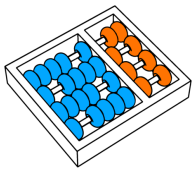
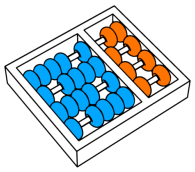


Nikolas Moya

"Interactive Segmentation of Multiple 3D Objects in Medical Images by Optimum Graph Cuts"

"*Segmentação Interativa de Múltiplos Objetos 3D em Imagens Médicas por Cortes Ótimos em Grafo*"

CAMPINAS
2015



University of Campinas
Institute of Computing

Universidade Estadual de Campinas
Instituto de Computação

Nikolas Moya

"Interactive Segmentation of Multiple 3D Objects in Medical Images by Optimum Graph Cuts"

Supervisor:
Orientador(a): **Prof. Dr. Alexandre Xavier Falcão**

"Segmentação Interativa de Múltiplos Objetos 3D em Imagens Médicas por Cortes Ótimos em Grafo"

MSc Dissertation presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a Master degree in Computer Science.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

This volume corresponds to the version of the Dissertation submitted to examining board by Nikolas Moya, under the supervision of Prof. Dr. Alexandre Xavier Falcão.

Este exemplar corresponde à versão da Dissertação apresentada à banca examinadora por Nikolas Moya, sob orientação de Prof. Dr. Alexandre Xavier Falcão.

Supervisor's signature / *Assinatura do Orientador(a)*

CAMPINAS
2015

Institute of Computing /*Instituto de Computação*
University of Campinas /*Universidade Estadual de Campinas*

Interactive Segmentation of Multiple 3D Objects in Medical Images by Optimum Graph Cuts

Nikolas Moya[1]

March 12, 2015

Examiner Board/*Banca Examinadora*:

- Prof. Dr. Alexandre Xavier Falcão (Supervisor/*Orientador*)
- Prof. Dr. Neucimar Jerônimo Leite
  Institute of Computing - UNICAMP
- Prof. Dr. Bruno Motta de Carvalho
  Department of Computer Science and Applied Mathematics - UFRN
- Prof. Dr. Guido Araújo
  Institute of Computing - UNICAMP (Substitute/*Suplente*)
- Prof. Dr. Paulo André Vechiatto de Miranda
  Institute of Mathematics and Statistics - USP (Substitute/*Suplente*)

[1]Financial support: CAPES 2013, FAPESP (2013/17991-0) 2014-2015

Abstract

Medical image segmentation is crucial to extract measures from 3D objects (body anatomical structures) that are useful for diagnosis and treatment of diseases. In such applications, interactive segmentation is necessary whenever automated methods fail or are not feasible. Graph-cut methods are considered the state of the art in interactive segmentation, but most approaches rely on the min-cut/max-flow algorithm, which is limited to binary segmentation while multi-object segmentation can considerably save user time and effort. This work revisits the differential image foresting transform (DIFT) – a graph-cut approach suitable for multi-object segmentation in linear time – and solves several problems related to it. Indeed, the DIFT algorithm can take time proportional to the number of voxels in the regions modified at each segmentation execution (sublinear time). Such a characteristic is highly desirable in 3D interactive segmentation to respond the user's actions as close as possible to real time. Segmentation using the DIFT works as follows: the user draws labeled markers (strokes of connected seed voxels) inside each object and background, while the computer interprets the image as a graph, whose nodes are the voxels and arcs are defined by neighboring voxels, and outputs an optimum-path forest (image partition) rooted at the seed nodes in the graph. In the forest, each object is represented by the optimum-path trees rooted at its internal seeds. Such trees are painted with a same color associated to the label of the corresponding marker. By adding/removing markers, the user can correct segmentation until the forest (its object label map) represents the desired result. For the sake of consistency in segmentation, similar seed-based methods should always maintain the connectivity between voxels and seeds that have labeled them. However, this does not hold in some approaches, such as random walkers, and when the output of segmentation is filtered to smooth object boundaries, better matching the result with the users' expectations. That connectivity is also paramount to make corrections without starting over the process at each user intervention. However, we observed that the DIFT algorithm fails in maintaining segmentation consistency in some cases. We have fixed this problem in the DIFT algorithm and when the obtained object boundaries are smoothed. These results are presented and evaluated on several 3D body anatomical structures from MR and CT images.

Resumo

Segmentação de imagens médicas é crucial para extrair medidas de objetos 3D (estruturas anatômicas) que são úteis no diagnóstico e tratamento de doenças. Nestas aplicações, segmentação interativa é necessária quando métodos automáticos falham ou não são factíveis. Métodos por corte em grafo são considerados o estado da arte em segmentação interativa, mas diversas abordagens utilizam o algoritmo min-cut/max-flow, que é limitado à segmentação binária, sendo que segmentação de múltiplos objetos pode economizar tempo e esforço do usuário. Este trabalho revisita a transformada imagem floresta diferencial (DIFT, em ingles) – uma abordagem por corte em grafo adequada para segmentação de múltiplos objetos – resolvendo problemas relacionados a ela. O algoritmo da DIFT executa em tempo proporcional ao número de voxels nas regiões modificadas em cada execução da segmentação (sublinear). Tal característica é altamente desejável em segmentação interativa de imagens 3D para responder as ações do usuário em tempo real. O algoritmo da DIFT funciona da seguinte forma: o usuário desenha marcadores (traço com voxels de semente) rotulados dentro de cada objeto e o fundo, enquanto o computador interpreta a imagem como um grafo, cujos nós são os voxels e os arcos são definidos por pixels vizinhos, produzindo como resultado uma floresta de caminhos ótimos (partição na imagem) enraizada nos nós sementes do grafo. Nesta floresta, cada objeto é representado pela floresta de caminhos ótimos enraizado em suas sementes internas. Tais árvores são pintadas com a mesmo cor associada ao rótulo do marcador correspondente. Ao adicionar ou remover marcadores, é possível corrigir a segmentação até o mapa de rótulo de objeto representar o resultado desejado. Para garantir consistência na segmentação, métodos baseados em semente sempre devem manter a conectividade entre os voxels e suas sementes. Entretanto, isto não é mantido em algumas abordagens, como random walkers ou quando o mapa de rótulos é filtrado para suavizar a fronteira dos objetos. Esta conectividade é primordial para realizar correções sem recomeçar o processo depois de cada intervenção do usuário. Todavia, foi observado que a DIFT falha em manter consistência da segmentação em alguns casos. Consertamos este problema tanto no algoritmo da DIFT, quanto após a suavização dos objetos. Estes resultados comparam diversas estruturas anatômicas 3D de imagens de ressonância magnética e tomografia computadorizada.

Acknowledgements

In the past two years I had the opportunity to meet many people that were involved in different steps of my work. It was an amazing experience.

First of all, I would like to thank Prof. Alexandre Xavier Falcão for his dedication while supervising me. His guidance, availability, patience, and enthusiasm were fundamental within this period.

Many thanks to Prof. Krzysztof Ciesielski (West Virginia University) and Prof. Jayaram Udupa (University of Pennsylvania) for comments, suggestions, and email discussions.

I would like to thank Prof. Luiz Eduardo Buzato and all other professors in the Institute of Computing at UNICAMP that taught me excellent classes. I will definitely take this knowledge with me along my career.

A special gratitude to FAPESP and CAPES for funding my studies.

A special thanks to my day to day friends, Paulo, John, Samuel, Spina, David, and Renzo. It was immeasurable the importance of our discussions about project ideas, coding, and writing skills. I believe that we all grew up together in this journey.

I do not want to forget my friends from Curitiba for their support and friendship along these academic years. Gregory, Mattioli, Pâmela, Guilherme, and Michele, the moments we spent together were great.

Most importantly, I am deeply grateful to my family for their absolute support. Special regards to my grandmother Ziloá and my childhood friend Thiago. At last, my sincerely acknowledgment to my adorable Marilia for her encouragement and her unconditional love.

"Computer science is no more about computers than astronomy is about telescopes."

Edsger W. Dijkstra

Contents

List of Tables

List of Figures

Chapter 1

Introduction

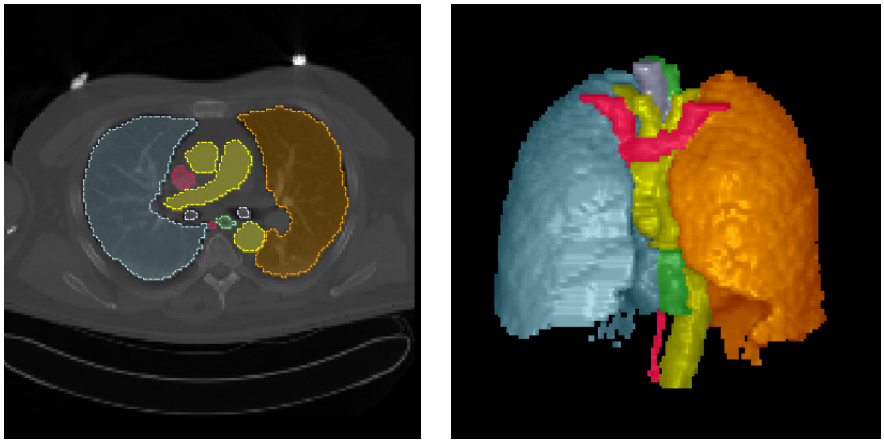
This chapter offers an overview of the present dissertation. It initially provides the motivation and explains the context of the work, which focus on tridimensional (3D) interactive segmentation of medical images by optimum graph cuts. The main research challenges and objectives on this topic are presented secondly. Finally, it outlines the main contributions and describes the structure of the text.

1.1 Context of the work

Medical image segmentation is crucial to analyze the shape and texture of body anatomical structures (3D objects) in computer-based systems for diagnosis and treatment of diseases [1]. Automated methods are desirable for these systems, but they can fail in several situations and are not viable in others. Despite the progress of object shape-texture models [2, 3, 4, 5] for automated medical image segmentation, the above difficulties make user (expert) interaction a necessity. Interactive segmentation methods are also needed to create object examples for shape-texture model construction.

Therefore, the present work focus on 3D interactive segmentation methods for anatomical medical images — i.e., Computer Tomography (CT) and Magnetic Resonance (MR) images. In this context, the challenges stem from artifacts in image acquisition, increasing image resolution when the response time to the user's actions need to be interactive, similar tissue properties that limit the image contrast between objects, and visual limitations of displaying 3D objects on a 2D screen (Figure 1.1). These difficulties challenge the methods and the user, who must verify the segmentation results and interact with the computer for correction.

In interactive segmentation, it is very important to minimize the user's time and effort and, at the same time, maintain the user's control over the segmentation process. For any input provided by the user, the computer should respond as close as possible to real



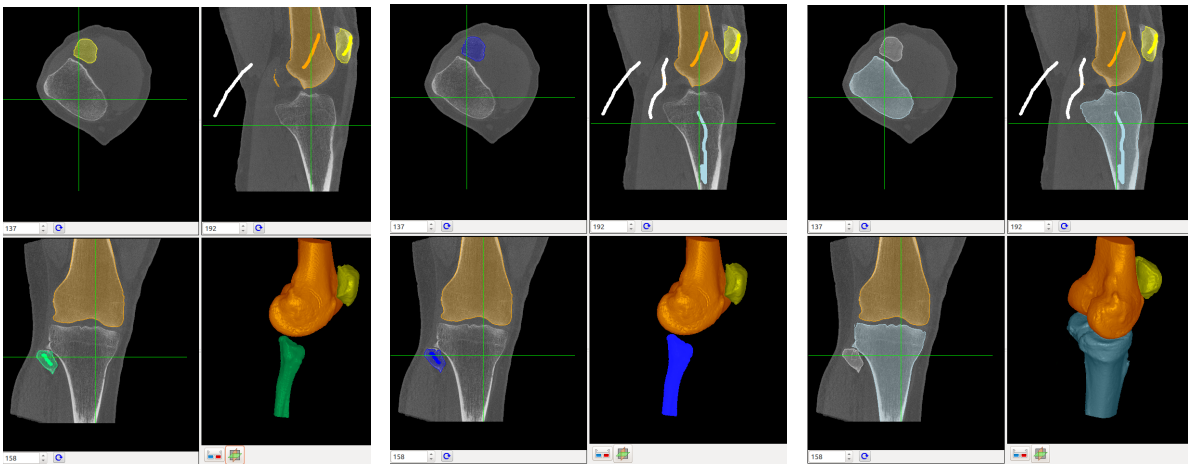
(a) (b)

Figure 1.1: (a) Axial slice of a 3D CT image from a human thorax. Each object is painted in a different color. (b) The 3D rendition of the objects. The structures are touching each other and their boundaries are hard to be visually distinguished.

time. In this context, graph-cut interactive segmentation methods are certainly among the state-of-the-art approaches [6, 7, 8, 9, 10, 11], being able to produce a fast response to the user's actions. These methods interpret the image as a graph, whose voxels are the nodes and arcs are defined by an *adjacency relation* between voxels. The graph is weighted on the arcs by some similarity (dissimilarity) measure, according to local image properties of their nodes, and object image properties (when available).

In order to make the human-computer interface more efficient and intuitive (Figure 1.2), the user input can be a set of *colored markers* (user-drawn strokes of connected seed voxels) selected inside and outside each object. The computer finds a minimum (maximum) energy graph cut based on the arc weights, which includes the internal seeds and excludes the external ones, separating objects and background. The color of the marker indicates the object and the user can add/remove markers to correct segmentation. However, the correction must be presented in interactive time, which requires a solution that does not start over the segmentation process for every new marker set [6, 9].

Figure 1.2 illustrates the behavior of a canonical interactive segmentation algorithm. Let $\hat{I} = (D, I)$ be a medical image, where $D \subset Z^3$ is its discrete domain and $I : D \to Z$ assigns an integer value to every voxel in D, representing some physical property related to the medical imaging modality (CT or MR). The body anatomical structures (3D objects) are subsets of D. By drawing colored markers in each object and background on image slices (coronal, sagittal, and axial), the user indicates a seed set $\mathcal{S}$ in which the object label of each voxel $s \in \mathcal{S}$ is known by $\lambda(s) \in \{0, 1, \ldots, n\}$ as 0 for background and $1 \leq i \leq n$ for object i. Additionally, each marker can be identified by a number $id(s) \in \{1, \ldots, m\}$,



(a) (b) (c)

Figure 1.2: (a) The user examines orthogonal slices and draws green, yellow and orange markers (connected seed voxels) to indicate three bones of interest in a CT image of the knee. A white marker is also selected in the background. The bottom-right part displays the 3D rendition of the resulting label map. (b) The user verifies the result and changes mind, picking any voxel in the influence zone of the green marker in (a) to remove a bone selected by mistake (the dark blue color indicates marked for removal). At the same time, the user inserts new markers (light blue for the correct bone and white for background) to adjust segmentation.(c) The algorithm removes the green bone and creates a new result for user verification.

where $m > n$. Note that one object may be defined from multiple markers. All these markers have the same object label, given by , but each marker has a distinct identification number, given by id. For a given adjacency relation $\mathcal{A}$, the pair $(D, \mathcal{A})$ is a graph whose arcs $(s,t) \in \mathcal{A}$ are weighted by a function $w(s,t) \geq 0$ based on properties extracted from I.

The seeds compete among themselves and their object labels are propagated in D in order to minimize (maximize) some energy function E. Let $L_O(s) \in \{0, 1, \ldots, n\}$ be the object label propagated to s from some seed in $\mathcal{S}$. The label map L_O defines a *cut* $\mathcal{C}$ in the graph, as the maximal subset of arcs $(s,t) \in \mathcal{A}$ such that $L_O(s) \neq L_O(t)$. The corresponding marker label $id(s)$ of each seed $s \in \mathcal{S}$ is also propagated in a marker label map $L_M \colon D \to \{1, 2, \ldots, m\}$. By doing that, the user has control over the markers, which can be selected for removal as well, by clicking on any voxel of their influence zones in L_M. This user action provides a set $\mathcal{M}$ with all voxels from the markers selected for removal in order to eliminate the influence zones of their seeds in L_O. Therefore, in a next execution of the algorithm, new seeds in $\mathcal{S}$ and marker voxels in $\mathcal{M}$ are provided by the user, and the algorithm must update L_O and L_M without starting over the process. In each execution,

the canonical algorithm searches for the graph cut with minimum (maximum) energy $E(\mathcal{C})$. This algorithm is presented next.

Algorithm 1 – Canonical Interactive Segmentation Algorithm

Input: An image $\hat{I} = (D, I)$, adjacency relation $\mathcal{A}$, arc-weight function w, and energy function E.

Output: Object label map $L_O: D \rightarrow \{0, 1, \ldots, n\}$ and marker label map $L_M: D \rightarrow \{1, 2, \ldots, m\}$.

1. ***do***
2. | *Get from the user, seeds in $\mathcal{S}$ labeled by λ and id, and marker voxels in $\mathcal{M}$.*
3. | *Run graph-cut segmentation with $\hat{I}$, $\mathcal{A}$, $\mathcal{S}$, $\mathcal{M}$, and w*
4. | *to minimize E and propagate labels to get L_O and L_M.*
5. | *Present the label map L_O to the user.*
6. ***While*** *User is not satisfied with L_O.*

The above algorithm provides full control to the user, who can add and remove markers until satisfaction. The graph-cut segmentation method should prepare the influence zones of markers for removal to be reconquered. The previous and added markers must compete in the current segmentation execution, but without starting over segmentation process.

Markers from a same color indicate the same object in L_O, which may consist of multiple connected components according to $\mathcal{A}$. However, for consistency, each voxel $t \in D$ labeled by a seed $s \in \mathcal{S}$ must remain connected to that seed according to $\mathcal{A}$. That is, there must exist a sequence $\pi_{s \rightarrow t} = \langle s_1, s_2, \ldots, s_k \rangle$, $s_1 = s$, $s_k = t$, $(s_i, s_{i+1}) \in \mathcal{A}$, $i = 1, 2, \ldots, k-1$, of distinct nodes in the graph (simple path) such that $L_O(s_i) = \lambda(s)$, $i = 1, 2, \ldots, k$. If this condition is not satisfied for some voxel in D, the segmentation is said *inconsistent*.

1.2 Objectives

This work aims to develop efficient and effective graph-cut segmentation methods for Algorithm 1. Efficient in the sense that it

- minimizes the number of markers and executions to achieve satisfactory segmentation,
- updates segmentation without starting over the process, and
- provides response time proportional to the number of modified voxels when updating segmentation (sub linear time in practice).

Effective in the sense that it provides consistent segmentation results that best match with the users' expectations.

In [12, 13], it was shown that, for real images, graph-cut segmentation requires hard constraints (internal and external seeds) to be effective. Ciesielski et al. [14] showed that several approaches, such as power watersheds [15], watershed cuts [16], iterative relative fuzzy connectedness [17, 18, 19], grow-cut [20], actually minimize one of two types of energy functions, E_{sum} and $E_{\max}$.

$$E_{sum}(\mathcal{C}) = \sum_{\forall(s,t)\in\mathcal{C}} w^{\alpha}(s,t), \tag{1.1}$$

$$E_{\max}(\mathcal{C}) = \max_{\forall(s,t)\in\mathcal{C}} w(s,t), \tag{1.2}$$

where $\alpha \geq 1$. Miranda and Falcão [21] also observed that the previous formulation used in [12] must actually be reduced to Equation 1.1 when internal and external seeds are used as hard constraints for the sake of effectiveness. Moreover, when α increases, both energy functions produce similar results.

Graph-cut segmentation methods based on E_{sum} are named GC_{sum} while methods based on $E_{\max}$ are named GC_{max} in [14]. GC_{sum} approaches can produce objects with smoother boundaries than $GC_{\max}$, and that is the motivation for hybrid approaches [8, 10]. On the other hand, GC_{sum} is limited to binary segmentation, being an NP-hard problem for multi-object segmentation. Therefore, solutions for Algorithm 1 based on GC_{sum} are suboptimal in the sense that each object must be segmented by turn, also compromising the efficiency of the algorithm. On the other hand, $GC_{\max}$ can segment multiple objects simultaneously, satisfying Equation 1.2. Moreover, $GC_{\max}$ usually requires less seeds (user effort) and is more robust to seed location than GC_{sum} (Figure 1.3).

The polynomial-time max-flow/min-cut algorithm [22] is the standard choice to implement methods based on GC_{sum} while $GC_{\max}$ can be implemented in linear time by several algorithms [19, 9], being the most efficient implementations easily reduced to an Image Foresting Transform (IFT) [23]. Indeed, in order to achieve respose time proportional to the modified regions during segmentation updates and avoid to start over the process for every new seed set, one should use the Differential IFT (DIFT) algorithm [9].

Therefore the solutions proposed in this work are based on the DIFT algorithm. It is also desirable to obtain objects with smooth boundaries. Given that $GC_{\max}$ tends to produce more irregular boundaries than GC_{sum}, the label map L_O can be filtered by diffusion [24], for instance, after each segmentation execution or at the end of Algorithm 1. Figure 1.4, however, shows that shape relaxation can destroy connectivity between seeds and voxels labeled by them, causing inconsistent segmentation. This also makes segmentation correction ineffective with the DIFT algorithm.

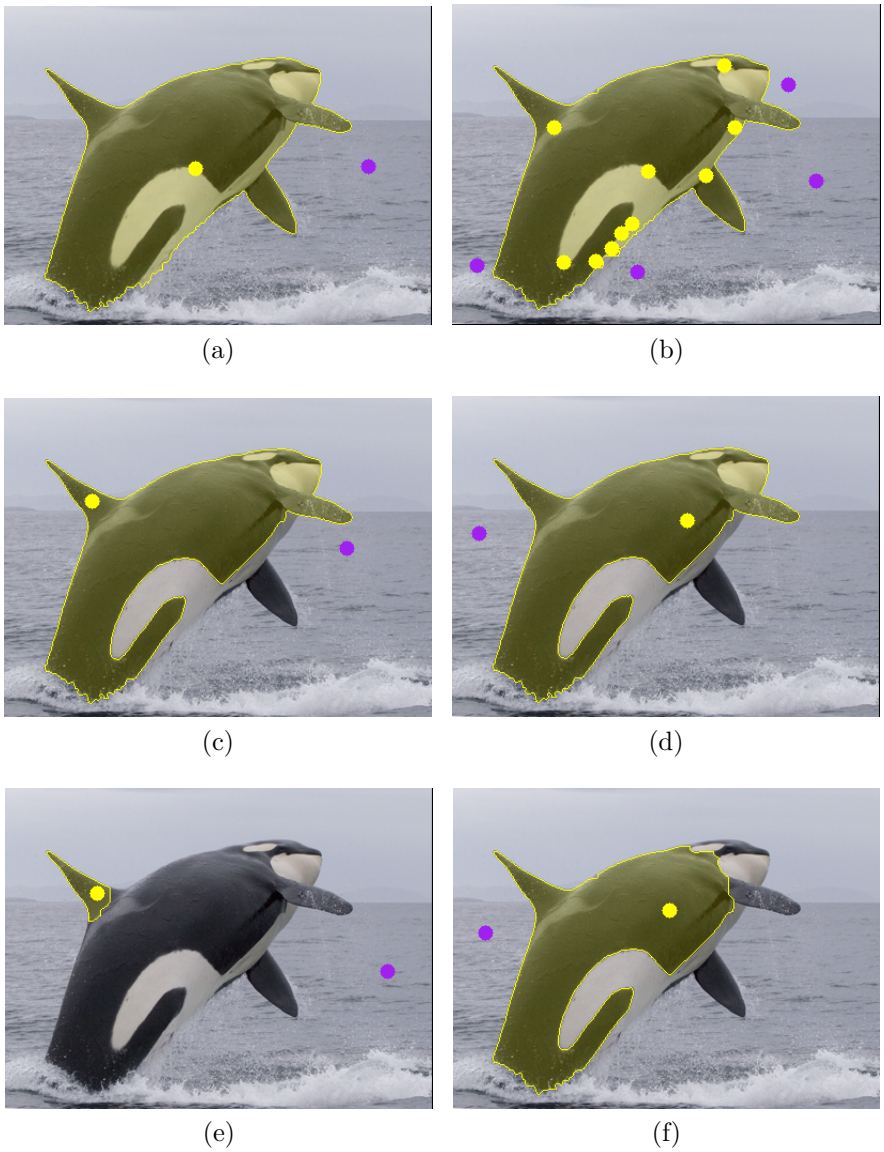
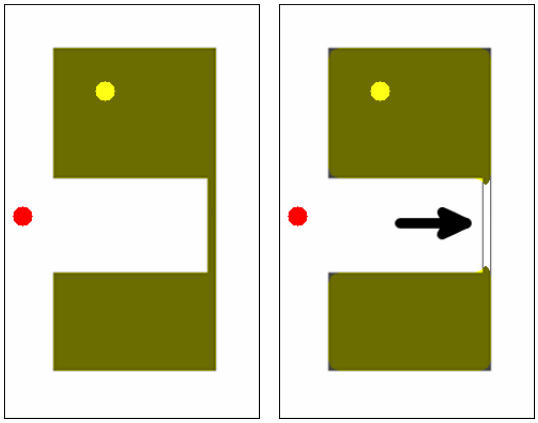


Figure 1.3: (a) shows that GC_{max} requires less seeds than (b) GC_{sum}; and (c) and (d) show that GC_{max} is more robust to seed positioning than (e) and (f) GC_{sum}.

1.3 Contributions

This dissertation proposes and evaluates the following solutions for Algorithm 1: $GC_{\max}$ approaches based on the DIFT algorithm without boundary relaxation, with boundary relaxation during segmentation, and with boundary relaxation afterwards. The baseline is the Dynamic Graph Cut method (GC_{sum}) [6], which represents the most efficient implementation of the max-flow/min-cut algorithm. The DGC algorithm can update seg-



(a) (b)

Figure 1.4: (a) Segmentation without boundary relaxation. (b) Depending on the relaxation parameters, the connection between marker and voxels labeled by it may be lost, causing inconsistent segmentation.

mentation for a new seed set without starting over the process. During the development of this work, we have also observed that the original DIFT algorithm [9] does not produce a consistent segmentation in all situations. Therefore, the main contributions of this work are:

1. A comparison between GC_{max} and GC_{sum} approaches.
2. A new DIFT Algorithm that produces consistent segmentation.
3. A much faster algorithm than the one proposed in [24] for boundary relaxation by diffusion.
4. A procedure to fix segmentation inconsistency caused by diffusion, which can also be used after any other relaxation filter.
5. An interactive tool with graphical user interface for multi-object medical image segmentation using the proposed algorithms.

The comparison in Item 1 used a geodesic robot [25], which simulates the actions of an untrained user during segmentation, without the bias from different users. A preliminary comparison was published in [7], but the proposed algorithms have been modified here to guarantee segmentation consistency.

Note that the extension of the proposed algorithms to n-D images from any application ($n = 2, 3, 4, \ldots$) is straightforward, given that it depends only on a suitable definition of $\mathcal{A}$ and w.

1.4 Organization

This dissertation is organized as follows: Chapter 2 presents the basic concepts, such as representing an image as a graph, discusses in more details the related graph-cut segmentation methods, such as the Image Foresting Transform. Chapter 3 covers the proposed solutions and implementation details. The experiments made for validation are shown in Chapter 4. The conclusive remarks and discussion about future work are presented in Chapter 5.

Chapter 2

Image Segmentation

Segmentation can be defined as the process of identifying and separating relevant objects in a given image. It is a fundamental image analysis problem and constitutes one of the main challenges in Image Processing and Computer Vision. This chapter presents the main tasks and methods for image segmentation, and subsequently discusses interactive segmentation from the viewpoint of user-given constraints. We are interested in region-based constraints and image segmentation by optimum graph cuts. Therefore, the chapter describes how to interpret images as graphs and reduce segmentation into an optimum graph-cut problem. It addresses two main algorithms for interactive graph-cut segmentation and concludes by justifying our choice for these algorithms.

2.1 Main tasks and methods

The segmentation process consists of two tightly coupled tasks: *recognition* and *delineation* [26]. Recognition is the task of roughly determining the location of a desired object in the image, while delineation consists in determining its exact spatial extent. Humans outperform computers in recognition, but the other way around is observed for delineation. As consequence, interactive (semi-automatic) segmentation methods usually combine the superior abilities of humans for recognition with a more precise object delineation by the computer. The most successful approaches, however, can provide user control over the process, to guarantee the desired result, and simultaneously minimize user time and effort in segmentation. Ciesielski et al.[14] classify segmentation methods into three groups: purely image-based, object model-based, and hybrid approaches. Figure 2.1 illustrates this classification, with examples of well known segmentation methods.

Image-based methods exploit local image properties basically for object delineation, with user input usually taken for object recognition. They can be further divided according to their delineation strategy into *region-based*, *boundary-based*, and *hybrid* approaches.

Optimum thresholding [27] and pixel classification [28] are the simplest image-based approaches for region-based delineation. However, the absence of object recognition in these approaches make their use limited to situations where object and background do not present similar image properties. More effective region-based methods, such as random walkers [29], geodesic-path segmentation [30], watersheds [31, 32, 33, 34, 15, 16], max-flow/min-cut segmentation[12], fuzzy connectedness algorithms [17, 18, 19], and level sets [35], define the regions that constitute the interior of the object based on some hard constraint. For example, they usually take internal and/or external markers as user input for delineation. In boundary-based approaches, such as live wire [36] and 3D live wire [37], user input is a set of anchor points selected along the object's boundary during delineation. Another example of user input for recognition in boundary-based approaches is the initialization of a parametric curve, such as a "snake model" [38], which subsequently deforms to delineate the object. Hybrid approaches can combine different types of user input and delineation methods to explore the best features of each [39, 40].

In some controlled situations, image-based methods can be used with object location based on image properties only [41]. The user input (e.g., internal and external markers) can also improve delineation by taking into account the image properties that best distinguish the object from its surrounding background [42]. However, image-based methods do not build any object model to assist segmentation. Methods based on object models, such as active appearance models (AAM) [4], active shape models (ASM) [5], and atlas-based methods (AM) [43], encode texture and/or shape information about the object, as obtained from interactively segmented images. In AAM and ASM, recognition is usually helped by the user, who initializes the model close to the object, and delineation can be very simple, by searching for high-contrast points that match with the control points of the model. In AM, recognition is based on deformable image registration. By placing the image in the same coordinate system of the atlas, the methods assume that the object can be delineated by thresholding a probabilistic map. However, more effective approaches are hybrid, since they exploit the above methods for image-based delineation while the object model helps on recognition. Examples are cloud models (CM) [2], fuzzy object models (FOM) [3], oriented active shape models (OASM) [44], and atlas with local object search [45].

2.2 Interactive image segmentation

At the most basic level, the user can interact with the computer by tunning the parameters that control the delineation algorithm. However, this type of interaction does not provide the full user control as desired for the interactive segmentation process. Instead, we are more interested in pictorial input [46], for example, methods where the user guides the

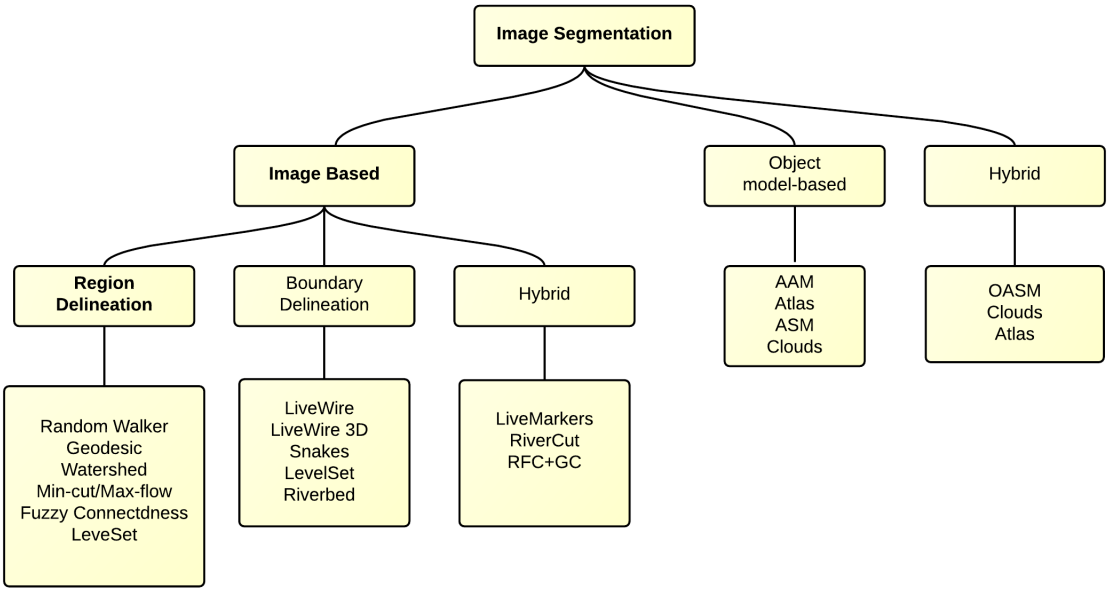
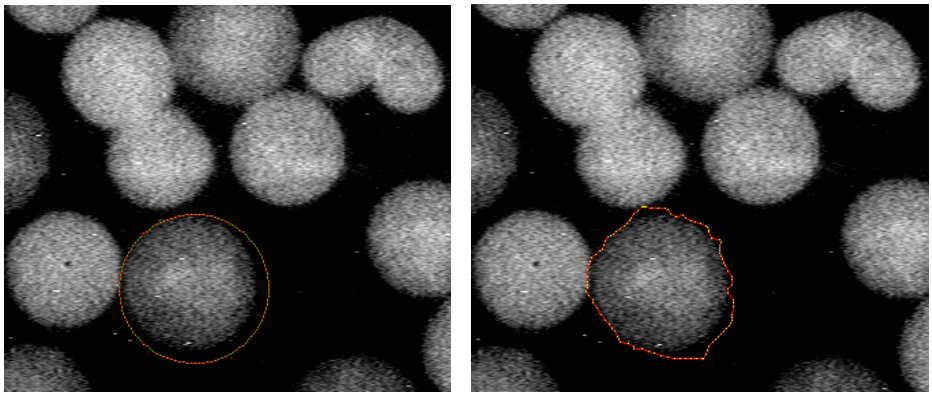


Figure 2.1: Image segmentation algorithms can be classified into three groups: purely image-based, object model-based, and hybrid. Image-based methods can be further divided according to the delineation strategy into region-based, boundary-based, and hybrid.

segmentation, by making some annotation in the image domain. These methods can be divided into two paradigms: approaches based on *soft constraints* and *hard constraints*. A soft constraint tends to guide the delineation method towards a particular solution, but it cannot reduce the set of feasible segmentation solutions in any way. As an example, the user can provide a parametric curve as close as possible to the desired boundary, but there is no guarantee that the automatic deformation of that curve towards the desired boundary will succeed. Besides that, once the final segmentation is achieved by optimizing some energy function, there is no mechanism to override this result. Active contour [38] and level-set methods [35] are examples of such soft constraints (Figure 2.2). Note that, they cannot accomplish with the aim of full user control over the process.

On the other hand, hard constraints can provide a higher degree of user control than soft constraints, making sure that the final segmentation is accepted by the user. They can be divided into boundary-based or regional-based constraints. For boundary-based constraints, all boundary elements (bels) specified by the user must be included in the final segmentation boundary. Live wire [36] is an example of method with boundary-based constraints — i.e., anchor points selected by the user along the object's boundary as the computer delineates optimum segments between subsequent points. The user has full



(a) (b)

Figure 2.2: (a) Initialization of a parametric curve (snake) around an object of interest. (b) Final segmentation after convergence, using the software in [47]. In order to correct segmentation, the process has to start over, sometimes from a different position of the curve and with adjustment of the parameters.

control over the process, because a new point is only selected when the segment from the previous point and the current position of the mouse's cursor describes a desired boundary segment, according to the user's judgement.

For region-based constraints, the object labels provided by the user must be preserved inside the objects in the final segmentation. For example, segmentation methods based on geodesic paths [30], watershed transform [48], and fuzzy connectedness [19]. Labeled (colored) markers are drawn by the user in each object (and background) and the segmentation result must include those markers. Such hard constraints aim at reducing the search space for a desired segmentation, but at the same time, they must reduce user time and effort, avoiding exhaustive and extensive marker selection. In *grabcut* [49], for instance, the user draws a rectangle around the desired object (Figure 2.3). By doing that, the user indicates background image properties, which are used to estimate object markers automatically. For 3D images, however, it is not so simple to select a bounding box for a given body anatomical structure. In such a case, the most suitable and intuitive hard constraints are colored markers (scribbles) drawn inside and outside the object (Figure 2.4). It is also not intuitive for the user to impose boundary-based constraints by examining cross sectional slices. This naturally guide us to the choice of region-based approaches for object delineation.

The interpretation of the input image as a graph is a powerful abstraction to adapt the vast literature of graph algorithms for object delineation. The most efficient examples from the state-of-the-art are the dynamic version of the max-flow/min-cut algorithm [6] and the differential version of the generalized Dijkstra's algorithm, as proposed by using

(a) (b)

Figure 2.3: (a) The user draws a rectangle around the object of interest and the object markers are estimated automatically (b) Delineation result from the hard constraints provided by the user [49].

(a) (b)

Figure 2.4: (a) The user draws yellow markers inside the object and black markers in the background. (b) Delineation result from the hard constraints provided by the user [9].

the Image Foresting Transform methodology [23, 9]. We evaluate both algorithms in this dissertation and propose considerable improvements in the second one.

2.3 Images as graphs

A digital image $\hat{I}$ can be defined as a pair $(D, \vec{I})$, where D is a set of points in Z^n (image domain), denominated *spels*, and $\vec{I}$ is a vectorial mapping that assigns to each spel $s \in D$ a vector $\hat{I}(s) = (I_1(s), I_2(s), \ldots, I_k(s))$ of scalar values associated with physical properties under study. The image dimension n and the number of bands k depend on the imaging modality. The methods in this work can be adapted to any image modality, but we will focus on medical images suitable to study body anatomical structures, such as Computerized Tomography (CT) and Magnetic Resonance Imaging (MRI).

CT and MR images are routinely used to generate high resolution 3D images of the

human body, wherein the spels are referred to as volume picture elements (voxels). Since they are acquired in grayscale, we will refer to them as a pair $\hat{I} = (D, I)$ with image domain $D \subset Z^3$ and mapping function $I : D \to Z$ that assigns an integer value $I(s)$ to each voxel $s \in D$. Higher values are displayed as brighter voxels and lower values as darker ones. In CT images, voxel intensities are proportional to the tissue density. We will use only MR images whose intensities measure the differences in the T1 relaxation time of tissues. In this case, the contrast between different tissues is determined by the rate at which excited hydrogen atoms from water molecules return to the equilibrium state, after being subjected to one of the basics pulse sequences of the magnetic field.

In order to interpret a 3D image as a weighted graph, one needs to define an *adjacency relation* between voxels and an *arc-weight function*. For image segmentation, the arc-weight function should assign lower (higher) weights to arcs across the object boundaries and higher (lower) weights elsewhere. This helps the algorithms to separate objects and background by either following the arcs along the object's boundary or by avoiding them during delineation (region-based approach). In both cases, the arcs along the object's boundary must be cut according to some minimization (maximization) criterion in order to separate objects and background. The image-graph definition can then be presented as follows.

2.3.1 Adjacency relation

An adjacency relation $\mathcal{A}$ is a binary relation which takes into account a distance criterion between voxels. It is also possible to use additional image properties, but they might output disconnected graphs with no relation to the object definition. From the practical viewpoint of user control, it is desirable that the adjacency relation be symmetric and shift-invariant, forming a connected graph.

It is also common to define the adjacency relation based on the Euclidean metric. A voxel t belongs to the adjacent set $\mathcal{A}(s)$ of a voxel s when the distance between them is less than a given scalar $\rho > 0$.

$$\forall s \in D, \mathcal{A}(s): \quad \{t \in D, \|t - s\| \leq \rho\} \tag{2.1}$$

This implies an image graph $(D, \mathcal{A})$, where the arc $(s, t) \in \mathcal{A}$. Figure 2.5 shows examples for 2D and 3D images. For 2D images, $\rho = \sqrt{2}$ defines an 8-neighborhood image graph. For 3D images, we use $\rho = \sqrt{3}$ to connect all 26 neighbors of each voxel.

2.3.2 Arc-weight function

Arc-weight estimation is actually an application-dependent pre-processing step for any object delineation algorithm. The weight may be referred to as distinct attribute func-

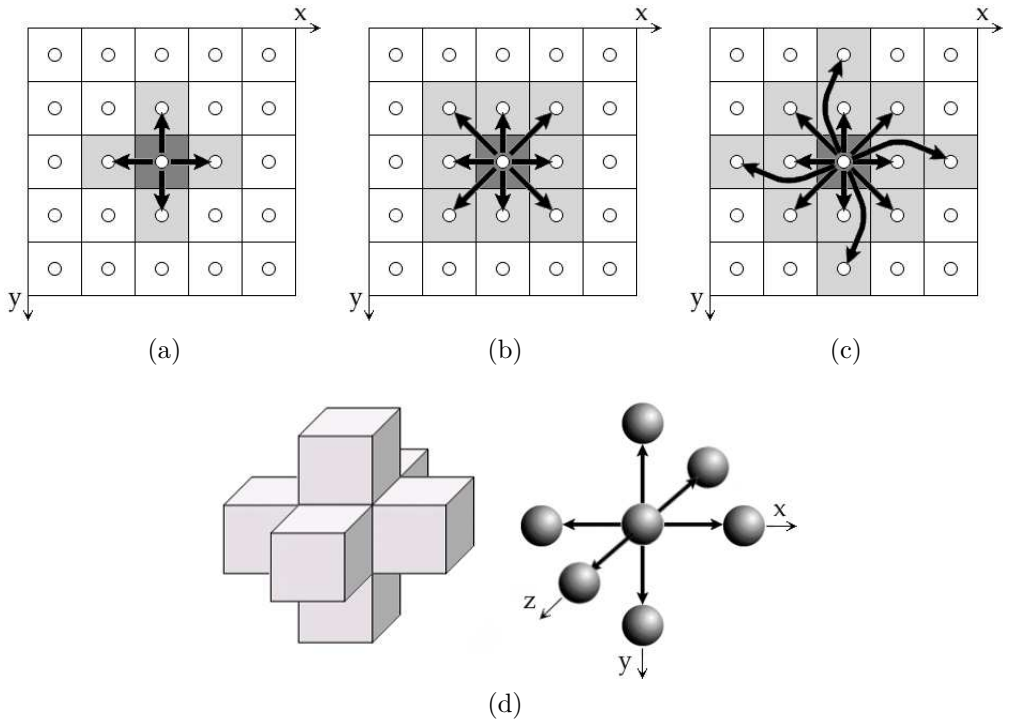


Figure 2.5: Euclidean adjacencies (a) 2D with $\rho = 1$. (b) 2D with $\rho = \sqrt{2}$. (c) 2D with $\rho = 2$. (d) 3D with $\rho = 1$

tionals, such as cost, gradient value, affinity, similarity, speed function, distance, local external energy, etc; depending on different frameworks used, such as watershed transforms, active contours, level sets, fuzzy connectedness, graph cuts, random warkers, etc. The accurate delineation by these methods with minimum user intervention strongly depends on a suitable arc-weight estimation, which usually takes into account local image properties and/or object information often obtained from markers selected by the user during segmentation. Since objects in medical images appear with similar image properties to parts of the background that are connected to those objects, it is difficult to make arc-weight estimation effective with object information from markers, as proposed in [42]. Therefore, we use a smoothing (Gaussian) filter to reduce noise followed by a local gradient vector $\vec{G}(s)$, computed on the resulting image $\hat{I} = (D, I)$.

$$\vec{G}(s) = \frac{\sum_{t \in \mathcal{A}(s)} [I(t) - I(s)]\vec{st}}{|\mathcal{A}(s)|}, \tag{2.2}$$

for all $s \in D$, where $\vec{st}$ is the unit vector from voxel s to voxel t. The arc weights are defined by $w(s,t) = \|\vec{G}(t)\|$ — i.e., $(D, \mathcal{A})$ is a directed and weighted image graph, suitable

for image segmentation by watershed transform, as we will see. Other algorithms require lower and symmetric arc weights along the object's boundary than inside and outside it. In this case, we use $\bar{w}(s,t) = \bar{w}(t,s) = G_{\max} - \frac{\|\vec{G}(s)\|+\|\vec{G}(t)\|}{2}$, where $G_{\max}$ is the maximum value of $\|\vec{G}(s)\|$, for all $s \in D$.

Depending on the body anatomical structure and imaging modality, the arc-weight estimation can be considerably improved. For instance, diffuse filtering could be used before arc-weight assignment to reduce noise in MR images without blurring the edges. The values of the gradient image, resulting from Equation 2.2, can be transformed to enhance the low-contrast boundary of a desired object, for instance. In this work, we simply assume a same gradient image, computed by Equation 2.2, for all methods, objects, and imaging modalities.

Despite the effort to improve arc-weight estimation for a given application, the possibility of displaying arc weights around each voxel s as a gradient image $\vec{G}(s)$ allows to assess the quality of the arc-weight assignment. After applying Equation 2.2, for instance, it is expected higher values along the object boundaries than inside and outside them (Figure 2.6).

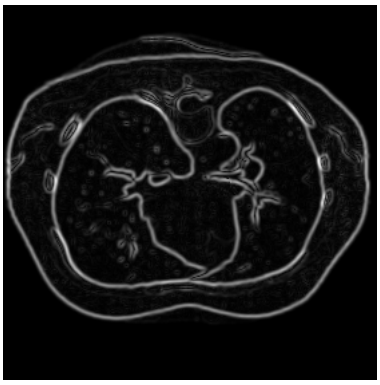


Figure 2.6: Example of arc-weight assignment suitable for lung delineation in CT images of the thorax.

2.4 Graph-based object delineation

Approaches for graph-cut segmentation are based on criterion functions that measure some global property of the object's boundary from the arc-weight assignment. The idea is to define a *cut boundary* in the graph — i.e., a set $\mathcal{C}$ of arcs linking nodes labeled as object and nodes labeled as background — that represents the desired segmentation when a graph-cut measure is minimum (maximum).

Wu and Leahy [50] were the first to propose a solution for graph-cut image segmentation using as measure the sum of the arc weights in the cut boundary (Equation 2.3). This cut measure has the bias towards small boundaries and to circumvent this problem, other objective functions were proposed, such as average cut [51], mean cut [52], average association [53], ratio cut [54], and normalized cut [55].

$$E_1(\mathcal{C}) = \sum_{\forall(s,t)\in\mathcal{C}} \bar{w}(s,t) \tag{2.3}$$

Unfortunately, the problem of segmenting a desired object in a given image cannot be simply reduced to finding a minimum of a criterion function in the entire search space, since false-cut boundaries due to similarities between object and background are very common in practice. Indeed, the problem of finding the minimum cut in a generic graph is NP-hard. Heuristic solutions have been proposed in polynomial time [56], but with poor computational performance and the results are sometimes far from the desired segmentation. That is, the running time of the graph-cut algorithms is usually $O(mn^2)$, being m the number of arcs and n the number of nodes.

Energy functions that incorporate object and background information in their formulation are more recent alternatives to reduce the small cut problem. For example, two terminal nodes (*source* and *sink*) can be added to the image graph, representing the object and background respectively [12, 57, 58]. These nodes are directly connected to all voxels by arcs whose weights reflect penalties for assigning a voxel to object and background based on image region properties (*probability maps*). A min-cut/max-flow algorithm from source to sink [22, 59] is used to compute the minimum-cut boundary, according to Equation 2.4.

$$E_2(\mathcal{C}) = E_1(\mathcal{C}) + \gamma\Big(\sum_{\forall(s,t)\in\mathcal{C}} \bar{P}_o(s) + \sum_{\forall(s,t)\in\mathcal{C}} \bar{P}_b(t)\Big) \tag{2.4}$$

where $\bar{P}_o(s)$ and $\bar{P}_b(t)$ are the complement of the probability maps P_o and P_b that measure how well the intensities of voxels s and t fit into a known intensity model of the object and background, respectively.

If the method fails the detection of the desired boundary, the user can impose the weights of the arcs that connect voxels with the source and sink by selecting seeds inside and outside the object. After a sufficient number of user intervetions, increasing the number of seeds at each interaction, the method can converge to the desired solution. However, this approach adds a lot of difficulties in Equation 2.4. The parameter γ specifies the importance of the region property term versus the boundary property term. If γ is

low, the cut is the same as in Equation 2.3, and if γ is high the method becomes very dependent on the probability maps (it becomes an optimum thresholding). That is, the pixels are not guaranteed to be connected to the markers that have labeled them in the image domain (a segmentation inconsistency), whenever object and background present regions with similar image properties.

A solution to overcome these problems is to drop the terms with probability maps (as in Equation 2.5) and force the entire simulated flow from source to pass through the internal markers, travel through the image graph, and achieve the sink through the external markers. Note that, higher values of α favor larger cut boundaries in the minimization of Equation 2.5. As the parameter α is lowered, object delineation requires more user effort in marker imposition (Figure 1.3b) to avoid small cuts. An interesting particular case, however, is the equivalence between the minimization of Equations 2.5 and 2.6, when $\alpha \to +\infty$ — i.e., $\lim_{\alpha\to+\infty} E_{sum}(\mathcal{C}) = E_{\max}(\mathcal{C})$.

$$E_{sum}(\mathcal{C}) = \sum_{\forall(s,t)\in\mathcal{C}} \bar{w}^{\alpha}(s,t) \tag{2.5}$$

$$E_{\max}(\mathcal{C}) = \max_{\forall(s,t)\in\mathcal{C}} \bar{w}(s,t) \tag{2.6}$$

This equivalence makes the cut boundary obtained by the min-cut/max-flow algorithm be the same as obtained by the image foresting transform (IFT) algorithm, when used to implement the relative fuzzy connectedness segmentation [21]. In the case of segmentation by watershed transform, this is also equivalent to the maximization of $E_{\min}(\mathcal{C})$.

$$E_{\min}(\mathcal{C}) = \min_{\forall(s,t)\in\mathcal{C}} w(s,t). \tag{2.7}$$

According to Ciesielski et al. [14], many segmentation methods from the state-of-the-art can be reduced to a graph-cut problem, whose solution is obtained by the minimization of either E_{sum} or $E_{\max}$ using internal and external markers as region-based hard constraints. Moreover, the min-cut/max-flow algorithm and the IFT algorithm are among the most efficient solutions to implement those segmentation methods. Therefore, the next sections present more details about these solutions.

2.4.1 Selecting region-based hard constraints

We must recall Algorithm 1 in this section. The user draws colored markers in each object, including background. For 3D images, the user interface shows three orthogonal

slices (axial, coronal and sagittal) for marker selection and the user can scroll these cross-sectional slices along the axes x, y, and z. Each marker is a connected set of *seed voxels* in D. The colors indicate object labels $\lambda(s) \in \{0, 1, \ldots, n\}$, being 0 for markers in the background and $1 \leq i \leq n$ for markers in the object i. For the sake of efficiency and user control, the markers are also labeled by an id function. Therefore, the user provides a seed set $\mathcal{S}$ from all markers, where each seed $s \in \mathcal{S}$ has object label $\lambda(s)$ and marker label $id(s) \in \{1, 2, \ldots, m\}$, $m > n$.

The markers propagate their labels in Lines 3-4 of Algorithm 1, while minimizing a given energy function, E_{sum} or $E_{\max}$ (or maximizing $E_{\min}$), such that the cut boundary will be defined by arcs $(s, t) \in \mathcal{A}$ where $L_O(s) \neq L_O(t)$. Therefore, each marker will label a connected region of voxels (*influence zone*) in the resulting label maps L_O and L_M, by propagating object label $L_O(s) \in \{0, 1, \ldots, n\}$ and marker label $L_M(s) \in \{1, \ldots, m\}$ to each voxel $s \in D$.

For segmentation in Lines 3-4 using the min-cut/max-flow algorithm, the method must first obtain the binary segmentation of each object $1 \leq i \leq n$ — i.e., label maps L_{O_1}, L_{O_2}, $\ldots$, L_{O_n} — from internal markers with label i and external markers with label $j \neq i$, $0 \leq j \leq n$. The multi-label map L_O can then be defined as $L_O(s) = \max_{i=1,2,\ldots,n} L_{O_i}(s)$ for each $s \in D$, assuming that the intersection between any pair of objects is empty. Similarly, it must create an unique marker label map L_M.

Given that marker selection may not be effective to solve segmentation in a single execution of Algorithm 1, the user should be able to add and remove markers to correct segmentation in a sequence of executions. The set $\mathcal{M}$ provides all voxels from markers selected for removal, which implies that the entire influence zone of removed markers must disappear from L_O and L_M after a next segmentation execution in Lines 3-4.

In order to achieve interactive response time to the user's actions in 3D image segmentation, it is paramount to correct segmentation in time proportional to the modified influence zones between executions. Object delineation by the min-cut/max-flow algorithm and the IFT algorithm cannot satisfy this condition. However, their variants, Dynamic Graph-Cut (DGC)[6] and Differential IFT[9], can update the label maps L_O and L_M without starting over the segmentation process, as required in the desired canonical solution (Algorithm 1).

The marker label map L_M and marker identification function id are not necessary when segmentation is computed from the beginning at each execution of Lines 3-4 in Algorithm 1. Therefore, they will be omitted in the next sections.

2.4.2 Object delineation by the min-cut/max-flow algorithm

Given an image graph $(D, \mathcal{A})$, whose arcs (s,t) are weighted by a function $\bar{w}(s,t)$ suitable to separate an object $1 \leq i \leq n$ from the background, the user provides a λ-labeled seed set $\mathcal{S} = \mathcal{S}_o \cup \mathcal{S}_b$, with object seeds in $\mathcal{S}_o$ and background seeds in $\mathcal{S}_b$. For object delineation using the min-cut/max-flow algorithm[57], the graph must be extended by two virtual nodes, a source o and a sink b, with arcs (o,s) and (s,b) connecting them to each voxel $s \in D$ (Figure 2.7). This algorithm is based on the Ford-Fulkerson theorem[22].

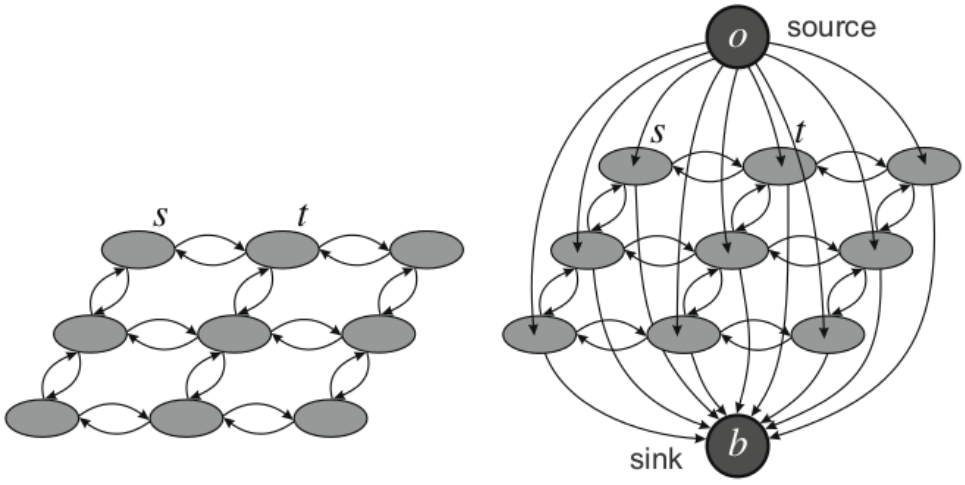
The delineation method simulates a flow from o to b, which should reach the object through its internal markers; visit the object's nodes, cross the object's boundary, and visit the background's nodes by passing through the arcs in the image domain; and finally reach b through its external markers. If the arc weights $\bar{w}(s,t)$ are lower across the object's boundary than inside and outside it; the arc weights $\bar{w}(o,s) = 0$ for $s \in \mathcal{S}_b$ and $\bar{w}(o,s) = +\infty$ for $s \in \mathcal{S}_o$; and the arc weights $\bar{w}(s,b) = 0$ for $s \in \mathcal{S}_o$ and $\bar{w}(s,b) = +\infty$ for $s \in \mathcal{S}_b$, the minimization of $E_{sum}(\mathcal{C}_i)$ will clearly insert in the cut $\mathcal{C}_i$ the arcs that connect $\mathcal{S}_o$ to background voxels; object voxels to $\mathcal{S}_b$; and internal voxels s to external voxels t. This process propagates object labels from the internal markers to all voxels inside the cut $\mathcal{C}_i$, leaving the remaining voxels with label 0. By eliminating from $\mathcal{C}_i$ the arcs with o and b, the final cut boundary $\mathcal{C}_i = \{(s,t) \in \mathcal{A}, L_{O_i}(s) \neq L_{O_i}(t)\}$ is implicitly represented in the resulting label map L_{O_i}.

Given that arc-weight assignment is never perfect, the user can add/remove markers, enforcing the object's voxels to be inside the cut. Such segmentation correction can be more efficiently by Markov random fields[6]. The algorithm, called Dynamic Graph-Cut (DGC), stores the flow obtained in the previous execution and uses it to initialize the subsequent execution, taking time proportional to the number of arcs in the modified regions of the label map L_{O_i}.

2.4.3 Object delineation by the IFT algorithm

The *Image Foresting Transform* (IFT) is a methodology to the design of image operators based on optimum connectivity [23]. For a given adjacency relation and connectivity function (path-value function), the IFT algorithm minimizes (maximizes) a *connectivity map* to partition the image graph into an *optimum-path forest* rooted at the minima (maxima) of the resulting connectivity map. The image operation is then obtained by a local processing on attributes of the forest (e.g., root labels, optimum paths, connectivity values). Examples can be found for image filtering [60], segmentation [61, 9], representation [62, 63], analysis [64], classification [65], and clustering [66].

In this section, we instantiate the IFT algorithm for multi-object image segmentation from a λ-labeled seed set $\mathcal{S}$, such that $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_n$ with seeds for object



(a) (b)

Figure 2.7: (a) A 2D image graph with 4-adjacent pixels s and t. (b) An extended graph obtained by adding two terminal nodes (source o and sink b), which represent object and background, respectively.

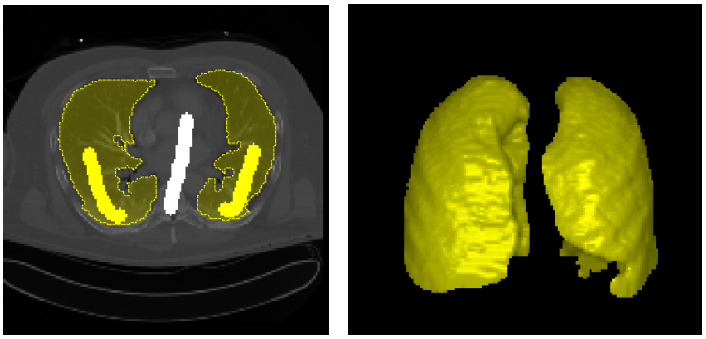
i in each set $\mathcal{S}_i$ and background seeds in $\mathcal{S}_0$. The method promotes an optimum seed competition wherein each seed in $\mathcal{S}$ conquers its most closely connected voxels in the image domain, according to the connectivity function. In the resulting optimum-path forest, an object i may be defined by multiple connected components, as represented by the optimum-path trees rooted at its internal seeds in $\mathcal{S}_i$ (Figure 2.8). The root labels are propagated during the IFT algorithm such that a resulting label map L_O defines a graph cut $\mathcal{C} = \{(s,t) \in \mathcal{A}, L_O(s) \neq L_O(t)\}$ with maximum value of $E_{\min}(\mathcal{C})$ (Equation 2.7). The details about this process are given next.

Paths

For a given 3D image graph $(D, \mathcal{A})$, a *simple path* π_t with terminus t is a sequence of distinct nodes $\langle t_1, t_2, \ldots, t_k \rangle$ with $(t_i, t_{i+1}) \in \mathcal{A}$, $1 \leq i \leq k-1$, and $t_k = t$. The concatenation of a path π_s and an arc (s,t) is denoted by $\pi_s \cdot \langle s,t \rangle$, being $\pi_t = \langle t \rangle$, i.e. $k = 1$, called a *trivial path*.

Connectivity function

A connectivity function assigns a value $f(\pi)$ to any path π in the image graph $(D, \mathcal{A})$. A path π_t is optimum if $f(\pi_t) \leq f(\tau_t)$ for every other path τ_t, disregarding the origin. The IFT algorithm is essentially Dijkstra's algorithm generalized for multiple sources (seeds)



(a) (b)

Figure 2.8: (a) Markers with the same label can indicate a single object with multiple connected components. (b) The resulting label map shows the object as composed by voxels in the optimum-path forest rooted at those markers.

and more general connectivity functions. However, it requires that function f be *smooth*. This means that for any voxel $t \in D$, there must exist an optimum path π_t, which is either trivial or has the form of $\tau_s \cdot \langle s, t \rangle$, where:

1. $f(\tau_s) \leq f(\pi_t)$
2. τ_s is optimal.
3. for any optimum path τ'_s, $f(\tau'_s \cdot \langle s, t \rangle) = f(\pi_t)$.

Optimum-path forest

For image segmentation from a λ-labeled seed set $\mathcal{S}$, we will consider arc weights $w(s,t) = ||\vec{G}(t)||$ — a gradient image value — such that the weights of arcs across the object boundaries are meant to be higher than the weights of internal and external arcs (Figure 2.6). Therefore, we may think of the image $\hat{G} = (D, G)$, where $G(s) = ||\vec{G}(s)||$, as the gradient image used for arc-weight assignment. The connectivity function is defined to penalize paths that cross the object boundaries.

$$\begin{aligned} f(\langle t \rangle) &= h(t) \\ f(\pi_s \cdot \langle s, t \rangle) &= \max\{f(\pi_s), G(t)\} \end{aligned} \tag{2.8}$$

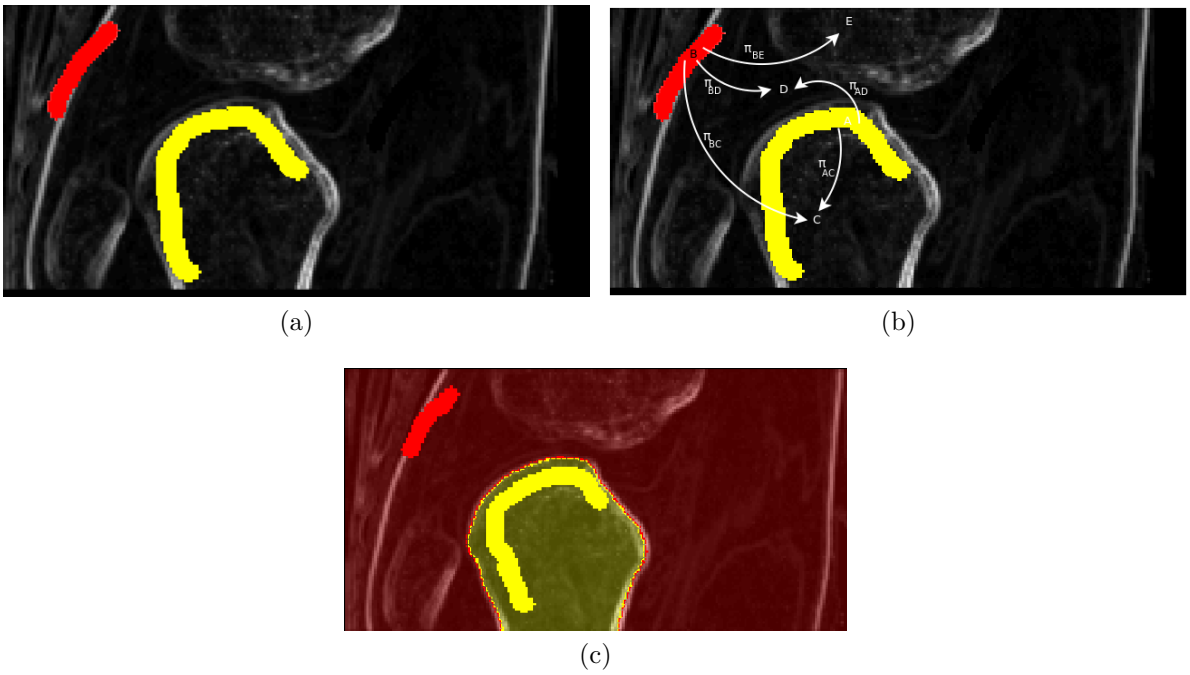
where $h(t)$ is a finite *handicap value*, usually defined as $h(t) = 0$ if $t \in \mathcal{S}$ and $h(t) = +\infty$ otherwise, to force optimum paths to start in $\mathcal{S}$.

The IFT algorithm minimizes a connectivity (cost) map C, by assigning to each node $t \in D$, the path π_t of minimum cost value.

$$C(t) = \min_{\forall \pi_t \in \Pi_t(D,\mathcal{A})} \{f(\pi_t)\} \tag{2.9}$$

where $\Pi_t(D, \mathcal{A})$ is the set of all possible paths with terminus t in the image graph. If the arc-weight assignment were ideal, it would be enough for segmentation to place a single seed per object and background. All optimum paths from the internal seed and other object voxels would have lower values than paths from other seeds, since paths with external source would be penalized by the higher arc weight across the boundary (Figure 2.9). Given that arc-weight assignment is never ideal, the segmentation requires more seeds per object.

The above solution is essentially a watershed transform from labeled markers [33]. The gradient image $\hat{G} = (D, G)$ may be interpreted as a surface where the object is formed by basins and its boundary is expected to be at the ridges between basins. Markers with distinct colors are holes from which colored water floods the surface. When waters from distinct colors meet at the ridges (boundary), a barrier is built to avoid their misture[48, 31]. However, the IFT algorithm also outputs an *optimum-path forest* — an acyclic predecessor map P that assigns to every voxel $s \in D \backslash \mathcal{S}$ its predecessor $P(s) \in D$ in the optimum path from $\mathcal{S}$, or $P(s) = nil \notin D$ when $s \in \mathcal{S}$. As we will see in the next chapter, since the predecessor map stores the connectivity between $\mathcal{S}$ and the remaining voxels, it allows to correct segmentation without starting over the process. This is a unique feature of the region-based image segmentation by IFT as compared to most watershed algorithms [67, 68]. The only exception is the watershed-cut algorithm [16], which is based on the relation between minimum spanning trees and optimum-path trees for the connectivity function f used in this section. Its results are equivalent to those of the IFT algorithm as long as the arc weight assignment does not create *tie zones* — regions where voxels can be conquered by multiple seeds with the same optimum path cost. It is not clear in [16], how they solve tie zones and removal of markers.



(a) (b)

(c)

Figure 2.9: (a) Arc weights are meant to be higher on object boundaries than inside and outside them. (b) Markers A (yellow) and B (red) compete by offering optimum paths to the remaining voxels. Due to the arc-weight assignment, $f(\pi_{AC}) < f(\pi_{BC})$, making C to be conquered by A. Similarly, B conquers D due to $f(\pi_{BD}) < f(\pi_{AD})$. (c) B conquers all voxels around the object, before A can reach E due to the ordered path-cost propagation in the IFT algorithm. This provides a consistent solution for tie zones. That is, even for $f(\pi_{BE}) = f(\pi_{AE})$, voxel E is conquered by B.

The IFT-watershed algorithm

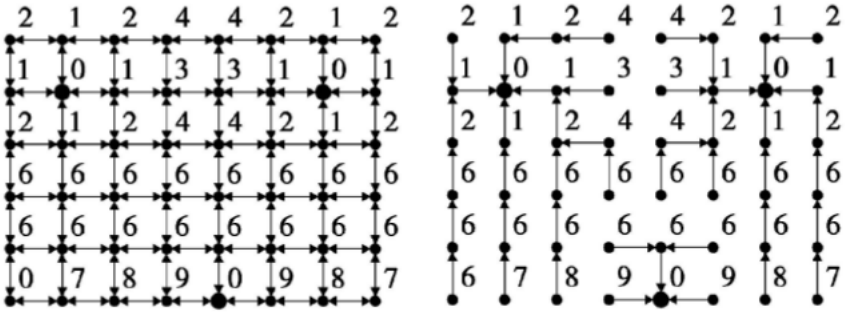
Algorithm 2 presents a Watershed Transform from Labeled Markers using the IFT algorithm. This formulation was first presented in [63].

Algorithm 2 – IFT-watershed

Input: Gradient image $\hat{G} = (D, G)$, adjacency relation $\mathcal{A}$, seed set $\mathcal{S} \subset D$ labeled by λ.
Output: Optimum path forest P, connectivity C, root R, and object label L_O maps.
Auxiliary: Priority queue Q and a variable tmp.

```
1.  For each t ∈ D, do
2.      P(t) ← nil, R(t) ← t, C(t) ← +∞
3.      If t ∈ S, then
4.          C(t) ← 0, L_O(t) ← λ(t)
5.          Insert t in Q
6.  While Q ≠ ∅, do
7.      Remove s from Q such that C(s) is minimum
8.      For each t ∈ A(s), such that C(t) > C(s), do
9.          tmp ← max{C(s), G(t)}.
10.         If tmp < C(t), then
11.             If C(t) ≠ +∞, then remove t from Q
12.             Set P(t) ← s, R(t) ← R(s), C(t) ← tmp, L_O(t) ← L_O(s)
13.             Insert t in Q
14. Return (P, C, R, L_O)
```

Lines 1–5 initialize maps and insert seeds in the priority queue Q. The roots of the forest will come from the minima of the handicap function h. In this case, $h(t) = 0$ (Line 4) if $t \in \mathcal{S}$, or $h(t) = +\infty$ (Line 2) otherwise. When this is not the case, it is possible that a seed be conquered by another seed which offers to it a path whose cost is less than its handicap value. The algorithm not only propagates optimum paths in P, their costs in C, and root labels in L_O, but it also propagates the root $R(t)$ that has conquered each voxel $t \in D$. The main loop computes an optimum path from $\mathcal{S}$ to every voxel s in a non-decreasing order of cost (Lines 6–13). At each iteration, a path of minimum cost $C(s)$ is obtained in P when we remove its last voxel s from Q (Line 7). Ties are broken in Q using first-in-first-out (FIFO) policy. That is, when two optimum paths reach an ambiguous voxel s with the same minimum cost, s is assigned to the first path that reached it. The rest of the lines evaluate if the path that reaches an adjacent voxel t through s is cheaper than the current path with terminus t and update Q and all maps accordingly. Actually, according to [33], Line 11 is not necessary for the case $w(s,t) = G(t)$, but we kept it in case the reader decides to implement the algorithm with other arc-weight function $w(s,t)$. Figure 2.10 exemplifies the execution of the IFT algorithm.



(a) (b)

Figure 2.10: (a) An image graph with 4-connected adjacency, where the integers are the gradient values $G(s)$. (b) An optimum-path forest for the connectivity function f (Equation 2.8), where $w(s,t) = G(t)$ and $h(t) = 0$, if t is one of the three seed pixels represented by bigger dots, and $h(t) = +\infty$ otherwise. [23]

Priority queue

Asymptotically, the bottleneck of Algorithm 2 lies in Line 7 — the selection of the minimum-cost voxel $s \in Q$. If Q is implemented as a balanced heap data structure, the total running time will be $O(|\mathcal{A}| + |D| \log |D|)$. Given that the connectivity costs $f(\pi_t)$ can be integers, Q can be implemented as a circular queue of $K + 1$ entries, each pointing to a circular doubly linked list of voxels. Let K be a fairly small upper bound to the incremental cost $f(\pi_s \cdot \langle s,t \rangle) - f(\pi_s)$ of extending an optimum path π_s by an arc $(s,t) \in \mathcal{A}$, and to the maximum difference $f(\langle t \rangle) - f(\langle t' \rangle)$ between the costs of trivial paths (excluding infinite values). The insertion, deletion, and cost update of a voxel can be done in $O(1)$ time. Finding the next minimum-cost voxel in Q may require to skip over several empty entries; however, since the cost of the next voxel never decreases, the total time of Line 7 is $O(K)$. Algorithm 2 will then run in $O(|\mathcal{A}| + |D| + K)$ time and $O(|D| + K)$ space. Figure 2.11 shows this data structure.

2.5 Complementary comments

The chapter has reduced our choices for seed-based interactive segmentation of medical images to two main approaches: one that minimizes E_{sum} (Equation 2.5) and the other that maximizes $E_{\min}$ (Equation 2.7 — i.e., it minimizes $E_{\max}$, Equation 2.6). The Dynamic Graph cut algorithm [6] is the best current surrogate for the minimization of E_{sum}, rather than the min-cut/max-flow algorithm, since it takes time proportional to the number of arcs in the modified regions between consecutive segmentation executions with different seed sets. Similarly, the Differential IFT algorithm [9] can take time proportional to the

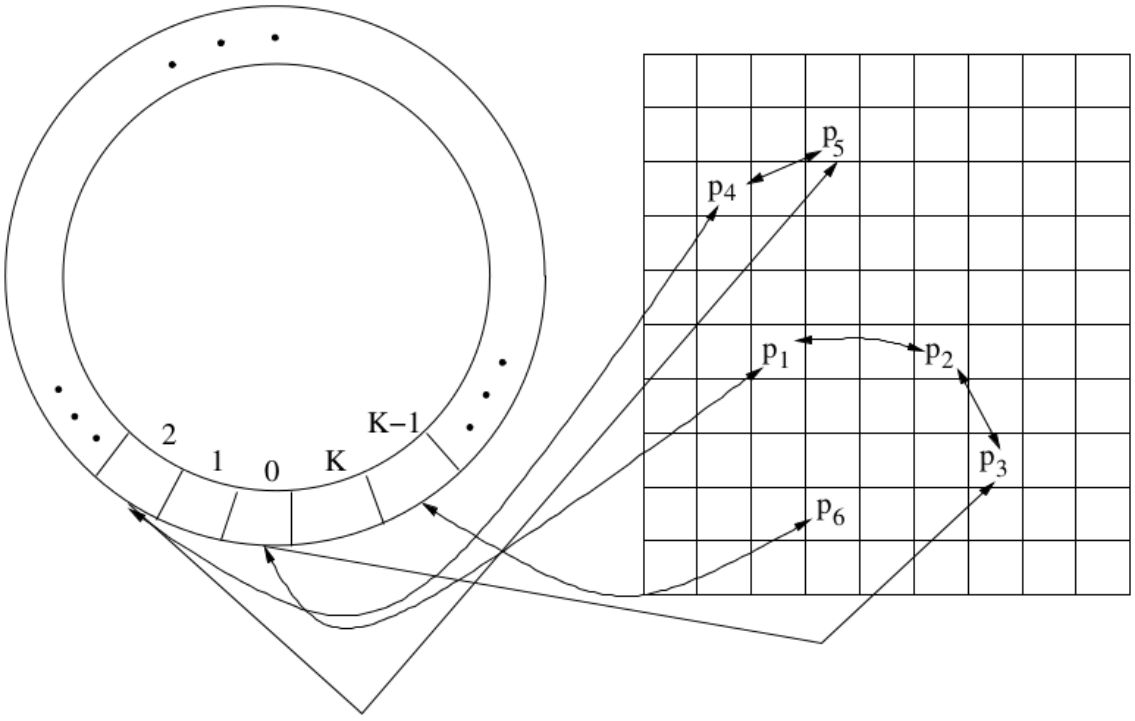


Figure 2.11: Nodes t are inserted in bucket $C(t)\%(K+1)$ (left), forming $K+1$ lists (right). The property $f(\pi_s \cdot \langle s,t \rangle) - f(\pi_s) \in [0,K]$ guarantees that nodes with different values are never in a same bucket.

number of voxels in the modified regions between consecutive segmentation executions, which makes it the best choice of $E_{\min}$ maximization. However, as presented in [9], the DIFT algorithm does not guarantee that all voxels are connected to the roots that labeled them, when distinct markers are addeed and removed at the same time. The same segmentation consistency problem may occur when the resulting label map L_O is filtered to smooth object boundaries. Therefore, the next chapter shows how to fix the problem for both situations.

Chapter 3

Differential and Relaxed Image Foresting Transform

In this chapter we revisit the Differential Image Foresting Transform (DIFT) algorithm [9] and a shape filtering procedure for object boundary relaxation [24] to fix inconsistencies that might be created by both methods in seed-based image segmentation. Additionally, for interactive segmentation, object relaxation must be much faster in responding to the user's actions in interactive time. We solve the problem such that the user can decide when and how many times the relaxation procedure will be applied during the segmentation process. The improvements made on both methods result into a fast interactive image segmentation algorithm, named *Differential and Relaxed Image Foresting Transform* (DRIFT), able to produce consistent label maps with smooth object boundaries.

3.1 Inconsistency in seed-based image segmentation

Let L_O be a label map resulting from a seed-based image segmentation procedure using adjacency relation $\mathcal{A}$ to propagate labels from the seed set. The label map L_O is said *inconsistent* when a voxel s was labeled by a seed $R(s)$ — i.e., $L_O(s) = L_O(R(s))$ — but there is no path using the same adjacency relation that connects s and $R(s)$ by a sequence of voxels with the same label $L_O(R(s))$. Note that this definition applies to any seed-based segmentation algorithm and, according to it, a random walker cannot guarantee a consistent label map.

It should be clear by definition that the IFT-watershed algorithm (Algorithm 2) generates consistent label maps L_O. In order to achieve interactive response time when the user corrects segmentation by adding new seeds, Algorithm 2 must be modified such that the optimum-path forest and its attributes can be updated without starting over the process. Therefore, we wish to execute a variant of the IFT-watershed algorithm multiple times

for different seed sets, using the forest and its path connectivity values of the previous execution as initial condition for seed competition in the next execution. However, a simple change in Lines 1-5 after the first execution, eliminating Line 2 and adding to Line 4 $R(t) \leftarrow t$ and $P(t) \leftarrow nil$, is not enough to guarantee consistent label maps.

Figure 3.1 illustrates a possible label inconsistency problem. Figure 3.1a presents a 4-adjacent image graph, where the numbers indicate the gradient values $G(s)$ of the nodes s (pixels) in Algorithm 2. Higher values (darker gray) are the ridges between basins, where the object boundaries are expected to be. For a single yellow seed, Figure 3.1b shows a possible optimum-path forest with one tree rooted at that seed. The numbers now indicate the connectivity values of the optimum paths from the selected seed. Suppose that a second seed (darker red) is added to the competition process. Due to the previous connectivity values, the modified Algorithm 2 will only propagate optimum paths with red labels to nodes reached by better connectivity values (smaller costs). Although the optimum-path forest is always correctly updated, the tie zones may cause an inconsistent label map. Figure 3.1c shows the problem for a tie zone of value 10 — it remains yellow when the optimum paths of its nodes have been updated to the red root. The situation with seed addition was foreseen in the original DIFT algorithm [9] and treated by adding a predecessor test, $P(t) = s$, in Line 10 of the modified Algorithm 2. The test forces to visit the nodes of the tie zones and update their labels as shown in Figure 3.1d.

Alternatively, one can think of updating the forest and connectivity values during the DIFT, and propagating labels and roots only afterwards. However, the main idea of interactive response time is to visit only nodes that require some modification. In this case, we wish to propagate labels and roots correctly on-the-fly. Note that, the DIFT algorithm is general and can also be used for applications with distinct connectivity functions. Some of them, as for example the Euclidean distance transform, require the root information to compute connectivity values.

In spite of the seed addition has been treated correctly in the DIFT algorithm [9] and in most situations the user corrects segmentation by adding new markers, we found in this work the inconsistency problem when markers are added and removed at the same time. Note that, when marker addition and removal are treated in separated DIFT executions, the problem does not appear. However, the simultaneous addition and removal of markers can create a situation where a tie-zone node is removed from the priority queue before its label is updated with respect to a new root. Figure 3.2 illustrates the problem. Figure 3.2a shows a 4-adjacency image graph, where numbers indicate the gradient values of the pixels, and Figure 3.2b shows a possible optimum-path forest, labels and connectivity values for Algorithm 2 from three seeds (darker red, yellow, and blue). In the DIFT algorithm [9], when the user removes a seed, the nodes of its tree become available to be reconquered by new seeds and also by non-removed seeds, as represented by their *frontier*

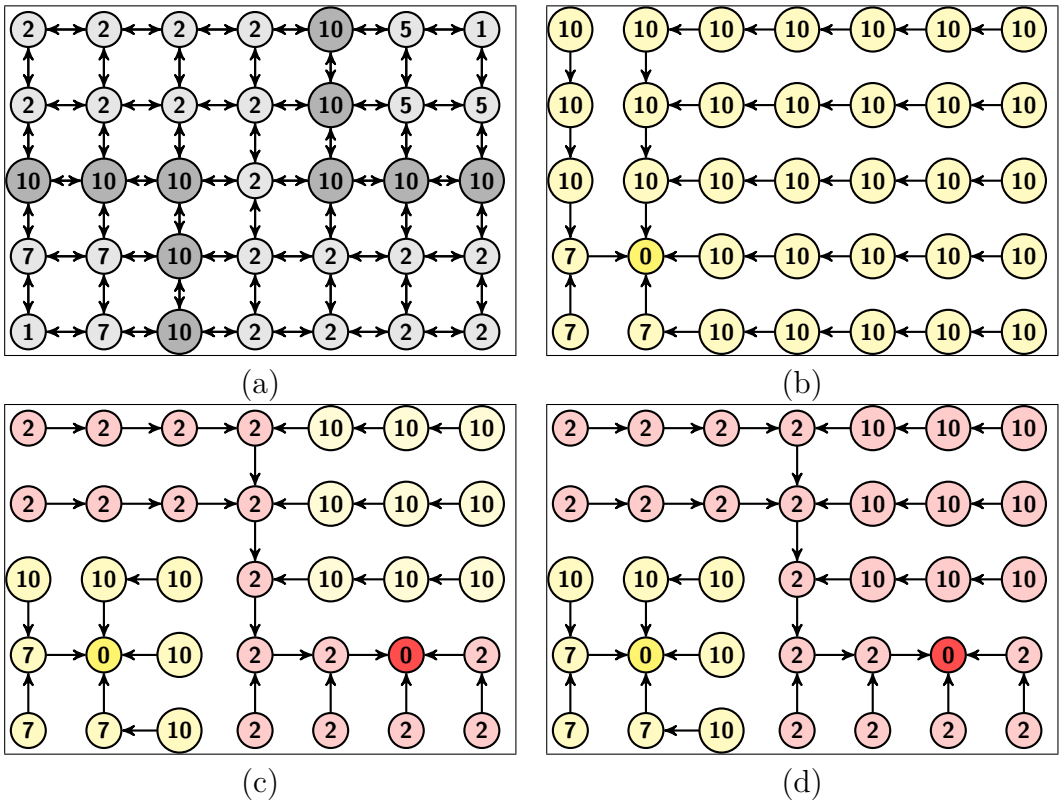


Figure 3.1: (a) A 4-adjacent image graph with numbers indicating gradient values for the IFT-watershed transform. (b) An optimum-path forest with a single tree rooted at a yellow seed, where the numbers indicate the connectivity values. (c) An optimum-path forest with two trees, being the second rooted at a darker red seed added to the segmentation in (b), and an inconsistent label map where the yellow tie zone with connectivity value 10 is not visited for label update with respect to the new root (darker red seed). (d) The correct result with consistent label map when using a predecessor test, $P(t) = s$, in Line 10 of Algorithm 2.

nodes — i.e., nodes of non-removed trees that are adjacent to those available nodes. The new seeds are inserted in the priority queue Q with handicap value $h(t) = 0$, as always, but the frontier nodes are inserted in Q with $C(t)$ equal to their final connectivity values from the previous execution. Figure 3.2c shows one example where the yellow tree is removed and new darker blue seeds are added. The frontier pixels are indicated by squares. This represents an instant before executing the DIFT algorithm for the second time. Figure 3.2d shows a possible optimum-path forest, connectivity values and labels resulting from the second execution. Note that some frontier pixels may be conquered by new seeds. The predecessor test can partially correct the label inconsistency problem (red

components with darker blue roots), but tie-zone nodes with value 2, which are frontier pixels, will leave Q before other nodes in the same tie zone due to the FIFO tie-breaking policy. Given that the predecessor test does not account for removed nodes, the subtrees of these frontier pixels will remain with their previous labels, creating an inconsistent label map.

This would not be a problem if we had implemented the DIFT algorithm with *Last-In-First-Out* (LIFO) tie-breaking policy in Q, as described in [23], but such a policy cannot equally share tie-zone nodes among disputing seeds and so, visually, the FIFO policy better matches the label map with the users' expectations. Most methods assume that tie zones do not represent relevant components in seed-based segmentation. This is not our experience, specially in the case of 3D medical images using local gradient estimators. Figure 3.2e shows the expected result and the complete solution that corrects label inconsistencies in the DIFT algorithm is presented next.

3.2 Differential Image Foresting Transform

User corrections in seed-based image segmentation are usually necessary, given that generally there is no way to know *a priori* where seeds must be placed with 100% effectiveness. The user very often needs to add markers (connected seed sets) and, to correct possible user mistakes, markers must be removed. The predecessor map of the Image Foresting Transform (IFT) [23] stores the connectivity information between seeds and voxels, which makes it possible to add and remove markers without starting over the process when its differential version [9] is used. Other seed-based approches have to recompute segmentation for every new seed set, with exeption of watershed cuts [16] which can store connectivity information in a minimum spanning forest. Considering a 3D image with 512^3 ($\approx$ 134 million) voxels and a linear-time algorithm running on a modern PC, each segmentation execution takes about 55 seconds making the interactive process unfeasible. In practice, the DIFT algorithm [9] can considerably reduce the computational time for marker addition and removal to a few seconds (e.g., less than 3s) per execution.

It is also worth mentioning that automatic solutions based on seed-based image segmentation may use fuzzy/statistical object models to derive the seeds [2, 45]. However, the best seed location requires some optimization algorithm based on an effective criterion function. When these methods fail, most interactive approaches can only start over the segmentation process, disregarding the fact that the segmentation errors might be localized in a few regions of the label map. Miranda et al. [69] presented a solution to transform any label map into an optimum-path forest with a minimum number of trees ("supposedly less user effort") for segmentation correction using the DIFT algorithm. In this case, tree removal is usually required since their choice was part of the automatic

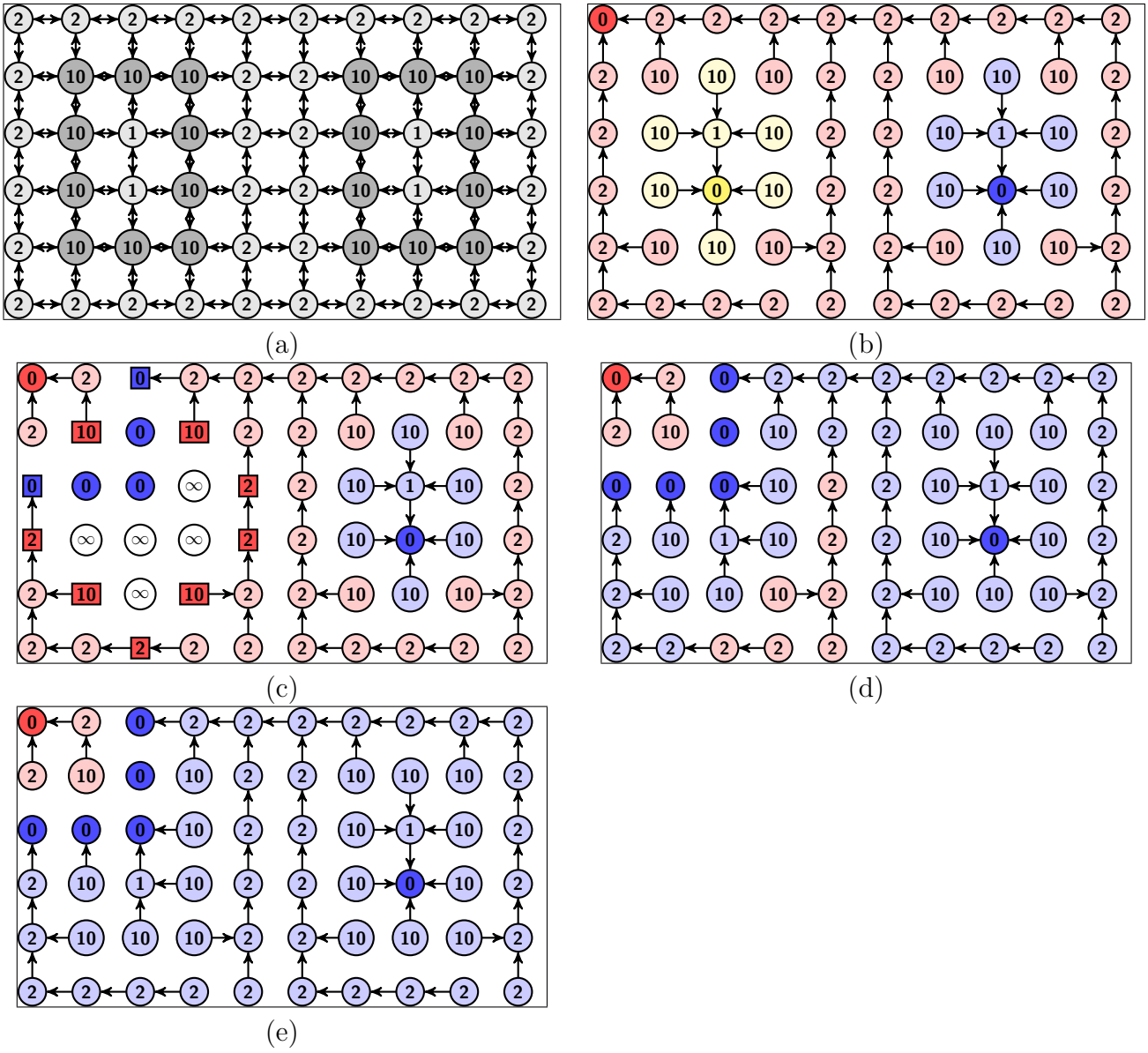


Figure 3.2: (a) A 4-adjacent image graph whose numbers indicate gradient values for IFT-watershed transform. (b) A possible optimum-path forest with connectivity values and labels from three seeds (darker yellow, red, and blue). (c) The user marks the yellow tree for removal and adds new darker blue seeds. Frontier pixels are indicated by squares. (d) Even with the predecessor test, the DIFT algorithm [9] cannot avoid label inconsistency (the two red components with darker blue roots), since frontier pixels in tie zones may leave Q before other tie-zone nodes, preventing their subtrees to have their labels fixed with respect to the new roots. (e) The desired result.

segmentation process.

In this section, we present a path forest data structure and the DIFT-watershed algorithm with the proposed correction to compute an optimum-path forest with its attributes in that data structure in a consistent way.

3.2.1 Path forest data structure

A path forest data structure can be defined as an image $\hat{F} = (D, \vec{F})$, as follows.

Path forest $\hat{F} = (D, \vec{F})$	where	$\vec{F}(s) = (F_1(s), F_2(s), \ldots, F_5(s))$	$\forall s \in D$
	D:	is the image domain in $\mathbb{Z}^3$,	
	F_1:	Predecessor map $\hat{F}.P : D \to D \cup \{nil\}$,	
	F_2:	Connectivity map $\hat{F}.C : D \to \mathbb{Z}$,	
	F_3:	Root map $\hat{F}.R : D \to D$,	
	F_4:	Object label map $\hat{F}.L_O : D \to \mathbb{Z}$, and	
	F_5:	Marker label map $\hat{F}.L_M : D \to \mathbb{Z}$.	
end			

The DIFT-watershed algorithm is a variant of Algorithm 2 that allows multiple executions of the image foresting transform in $\hat{F}$, using the attributes of the previous execution as initial values and modifying them by influence of the new seeds. Given that the connectivity values are integers, the priority queue Q can be implemented as described in Section 2.4.3, achieving time proportional to the number of voxels in D in the first execution and time proportional to the size of the modified regions in the subsequent executions.

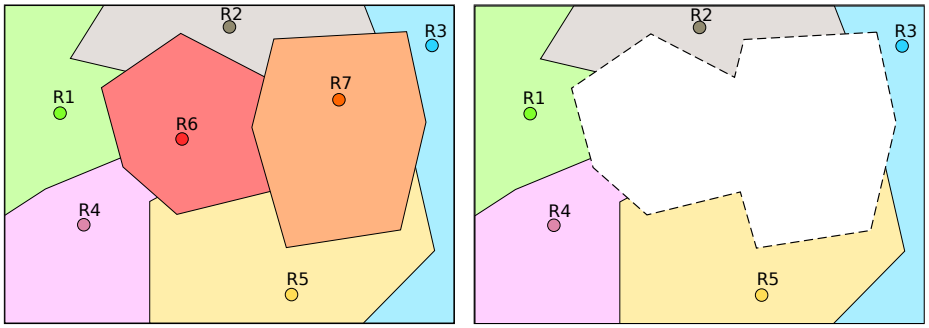
As input, the algorithm receives $\hat{F}$, a gradient image $\hat{G} = (D, G)$, an adjacency relation $\mathcal{A}$, an initial seed set $\mathcal{S}$ with voxels from all markers, labeled by the user via λ and id functions, and a set $\mathcal{M}$ of voxels for marker removal (empty in the first execution). The labeling function $\lambda(s) \in \{0, 1, \ldots, n\}$ assigns to seed voxels $s \in \mathcal{S}$ their object labels in $\{1, 2, \ldots, n\}$ or a background label 0. The labeling function id assigns to each voxel $s \in \mathcal{S}$ its corresponding marker label $id(s) \in \{1, 2, \ldots, m\}$.

The algorithm essentially propagates the predecessor $\hat{F}.P(t)$ in the optimum path with terminus t, its connectivity value $\hat{F}.C(t)$, the root node $\hat{F}.R(t) \in D$, object label $\hat{F}.L_O(t) \in \{0, 1, 2, \ldots, n\}$, and marker label $\hat{F}.L_M(t) \in \{1, 2, \ldots, m\}$ to each node $t \in D$ in a seed competition process, such that $\hat{F}.P$ is an optimum-path forest, where voxels t are conquered by their most closely connected seed $\hat{F}.R(t)$.

For marker removal, the user clicks on a single voxel of the influence zone in $\hat{F}.L_M$ conquered by a marker selected for removal. In the next execution, the input set $\mathcal{M}$ must contain all voxels from the markers selected for removal.

For the first execution, the attributes of $\hat{F}$ are initialized to every $t \in D \backslash \mathcal{S}$ as $\hat{F}.P(t) \leftarrow nil$ (trivial path) and $\hat{F}.C(t) \leftarrow +\infty$. In the case of seeds $t \in \mathcal{S}$, $\hat{F}.C(t) \leftarrow 0$, $\hat{F}.P(t) \leftarrow nil$, $\hat{F}.L_O(t) \leftarrow \lambda(t)$, $\hat{F}.L_M(t) \leftarrow id(t)$, and $\hat{F}.R(t) \leftarrow t$. The initialization of non-seed voxels must be done outside the algorithm, before $\hat{F}$ be given as input. In the algorithm, all seeds are initialized and inserted in Q for competition.

For the subsequent executions, each voxel $s \in \mathcal{M}$ is the root of a tree from the optimum-path forest of a marker that was selected for removal. The trees rooted in $\mathcal{M}$ are removed by setting $\hat{F}.P(t) \leftarrow nil$ and $.C(t) \leftarrow +\infty$ to their nodes $t \in D$. Such a tree removal procedure will also be useful to remove trees created by the object boundary relaxation process in Section3.3.1. The frontier voxels — nodes of non-removed trees that are adjacent to those of removed trees — return in a frontier set $\mathcal{S}_f$ to participate of the seed competition process as representatives of their root nodes (Figure 3.3). The voxels in $\mathcal{S}_f \backslash \mathcal{S}$ are inserted in Q with their current attribute values from the previous execution. The seed voxels $t \in \mathcal{S}$ have their attributes initialized as $\hat{F}.P(t) \leftarrow nil$, $\hat{F}.C(t) \leftarrow 0$, $\hat{F}.L_O(t) \leftarrow \lambda(t)$, $\hat{F}.L_M(t) \leftarrow id(t)$, and $\hat{F}.R(t) \leftarrow t$, in order to be inserted in Q for seed competition.



(a) (b)

Figure 3.3: (a) Initial segmentation with seven trees. (b) When the trees of roots $R6$ and $R7$ are marked for removal, their nodes become available to be reconquered (white). The dashed frontier $\mathcal{S}_f$ indicates the adjacent nodes from non-removed trees.

3.2.2 Optimum path propagation

The DIFT-watershed algorithm relies on two important procedures: (a) the main procedure for propagation of optimum paths and their attributes, where new markers compete with previous markers, and propagate their labels and the labels of their seeds together with the remaining attributes of the forest, and (b) the tree removal procedure, which resets the voxel regions that will be reconquered by the new seeds and roots from the

previous execution, as represented by the frontier voxels. This is consistent with the connectivity function f (Equation 2.8, with $h(t) = 0$ when $t \in \mathcal{S}$, and $h(t) = +\infty$ otherwise).

Optimum path propagation in the DIFT starts from the seeds in $\mathcal{S}$, as in Algorithm 2, but whenever a node s cannot offer a better path to an adjacent node t (no matter t is in the priority queue Q or has already been removed from it), the predecessor test, $\hat{F}.P(t) = s$, must be applied to avoid segmentation inconsistencies (Figure 3.1). This process updates the maps in $\hat{F}$ and, since we wish to relax boundaries between regions of processed voxels with distinct labels, a set $\mathcal{P}$ of voxels processed in the current execution also returns from the algorithm. The DIFT-watershed algorithm from labeled markers is presented below.

Algorithm 3 – DIFT-watershed

Input: Path forest $\hat{F} = (D, \vec{F})$, gradient image $\hat{G} = (D, G)$, adjacency relation $\mathcal{A}$, seed set $\mathcal{S} \subset D$ labeled by λ and id, and a voxel set $\mathcal{M}$ for tree (marker) removal.
Output: Optimum-path forest $\hat{F}$ and set $\mathcal{P}$ of processed voxels.
Auxiliary: Priority queue Q and variable tmp.

```
1.  Set P ← ∅
2.  If M ≠ ∅, then
3.      (F̂, S_f) ← DIFT-TreeRemoval(F̂, A, M)
4.      While S_f ≠ ∅, do
5.          Remove t from S_f and insert t in Q
6.  While S ≠ ∅, do
7.      Remove t from S
8.      If t ∈ Q, then Remove t from Q
9.      F̂.C(t) ← 0, F̂.L_O(t) ← λ(t), F̂.L_M(t) ← id(t), F̂.R(t) ← t, and F̂.P(t) ← nil
10.     Insert t in Q
11. While Q ≠ ∅, do
12.     Remove s from Q, such that F̂.C(s) is minimum and set P ← P ∪ {s}
13.     For each t ∈ A(s) do
14.         tmp ← max{F̂.C(s), G(t)}
15.         If tmp < F̂.C(t) or F̂.P(t) = s, then
16.             If t ∈ Q, then Remove t from Q
17.             Set F̂.P(t) ← s, F̂.C(t) ← tmp, F̂.R(t) ← F̂.R(s),
18.             F̂.L_M(t) ← F̂.L_M(s), and F̂.L_O(t) ← F̂.L_O(s)
19.             Insert t in Q
20. Return (F̂, P)
```

Line 1 initializes the set of processed nodes as empty. Subsequently, the DIFT-TreeRemoval procedure (Algorithm 4) is executed and frontier voxels are inserted in

Q with their attributes unchanged (Lines 2–5). Lines 6–10 initialize the new seeds and insert them in Q, such that they have precedence over frontier voxels (Line 8). The main loop (Lines 11–19) propagates optimum paths from seeds and frontier voxels (by extending their current paths) as described for Algorithm 2, except for two details. Set $\mathcal{P}$ is constructed in Line 12 for subsequent object relaxation (Section 3.3.1). The predecessor test, $\hat{F}.P(t) = s$, in Line 15 accounts for the cases where $tmp = \hat{F}.C(t)$ (tie zone) by forcing consistent label propagation along optimum paths, no matter t has already been removed or not from Q (Figure 3.2e). Algorithm 4 for tree (marker) removal is presented next.

Algorithm 4 – DIFT-TreeRemoval

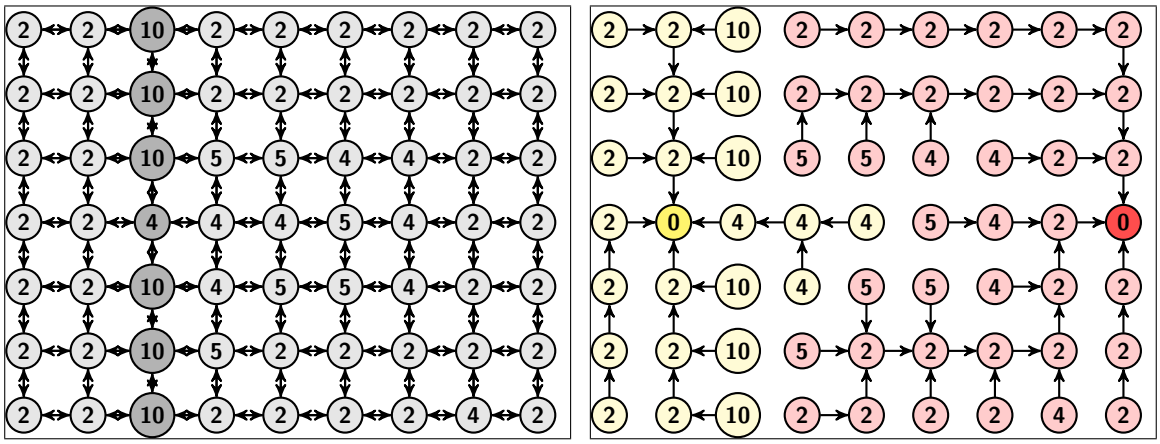
Input:	Path forest $\hat{F}$, adjacency $\mathcal{A}$, and voxel set $\mathcal{M}$ for tree removal.
Output:	$\hat{F}$ with removed trees and set $\mathcal{S}_f$ of frontier voxels.
Auxiliary:	Sets $\mathcal{T}_1$ and $\mathcal{T}_2$.

1. $\mathcal{S}_f \leftarrow \emptyset$, $\mathcal{T}_1 \leftarrow \emptyset$, *and* $\mathcal{T}_2 \leftarrow \emptyset$
2. **For each** $s \in \mathcal{M}$, **do**
3. **If** $\hat{F}.C(R(s)) \neq +\infty$, **then**
4. Set $r \leftarrow R(s)$, $\hat{F}.C(r) \leftarrow +\infty$, $\hat{F}.P(r) \leftarrow nil$, *and* $\mathcal{T}_1 \leftarrow \mathcal{T}_1 \cup \{r\}$
5. **While** $\mathcal{T}_1 \neq \emptyset$, **do**
6. Remove s *from* $\mathcal{T}_1$ *and set* $\mathcal{T}_2 \leftarrow \mathcal{T}_2 \cup \{s\}$
7. **For each** $t \in \mathcal{A}(s)$, **do**
8. **If** $\hat{F}.C(t) \neq +\infty$ *and* $\hat{F}.P(t) = s$, **then**
9. Set $\hat{F}.C(t) \leftarrow +\infty$, $\hat{F}.P(t) \leftarrow nil$, *and* $\mathcal{T}_1 \leftarrow \mathcal{T}_1 \cup \{t\}$
10. **While** $\mathcal{T}_2 \neq \emptyset$, **do**
11. Remove s *from* $\mathcal{T}_2$
12. **For each** $t \in \mathcal{A}(s)$, **do**
13. **If** $\hat{F}.C(t) \neq +\infty$, **then**
14. Set $\mathcal{S}_f \leftarrow \mathcal{S}_f \cup \{t\}$
15. *Return* $(\hat{F}, \mathcal{S}_f)$

Line 1 initializes the auxiliary and frontier sets to empty. Lines 2-9 visit all nodes from each tree rooted at $\hat{F}.R(s)$ for each voxel $s \in \mathcal{M}$ to reinitialize their path and connectivity value. Note that, even if $s \in \mathcal{M}$ is not a root voxel, the tree it belongs to is removed. The visited region, set $\mathcal{T}_2$ (Line 6), consists of the union of all removed trees. Finally, the frontier voxels are identified in Lines 10-14 and inserted in $\mathcal{S}_f$ (Line 14), as voxels $t \notin \mathcal{T}_2$ that are adjacent to voxels $s \in \mathcal{T}_2$.

3.3 Smoothing object boundaries

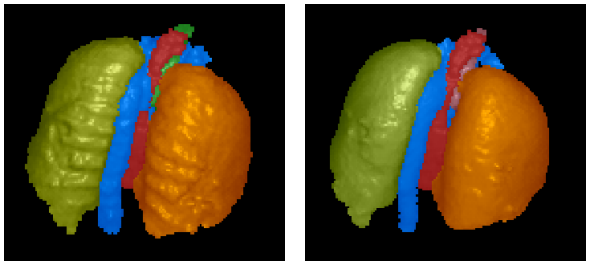
Segmentation methods that minimize E_{max} in Equation 2.6 (maximize E_{min} in Equation 2.7) are more susceptible to *leaking* [16, 33, 19] — errors when the label of the object (background) propagates to background (object) voxels, as an analogy to the water that fills up the object (background) basins in a gradient image and leaks to the background (object). The correction of such segmentation errors is more effective when markers are drawn around the lowest gradient parts of the object's boundary, where the leakings occur (Figure 3.4). However, small leakings are usually common in such approaches, creating more irregular ("jagged") segmentation boundaries than methods that minimize E_{sum} in Equation 2.5 [12, 49, 8].



(a) (b)

Figure 3.4: (a) A 4-adjacent image graph with numbers indicating gradient values. The desired boundary is represented by darker nodes. (b) The plateaus of optimum connectivity value equal to 4 with respect to the yellow seed provokes the leaking, since the red seed can only reach it by paths of value equal to 5.

Human body structures usually present regular ("smooth") boundaries and even E_{sum}-based segmentation methods cannot provide user control over the desired degree of object smoothness. This can be achieved by controlling the parameters of a diffusion filter [24]. However, any relaxation filter cannot guarantee segmentation consistency and this operation needs to respond in interactive time. We present a fast scheme for object boundary relaxation by diffusion filtering followed by label consistency correction (Figure 3.5).



(a) (b)

Figure 3.5: (a) DIFT-watershed segmentation of multiple body anatomical structures in the thorax. (b) Our approach can relax multiple object boundaries in a consistent way, within interactive response time, and under user control.

3.3.1 Object boundary relaxation

The irregular segmentation boundaries in the object label map $\hat{F}.L_O$ can be represented by a set $\mathcal{B}$ of voxels $s \in \mathcal{P}$ such that $\hat{F}.L_O(s) \neq \hat{F}.L_O(t)$ for some $t \in \mathcal{A}(s)$. The set $\mathcal{P}$ of processed voxels is equal to the image domain D after the first execution of the DIFT-watershed algorithm (Algorithm 3), but $\mathcal{P} \subseteq D$ is usually much smaller than D in the subsequent executions. The user can also decide when and how much the boundary set $\mathcal{B}$ will be relaxed, by calling the object boundary relaxation procedure described in this section with given parameters.

The boundary relaxation procedure is a variant of the diffusion filtering algorithm proposed by Filip et al. [24]. The original algorithm reprocesses all voxels in D at every iteration of diffusion, while our variant processes only voxels in a set $\mathcal{B}_d \subseteq D$ that dilates from $\mathcal{B}$ by the adjacency relation $\mathcal{A}$ (planar structure element) at each iteration. If the number T of iterations is small, as chosen by the user, the dilated boundary set $\mathcal{B}_d$ will still be much smaller than D. As mentioned earlier, the procedure can create inconsistencies in $\hat{F}.L_O$, but those segmentation inconsistencies can be efficiently corrected from voxels in $\mathcal{B}_d$ (Section 3.3.2).

The diffusion process starts from a previous map M_p that assigns membership value $M_p(s) \leftarrow 1$, $\forall s \in D$, with respect to the previous object label $L_{O_p}(s) = \hat{F}.L_O(s)$, as initially chosen by the DIFT-watershed algorithm, to decide a next object label $L_{O_n}(s) \leftarrow l_{\max} \in \{0, 1, 2, \ldots, n\}$ for s of maximum membership $\mu(l_{\max}) = \max_{l=0,1,\ldots,n}\{\mu(l)\}$. The membership values $\mu(l)$ are defined based on $M_p(t)$ and diffusion weights $W(t)$ of adjacent voxels $t \in \mathcal{A}(s)$.

$$\mu(l) \;=\; \frac{1}{N_f(s)} \sum_{\forall t \in \mathcal{A}(s) | L_{O_p}(t) = l} M_p(t) W(t) \tag{3.1}$$

$$N_f(s) = \sum_{\forall t \in \mathcal{A}(s)} W(t), \tag{3.2}$$

$$W(s) = \frac{1}{1+\beta G'(s)}, \tag{3.3}$$

where $N_f(s)$ is a normalization factor of the diffusion weights around s, $0 \leq \beta \leq 1$ is an user-defined relaxation factor, and $G'(s) = \frac{G(s)}{\max_{\forall s \in D}\{G(s)\}}$ is the normalized gradient value at s. The maximum membership value $\mu(l_{\max})$ computed for s must then be stored in a next membership map as $M_n(s)$. Given that each voxel $s \in D$ is also associated to a previous marker label $L_{M_p}(s) = \hat{F}.L_M(s)$, as initially chosen by the DIFT-watershed algorithm, the marker label $L_{M_p}(t)$ of the adjacent voxel $t \in \mathcal{A}(s)$ is stored in $mk(l)$, where $l = L_{O_p}(t)$. Therefore, $mk(l_{\max})$ must then be stored in a next marker label map as $L_{M_n}(s)$. This diffusion process, however, is only applied to voxels in a previous boundary set $\mathcal{B}_p$ (initially equal to $\mathcal{B}$), whose dilation by $\mathcal{A}$ creates voxels stored in a next boundary set $\mathcal{B}_n$. The process is then repeated by T iterations, by making $M_p(s) \leftarrow M_n(s)$, $L_{O_p}(s) \leftarrow L_{O_n}(s)$, $L_{M_p}(s) \leftarrow L_{M_n}(s)$ for all $s \in \mathcal{B}_p$, and then $\mathcal{B}_p \leftarrow \mathcal{B}_n$.

The parameters that do not change in the algorithm can be previously computed only once. They are stored in a quadruple $\Omega = (W, N_f, \beta, T)$ defined as follows: The diffusion

Relaxation Ω	is	
	$\Omega.W$:	Diffusion weights $\Omega.W : D \to \mathbb{R}$
	$\Omega.N_f$:	Normalization factors $\Omega.N_f : D \to \mathbb{R}$
	$\Omega.\beta$:	Smoothing factor within $[0, 1]$.
	$\Omega.T$:	Integer for the number of relaxation iterations.
end		

procedure is presented in Algorithm 5.

Algorithm 5 – ObjectRelaxation

Input: Consistent path forest $\hat{F} = (D, \vec{F})$, adjacency relation $\mathcal{A}$, set $\mathcal{P}$ of processed voxels by Algorithm3, relaxation quadruple Ω.

Output: Relaxed object label map L'_O, relaxed marker label map L'_M, and dilated boundary set $\mathcal{B}_d$.

Auxiliary: Previous and next object label maps L_{O_p} and L_{O_n}, previous and next marker label maps L_{M_p} and L_{M_n}, previous and next boundary sets $\mathcal{B}_p = \emptyset$ and $\mathcal{B}_n = \emptyset$, previous and next membership maps M_p and M_n (set to 1 for all $s \in D$), label membership $\mu(l)$ and marker membership $mk(l)$ with respect to each label $l = 0, 1, 2, \ldots, n$.

1. **For each** $s \in \mathcal{P}$, **do**
2. $\quad$ **For each** $t \in \mathcal{A}(s)$, **do**
3. $\quad\quad$ **If** $\hat{F}.L_O(s) \neq \hat{F}.L_O(t)$
4. $\quad\quad\quad$ $\mathcal{B}_p \leftarrow \mathcal{B}_p \cup \{s, t\}$
5. $L_{O_p} \leftarrow \hat{F}.L_O$, $L_{O_n} \leftarrow \hat{F}.L_O$, $L_{M_p} \leftarrow \hat{F}.L_M$, $L_{M_n} \leftarrow \hat{F}.L_M$
6. **For each** *relaxation iteration from* 1 *to* $\Omega.T$, **do**
7. $\quad$ $\mathcal{B}_n \leftarrow \emptyset$
8. $\quad$ **For each** s *in* $\mathcal{B}_p$, **do**
9. $\quad\quad$ $\mathcal{B}_n \leftarrow \mathcal{B}_n \cup \{s\}$
10. $\quad\quad$ **For each** $l \in \{0, 1, \ldots, n\}$, **do**
11. $\quad\quad\quad$ $mk(l) \leftarrow -1$ *and* $\mu(l) \leftarrow 0$
12. $\quad\quad$ **For each** $t \in \mathcal{A}(s)$, **do**
13. $\quad\quad\quad$ *Set* $l \leftarrow L_{O_p}(t)$, $\mu(l) \leftarrow \mu(l) + \Omega.W(t)M_p(t)$
14. $\quad\quad\quad$ $mk(l) \leftarrow L_{M_p}(t)$, *and* $\mathcal{B}_n \leftarrow \mathcal{B}_n \cup \{t\}$
15. $\quad\quad$ **For each** $l \in \{0, 1, \ldots, n\}$, **do**
16. $\quad\quad\quad$ $\mu(l) \leftarrow \mu(l)/\Omega.N_f(s)$
17. $\quad\quad$ *Compute* $l_{\max}$ *such that* $\mu(l_{\max}) = \max_{l=0,1,\ldots,n}\{\mu(l)\}$.
18. $\quad\quad$ $L_{O_n}(s) \leftarrow l_{\max}$, $M_n(s) \leftarrow \mu(l_{\max})$, *and* $L_{M_n}(s) \leftarrow mk(l_{\max})$
19. $\quad$ **While** $\mathcal{B}_p \neq \emptyset$, **do**
20. $\quad\quad$ *Remove* s *from* $\mathcal{B}_p$
21. $\quad\quad$ *Set* $L_{O_p}(s) \leftarrow L_{O_n}(s)$, $L_{M_p}(s) \leftarrow L_{M_n}(s)$, *and* $M_p(s) \leftarrow M_n(s)$.
22. $\quad$ $\mathcal{B}_p \leftarrow \mathcal{B}_n$
23. $L'_O \leftarrow L_{O_n}$, $L'_M \leftarrow L_{M_n}$, $\mathcal{B}_d \leftarrow \mathcal{B}_n$
24. *Return* $(L'_O, L'_M, \mathcal{B}_d)$

Lines 1–4 compute the initial boundary set $\mathcal{B}_p$ from $\mathcal{P}$, and Line 5 initializes the auxiliary maps. Lines 6–22 represent the diffusion filtering process on the label maps, according to the earlier description based on computed object memberships Equation 3.1. The number $\Omega.T$ of relaxation iterations is an important parameter. If it is too high, the object may deform too much, loosing shape indentations and protrusions, and so increasing the segmentation error. Set $\mathcal{B}_n$ holds the dilated boundary set (Lines 9 and 14) along the iterations. The output variables L'_O, L'_M, and $\mathcal{B}_d$ (Line 23) are used next to correct possible label inconsistencies created by the diffusion process.

3.3.2 Correcting label inconsistencies

Let π_t be the optimum path with terminus t in $\hat{F}.P$ for some $t \in \mathcal{B}_d$. A label inconsistency due to relaxation occurs when exists an inconsistent node r in the path π_t, such that

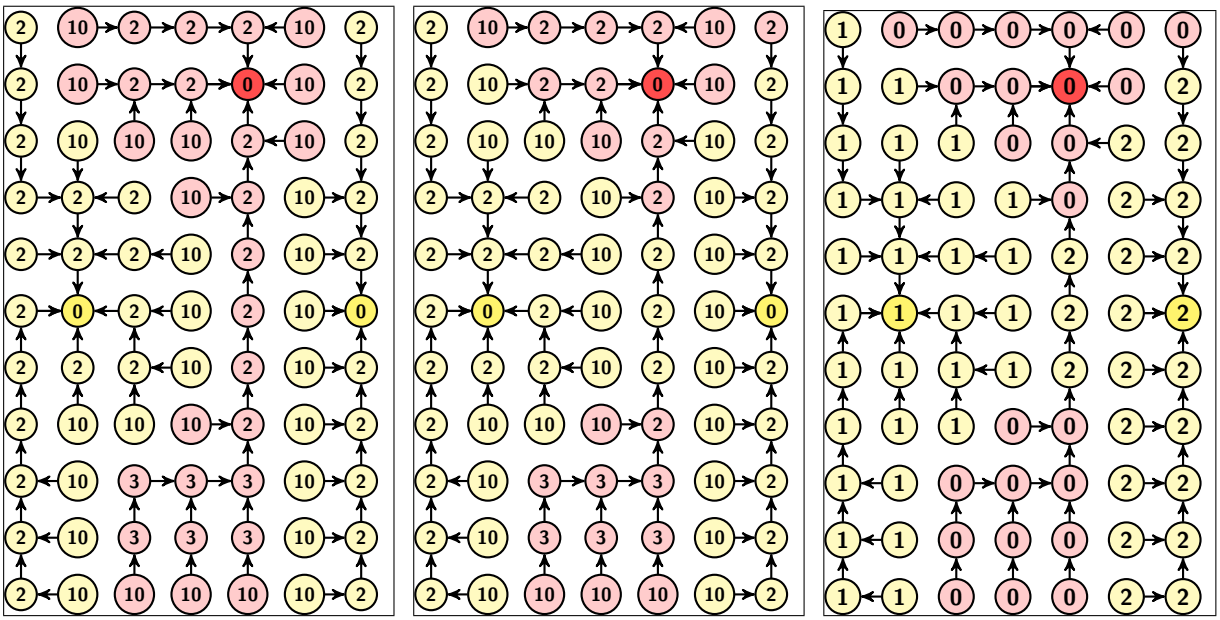
- either $r = \hat{F}.R(t)$ is the root of π_t and it changed label, $L'_O(r) \neq \hat{F}.L_O(r)$,

- or the new label of r is different from the new label of its predecessor in π_t, $L'_O(r) \neq L'_O(\hat{F}.P(r))$.

In any case, the sub-tree of r is inconsistent, irrespective to the new labels of its nodes. This inconsistency can be corrected by

- finding the inconsistent node r in π_t, which is the closest to $\hat{F}.R(t)$;
- making it a new root, $\hat{F}.P(r) \leftarrow nil$ and $\hat{F}.R(r) \leftarrow r$, called *relaxation root*; and
- propagating the new attributes of r to each node s of its subtree, including the new label maps, as $\hat{F}.L_O(s) \leftarrow L'_O(s) \leftarrow L'_O(r)$, $\hat{F}.L_M(s) \leftarrow L'_M(s) \leftarrow L'_M(r)$, and $\hat{F}.R(s) \leftarrow r$.

Note also that only nodes $t \in \mathcal{B}_d$ such that $L'_O(t) \neq \hat{F}.L_O(t)$ need to be verified and since we also update previous and current label maps for each first detected inconsistent tree, we can avoid verification for several nodes t that initially had to be verified. One example of a consistent segmentation turning into an inconsistent segmentation due to relaxation is presented in Figure 3.6 and the label correction procedure is presented in Algorithm 6.



(a) (b) (c)

Figure 3.6: (a) Segmentation with two seeds: background (yellow) and object (red). The numbers are the connectivity costs from the roots to each node. (b) and (c) Inconsistent object, cost and maker segmentation. Note that the thin path of nodes connecting the red components was labeled as background, causing an inconsistency.

Algorithm 6 – LabelCorrection

Input: Path forest $\hat{F}$, adjacency relation $\mathcal{A}$, relaxed object label map L'_O, relaxed marker label map L'_M, and dilated border $\mathcal{B}_d$ computed during relaxation.

Output: Relaxed and consistent forest $\hat{F}$.

Auxiliary: Set $\mathcal{T}$ of new roots and $color \in \{white, gray, black\}$.

```
1. While B_d ≠ ∅
2.     Remove t from B_d
3.     If L'_O(t) ≠ F̂.L_O(t) then
4.         color ← white
5.         (r, color) ← SearchRelaxationRoot(F̂, L'_O, t, color)
6.         If color = black then
7.             PropagateRootAttributes(F̂, A, L'_O, L'_M, r)
8.             F̂.P(r) ← nil
```

Each node $t \in \mathcal{B}_d$ is checked if its label changed during relaxation. In such cases, function SearchRelaxationRoot (Line 5) searches if there is any inconsistency in the path with terminus t and returns the inconsistency node r with $color = black$ or $color \neq black$ when the path is consistent. Line 8 creates a relaxation tree by making r a root of the forest with its subtree the new tree. Figure 3.7a shows the expected consistent output, with relaxation roots denoted by the rectangles. The numbers are the connectivity cost of each path. In Figure 3.7b, the numbers are the markers identification after the relaxation that were propagated from existing markers to the corresponding relaxation roots. If the relaxation roots were assigned with a new id, these roots would have to be removed manually one by one compromising the user control over the process.

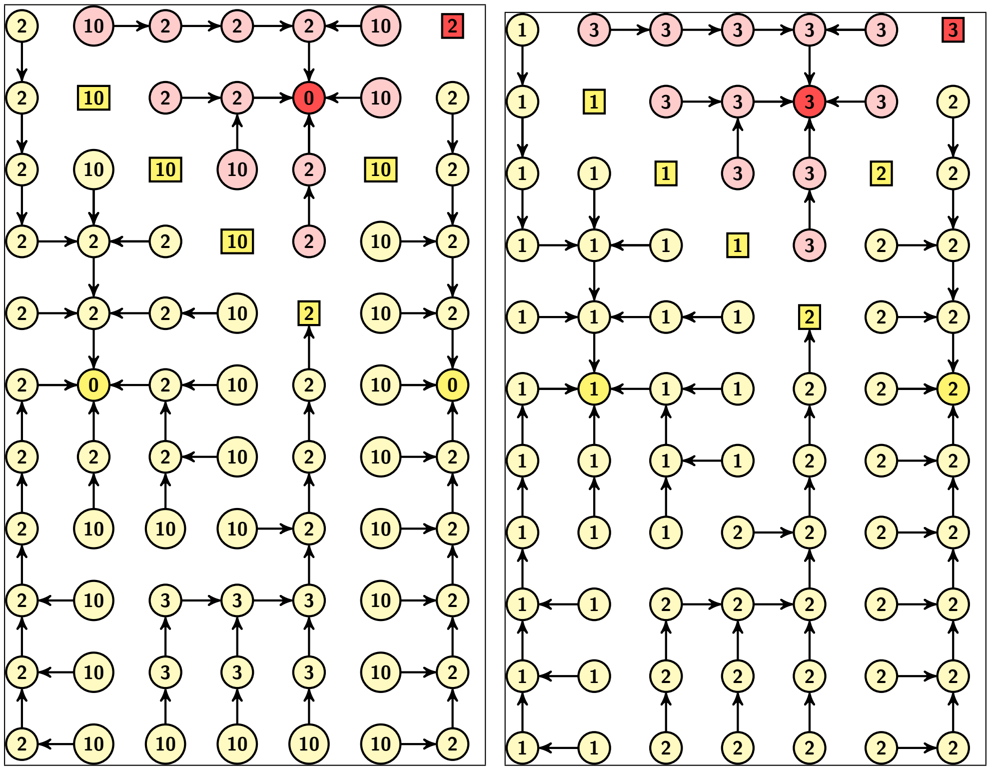
The SearchRelaxationRoot and PropagateRootAttributes procedures are presented in Algorithms 7 and 8.

Algorithm 7 – SearchRelaxationRoot

Input: Path forest $\hat{F}$, relaxed object label map L'_O, voxel t, and $color$ variable

Output: A tuple $(r, color)$, with $color = black$ or $color \neq black$.

```
1. If F̂.P(t) = nil, then
2.     color ← gray
3.     If L'_O(t) ≠ F̂.L_O(t), then
```



(a) (b)

Figure 3.7: (a) Obtained cost and label map after execution of the LabelCorrection procedure. The relaxation roots are denoted by the rectangles. (b) Marker map after the correction. The marker *ids* are propagated from existing markers to the corresponding relaxation roots.

└ └ $r \leftarrow t$, $color \leftarrow black$

Else

│ $(t, color) \leftarrow SearchRelaxationRoot(\hat{F}, L'_O, \hat{F}.P(t), color)$

│ ***If*** $color = gray$, ***then***

│ │ ***If*** $L'_O(t) \neq L'_O(\hat{F}.P(t))$ ***then***

└ └ └ $r \leftarrow t$, $color \leftarrow black$

Return $(r, color)$

Algorithm 8 – PropagateRootAttributes

Input: Path forest $\hat{F}$, adjacency relation $\mathcal{A}$, relaxed object label map L'_O, relaxed marker label map, L'_M, and root r

Output: Path forest $\hat{F}$, maps L'_O, and L'_M are updated.

Auxiliary: Subtree set $\mathcal{T}$

```
1. T ← {r}
2. While T ≠ ∅
3.     Remove s from T
4.     F̂.L_O(s) ← L'_O(r), L'_O(s) ← L'_O(r), F̂.L_M(s) ← L'_M(r), L'_M(s) ← L'_M(r), F̂.R(s) ← r
5.     For each t ∈ A(s), do
6.         If P(t) = s then
7.             T ← T ∪ {t}
```

Algorithm 7 is a recursive method that traverses from a node t searching for a node r that satisfies any of the conditions described in the beginning of this section. That is, $L'_O(r) \neq \hat{F}.L_O(r)$ or $L'_O(r) \neq L'_O(\hat{F}.P(r))$. This procedure also ensures that the relaxation root r is the closest one to $\hat{F}.R(t)$. The variable *color* is used to check if any of the conditions were satisfied and in case $color \neq black$, the path with terminus in t is said consistent.

In case $color = black$, Algorithm 8 updates the attributes L_O, L'_O, L_M, L'_M, and R (Line 4) for all nodes in the subtree of r (Lines 2–7).

3.4 Segmentation using relaxation

The segmentation process using relaxation can be divided in two paradigms: after each execution of DIFT-watershed the relaxation is applied in the propagated region; or the relaxation is applied only once, at the end of the process. We will present both versions based in the canonical Algorithm 1 mentioned in Chapter 1.

Both Algorithms receives as input a gradient image, an appropriate adjacency relation, the relaxation factor β, and the number of relaxation iterations T. More user input is required during the execution, in order to set new seeds and mark voxels for removal. Differential and Relaxed Image Foresting Transform (DRIFT) (Algorithm 9) applies the relaxation several times and DIFT Terminus Relaxation (DIFT-TR) (Algorithm 10) only once at the end. In order to use the same ObjectRelaxation (Algorithm 5) procedure for both Algorithms we set the list of processed nodes $\mathcal{P}$ with the image domain D so that the relaxation is applied in the entire image.

Algorithm 9 – DRIFT

Input: An image $\hat{G} = (D, G)$, adjacency relation $\mathcal{A}$, relaxation factor β, and number of relaxation iterations T.

Output: Object label map $L_O{:}\, D \rightarrow \{0, 1, \ldots, n\}$ and marker label map $L_M{:}\, D \rightarrow \{1, 2, \ldots, m\}$.

1. *Create a path forest structure* $\hat{F}$ *using* D.
2. *Create relaxation structure* Ω *from* $\hat{G}$, β, *and* T.
3. ***do***
4. | *Get from the user, labeled seeds* $\mathcal{S}$ *with* λ *and id, marker voxels in* $\mathcal{M}$
5. | $(\hat{F}, \mathcal{P}) \leftarrow$*DIFT-watershed*$(\hat{F}, \hat{G}, \mathcal{A}, \mathcal{S}, \mathcal{M})$.
6. | $(L'_O, L'_M, \mathcal{B}_d) \leftarrow$ *ObjectRelaxation*$(\hat{F}, \mathcal{A}, \mathcal{P}, \Omega)$.
7. | $\hat{F} \leftarrow$ *LabelCorrection*$(\hat{F}, \mathcal{A}, L'_O, L'_M, \mathcal{B}_d)$.
8. | *Present the label map* $\hat{F}.L_O$ *to the user.*
9. ***While*** *User is not satisfied with* $\hat{F}.L_O$.
10. *Return* $(\hat{F}.L_O, \hat{F}.L_M)$

Algorithm 10 – DIFT-TR

Input: An image $\hat{G} = (D, G)$, adjacency relation $\mathcal{A}$, relaxation factor β, and number of relaxation iterations T.

Output: Object label map $L_O{:}\, D \rightarrow \{0, 1, \ldots, n\}$ and marker label map $L_M{:}\, D \rightarrow \{1, 2, \ldots, m\}$.

1. *Create a path forest structure* $\hat{F}$ *using* D.
2. *Create relaxation structure* Ω *from* $\hat{G}$, β, *and* T.
3. ***do***
4. | *Get from the user, labeled seeds* $\mathcal{S}$ *with* λ *and id, marker voxels in* $\mathcal{M}$
5. | $(\hat{F}, \mathcal{P}) \leftarrow$*DIFT-watershed*$(\hat{F}, \hat{G}, \mathcal{A}, \mathcal{S}, \mathcal{M})$.
6. | *Present the label map* $\hat{F}.L_O$ *to the user.*
7. ***While*** *User is not satisfied with* $\hat{F}.L_O$.
8. $\mathcal{P} \leftarrow D$.
9. $(L'_O, L'_M, \mathcal{B}_d) \leftarrow$ *ObjectRelaxation*$(\hat{F}, \mathcal{A}, \mathcal{P}, \Omega)$.
10. $\hat{F} \leftarrow$ *LabelCorrection*$(\hat{F}, \mathcal{A}, L'_O, L'_M, \mathcal{B}_d)$.
11. *Return* $(\hat{F}.L_O, \hat{F}.L_M)$

The operations in Algorithm 6 are sufficient to circumvent possible inconsistencies during the relaxation while maintaining forest topology (predecessor map) and connectivity costs, but the sub-trees affected by diffusion now have a new root.

The drawback of this solution is the relaxation trees are usually small (trivial in many cases). To facilitate user corrections, a few implementation rules had to be defined:

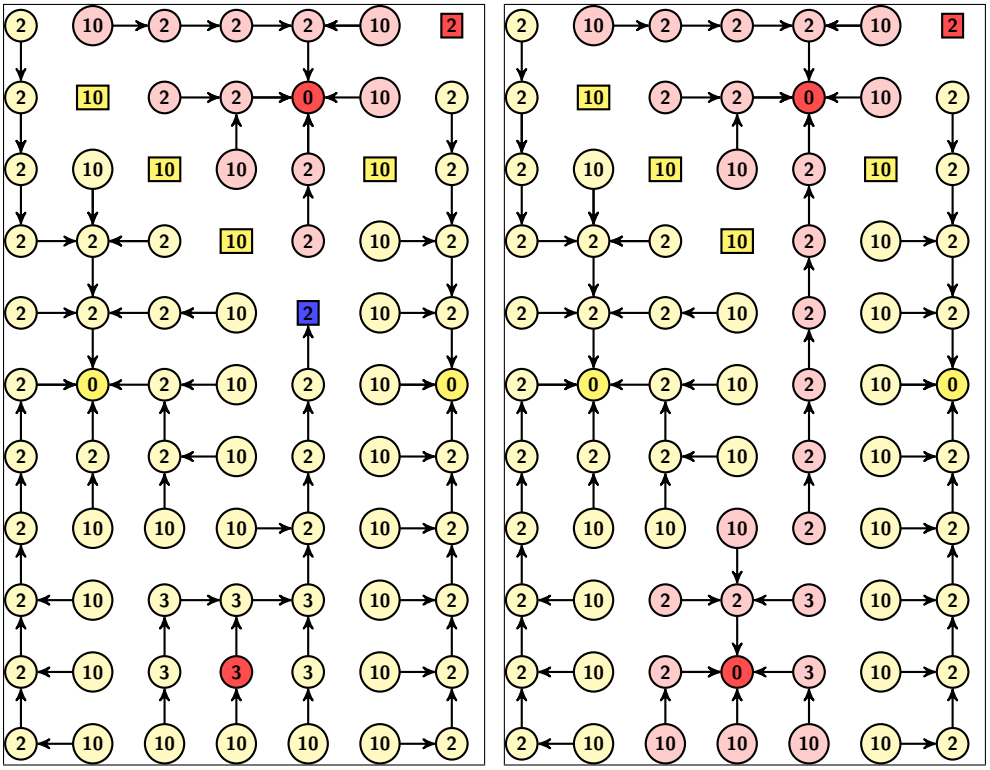
1. Relaxation trees are associated with user-drawn markers from which they inherit the new labels and marker *ids* during diffusion.

2. User-drawn markers have higher priority than relaxation roots.

The first rule is important to avoid manual removal of relaxation trees, as explained in Section 3.3.2. The second rule is exemplified in Figure 3.8. Given the expected result after the LabelCorrection procedure (Figure 3.7a), suppose that the user wants to override regions represented by relaxation trees. New seeds in those regions would have to compete with the relaxation roots. This can be avoided by automatically inserting the respective relaxation roots into the deletion set $\mathcal{M}$. Figure 3.8a shows an example where a seed is inserted in a relaxation tree rooted at a blue node. Since the blue node will be inserted in $\mathcal{M}$ there will be no competition between it and the new seed. The corresponding result of the DIFT-watershed is shown in Figure 3.8b.

We noticed that by applying the relaxation procedure at every iteration, several relaxation roots are created (Figure 3.9). Rule 2 then allows the user to easily correct this effect by drawing new markers on influence zones of relaxation roots, which will cause the automatic insertion of those roots in the deletion set $\mathcal{M}$. However, this is not easy when the relaxation trees are trivial. In this case, it is better to select their corresponding markers for removal, turn off the relaxation procedure, and add new markers to correct segmentation. The relaxation procedure can be turned on again in subsequent executions. Since set $\mathcal{P}$ of processed nodes will be different, future corrections will not make the relaxation roots to reappear.

We have implemented the DRIFT Algorithm in a software tool, called Volumetric Image Segmentation for Visualization and Analysis (VISVA), and performed a preliminary evaluation of it using a non-expert user and an MR-Brain image. Figure 3.10 shows the ground truth, the resulting segmentation of the DRIFT algorithm, and the relaxed segmentation of the DIFT-watershed Algorithm (i.e., the relaxation is only applied at the end). The ground truth (Figure 3.10a) shows the DIFT-watershed segmentation of the cerebellum, left and right hemispheres as obtained by a neurologist. Figures 3.10b and 3.10c show the segmentation results as obtained by a non-expert user. Given that the hemispheres are connected by the corpus callosum, and the cerebellum is connected to them by the spinal cord and medulla, which must not be considered in the segmentation, the markers must be selected around those connected components. Moreover, due to partial volume, these objects must also be disconnected in other parts of the image. The non-expert user has found difficult to select markers for segmentation, which indicates that intelligent techniques to suggest potential marker locations are important to be investigated in a future work. Experts trained in using the software tool may show a different experience and this has to be evaluated in a future work. The user could finish



(a) (b)

Figure 3.8: (a) This image represents the instant before executing the DIFT-watershed algorithm after a user correction. When the user adds a new marker (red), since this marker is placed under a relaxation tree, the root of this tree is inserted into set $\mathcal{M}$ in order to avoid competition with the new user marker. (b) The result after the execution of DIFT-watershed with the relaxation root $r \in \mathcal{M}$ and the red user marker in $\mathcal{S}$.

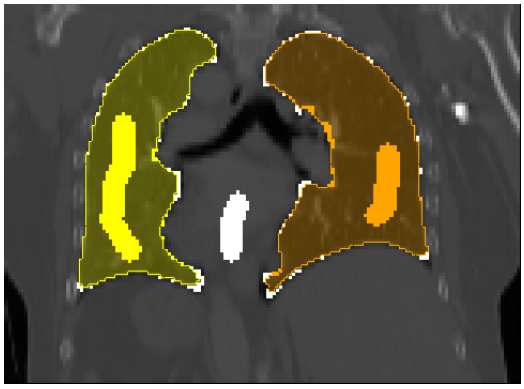
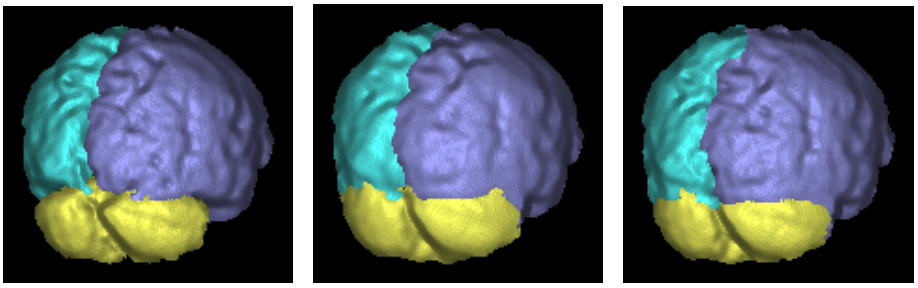


Figure 3.9: By applying the relaxation procedure at every iteration, several relaxation roots are created.

segmentation in about 10 executions of the algorithms, taking about 4 minutes mostly to choose marker locations. On the other hand, the results have shown to be very accurate according to Dice metric [70]: the mean accuracies of DRIFT and DIFT-TR (DIFT with terminal relaxation) were 0.92 and 0.93, respectively. For the DIFT algorithm 5 relaxation iterations were applied at the end of the process, increasing Dice's value to 0.93. The DIFT algorithm required more corrections, markers, user's time but achieved a better accuracy at the end. Table 3.1 presents all the metrics that were evaluated. A correction is a user intervention to place more markers followed by a another execution of the algorithm.

Algorithm	Markers	Corrections	Total time	Dice
DRIFT	17	6	$3m21s$	0.92
DIFT-TR	27	13	$6m50s$	0.93

Table 3.1: Evaluation of DIFT-TR (DIFT with terminal relaxation) and DRIFT algorithms by a non-expert user.



(a) (b) (c)

Figure 3.10: (a) Ground truth of a MR-brain: left brain hemisphere, right brain hemisphere, and cerebellum. (b) Interactive segmentation of the brain image using the DIFT algorithm with relaxation at the end.(c) Interactive segmentation of the brain image using the DRIFT algorithm.

In Chapter 4 we perform more experiments with DIFT, DIFT with terminal relaxation, DRIFT, and DGC on three distinct medical image datasets. Table 3.1 raises the question if the best approach is to apply the relaxation only at the end of the process or during segmentation.

Chapter 4

Experiments

In this chapter we present a experimental setup to compare four graph-cut segmentation methods using three datasets. In each dataset, there are multiple objects of interest for segmentation. The main challenge comes from the absence of contrast in several parts, which makes difficult to disconnect objects and eliminate background.

In order to avoid bias from distinct experts, the experiments adopted a geodesic robot user [25] — an algorithm that selects markers on error components for each new execution of the segmentation method until a convergence criterion to the available ground-truth image is reached. Finding several experts to perform segmentation experiments is also very difficult. The bias come from possible disagreements among them with respect to the delineation of a same object. Robot users avoid such a problem and make it possible to repeat the experiments many times. Indeed, robot users have been proposed and exploited in several works of the segmentation literature[71, 72, 25, 11]. Robot users are also not free of bias since they can favor specific methods[11]. The geodesic robot user places the markers at the center of the error components and this certainly does not favor any of the methods under comparison. On the other hand, once we choose the best approach, it is also important to perform experiments with real users (experts) in order to further improve and validate the practical use of the method.

4.1 Datasets

The datasets cover three distinct regions of the human body: thorax, brain, and foot[1]. They consist of CT and MR images and the corresponding multiple-label images of the interest objects, as interactively segmented by experts.

[1]We would like to thank Prof. Udupa from the University of Pennsylvania for the thorax and foot datasets, and Prof. Cendes from the University of Campinas for the brain dataset.

4.1.1 Thorax dataset

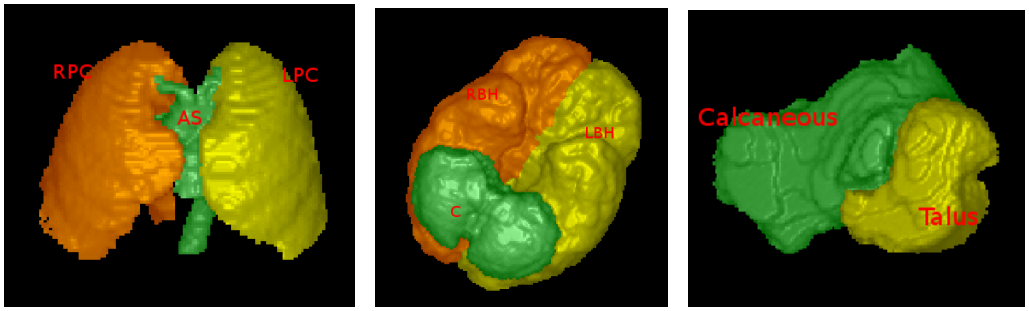
The thorax dataset contains 36 CT images with three body anatomical structures (objects) per image (Figure 4.1a): the arterial system (AS), right pleural cavity (RPC), and left pleural cavity (LPC). These images were acquired with voxel size $0.5 \times 0.5 \times 5\ mm^3$ and were interpolated to form an isotropic image with voxel size $2.5 \times 2.5 \times 2.5\ mm^3$ ($\approx$ 2.9 M voxels).

4.1.2 Brain dataset

The brain dataset contains 40 images with three objects of interest each (Figure 4.1b): the cerebellum (C), left brain hemisphere (LBH), and the right brain hemisphere (RBH). These images were acquired with voxel size $0.98 \times 0.98 \times 0.98 mm^3$ ($\approx$ 6.9 M voxels).

4.1.3 Foot dataset

The foot dataset has 20 MR images with two body structures per image (Figure 4.1c): the calcaneous and talus bones. The images were acquired with voxel size $0.54 \times 0.54 \times 0.54$ mm^3 ($\approx$ 10.8 M voxels).



(a) (b) (c)

Figure 4.1: (a-c) Images presenting the difficulty to segment the objects in the thorax, brain, and foot dataset respectively.

4.2 Methods

The experiments evaluate the accuracy and efficiency of the following methods: DRIFT (Algorithm 9), DIFT (Algorithm 9 with no relaxation), DIFT-TR (Algorithm 10 — DIFT with terminal relaxation), and MODGC (a modified version of Dynamic Graph Cut for multiple objects). As described in Chapter 1, MODGC requires the application of the

Dynamic Graph Cut algorithm[6] for each object separately, followed by merging of their binary label maps into a multi-object label map. The MODGC algorithm is presented below.

Algorithm 11 – MODGC

Input: An image $\hat{G} = (D, G)$ and adjacency relation $\mathcal{A}$.
Output: Object label map $L_O: D \to \{0, 1, \ldots, n\}$.

1. ***For each*** *object i from 1 to n,* ***do***
2. ***do***
3. *Get from the user, labeled seeds $\mathcal{S}$ with λ to segment i*
4. $L_{Oi} \leftarrow$*DynamicGraphCut*$(\hat{G}, \mathcal{A}, \mathcal{S})$ *as in [6].*
5. *Present the label map L_{Oi} to the user.*
6. ***While*** *User is not satisfied with L_{Oi}.*
7. ***For each*** $p \in D$, ***do***
8. $L_O(p) \leftarrow \max_{i=1,2,\ldots,n} L_{O_i}(p)$
9. *Return L_O*

All methods executed in the same graph, whose nodes are the voxels of the input 3D image.

4.2.1 Geodesic robot

A robot can be used to simulate the behavior of a real user during segmentation. More specifically, it was observed that real users tend to place markers at big error components and usually in the center of these components. More specifically, a geodesic robot [73] creates after each segmentation execution a sorted list L that defines the order in which markers should be placed. In order to achieve that, the robot computes an image $\hat{E} = (D, E)$, with $E(t) = -1$ if voxel t was correctly labeled or $E(t) = \lambda(t)$ if the correct label for t should be $\lambda(t)$. By using $\hat{E}$, the robot finds every connected component with the same label through a breadth-first search, ignoring the label -1. These components are called error components. Using the Euclidean Distance Transform [63], the list L is populated with seed voxels at the geometric centers of the error components in the decreasing order of distance to the border of the component. This way, the larger is the component, higher is the priority of placing a marker in its geodesic center. Figure 4.2 illustrate some markers (colored spheres) created by the geodesic robot in the task of thoracic image segmentation into three objects of interest and background. The color of the markers indicates object membership.

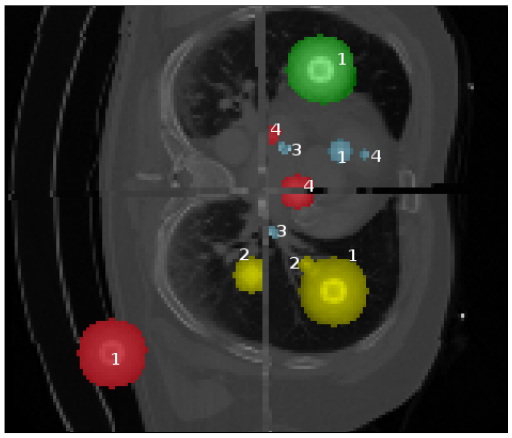


Figure 4.2: Spheres centered on seed voxels generated by the geodesic robot. At each iteration, indicated by the numbers, the error components are found and a number (up to a fixed limit) of seeds is chosen in their geodesic centers.

The geodesic robot requires several parameters to configure its behavior. The number of segmentation executions was set to 50, but after each execution a convergence check is performed. If the robot did not insert any new marker, then the segmentation result has converged. This requires to check if the error did not change from one iteration to another, since the object relaxation process can rearrange the border labels, possibly changing the error components, which may cause the robot to insert a new marker. In some cases, due to the relaxation process, the error values may alternate between consecutive executions, causing the robot to perform 50 executions, and increasing the mean number of executions for convergence in a given image database. The markers placed by the robot are spheres whose center is at a seed voxel and radius is set to five voxels. The spheres may also be reduced up to one-voxel radius whenever the seed is too close a ground-truth border. The idea is to avoid seeds outside regions of the same label. It is not guaranteed that the robot will perform a perfect segmentation, because error components may be formed in regions that the robot can not insert a new marker. For example, an error component closer than 2 voxels of the object border or too small to fit a marker.

The above parameters were chosen by following the strategy described in Section 4.4.

4.3 Efficiency and accuracy measures

For efficiency evaluation, we measured for each method and dataset: the mean total number of markers selected by the robot user to complete segmentation of all objects in the dataset, the mean total number of executions (user interventions for correction), and the mean response clock time per execution. Note that these measures are not

separated by object, therefore the efficiency gain of multi-object segmentation methods over MODGC increases with the number of objects.

For accuracy evaluation, we used the mean values of the Average Symmetric Surface Distance [74] and Dice[70] measures. Since all methods, including MODGC (Algorithm11), output a multi-object label map, these measures are presented for each object separately, and also their average for the corresponding dataset.

Dice coefficient offers a global and intuitive idea of the correspondence between the resulting segmentations. However, the surface distance is better when analyzing local differences. Both metrics are often used in the scientific literature [74, 45] and segmentation challenges [2].

4.3.1 Average Symmetric Surface Distance

Average Symmetric Surface Distance (ASSD) is the distance between two surfaces. It is averaged because it computes the average from surface A to surface B and from surface B to surface A. Let $\mathcal{B}_l$ be the border set of each object $l = 1, 2, \ldots, n$ under segmentation, $\mathcal{G}_l$ be the set with the real borders of each one of those objects. Let $D(t, \mathcal{G}_l)$ the Euclidean Distance value [63] of each voxel $t \in \mathcal{B}_l$ related to its closest voxel in $\mathcal{G}_l$. The mean error for each object is measured by Equation 4.1. This metric is only symmetric if it is computed the average from all $t \in \mathcal{B}_l$ to $\mathcal{G}_l$ and for all $t \in \mathcal{G}_l$ to $\mathcal{B}_l$.

$$ASSD = \frac{\frac{1}{n}\sum_{l=1}^{n}\left[\frac{1}{|\mathcal{B}_l|}\sum_{\forall t\in\mathcal{B}_l} D(t,\mathcal{G}_l)\right] + \frac{1}{n}\sum_{l=1}^{n}\left[\frac{1}{|\mathcal{G}_l|}\sum_{\forall t\in\mathcal{G}_l} D(t,\mathcal{B}_l)\right]}{2} \tag{4.1}$$

This value is multiplied by the voxel dimension to convert the results to millimeters (mm). When computing the distance, it is important to compute the closest distance to a border voxel with the same label, instead of considering only the closest voxel disregarding its label.

This operation consists of computing the distance between two surfaces, therefore in a perfect segmentation the distance is 0.

Figure 4.3 shows a segmentation in green and the correct segmentation in red. Voxels are only painted in red when they were segmented incorrectly. In this example, the first and second rows show the best and worst segmentation results with ASSD 0.65mm and 1.55mm, respectively.

[2] http://mbi.dkfz-heidelberg.de/grand-challenge2007/sites/eval.htm

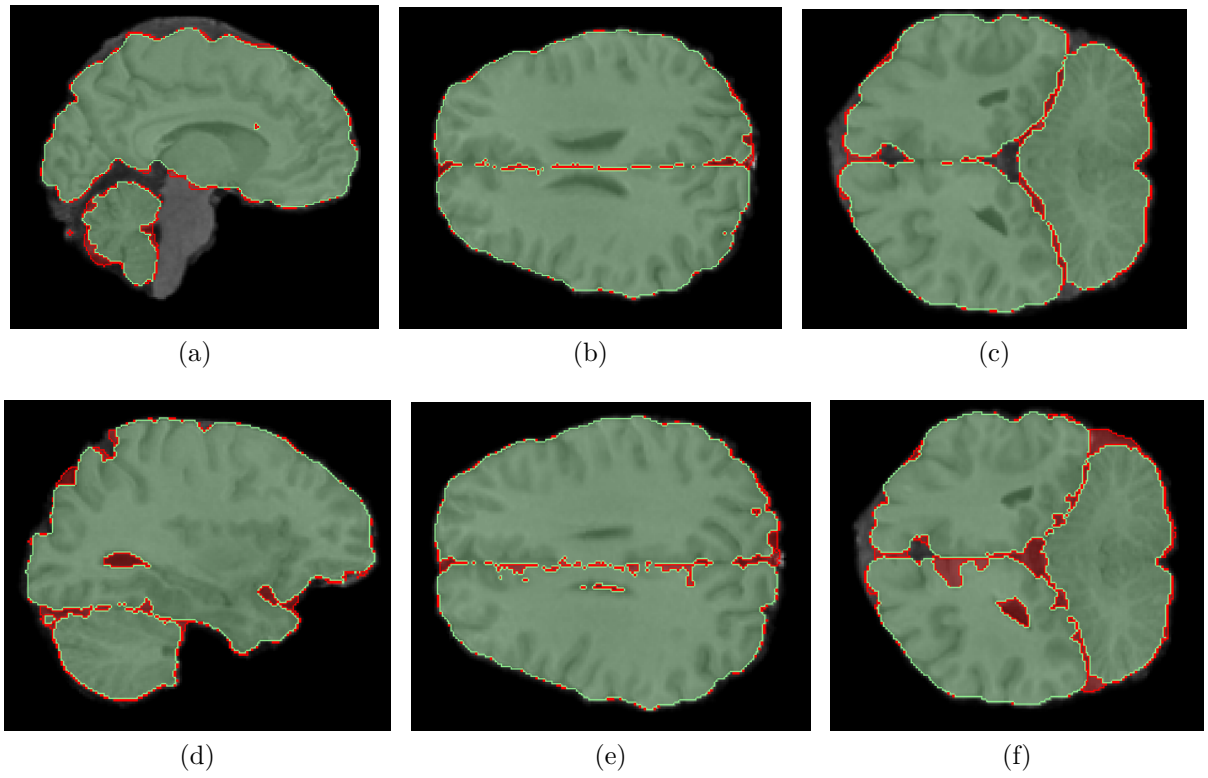


Figure 4.3: Three orthogonal slices of an MR-brain image showing the best with ASSD 0.65 in (a-c) and the worst with ASSD 1.55 in (d-f) segmentation results. Voxels correctly labeled are shown in green and errors in red.

4.3.2 Dice coefficient

Sørensen–Dice coefficient [70], is a statistical measure to compute the similarity between two samples. The similarity between the obtained segmentation label map and the ground truth can be computed with Equation 4.2. Let L_O be the achieved segmentation and $\mathcal{G}$ the correct segmentation (ground truth). Dice coefficient is the ratio between the intersection of both label maps divided by the size of each label map. In our case, $|L_O| = |\mathcal{G}|$.

$$dice = \frac{2 \times |L_O \cap \mathcal{G}|}{|L_O| + |\mathcal{G}|} \tag{4.2}$$

Dice value is a number within $[0, 1]$ with 1 being the best possible value. This metric can be misleading in the sense that a small object can be completely misplaced but since there is high intersection of the background pixels, it would still result in a high dice value.

4.4 Choice of parameters

For each dataset a training was performed on 25% of the images, randomly chosen to find the best parameter values for each method. This choice aimed to maximize the mean accuracy of the method, including the robot user, with respect to the ground-truth of the training images. The remaining 75% images were used as test for comparison among the methods.

In the case of MODGC, the parameter α in Equation 2.5 was optimized within $[1, 9]$ in order to avoid its reduction to Equation 2.6. For DRIFT and DIFT-TR, we optimized the number T of relaxation iterations and fixed the relaxation factor $\Omega.\beta$ at 0.5 (Section 3.3.1).

4.5 Experiments and Results

We divided the experiments into two parts. The first part aimed at showing that multi-object image segmentation is more effective than single-object segmentation. The second part compared DRIFT, DIFT, DIFT-TR, and MODGC, with statistical significance test. We used one-way Analysis of Variance (ANOVA) followed by a Tukey's HSD test. An interval of 95% confidence was considered. The tables in Section**??** show in bold the best approach according to the statistical test. No values in bold are presented when the methods are equivalent.

4.5.1 Single-Object versus Multi-Object Segmentation

In order to justify a multi-object segmentation, we executed the experiments for each object individually and a multi-object segmentation using the DIFT algorithm on the thorax dataset.

Regarding the ASSD metric, the thoracic images were benefited with the multi-object algorithm as shown in Table 4.1.

DIFT Single Object (mm)				DIFT Multiple Object (mm)
AS	RPC	LPC	Average	Average
8.73	1.57	1.94	4.08	1.29 $\pm$0.20

Table 4.1: Comparison between single-object and multi-object segmentation in the thorax dataset.

Since the execution time of the segmentation algorithms using the IFT framework is independent of the number of objects, segmenting each object separately yields to a

considerably slower process than segmenting objects simultaneously. Also, since an object may be close to another, its easier for a real user to determine the boundaries between them in the same image than in separated segmentation tasks.

4.5.2 Comparison among Multi-Object Segmentation Methods

The ASSD values in Table 4.2 are in *mm* and they correspond to the mean error over the 27 images (75% of the images) of the thorax dataset. Table 4.3 and Table 4.4 present the dice result and the user effort (efficiency) measurements, respectively. Values in bold represent statistical significance among the other values, according to ANOVA and Tukey's HSD test.

For the thorax dataset, the best number of relaxation iterations was 1 for DRIFT and 5 for DIFT-TR. The best α parameter in MODGC was 9.

ASSD - Thorax

	Average	AS	RPC	LPC
DIFT	0.67 ± 0.06	1.09 ± 0.1	0.46 ± 0.08	0.46 ± 0.06
DIFT-TR	**0.60 ± 0.05**	0.93 ± 0.1	0.44 ± 0.07	0.43 ± 0.05
DRIFT	**0.63 ± 0.13**	1.08 ± 0.3	0.41 ± 0.06	0.40 ± 0.03
MODGC	0.80 ± 0.05	1.19 ± 0.1	0.62 ± 0.06	0.59 ± 0.04

Table 4.2: ASSD Error in millimeters for the thorax dataset.

Dice - Thorax

	Average	AS	RPC	LPC
DIFT	0.90 ± 0.01	0.77 ± 0.02	0.96 ± 0.01	0.96 ± 0.01
DIFT-TR	0.90 ± 0.01	0.79 ± 0.02	0.96 ± 0.01	0.96 ± 0.01
DRIFT	0.90 ± 0.01	0.77 ± 0.02	0.96 ± 0.01	0.96 ± 0.01
MODGC	0.89 ± 0.05	0.76 ± 0.03	0.96 ± 0.01	0.96 ± 0.01

Table 4.3: Dice coefficient for the multiple object thorax dataset.

The ASSD values in Table 4.5 are in *mm* and they correspond to the mean error over the 30 images (75% of the images) of the brain dataset. Table 4.6 and Table 4.7 present the dice result and the user effort (efficiency) measurements, respectively. Values in bold represent statistical significance among the other values, according to ANOVA and Tukey's HSD test.

Efficiency Measures - Thorax

	Num Markers	Num Executions	Response time (s)
DIFT	70 ± 12	16 ± 4	**0.40 ± 0.08**
DIFT-TR	70 ± 12	16 ± 4	**0.49 ± 0.10**
DRIFT	**57 ± 12**	17 ± 4	**0.51 ± 0.11**
MODGC	88 ± 16	14 ± 3	7.08 ± 1.69

Table 4.4: Mean number of markers, mean number of corrections, and the mean waiting time for the user after each correction in the thorax dataset.

For the brain dataset, the best number of relaxation iterations was 1 for DRIFT and 5 for DIFT-TR. The best α parameter in MODGC was 9.

ASSD - Brain

	Average	C	LBH	RBH
DIFT	0.73 ± 0.05	0.91 ± 0.1	0.64 ± 0.07	0.64 ± 0.06
DIFT-TR	**0.63 ± 0.04**	0.83 ± 0.1	0.53 ± 0.07	0.53 ± 0.05
DRIFT	0.73 ± 0.06	0.90 ± 0.1	0.64 ± 0.07	0.64 ± 0.06
MODGC	0.85 ± 0.07	0.91 ± 0.1	0.75 ± 0.18	0.89 ± 0.08

Table 4.5: ASSD Error in millimeters for the brain dataset.

Dice - Brain

	Average	C	LBH	RBH
DIFT	0.96 ± 0.00	0.93 ± 0.01	0.97 ± 0.00	0.97 ± 0.00
DIFT-TR	0.96 ± 0.00	0.94 ± 0.01	0.97 ± 0.00	0.97 ± 0.00
DRIFT	0.96 ± 0.00	0.93 ± 0.01	0.97 ± 0.00	0.97 ± 0.00
MODGC	0.96 ± 0.01	0.94 ± 0.01	0.97 ± 0.01	0.96 ± 0.00

Table 4.6: Dice coefficient for the multiple object brain dataset.

The ASSD values in Table 4.8 are in *mm* and they correspond to the mean error over the 15 images (75% of the images) of the foot dataset. Table 4.9 and Table 4.10 present the dice result and the user effort (efficiency) measurements, respectively. Values in bold represent statistical significance among the other values, according to ANOVA and Tukey's HSD test.

Efficiency Measures - Brain

	Num Markers	Num Executions	Response time (s)
DIFT	335 ± 79	26 ± 8	**0.17 ± 0.04**
DIFT-TR	335 ± 79	26 ± 8	**0.22 ± 1.25**
DRIFT	**296 ± 84**	24 ± 7	3.08 ± 2.03
MODGC	332 ± 47	28 ± 6	15.91 ± 7.29

Table 4.7: Mean number of markers, mean number of corrections, and the mean waiting time for the user after each correction in the brain dataset.

For the foot dataset, the best number of relaxation iterations was 5 for DRIFT and 50 for DIFT-TR. The best α parameter in MODGC was 9.

ASSD - Foot

	Average	Calcaneous	Talus
DIFT	1.30 ± 0.10	1.44 ± 0.12	1.17 ± 0.11
DIFT-TR	**0.90 ± 0.06**	0.96 ± 0.08	0.85 ± 0.06
DRIFT	1.05 ± 0.09	1.17 ± 0.12	0.93 ± 0.10
MODGC	1.52 ± 0.20	1.99 ± 0.41	1.05 ± 0.06

Table 4.8: ASSD Error in millimeters for the foot dataset.

Dice - Foot

	Average	C	Talus
DIFT	0.93 ± 0.01	0.93 ± 0.00	0.93 ± 0.01
DIFT-TR	0.94 ± 0.00	0.94 ± 0.00	0.94 ± 0.01
DRIFT	0.93 ± 0.01	0.92 ± 0.01	0.93 ± 0.01
MODGC	0.90 ± 0.01	0.87 ± 0.02	0.92 ± 0.01

Table 4.9: Dice coefficient for the multiple object foot dataset.

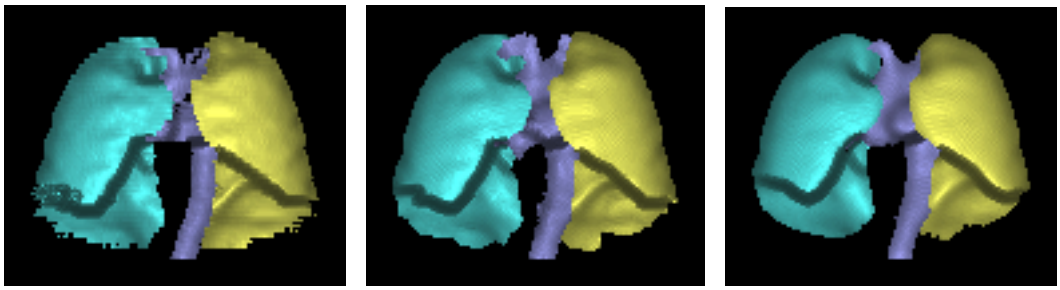
4.6 Visual Performance

This presents a visual (Figure 4.4, Figure 4.5, Figure 4.6) comparison between DRIFT/DRIFT-TR (the best for each dataset according to ASSD) and DIFT with respect to the ground truth segmentation.

Efficiency Measures - Foot

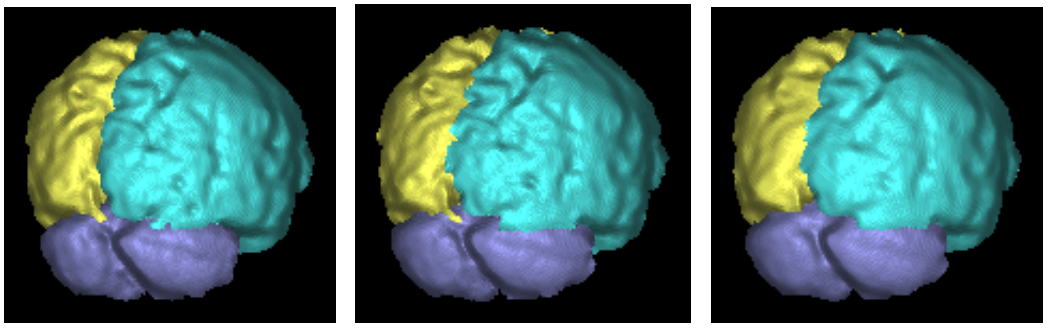
	Num Markers	Num Executions	Response time (s)
DIFT	393 ± 144	45 ± 4	**0.5 ± 0.1**
DIFT-TR	393 ± 144	45 ± 4	1.8 ± 0.2
DRIFT	**243 ± 116**	47 ± 3	2.4 ± 0.4
MODGC	436 ± 17	55 ± 4	19.5 ± 6.6

Table 4.10: Mean number of markers, mean number of corrections, and the mean waiting time for the user after each correction in the foot dataset.



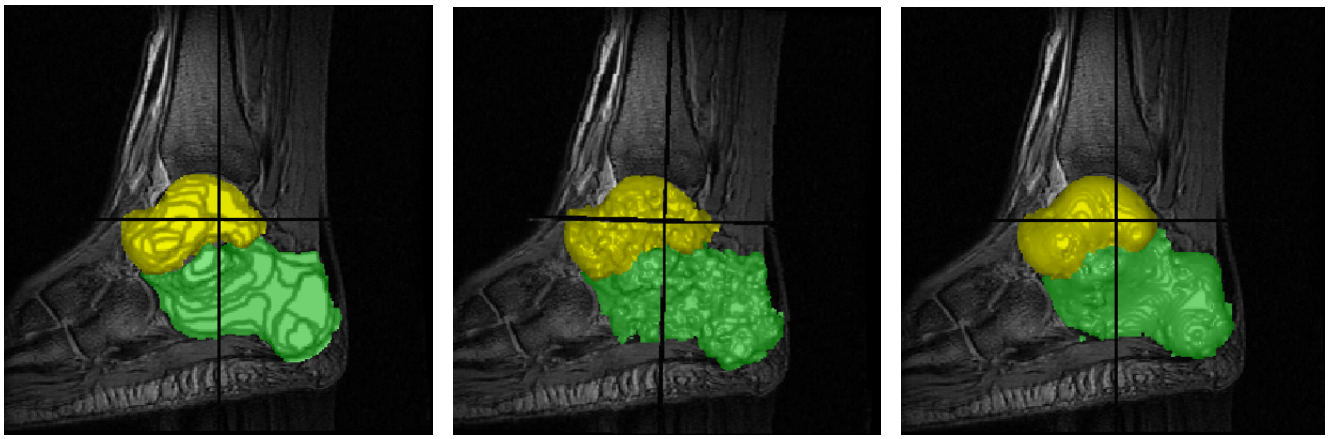
(a) (b) (c)

Figure 4.4: (a) Ground truth segmentation of the left pleural cavity, right pleural cavity, and arterial system. (b) Non relaxed segmentation of the thorax CT using the geodesic robot. (c) Relaxed segmentation of the thorax CT using 1 relaxation iteration per execution.



(a) (b) (c)

Figure 4.5: (a) Ground truth segmentation of the left brain hemisphere, right brain hemisphere and cerebellum. (b) Non relaxed segmentation of the MR-brain using the geodesic robot. (c) DIFT-TR segmentation of the brain with 5 relaxation iterations.



(a) (b) (c)

Figure 4.6: (a) Ground truth segmentation of the talus and calcaneous. (b) Non relaxed segmentation of the foot MRI using the geodesic robot. (c) Relaxed segmentation of the brain MRI using 50 relaxation iterations at the end of the process.

4.7 Discussion

The first part of the experiments show that multi-object segmentation can provide more accurate results than single-object segmentation. This is possibly due to the lack of precision in merging the individual label maps into a multi-object label map. Although we did not account for efficiency it is possible to observe from the efficiency tables in the previous section that the multi-object segmentation methods are significantly more efficient than MODGC.

During the experiments it was noticed that $E_{\min}$-based methods are more accurate than E_{sum}-based. In practical terms, small differences in ASSD may represent significant segmentation errors (Figure 4.3). The efficiency gain of $E_{\min}$-based methods over E_{sum}-based is clearly due to the multi-object segmentation paradigm.

Object relaxation is another relevant factor to be considered. Applying object relaxation after each execution (DRIFT) consistently reduces the number of required markers without compromising the response time, matching our experience in the interactive segmentation test presented in Section 3.4. This also indicates that the high number of relaxation trees created by DRIFT does not affect user control over the process. However, DIFT-TR was consistently more accurate (at least equivalent to) than DRIFT. This was achieved at the price of more markers, indicating that the user can also put more effort in segmentation without worrying about small corrections and leave those corrections to be automatically done at the end by the relaxation procedure.

The Dice coefficient is fast to compute and a good metric to provide a preliminary

evaluation, but it is not precise enough to compare methods in our experience. All results turned out to be equivalent in this metric.

Ideally, a suitable gradient image for each dataset should be created focused in the image modality and in the image properties of the objects. For instance in the brain dataset, it is possible to provide a better gradient by degrading the contrast between pia mater and CSF (Cerebral Spinal Fluid) in favor to the contrast between CSF and gray mater (the boundary between hemispheres and CSF) [2].

The foot dataset contains very noisy images and a low-pass filter was applied before computing the gradient image by Equation 2.2. However, this was not enough to avoid leaking in several parts which, required a considerably higher number of relaxation iterations in DIFT-TR.

Chapter 5

Conclusion and future works

According to [14], graph-cut image segmentation algorithms may be divided into two types of energy minimizers, $E_{\max}$ and E_{sum}. Approaches based on $E_{\max}$ minimize Equation 2.6 (the same as maximize Equation 2.7), while approaches based on E_{sum} minimize Equation 2.5. In the context of interactive segmentation, where the user adds and removes markers to indicate the objects of interest, these approaches can be more efficiently implemented with the Differential Image Foresting Transform (DIFT) algorithm [9] and Dynamic Graph Cut (DCG) algorithm [6], respectively. While the latter is limited to binary segmentation, the former provides more irregular object boundaries. Moreover, we observed that the original DIFT algorithm could not guarantee a consistent segmentation when marker addition and removal were performed simultaneously.

In this work, we first fixed the inconsistency problem by presenting a new version of the DIFT algorithm. In order to provide more regular object boundaries and inspired on the previous work of Filip Malmberg [24], we proposed a fast diffusion filtering algorithm for object boundary relaxation on the results of the DIFT algorithm. The diffusion filtering, as any other independent boundary regularization procedure, may make the segmentation result inconsistent with respect to the user-selected markers, but we presented a solution that adds relaxation trees to the resulting optimum-path forest. As side effect, the relaxation trees tend to be small for user corrections, but we provided two rules that amend this problem: (1) relaxation trees are associated with user-drawn markers, so they can be deleted by removing their corresponding marker, and (2) user-drawn markers have higher priority than relaxation roots, so relaxation trees are automatically removed when the user draws markers that intercept them. However, this is not easy when the relaxation trees are trivial. In this case, it is better to select their corresponding markers for removal, turn off the relaxation procedure, and add new markers to correct segmentation and turn on the relaxation procedure afterwards. The resulting interactive segmentation method, named DRIFT, relies on the new DIFT algorithm to segment multiple objects simultane-

ously and on the new diffusion filtering procedure for object boundary relaxation. It was compared to multi-object segmentation using MODGC, DIFT, and DIFT-TR (another proposed approach). The experiments involved eight body anatomical structures from several CT-thorax images, MR-brain images, and MR-foot images, and a geodesic robot user[73] to avoid human bias.

The experiments showed the advantages in accuracy and efficiency of the multi-object segmentation paradigm over single-object. We may conclude that DRIFT is the most efficient approach (less user effort), being only slightly less accurate than the most accurate one, DIFT-TR.

In [75] it is observed that unexperienced users tend to spend more time fixing small error components. In this context, the relaxation procedure seems to be helpful in DRIFT and DIFT-TR. However, the choice of the best approach may change depending on the application (e.g., different object shapes) and the human user. In this sense, it is important to consider experiments with real users in a future work.

It is also important to improve the gradient image depending on the application. All segmentation methods can benefit from gradient images that present higher values on the object boundaries than elsewhere. This aspect is also left for a future work.

DRIFT and DIFT-TR represent important contributions for interactive segmentation which can provide accurate results with minimum effect to the user controllability and waiting time between corrections. We intent to explore them in real medical image analysis applications. Moreover, the algorithm presented in [76] allows to convert the segmentation result of any method into an optimal path forest with minimal number of roots for interactive corrections. Therefore the proposed relaxation procedure can also be applied to relax those object boundaries.

Bibliography

[1] R. Audigier, R. Lotufo, and A. Falcão, "3D visualization to assist iterative object definition from medical images," *Computerized Medical Imaging and Graphics*, vol. 30, no. 4, pp. 217–230, 2006.

[2] P. Miranda, A. Falcão, and J. K. Udupa, "Cloud models: Their construction and employment in automatic mri segmentation of the brain," 2010.

[3] S. Kobashi and J. Udupa, "Fuzzy object model based fuzzy connectedness image segmentation of newborn brain mr images," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012, pp. 1422–1427.

[4] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," in *Computer Vision*, 1998, vol. 1407, pp. 484–498.

[5] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models – their training and application," *Computer Vision Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[6] P. Kohli and P. H. S. Torr, "Dynamic graph cuts for efficient inference in markov random fields," *IEEE Transactions Pattern Analysis Machine Intelligence*, vol. 29, no. 12, pp. 2079–2088, 2007.

[7] N. Moya, A. X. Falcão, K. C. Ciesielski, and J. K. Udupa, "Differential and relaxed image foresting transform for graph-cut segmentation of multiple 3d objects," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2014, vol. 8673, pp. 690–697.

[8] K. Ciesielski, J. Udupa, P. Miranda, and A. Falcão, "Joint graph cut and relative fuzzy connectedness image segmentation algorithm," *Medical Image Analysis*, vol. 17, no. 8, pp. 1046–1057, 2013.

[9] A. Falcão and F. Bergo, "Interactive volume segmentation with differential image foresting transforms," *IEEE Transactions on Medical Imaging*, vol. 23, no. 9, pp. 1100–1108, 2004.

[10] T. Spina, P. Miranda, and A. Falcão, "Hybrid approaches for interactive image segmentation using the live markers paradigm," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5756–5769, 2014.

[11] P. Rauber, A. Falcão, T. Spina, and P. de Rezende, "Interactive segmentation by image foresting transform on superpixel graphs," in *Brazilian Symposium on Graphics, Patterns and Images (SIBGRAPI)*, 2013, pp. 131–138.

[12] Y. Y. Boykov, "Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images." *International Conference on Computer Vision (ICCV)*, vol. 1, pp. 105–112, 2001.

[13] A. X. Falcão, P. A. V. Miranda, and A. Rocha, "A linear-time approach for image segmentation using graph-cut measures," in *Advanced Concepts for Intelligent Vision Systems*, 2006, pp. 138–149.

[14] K. C. Ciesielski, J. K. Udupa, A. X. Falcão, and P. A. V. Miranda, "Fuzzy connectedness image segmentation in graph cut formulation: A linear-time algorithm and a comparative analysis." *Journal of Mathematical Imaging and Vision*, vol. 44, no. 3, pp. 375–398, 2012.

[15] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest," in *IEEE International Conference onComputer Vision*, 2009, pp. 731–738.

[16] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Thinnings, shortest path forests, and topological watersheds," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, no. 5, pp. 925–939, 2010.

[17] P. K. Saha and J. K. Udupa, "Relative fuzzy connectedness among multiple objects: Theory, algorithms, and applications in image segmentation," *Computer Vision and Image Understanding*, vol. 82, no. 1, pp. 42–56, 2001.

[18] J. K. Udupa and S. Samarasekera, "Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation," *Graphical Models and Image Processing*, vol. 58, no. 3, pp. 246–261, 1996.

[19] K. C. Ciesielski, J. K. Udupa, P. K. Saha, and Y. Zhuge, "Iterative relative fuzzy connectedness for multiple objects with multiple seeds," *Computer Vision and Image Understanding*, vol. 107, no. 3, pp. 160 – 182, 2007.

[20] V. Vezhnevets and V. Konouchine, "Growcut - interactive multi-label n-d image segmentation by cellular," pp. 150–156, 2005.

[21] P. A. Miranda and A. X. Falcão, "Links between image segmentation based on optimum-path forest and minimum cut in graph," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 128–142, 2009.

[22] L. Ford and D. Fulkerson, "Flows in networks." 1962.

[23] A. Falcão, J. Stolfi, and R. de Alencar Lotufo, "The image foresting transform: theory, algorithms, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 19–29, 2004.

[24] F. Malmberg, I. Nyström, A. Mehnert, C. Engstrom, and E. Bengtsson, "Relaxed image foresting transforms for interactive volume image segmentation," in *Proceedings of SPIE on Medical Imaging*, vol. 7623, no. 7623, 2010.

[25] P.Kohli, H.Nickisch, C.Rother, and C.Rhemann, "Learning an interactive segmentation system," *International Journal of Computer Vision*, vol. 100, no. 3, pp. 261–274, 2012.

[26] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. de A. Lotufo, "User-steered image segmentation paradigms: Live wire and live lane," *Graphical Models and Image Processing*, vol. 60, no. 4, pp. 233 – 260, 1998.

[27] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[28] K. Abend, T. Harley, and L. Kanal, "Classification of binary random patterns," *IEEE Transactions Information Theory*, vol. 11, no. 4, pp. 538–544, 2006.

[29] L. Grady, "Random walks for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[30] X. Bai and G. Sapiro, "A geodesic framework for fast interactive image and video segmentation and matting," in *IEEE International Conference on Computer Vision (ICCV)*, 2007, pp. 1–8.

[31] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[32] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1163–1173, 1996.

[33] R. Lotufo and A. Falcão, "The ordered queue and the optimality of the watershed approaches," in *In Mathematical Morphology and its Applications to Image and Signal Processing*, 2000, pp. 341–350.

[34] R. A. Lotufo, A. X. Falcão, and F. Zampirolli, "IFT-Watershed from gray-scale marker," in *Proceedings of XV Brazilian Symposium on Computer Graphics and Image Processing*, 2002, pp. 146–152.

[35] J. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, 1999.

[36] A. Falcão, J. Udupa, and F. Miyazawa, "An ultra-fast user-steered image segmentation paradigm: live wire on the fly," *Medical Imaging, IEEE Transactions on*, vol. 19, no. 1, pp. 55–62, 2000.

[37] A. X. Falcão and J. K. Udupa, "A 3d generalization of user-steered live-wire segmentation," *Medical Image Analysis*, vol. 4, no. 4, pp. 389 – 402, 2000.

[38] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[39] T. Vallin Spina, A. Falcão, and P. Vechiatto Miranda, "User-steered image segmentation using live markers," in *Computer Analysis of Images and Patterns*, 2011, vol. 6854, pp. 211–218.

[40] P. A. V. Miranda, A. X. Falcão, and T. V. Spina, "Riverbed: A novel user-steered image segmentation method based on optimum boundary tracking." *IEEE Transactions on Image Processing*, vol. 21, no. 6, pp. 3042–3052, 2012.

[41] C. Suzuki, J. Gomes, A. Falcão, S. Shimizu, and J. Papa, "Automated diagnosis of human intestinal parasites using optical microscopy images," in *International Symposium on Biomedical Imaging (ISBI)*, 2013, pp. 460–463.

[42] P. de Miranda, A. Falcão, and J. Udupa, "Synergistic arc-weight estimation for interactive image segmentation using graphs," *Computer Vision and Image Understanding*, vol. 114, no. 1, pp. 85 – 99, 2010.

[43] Y. Gao and A. Tannenbaum, "Combining atlas and active contour for automatic 3d medical image segmentation." in *International Symposion on Biomedical Imaging (ISBI)*, 2011, pp. 1401–1404.

[44] J. Liu and J. Udupa, "Oriented active shape models," *IEEE Transactions on Medical Imaging*, vol. 28, no. 4, pp. 571–584, 2009.

[45] R. Phellan, A. Falcão, and J. Udupa, "Improving atlas-based medical image segmentation with a relaxed object search," in *Computational Modeling of Objects Presented in Images. Fundamentals, Methods, and Applications*, 2014, vol. 8641, pp. 152–163.

[46] S. D. Olabarriaga and A. W. M. Smeulders, "Interaction in the segmentation of medical images: A survey," *Medical Image Analysis*, vol. 5, no. 2, pp. 127–142, 2001.

[47] M. A. Schulze, "Active contours (snakes)," July 2003, available at http://www.markschulze.net/snakes/.

[48] F. Meyer and S. Beucher, "Morphological segmentation," *Journal of Visual Communication and Image Representation*, vol. 1, no. 1, pp. 21–46, 1990.

[49] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: interactive foreground extraction using iterated graph cuts," in *ACM SIGGRAPH Papers*, 2004, pp. 309–314.

[50] 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*, vol. 15, pp. 1101–1113, 1993.

[51] I. Cox, S. Rao, and Y. Zhong, "Ratio regions: a technique for image segmentation." *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 557–564, 1996.

[52] S. Wang and J. Siskind, "Image segmentation with minimum mean cut," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1, 2001, pp. 517–524.

[53] S. Sarkar and K. Boyer, "Quantitative measures of change based on feature organization: eigenvalues and eigenvectors," in *IEEE on Computer Vision and Pattern Recognition (CVPR)*, 1996, pp. 478–483.

[54] S. Wang and J. M. Siskind, "Image segmentation with ratio cut," *IEEE Transactions Pattern Analysis Machine Intelligence*, vol. 25, no. 6, pp. 675–690, 2003.

[55] J. Shi and J. Malik, "Normalized cuts and image segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 888–905, 2000.

[56] C. Fowlkes, S. Belongie, and J. Malik, "Efficient spatiotemporal grouping using the nyström method," in *Computer Vision and Pattern Recognition*, 2001, pp. 231–238.

[57] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1124–1137, 2004.

[58] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 65–81, 2004.

[59] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society*, vol. 51, no. 2, pp. 271–279, 1989.

[60] A. X. Falcão, B. S. da Cunha, and R. A. Lotufo, "Design of connected operators using the image foresting transform," in *Proceedings of SPIE on Medical Imaging*, vol. 4322, 2001, pp. 468–479.

[61] R. A. Lotufo, A. X. Falcão, and F. Zampirolli, "IFT-Watershed from gray-scale marker," in *Proceedings of XV Brazilian Symposium on Computer Graphics and Image Processing*, 2002, pp. 146–152.

[62] A. X. Falcão, L. F. Costa, and B. S. da Cunha, "Multiscale skeletons by image foresting transform and its applications to neuromorphometry," *Pattern Recognition*, vol. 35, no. 7, pp. 1571–1582, 2002.

[63] R. d. A. Lotufo, A. X. Falcão, and F. A. Zampirolli, "Fast euclidean distance transform using a graph-search algorithm," in *Proceedings of the 13th Brazilian Symposium on Computer Graphics and Image Processing*, 2000, pp. 269–275.

[64] A. Falcão, C. Suzuki, J. Gomes, J. Papa, L. Dias, and S. Shimizu, "A system for diagnosing intestinal parasites by computerized image analysis," 2008.

[65] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki, "Supervised pattern classification based on optimum-path forest," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 120–131, 2009.

[66] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falcão, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 50–68, 2009.

[67] R. Audigier and R. de Alencar Lotufo, "Tie-zone watershed, bottlenecks, and segmentation robustness analysis," in *Brazilian Symposium on Computer Graphics and Image Processing*, 2005, pp. 55–62.

[68] J. B. Roerdink and A. Meijster, "The watershed transform: definitions, algorithms and parallelization strategies," *Annales Societatis Mathematicae Polonae*, vol. 41, no. 1, 2, pp. 187–228, 2000.

[69] P. Miranda, A. Falcão, G. Ruppert, and F. Cappabianco, "How to fix any 3d segmentation interactively via image foresting transform and its use in mri brain segmentation," in *IEEE International Symposium on Biomedical Imaging*, 2011, pp. 2031–2035.

[70] T. Sørensen, *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons*. I kommission hos E. Munksgaard, 1948.

[71] K. McGuinness and N. E. O'Connor, "Toward automated evaluation of interactive segmentation," *Computer Vision and Image Understanding*, vol. 115, no. 6, pp. 868 – 884, 2011.

[72] N. E. O'Connor and K. McGuinness, "A comparative evaluation of interactive segmentation algorithms," *Pattern Recognition*, vol. 43, no. 2, pp. 434 – 444, 2010.

[73] P. Kohli, H. Nickisch, C. Rother, and C. Rhemann, "User-centric learning and evaluation of interactive segmentation systems," *International Journal of Computer Vision*, vol. 100, no. 3, pp. 261–274, 2012.

[74] T. R. Langerak, U. A. van der Heide, A. N. T. J. Kotte, F. F. Berendsen, and J. P. W. Pluim, "Evaluating and improving label fusion in atlas-based segmentation using the surface distance," pp. 796 226–796 226–7, 2011.

[75] T. V. Spina, P. A. V. de Miranda, and A. X. Falcão, "Intelligent understanding of user interaction in image segmentation," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 26, no. 2, p. 26, 2012.

[76] P. A. V. Miranda, A. X. Falcao, and G. C. S. Ruppert, "How to complete any segmentation process interactively via image foresting transform," in *Brazilian Symposium on Graphics, Patterns and Images (SIBGRAPI)*, 2010, pp. 309–316.