t = 0 \\ 0 \leq x \leq 1 \end{cases} \tag{6.5.4}$$

Area constraints are included in the linear programming framework by adding inequalities constraints on the number of nodes included in the set S : $\left(\sum_{1 \leq i \leq n} y_i \right) \leq \beta$. Since this is a simple linear inequality, the problem can be solved by any generic linear program solver. The area constraints now depends on the variables y_i which indicate the labeling corresponding to a graph cut. We recall here that $y_i = 1$ if $i \in S$ and $y_i = 0$ otherwise.

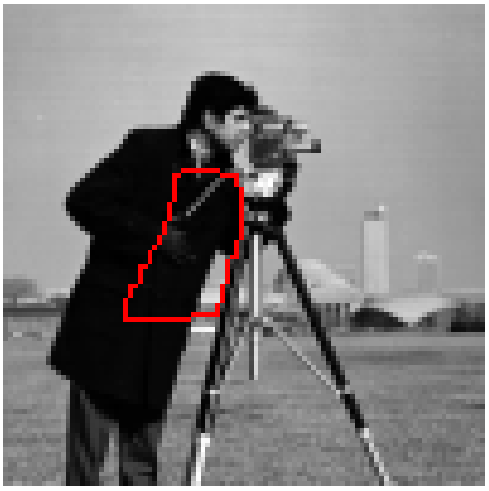
6.5.2 Example

This section illustrates the use of area constraints in addition with classical marker constraints for image segmentation purposes based on minimal cuts. The example, illustrated in figure 6.8, compares the segmentation results obtained by a classical minimal cut separating two sets of markers and the minimal cut with a maximal area constraint. The classical minimal cut produces a set S connected to the terminal s having an area of 2719 nodes. The set S is then constrained in the following examples to have a maximal area of 2200, 2000 and 1500 nodes. The segmentation results correspond to our expectations. However one should note that the solutions of the corresponding linear programs are not integer. The final segmentation is obtained by thresholding the solution at the value 0.5. The results do not totally respect the area constraints but are near optimal results and are in good concordance with the expected segmentations.



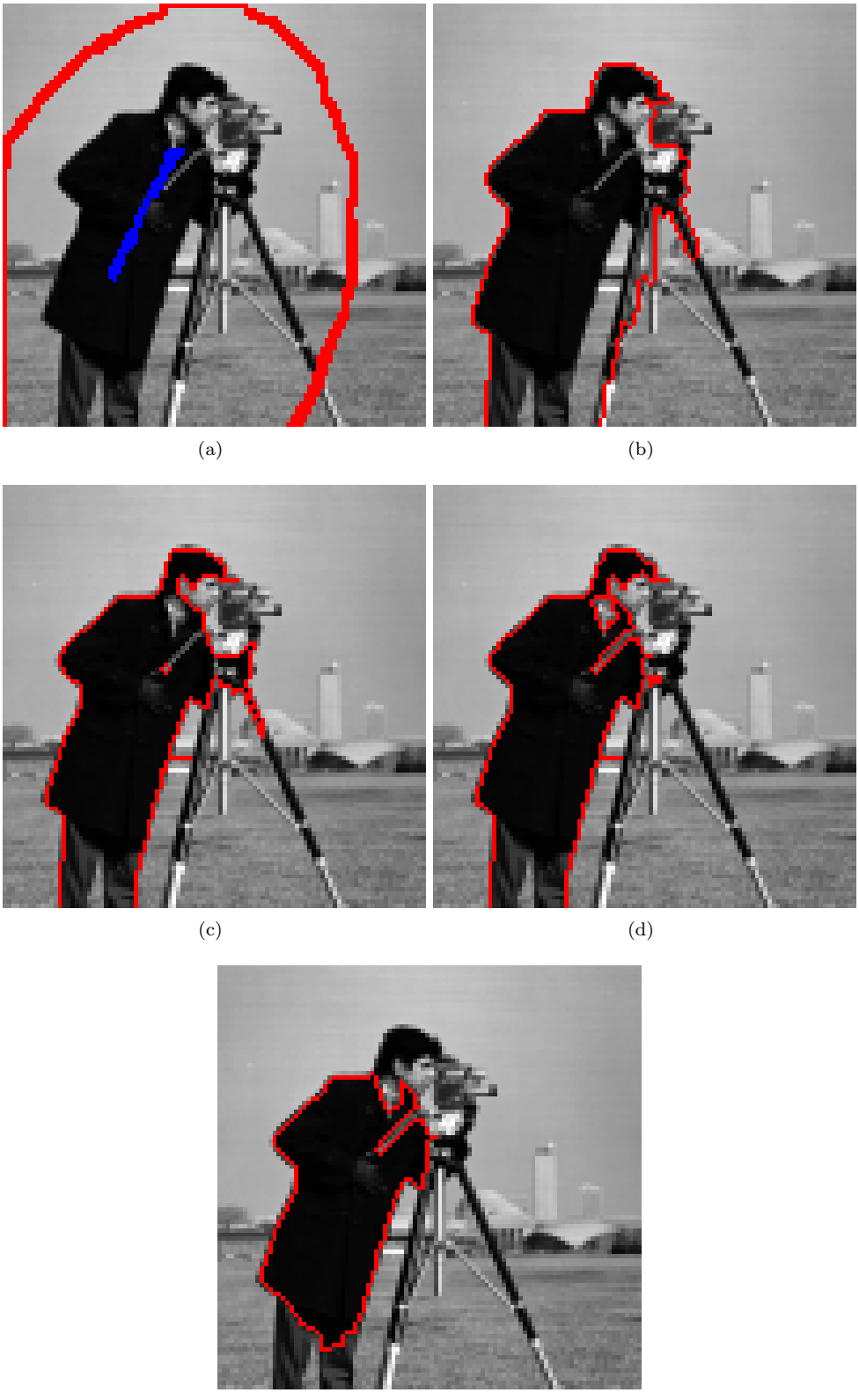
(a) (b)

(c) (d)



(e)

Figure 6.7: Perimeter constrained minimal cut. (a) Original image surimposed with the markers. (b) Classical minimal cut separating the markers. (c) Minimal cut separating the markers and having a perimeter (cardinality) limited to 250 edges. (d) Perimeter (cardinality) limited to 200 edges. (e) Perimeter (cardinality) limited to 150 edges.



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

Figure 6.8: Area constrained minimal cut. (a) Original Image surimposed with the markers. (b) Classical minimal cut separating the markers. (c) Minimal cut separating the markers and having an area limited to 2200 nodes. (d) Area limited to 2000 nodes. (e) Area limited to 1500 nodes

6.6 Conclusion

We have shown that minimal cuts can be extended to provide constrained segmentation models. These new constrained problems are solved in a linear programming framework. The linear programming framework offers interesting perspectives for medical image segmentation. Boundary constraints are particulary adapted to provide new interactive segmentation schemes based on user provided contours. The introduction of contours constraints in the graph cut framework naturally brings the classical minimal cut problem into the linear programming framework. This constrained segmentation schemes seem to be well adapted to provide new modes of interactivity for medical image segmentation.

Geometric constraints can also bring valuable contributions to graph based segmentation methods. These constraints can be used for model based segmentation by combining both perimeter and area constraints. Combining the linear programming framework with morphological segmentation can also offer a large variety of possibilities that have to be studied in future work. Linear programming methods are not yet used for image segmentation applications. The different models presented in this chapter are also new extensions of the minimal cut for image segmentation applications. This work is a first attempt to show the great potential of constrained minimal cuts and linear programming in general. This framework has to be further investigated in the future to identify the range of possibilities offered by these methods.

The major limitation of linear programming is the current inefficiency of generic solvers. Unfortunately we did not develop an efficient linear program solver that can handle graphs representing large images. However, it seems possible to develop efficient linear program solvers for imaging applications since the topology and the structure of the corresponding graphs are always the same. The main problem of the existing methods is mainly the memory requirements needed by the solver. The graphs and the constraints have to be given as matrices and most of the freely available solvers do not take into account the sparsity of these matrices, wasting a huge amount of memory only to store the zeros of the matrices. However we think that these methods will soon be leading techniques for image segmentation purposes as soon as some efficient solvers are available.

Conclusion

The main motivation of this thesis was to propose new interactive medical image segmentation techniques. In order to attain our goal, we have developed a graphical user interface dedicated to 3D medical image segmentation and visualization. Our software allows the user to explore a highly detailed view of the data-set for easy interpretation and validation of the segmentation tools. The developed tools, based on low-level segmentation and graph based methods, have been successfully used for various medical applications, including tumors delineation and model creation for surgery planning and simulation. Thanks to the help of experienced radiologists of the "Institut Gustave-Roussy", we have designed relevant image segmentation protocols which fulfill the necessary conditions to be used in a near future in a clinical context. The challenging problems presented by medical applications forced us to imagine innovative methods that can tackle the problems linked with clinical applications. The developed methods present all the characteristics to be used in real life applications: robustness, speed and precision. Moreover our work on linear programming and mathematical morphology opens new perspectives which could lead to valuable improvements of existing image segmentation techniques.

Contributions

Graph Theory:

We have detailed the use of minimal spanning trees, shortest paths trees and minimal cuts for image segmentation problems. We have established in the first chapter new theoretical links between these structures and have given a general framework unifying these graph based image segmentation methods. We have shown how shortest path forests and minimal cuts converge to minimal spanning forests under a particular weight map transform. We have also addressed the usual problems met when using graph based segmentation tools and have proposed solutions in concordance with our theoretical results. These results give a general view of the possibilities and the limitations of each technique for image segmentation problems. Finally we have highlighted a common formulation unifying random walkers, minimal cuts, minimal spanning forests and shortest path forests.

Combining Morphological and Graph Cuts Segmentation:

We have presented new theoretical and practical improvements of minimal graph cuts methods [101]. We have shown that the combination of morphological segmentation and graph cuts can be used to compute shape constrained minimal surfaces [100] as well as statistical and geometrical constrained estimates of Markov random fields [104, 105]. These extensions prove the rich potential of region adjacency graphs based methods over pixel graphs based methods. The combination of morphological and graph cuts segmentation has permitted us to speed up the process and define new classes of energy functions that can be minimized using graph cuts.

Application to Medical Image Segmentation:

Our main interest was to develop robust and fast segmentation tools for interactive medical image segmentation. We have shown by qualitative and quantitative analysis that the developed methods are leading

techniques for various medical applications including tumors segmentation [103, 102] and model creation for radiotherapy, surgery planning and simulation [96, 106]. We have also shown that our methods outperform many state of the art techniques such as machine learning, level sets and other semi-automatic or interactive methods for the segmentation of liver tumors [102]. The developed techniques present all the necessary characteristics to be used in a clinical routine. Moreover we have included all these methods in a graphical user interface dedicated to 3D medical image segmentation and visualization. This software should be soon available for research purposes.

Mathematical Morphology and Graphs:

We have then revisited the well known watershed transform in the path algebra framework. We have given some clear definitions of the watershed on graphs and vector valued images based on shortest path forests. We have especially shown how minimal spanning forests and shortest path forests can be extended to multi-spectral and color images by using graphs weighted by vectors. These new extensions offer alternatives for vector valued image segmentation.

Linear Programming:

Finally, some interesting and new perspectives have been presented for shape and boundary constrained minimal cuts on a pixel graph. We have shown that the linear programming framework has all the properties to be the next great topic of interest of the imaging community. This framework allows to model a large variety of complex image segmentation problems that can be advantageously used in various image segmentation applications. This framework provides various innovative segmentation tools such as contour, perimeter and area constrained minimal cuts. These different models are new extensions of the minimal cut problem and aim to highlight the great potential of linear programming for imaging applications. These constrained segmentation schemes seem to be well adapted to provide new modes of interactivity for medical image segmentation.

Perspectives

The combination of morphological low-level segmentations and graphs opens interesting perspectives. The energy functions defined on regions are richer than the properties defined on a single pixel. This property should be further investigated to design energy functions that can take into account high level and prior information such as explicit shape constraints models. It would also be possible to combine geometric and statistical properties of regions to design new energy functions for image segmentation, which should lead to powerful image segmentation methods. These methods have also to be extended to multi-dimensional images, such as 4D images.

Our work on the watershed transform motivates further research about mathematical morphology on graphs, in order to define new operators based on the rich properties of graph and tree structures. This field has been investigated during the thesis and we can already announce that this framework will soon offer new interesting alternatives to the classical mathematical morphology operators. Our main interest is now to develop efficient morphological anisotropic filters which take into account the structure of minimal spanning trees of an image. The use of graphs and trees can also be advantageously used for image filtering purposes in order to simplify the segmentation process.

On the other hand, the linear programming techniques have shown promising segmentation results. Geometric constraints have brought valuable contributions to graph based segmentation methods. These constraints can be used for model based segmentation by combining both perimeter and area constraints. Moreover this technique could be also advantageously combined with low-level morphological segmentation in order to speed up and extend its possibilities. Linear programming methods are not yet used for real image segmentation applications. This framework has to be further investigated in the future to identify the range of possibilities offered by these methods.

Bibliography

[1] C. Allène, J. Audibert, M. Couprie, J. Cousty, and R. Keriven, *Some links between min-cuts, optimal spanning forests and watersheds*, Proceedings of the The 8th International Symposium on Mathematical Morphology, Rio de Janeiro, Brazil **1** (2007), 253–264.

[2] R. Audigier and R. A. Lotufo, *Watershed by image foresting transform, tie-zone, and theoretical relationships with other watershed definitions*, Proceedings of the The 8th International Symposium on Mathematical Morphology, Rio de Janeiro, Brazil **1** (2007), 277–288.

[3] R. Beare, *Regularized seeded region growing*, Proceedings of the The 6th International Symposium on Mathematical Morphology, Sydney, Australia (2002), 91–99.

[4] ———, *A locally constrained watershed transform*, IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006), 1063–1074.

[5] J.-M. Beaulieu and M. Goldberg, *Hierarchy in picture segmentation : a stepwise optimization approach.*, IEEE Transactions on Pattern Analysis and Machine Intelligence **11** (1989), no. 2.

[6] I. Ben-Dan and E. Shenhav, *Liver tumor segmentation in ct images using probabilistic.*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[7] C. Berge, *Théorie des graphes et ses applications*, Dunod, Paris, 1958.

[8] J. Besag, *On the statistical analysis of dirty pictures*, Royal Journal of Statistics, Soc. B **48(3)** (1974), 259–302.

[9] ———, *Spatial interaction and the statistical analysis of lattice systems*, Royal Journal of Statistics, Soc. B **36** (1974), 192–236.

[10] S. Beucher, *Watershed, hierarchical segmentation and waterfall algorithm*, Mathematical Morphology and its applications to signal processing (Proceedings ISMM'94) (Fontainebleau, France) (J. Serra and P. Soille, eds.), Kluwer Academic Publishers, September 1994, pp. 69–76.

[11] S. Beucher and C. Lantuéjoul, *Use of watershed in contour detection*, International workshop on image processing, real time edge and motion detection/estimation. Rennes, France (1979).

[12] N. Biggs, E. Lloyd, and R. Wilson, *Graph theory, 1736-1936*, Oxford University Press, 1986.

[13] Y. Boykov and M.P. Jolly., *Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images*, Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada **1** (2001), 105–112.

[14] Y. Boykov and V. Kolmogorov, *Computing geodesics and minimal surfaces via graph cuts*, Proceedings of the 9th International Conference on Computer Vision, Nice, France **1** (2003), 26–33.

[15] ———, *An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (2004), no. 9, 1124–1137.

[16] Y. Boykov, O.Veksler, and R. Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001), no. 11, 1222–1239.

[17] Y. Boykov, O. Veksler, and R. Zabih, *Markov random fields with efficient approximations*, IEEE Conference on Computer Vision and Pattern Recognition (1998), 648–655.

[18] M. Bruglieri, F. Maffioli, and M. Ehrgott, *Cardinality constrained minimum cut problems: complexity and algorithms*, Discrete Applied Mathematics **137 (3)** (2004), 311–341.

[19] B. Carre, *Graphs and networks*, Oxford University Press, 1979.

[20] V. Caselles, R. Kimmel, and G. Sapiro, *Geodesic active contours*, International Journal of Computer Vision, 694–699.

[21] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert, *Minimal surfaces: A three dimensional segmentation approach*, Technion EE Pub. 973 (1995).

[22] A. Choudhary, N. Moretto, F. Pizzorni Ferrarese, and G. Zamboni, *An entropy based multithresholding method for semi-automatic segmentation of liver tumors*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[23] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to algorithms*, McGraw-Hill (1990).

[24] J. Crespo, R. Schafer, J. Serra, C. Gratin, and F. Meyer, *The flat zone approach : a general low-level region merging segmentation method*, Signal Processing **62** (1997), 37–60.

[25] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D Seymour, and M. Yannakakis, *The complexity of multiterminal cuts*, SIAM J. Comput. **23** (1994), 864–894.

[26] G.B. Dantzig, *Maximization of a linear function of variables subject to linear inequalities*, Activity of production and allocation (T.C.Koopmans, ed.), Wiley, New York (1951), 359–373.

[27] G.B. Dantzig and D.R. Fulkerson, *On the max-flow min-cut theorem of networks*, vol. 38, 1956, pp. 215–221.

[28] E. W. Dijkstra, *A note on two problems in connexion with graphs*, In Numerische Mathematik **1** (1959), 269–271.

[29] E. A. Dinic, *Algorithm for solution of a problem of maximum flow in a network with power estimation*, Soviet Math. Doklady **11** (1970), 1277–1280.

[30] J. Edmonds and R. M. Karp, *Theoretical improvements in algorithmic efficiency for network flow problems*, Journal of the ACM 19. **2** (1972), 248–264.

[31] A.N. Evans and X.U. Liu, *A morphological gradient approach to color edge detection*, IEEE Transactions on Image Processing **15** (2006), 1454–1463.

[32] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. A. Lotufo, *User-steered image segmentation paradigms: Live wire and live lane*, Graphical Models and Image Processing **60** (1998), no. 4, 233–260.

[33] A.X. Falcao, J. Stolfi, and R. Lotufo, *The image foresting transform: Theory, algorithms, and applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (2004), 19–29.

[34] J. Fan, G. Zeng, and M. Body M.-S. Hacid, *Seeded region growing: an extensive and comparative study*, Pattern Recognition Letters **26** (2005), 1139–1156.

[35] F. Ford and D. Fulkerson, *A simplex algorithm finding maximal networks flows and an application to the hitchock problem*, Rand Report Rand Corporation ,Santa Monica (1955).

[36] ———, *Flows in networks*, Princeton University Press, 1962.

[37] D. Freedman and P. Drineas, *Energy minimization via graph cuts: Settling what is possible*, D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In Proc. IEEE Conf. Com (b) (c) (2005), 939–946.

[38] S. Gaubert, *Two lectures on max-plus algebra*, In Proceedings of the 26th Spring School on Theoretical Computer Science and Automatic Control, Noirmoutier (1998).

[39] S. Gaubert and Max Plus, *Methods and applications of (MAX, +) linear algebra*, Symposium on Theoretical Aspects of Computer Science (1997), 261–282.

[40] S. Geman and D. Geman, *Stochastic relaxation, gibbs distributions, and the bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **6** (1984), no. 6, 721–741.

[41] B. Glocker, N. Komodakis, N. Paragios, G. Tziritas, and N. Navab, *Inter and intra-modal deformable registration: Continuous deformations meet efficient optimal linear programming.*, Proceedings of the 20th International Conference on Information Processing in Medical Imaging. (2008).

[42] A. Goldberg and R.Tarjan, *A new approach to the maximum-flow problem*, vol. 35(4), 1988, pp. 921–940.

[43] M. Gondran and M. Minoux, *Graphes et algorithmes*, Eyrolles, 1995.

[44] L. Grady, *Computing exact discrete minimal surfaces: Extending and solving the shortest path problem in 3d with application to segmentation*, Proceedings of CVPR 2006 (2006), 69–78.

[45] ———, *Random walks for image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006), no. 11, 1768–1783.

[46] L. Grady and E.L. Schwartz, *Isoperimetric partitioning: A new algorithm for graph partitioning*, SIAM Journal on Scientific Computing **27** (2006), no. 6, 1844–1866.

[47] L. Grady and A. K. Sinop, *A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm*, Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil (2007).

[48] D.M. Greig, B.T. Porteous, and A.H. Seheult., *Exact maximum a posteriori estimation for binary images*, Royal Journal of Statistics, Soc. B **51(2)** (1989), 271–279.

[49] O. Grygorash, Y. Zhou, and Z. Jorgensen, *Minimum spanning tree based clustering algorithms*, Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06) (2006), 73–81.

[50] Y. Hame, *Liver tumor segmentation using implicit surface evolution.*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[51] W.R. Hamilton, *Account of the icosian game*, Proceedings of the Royal Irish Academy **6**, 415–416.

[52] J.M. Hammersley and P. Clifford, *Markov field on finite graphs and lattices*, Unpublished.

[53] T.C. Hu, *The maximum capacity route problem*, Journal of Operations Research **9** (1961), no. 2, 898–900.

[54] H. Ishikawa, *Exact optimization for markov random fields with convex priors*, IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003), no. 10, 1333–1336.

[55] E. Ising, *Beitrag zur theorie des ferromagnetismus*, Z. Physik **31** (1925), 235–258.

[56] T.N. Janakiraman, *Image segmentation based on minimal spanning tree and cycles*, Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007) **3** (2007), 215–219.

[57] I.H. Jermyn and H. Ishikawa, *Globally optimal regions and boundaries as minimum ratio cycles*, IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001), 1075–1088.

[58] D. Karger, P. Klein, C. Stein, M. Thorup, and N. Young, *Rounding algorithms for a geometric embedding of minimum multiway cut.*, STOC'99 (1999), 668–678.

[59] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes : Active contour models*, International Journal of Computer Vision **1(4)** (1988), 321–331.

[60] R. Kindermann and J.L. Snell, *Markov random fields and their applications*, American Mathematical Society, Providence, R.I (1980).

[61] V. Kolmogorov and R. Zabih, *What energy functions can be minimized via graph cuts?*, IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (2004), 147–159.

[62] B. Korte and J. Vygen, *Combinatorial optimization. theory and algorithms*, Springer, 2005.

[63] A. H. Land and A. G. Doig, *An automatic method for solving discrete programming problems.*, Econometrica. **28** (1960), 497–520.

[64] Y. Li, S. Hara, and K. Shimura, *A machine learning approach for locating boundaries of liver tumors in ct images*, Proceedings of the 18th International Conference on Pattern Recognition (2006), 400–403.

[65] Y. Li, J. Sun, C. Tang, and H. Shum, *Lazy snapping*, SIGGRAPH 2004, ACM Transaction on Graphics **23** (2004), 303–308.

[66] R. Lotufo and A. Falcao, *The ordered queue and the optimality of the watershed approaches*, Proceedings of the International Symposium on Mathematical Morphology 2000, Palo Alto, CA, Jun 2000. (2000), 341–350.

[67] R. Lu, P. Marziliano, and C.H. Thng, *Liver tumor volume estimation by semi-automatic segmentation method*, Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference Shanghai, China, September 1-4. (2005), 3297–3299.

[68] B. Marcotegui, P. Correia, F. Marqués, R. Mech, R. Rosa, M. Wollborn, and F. Zanoguera, *A video object generator tool allowing friendly user interaction*, IEEE International Conference on Image Processing (Kobe, Japan), October 1999.

[69] C.E. Mathieu and I.E. Magnin, *On the choice of the first level on graph pyramids*, Journal of Mathematical Imaging and Vision. **6** (1996), 85–96.

[70] J. McQueen, *Some methods for classification and analysis of multivariate observations*, In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability **1** (1967), 281–297.

[71] F. Meyer, *Minimal spanning forests for morphological segmentation*, ISMM'94, Mathematical Morphology and its applications to Signal Processing (1994), 77–84.

[72] ———, *Topographic distance and watershed lines.*, Signal Processing **38** (1994), 113–125.

[73] ———, *Grey-weighted, ultrametric and lexicographical distances*, Computational Imaging and Vision, Mathematical Morphology: 40 Years On. Proceedings of the 7th International Symposium on Mathematical Morphology, vol. 30, 2005, pp. 289–298.

[74] F. Meyer and S. Beucher, *Morphological segmentation*, Journal of Visual Communication and Image Representation **1** (1990), no. 1, 21–46.

[75] _____, *The morphological approach to segmentation: the watershed transformation*, Mathematical Morphology in Image Processing (E.R. Dougherty, ed.), 1993, pp. 433–481.

[76] F. Meyer and C. Vachier, *Image segmentation based on viscous flooding simulation*, Proceedings of The 6th International Symposium on Mathematical Morphology, Sydney, Australia (2002), 69–77.

[77] M. Mohri, *Semiring frameworks and algorithms for shortest-distance problems*, Journal of Automata, Languages and Combinatorics **7**, 321–350.

[78] J. Moltz, L. Bornemann, V. Dicken, and H. Peitgen, *Segmentation of liver metastases in ct scans by adaptive thresholding and morphological processing*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[79] E.N. Mortensen and W.A. Barrett, *Interactive segmentation with intelligent scissors*, Graphical Models and Image Processing **60** (1998), no. 5, 349–384.

[80] L. Najman and M. Schmitt, *Watershed of a continuous function*, Signal Processing **38** (1994), 99–112.

[81] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*, Springer.

[82] M. Pardàs, *Segmentaciòn morfológica de secuencias de imàgenes: Aplicación a la codificación*, PhD thesis, Universitat Politcnica de Catalunya, Barcelona, Spain. (1995).

[83] T. Popa, L. Ibanez, E. Levy, A. White, J. Bruno, and K. Cleary, *Tumor volume measurement and volume measurement comparison plug-ins for volview using itk*, Proceedings of the SPIE : The International Society for Optical Engineering **6141** (2006), 395–402.

[84] F.A. Potra and S.J. Wright, *Interior-point methods*, Journal of Computational and Applied Mathematics (2000), 281–302.

[85] F. Preteux, *On a distance function approach for gray-level mathematical morphology*, In Mathematical Morphology in Image Processing, E. R. Dougherty, Ed. Marcel Dekker, New York **10** (1993), 323–349.

[86] Y. Qi, W. Xiong, W. Leow, Q. Tian, J. Zhou, and J. Liu, *Semi-automatic segmentation of liver tumors from ct scans using bayesian rule-based 3d region growing*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[87] J.B.T.M. Roerdink and A. Meijster, *The watershed transform: Definitions, algorithms and parallelization strategies*, Fundamenta Informatica **41** (2001), 187–228.

[88] P. Salembier and J. Serra, *Flat zones filtering, connected operators and filters by reconstruction*, IEEE Transactions on Image Processing **3** (1995), no. 8, 1153–1160.

[89] G. Schmidt, G. Binnig, M. Kietzmann, and J. Kim, *Cognition network technology for a fully automated 3d segmentation of liver tumors*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

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

[91] J.A. Sethian, *Level set methods and fast marching methods*, Cambridge University Press.

[92] J. Shi, S. Belongie, T. Leung, and J. Malik, *Image and video segmentation: The normalized cut framework*, IEEE Int'l Conf on Image Processing, Chicago, Illinois (1998), 943–947.

[93] A. Shimizu, T. Narihira, D. Furukawa, H. Kobatake, S. Nawano, and K. Shinozaki, *Ensemble segmentation using adaboost with application to liver lesion extraction from a ct volume*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[94] D. Smeets, B. Stijnen, D. Loeckx, B. De Dobbelaer, and P. Suetens, *Segmentation of liver metastases using a level set method with spiral-scanning technique and supervised fuzzy pixel classification.*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[95] T.S. Sørensen, G.F. Greil, O.K. Hansen, and J. Mosegaard, *Surgical simulation - a new tool to evaluate surgical incisions in congenital heart disease?*, Interactive Cardiovascular and Thoracic Surgery **5** (2006), 536–539.

[96] T.S. Sørensen, J. Stawiaski, and J. Mosegaard, *Virtual open heart surgery: Obtaining models suitable for surgical simulation*, Proceedings of Medicine Meets Virtual Reality 15. Stud Health Technol Inform. 2007;125:445-7 (2007).

[97] S.Sarkar and P. Soundararajan, *Supervised learning of large perceptual organization: Graph spectral partitioning and learning automata*, IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000), 504–525.

[98] J. Stawiaski, F. Bidault, E. Decencière, T. Girinsky, C. Le Pchoux, I. Ferreira, F. Dhermain, F. Meyer, J. Bourhis, and R. Sigal, *Analyse dimages ct 4d : Application à la segmentation des poumons.*, Journée française de Radiologie, Paris La Défense. (2007).

[99] J. Stawiaski, F. Bidault, E. Decencière, T. Girinsky, C. Le Péchoux, I. Ferreira, F. Dhermain, F. Meyer, J. Bourhis, and R. Sigal, *Segmentation interactive dimages médicales par hiérarchie de partitions, apport des marqueurs*, Journée française de Radiologie, Paris La Défense. (2006).

[100] J. Stawiaski and E. Decencière, *Computing approximate geodesics and minimal surfaces using watershed and graph-cuts*, Proceedings of the The 8th International Symposium on Mathematical Morphology, Rio de Janeiro, Brazil **1** (2007), 349–360.

[101] ———, *Combining graph-cuts and morphological segmentation*, Image Analysis and Stereolgy **27** (2008), no. 1, 39–46.

[102] J. Stawiaski, E. Decencière, and F. Bidault, *Interactive liver tumor segmentation using graph cuts and watershed*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[103] J. Stawiaski, E. Decencière, and F. Bidault., *Spatio-temporal segmentation of 4d ct images.*, The European Consortium For Mathematics In Industry 2008, London, England. (2008).

[104] J. Stawiaski, E. Decencière, and D. Jeulin, *Interactive segmentation of microstructures*, Workshop on 3D image analysis and modelling of microstructures, ITWM Fraunhofer, Kaiserslautern (2007).

[105] ———, *Segmentation semi-automatique dimages 3d de microtomographie*, Journées Annuelles de la Société Française de Métallurgie et de Matériaux. (2007).

[106] J. Stawiaski, J. Mosegaard, and T.S. Sørensen, *Virtual open heart surgery: Segmentation*, Proceedings of Medicine Meets Virtual Reality 15. Stud Health Technol Inform. 125:448-50 (2007).

[107] J. Stawiaski, T.S. Sørensen, E. Decencière, and F. Bidault, *Combining morphological and graph based methods for multi-label segmentation*, Workshop on interaction in medical image analysis and visualisation, MICCAI 2007, Brisbane, Australia (2007).

[108] J.J. Sylvester, *Chemistry and algebra*, Nature **17** (1878), 284.

[109] Y. Taieb, O. Eliassaf, M. Freiman, L. Joskowicz, and J. Sosna, *An iterative bayesian approach for liver analysis: tumors validation study.*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[110] K. Tsuda and G. Ratsch, *Image reconstruction by linear programming*, IEEE transactions on image processing. **14** (2005), 737–744.

[111] P. Tylsky, G. Bonniaud, E. Decencière, J. Stawiaski, D. Lefkopoulos, and M. Ricard, *FDG PET images segmentation using morphological watershed: a phantom study*, IEEE Nuclear Science Symposium and Medical Imaging Conference (San Diego (CA), USA), October 2006.

[112] O. Veksler, *Image segmentation by nested cuts*, IEEE Conf. Computer Vision and Pattern Recognition **1** (2000), 339–344.

[113] L. Vincent, *Algorithmes morphologiques à base de files d'attente et de lacets. extension aux graphes.*, Ph.D. thesis, Ecole des Mines de Paris, Paris, May 1990.

[114] W.E.Lorensen and H.E. Cline, *Marching cubes: A high resolution 3d surface construction algorithm*, Computer Graphics (Proceedings of SIGGRAPH '87) **21** (1987), 163–169.

[115] D. Wong, J. Liu, F. Yin, Q. Tian, W. Xiong, and J. Zhou, *A semi-automated method for liver tumor segmentation based on 2d region growing.*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).

[116] Z. Wu and R. Leahy, *An optimal graph theoretic approach to data clustering : theory and its applications to image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **15** (1993), no. 11, 1101–1113.

[117] F. Zanoguera, B. Marcotegui, and F. Meyer, *A tool-box for interactive image segmentation based on nested partitions*, IEEE International Conference on Image Processing (Kobe, Japan), October 1999.

[118] ———, *A segmentation pyramid for the interactive segmentation of 3-d images and video sequences*, Mathematical Morphology and its Applications to Image Processing, Proc. ISMM'00 (Palo Alto, California), Kluwer Ac. Publ., Nld, June 2000, pp. 263–272.

[119] J. Zhou, W. Xiong, Q. Tian, Y. Qi, J. Liu, and W. Leow, *Semi-automatic segmentation of 3d liver tumors from ct scans using voxel classification and propagational learning*, Workshop on 3D Segmentation in the Clinic: A Grand Challenge II. Liver Tumor Segmentation Challenge. MICCAI, New York, USA. (2008).