On some extensions of the level sets and the graph cuts towards their application to image and video segmentation

Olivier Juan

THÈSE

présentée pour l'obtention du titre de

DOCTEUR DE L'ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

Spécialité : Informatique

par

Olivier JUAN

Quelques Extensions des Level Sets et des Graph Cuts & Leurs Applications à la Segmentation d'Images et de Vidéos

On Some Extensions of Level Sets and Graph Cuts & Their Applications to Image and Video Segmentation

Soutenance le 15 mai 2006 devant le jury composé de :

Rapporteurs :	Michel BARLAUD Nikos PARAGIOS
Examinateurs :	Yuri BOYKOV Daniel CREMERS Renaud KERIVEN
Direction de thèse :	Olivier FAUGERAS

Abstract

Image processing techniques are now widely spread out over a large quantity of domains: like medical imaging, movies post-production, games... Automatic detection and extraction of regions of interest inside an image, a volume or a video is challenging problem since it is a starting point for many applications in image processing. However many techniques were developed during the last years and the state of the art methods suffer from some drawbacks: The Level Sets method only provides a local minimum while the Graph Cuts method comes from Combinatorial Community and could take advantage of the specificity of image processing problems. In this thesis, we propose two extensions of the previously cited methods in order to soften or remove these drawbacks.

We first discuss the existing methods and show how they are related to the segmentation problem through an energy formulation. Then we introduce stochastic perturbations to the Level Sets method and we build a more generic framework: the Stochastic Level Sets (SLS). Later we provide a direct application of the SLS to image segmentation that provides a better minimization of energies. Basically, it allows the contours to escape from local minimum. Then we propose a new formulation of an existing algorithm of Graph Cuts in order to introduce some interesting concept for image processing community: like initialization of the algorithm for speed improvement. We also provide a new approach for layer extraction from video sequence that retrieves both visible and hidden layers in it.

Résumé

Les techniques de traitement d'image sont maintenant largement répandues dans une grande quantité de domaines : comme l'imagerie médicale, la post-production de films, les jeux... La détection et l'extraction automatique de régions d'intérêt à l'intérieur d'une image, d'un volume ou d'une vidéo est réel challenge puisqu'il représente un point de départ pour un grand nombre d'applications en traitement d'image. Cependant beaucoup de techniques développées pendant ces dernières années et les méthodes de l'état de l'art souffrent de quelques inconvénients : la méthode des ensembles de niveaux fournit seulement un minimum local tandis que la méthode de coupes de graphe vient de la communauté combinatoire et pourrait tirer profit de la spécificité des problèmes de traitement d'image. Dans cette thèse, nous proposons deux prolongements des méthodes précédemment citées afin de réduire ou enlever ces inconvénients.

Nous d'abord discutons les méthodes existantes et montrons comment elles sont liées au problème de segmentation via une formulation énergétique. Nous présentons ensuite des perturbations stochastiques a la méthode des ensembles de niveaux et nous établissons un cadre plus générique : les ensembles de niveaux stochastiques (SLS). Plus tard nous fournissons une application directe du SLS à la segmentation d'image et montrons qu'elle fournit une meilleure minimisation des énergies. Fondamentalement, il permet aux contours de s'échapper des minima locaux. Nous proposons ensuite une nouvelle formulation d'un algorithme existant des coupes de graphe afin d'introduire de nouveaux concepts intéressant pour la communauté de traitement d'image : comme l'initialisation de l'algorithme pour l'amélioration de vitesse. Nous fournissons également une nouvelle approche pour l'extraction de couches d'une vidéo par segmentation du mouvement et qui extrait à la fois les couches visibles et cachées présentes.

Acknowledgments

First of all, I would like to thank Yuri Boykov, Jean-François Delmas and Maureen Clerc who each guided and directed my researches during the different periods of my PhD Studies. I would like to thank Renaud Keriven for his advices and supervision all along this journey.

I would like to express my gratitude to Olivier Faugeras for accepting me as one of his PhD student.

I would like to thank Nikos Paragios and Michel Barlaud, who accepted to be reviewers of my thesis. Many thanks also to committee members Yuri Boykov and Daniel Cremers.

This thesis was accomplished in the project Odyssée at the CERTIS lab in Ecole Nationale des Ponts et Chaussées (ENPC), Paris. It has been funded by the ENPC itself, by a RIAM project, a French national grant with Realviz and Duboi on rotoscoping, and by Yuri Boykov via a six months student exchange with the University of Western Ontario, London, Ontario, Canada.

The results within the context of the RIAM project are not presented here. This work represents six months of involvement for me but it motivated this thesis on segmentation (image and video).

I would like to express special thanks to Nikos Paragios for all his advices and for providing me the opportunity to meet and work with Yuri Boykov.

I would like to thank Yuri Boykov for welcoming me for six months at the Computer Science Department of the University of Western Ontario and for the fruitful work that happened there. While speaking of Canada, I would like to thank for their harmful welcome: Olga Veksler, Andrew Delong, Dan Santoni and Mike Burrell.

Several people have widely contributed to the work presented in this manuscript: Gheorghes Postelnicu on the Stochastic Level Sets, Yuri Boykov and Andrew Delong on the Active Cuts and Romain Dupont on Motion Layer Extraction. I greatly thank all those collaborators and friends.

I would like to thank my family, and in particular my parents, for their unconditional support and encouragement along my studies.

I would like also to thank some of my friends who supported and encouraged me during my thesis: Georges, Geoffray, Nathalie, Nicola, Romain.

A special thanks goes to Charlotte who always encouraged me during the difficult times. I will not have succeeded without her support. In this and many other things, I greatly thank her and acknowledge her patience.

Contents

List of Figures

Introduction

Position of segmentation in real life: critical ...

Image and signal processing tools get more and more into every day life: cellphones, DVD, digital camera... Image processing aims at extracting meaningful data from image or video. Human watch cannot achieve surveillance of large areas accurately; computers provide their calculation power to allow real-time analysis of video and detection of risky situations. Image processing is more and more used in Medical applications. It can be used for image enhancement on noisy image (ultrasound scan image) or volume segmentation for a given acquisition modality. Image or volume registration of different modalities is also a considerable challenge since different modalities reveal different information. Recently, Image Processing community undertakes to help doctors to diagnosis of diseases and to surgical operations.

In every day life, we use image-processing tools without noticing. Indeed image or video compression is a build-in part of common products: DVD, cellphones, digital cameras, video-talk... We could also cite watermarking technique that was developed to insert undeletable copyright signature directly inside image and video.

Another challenging application is image classification. Indeed image databases, even for a holiday photographer, are getting bigger and bigger and handmaid classification is no more usable. However, automatic image classification is far from a trivial process but the large number of application makes it even more attractive.

All those techniques make use of image segmentation during the process, mostly at the beginning. Segmentation stands for a generic problem that can be expressed in many different formulations. However, it can be defined by the following sentence. Segmentation consists into extracting and labeling regions inside an image. Many different features can describe regions depending on the desired application.

Features commonly refer to intensities, colors, textures, shapes, movements or even depth. However, one could design is own feature according to

the considered segmentation problem.

Usually the problem is posed as an energy minimization problem where the energy measures how good a given segmentation fits the image. Energies are designed to take into account the needed features. The generic problem of segmenting any given image is intractable since it should takes into account an incredible number of features. However, the generic problem can be exploded into a bunch of problems, each one segmenting its own restricted class of images. Then a supervisor process such as an expert system will mutualise them all in order to produce a robust generic segmentation process.

History of segmentation approaches

We refer the reader to the Computer Vision handbook [133] for any method cited in the following paragraphs.

Historically, image segmentation was based on local feature extraction such as corners or T-junctions. Some feature entities in the image are extracted using generic methods such as Canny filters (points of interest: corner...) or Hough transforms (lines or circles). By assuming that feature entities are located on object boundaries, one expected to rebuild contours of an object by assembling feature data.

Later contour evolution methods were introduced to manage segmentation without the fastidious assembling of features. Furthermore, the assembling step was quite weak. Those new methods consist into a (parametric) contour driven by the feature data usually image gradient. Indeed, by definition object boundaries match high image gradient. However, these methods depend on the local structure of the image and therefore they may be stuck on local minima. This problem motivated the use of more global features to drive contours such as region texture, region colors...

The Level Set method brought a general framework to evolve contours (in any dimension). With some sacrifices (mainly speed), it removes at lot of drawbacks of parametric methods. The Level Set method was widely spread over the community and in a large number of applications. Since it was widely studied over the years, it comes now with a bunch of stable numerical schemes. However, a standard gradient descent method is mainly used to minimize the energies. As many energies are not convex, this minimization process leads to a local minimum.

Recently Stochastic Active Contour was introduced to improve the accuracy of the minimizer. Stochastic Active Contour consists into bringing simulated annealing concept into Level Set framework. Even if no theoretic results are available about the decreasing temperature scheme to apply in

order to ensure a global minimum, it shows real improvements in practice.

In the same time, Markov Random Field methods were widely developed but the lack of good minimizer tools limited their application. Recently new minimizing methods based on Max-Flow/Min-Cut problem received a large attention from the Computer Vision community. Indeed those algorithms provide a global minimum in a relative small time for class of energies widely used in Computer Vision.

Organization

This thesis is organized as follow.

Chapter 1

In chapter 1, we review existing variational approaches for segmentation based on contour evolution. We mainly present the two main classes of evolution: Boundary [95] and Region [30, 82] based evolution. We also describe a widely used Bayesian inference scheme to integrate any region feature into a contour evolution.

In particular Geodesic Active Contour [26, 27, 98] and Geodesic Active Region [134, 136, 83] are presented for parametric contour model. Advantages and drawbacks are discussed and some solutions are given.

Chapter 2

In chapter 2, we present the Level Set framework [59, 60, 132] and how it suits well to contour evolution. Some numerical schemes are provided. We provide also some tips and tricks to implement Level Set with equanimity. Advantages (mainly build-in change of topology and stable numerical scheme) and drawbacks (mainly speed) are debated.

We also provide some implementations of classical curve evolutions, such as Mean Curvature Motion. For comparison, we succinctly describe Geodesic Active Contour and Region implementation.

Chapter 3

In chapter 3, we present the recently developed Graph Cut framework [18, 109]. This method has two main advantages: speed and Global optimality for binary segmentation. Indeed, in this chapter segmentation is not defined by regions interfaces but by pixel labeling. This leads to a Markov Random

Fields formulation. Then we review the existing minimizing method for MRF problems.

We present then the class of MRF energies that Graph Cut can minimize optimally. We also describe how Graph Cut minimization process can be used to approximate Multi-Way Cuts, a NP-hard problem.

We then review available Max-Flow/Min-Cut algorithms and discuss their advantages. For comparison, we provide implementations for equivalent problem of Geodesic Active Contour and Region.

Chapter 4

In chapter 4, we present an extension of the Level Sets: The Stochastic Level Sets (SLS), resulting of a join work with Gheorghe Postelnicu. It consists into bringing stochastic motion [90, 91, 92] to Level Set framework. Not all local perturbations are allowed since they have to fit some Level Set constraints. We present some theoretic results settling which stochastic calculus framework should be used for Level Set application. We describe an implementation for the classical Mean Curvature Motion.

Local perturbations of a Level Set allows in a Simulated Annealing framework to converge towards a better minimum. Even if no proof of global optimum is provided, in practice this method proved its usefulness. See chapter 6 for its application (Sec:6.1).

Chapter 5

In chapter 5, we present an extension of the Graph Cuts algorithm, Active Cuts (join work with Yuri Boykov [86, 87]). This new algorithm suits particularly well for Computer Vision problems. It consists into a new Symmetric Push-Relabel formulation. However, it also comes with a bunch of other interesting features such as initialization and some intermediate segmentation encountered during the optimization (local minima).

We then provide some proof of convergence and some properties of the intermediate cuts. We also give another implementation built for speed base don some Dynamic Tree structure.

Chapter 6

In chapter 6, we present some applications of the previously exposed extensions to image segmentation. In particular, we present Stochastic Active Contour [90, 91, 92], an application of the Stochastic Level Sets for region

based segmentation. In practice, it provides better results than standard Geodesic Active Region. We also discuss some other advantages.

Then we address a three regions segmentation problem for Digital Matting initialization [88, 89]. Digital Matting or Alpha Matting stands for extraction from a static image of foreground, background and alpha a mixing factor. Most of the available methods need to be initialized by a Trimap. A Trimap consists into three regions segmentation: Foreground, Background and a region formed of mixed pixels. A specific probability density function has been designed for the mixed region.

This problem is addressed with both Level Set [89] and Graph Cut [88]. We also discuss and compare the two methods.

We then propose some applications of the Active Cuts algorithm for image segmentation [87]. In particular, we show how our new algorithm allows segmenting image by a hierarchical approach without loosing the global optimality.

Chapter 7

In chapter 7, we present here some applications in video segmentation [87]. We briefly present the natural application of the Active Cuts algorithm to video sequence segmentation. Indeed, by using the segmentation result at the previous frame as an initial segmentation, the Active Cuts algorithm shows in this case great performance.

Then we proposed a new Motion Layer approach based on both segmenting and tracking layers in a video [62, 63]. More than extracting layers, we also track them behind each other. This means that we obtain both Visible and Hidden layers of a Video scene.

Introduction (version française)

Place de la segmentation dans la vie quotidienne : critique ?

Les outils de traitement du signal et d'image sont de plus en plus présents dans la vie courante : téléphones portables, DVD, appareil-photo numérique... Le traitement d'image vise principalement à extraire des données signicatives d'une image ou d'une vidéo. Par exemple, la surveillance humaine de larges secteurs est irréalisable convenablement ; les ordinateurs fournissent leur puissance de calcul et permettent l'analyse en temps réel de vidéos de surveillance et la détection des situations à risque. Le traitement d'image est aussi de plus en plus employé pour des applications médicales. Il peut être employé pour la correction d'images bruitées (image d'échographies) ou la segmentation de volume pour une modalité d'acquisition donnée. Le recalage d'image ou de volume de différentes modalités est également un défi considérable puisque les différentes modalités d'acquisition révèlent des informations différentes. Récemment, La communauté de traitement d'image s'est engagée dans l'assistance au diagnostic des maladies et aux opérations chirurgicales.

Dans la vie quotidienne, nous utilisons les outils de traitement d'images sans le savoir. En effet, la compression d'image ou de vidéo fait parti intégrante de produits communs : DVD, téléphone portables, appareils-photo numériques, vidéo-conférences... Nous peut également citer la technique de filigrane numérique qui a été développée pour protéger les droits d'auteurs pour les images et les vidéos en y insérant de manière irrémédiable et invisible une signature.

Un autre challenge est classification d'image. En effet les bases de données d'image, même pour un photographe amateur, deviennent de plus en plus grandes et la classification manuelle n'est plus utilisable. Cependant, la classification automatique d'image est loin d'être un processus trivial mais le grand nombre d'application le rend encore plus attrayant.

Toutes ces techniques font appel à la segmentation d'image au cours du

processus, la plupart du temps au début. La segmentation représente donc un problème générique qui peut être exprimé de beaucoup de manières différentes. Il peut cependant se définir par la phrase suivante : la segmentation consiste à extraire et marquer les différentes régions d'une image. Beaucoup de critères peuvent être utilisé pour décrire les différents objets d'une image et sont à choisir en fonction l'application désirée.

Ces caractéristiques envisagées sont généralement les intensités, couleurs, textures, formes, mouvements ou même profondeur des objets. Cependant, on peut être amené à concevoir son propre critère selon le problème considéré.

Habituellement le problème est posé comme problème de minimisation d'énergie où l'énergie mesure la qualité d'une segmentation donnée par rapport à l'image considérée. Les énergies sont conçues pour tenir compte des critères choisis. Le problème générique de segmentation de n'importe quelle image donnée est insurmontable puisqu'il devrait tenir compte d'un nombre incroyable de caractéristiques. Cependant, le problème générique peut être divisé en une multitude de problèmes, chacun segmentant sa propre classe restreinte d'images. Puis un processus décideur tel qu'un système expert, prendra en compte chaque résultat afin de produire un procédé générique robuste de segmentation.

Historique de la segmentation

Nous renvoyons le lecteur au livre de référence de vision par ordinateur [133] pour toute méthode citée dans les paragraphes suivants.

Historiquement, la segmentation d'image était basée sur l'extraction locale de caractéristiques telles que les coins ou les jonctions en T. Elles étaient extraites des images en utilisant des méthodes génériques telles que les filtres de Canny (points d'intérêt : coin...) ou transformée de Hough (lignes ou cercles). En supposant que les caractéristiques sont situées sur des frontières des objets, on s'attend à pouvoir reconstruire les contours d'un objet en assemblant des données issues des détecteurs.

Plus tard, des méthodes d'évolution de contours ont été proposées pour obtenir une segmentation sans avoir à réaliser la fastidieuse étape d'assemblage. En outre, cette étape est très peu robuste. Ces nouvelles méthodes consistent en un contour (paramétrique) déformé par rapport au critère choisi : généralement le gradient de l'image. En effet, par définition, les contours des objets correspondent aux zones de gradient élevé de l'image. Cependant, ces méthodes dépendent de la structure locale de l'image et donc elles peuvent être coincées sur des minima locaux. Ce problème a motivé l'utilisation pour déformer le contour de caractéristiques plus globales telles

que la texture de la région, les couleurs de la région...

La méthode des Ensemble de Niveaux ont amené un cadre général pour l'évolution de contours (pour n'importe quelle dimension). Au prix de quelques sacrifices (principalement la vitesse), cette approche résout la plus part des désavantages des méthodes paramétriques. La méthode des ensembles de niveaux a été largement rependue dans la communauté et a été largement utilisée dans toutes sortes d'applications. Puisqu'il a été largement étudié au cours des années, on possède maintenant un large nombre de schémas numériques stables. Cependant, on utilise principalement une méthode standard de descente de gradient pour minimiser les énergies. Comme beaucoup d'énergies ne sont pas convexes, ce processus de minimisation mène à un minimum local.

Récemment, les Contours Actifs Stochastiques ont été proposés pour améliorer les résultats de la minimisation. Les contours actifs stochastiques consistent à introduire le concept de recuit simulé dans le cadre de la méthode des ensembles de niveaux. Même si aucun résultat théorétique n'est disponible concernant la décroissance de la température à appliquer pour obtenir un minimum global, on obtient de réelles améliorations en pratique.

En même temps, les méthodes de champ de Markov aléatoire ont été largement développées mais le manque de bons outils de minimisation a limité leur application. Récemment, de nouvelles méthodes de minimisation basées sur le problème de flot maximal/coupe minimale ont obtenu une grande attention de la part de la communauté de vision par ordinateur. En effet ces algorithmes fournissent un minimum global pour une classe d'énergies couramment employées en la vision par ordinateur en un temps relativement cours.

Organisation

Cette thèse est organisée comme il suit.

Chapitre 1

Dans le chapitre 1, nous passons en revue les approches variationnelles existantes de segmentation basée sur l'évolution de contours. Nous présentons principalement les deux classes principales d'évolution : les évolutions basés Contours [95] et Régions [30, 82]. Nous décrivons également comment une inférence Bayésienne est généralement employée pour tenir compte de n'importe quelle caractéristique de région dans l'évolution du contour.

En particulier, nous présentons les Contours Actifs Géodésiques [26, 27, 98] et les Régions Actives Géodésiques [134, 136, 83] pour un modèle paramétrique de contours. Nous discutons rapidement des avantages et des inconvénients et nous rappelons quelques solutions.

Chapitre 2

Dans le chapitre 2, nous présentons la méthode des ensembles de niveaux [59, 60, 132] et comment il convient particulièrement à l'évolution de contours. Nous rappelons quelques schémas numériques. Nous présentons rapidement quelques astuces d'implémentation pour accélérer la méthode puis nous discutons brièvement des avantages (principalement la gestion intrinsèque du changement de topologie et des schémas numériques stables) et des inconvénients (principalement la lenteur).

Nous décrivons également quelques évolutions classiques de courbes, comme le mouvement de courbure moyenne. Pour comparaison, nous décrivons succinctement l'implémentation des Contours et Régions Actifs Géodésiques.

Chapitre 3

Dans le chapitre 3, nous présentons le cadre récemment développé des Coupe de Graphe [18, 109]. Cette méthode a deux avantages principaux : vitesse et optimum global pour la segmentation binaire. En effet, dans ce chapitre la segmentation n'est pas définie par les interfaces entre les différentes régions mais par un marquage des pixels. Ceci mène directement à une formulation par champs de Markov aléatoires. Puis nous passons en revue les méthodes existantes de minimisation pour des problèmes MRF.

Nous présentons alors la classe des énergies MRF que les coupes de graphe peuvent minimiser de façon optimale. Nous décrivons également comment la méthode de minimisation par coupes de graphe peut être employée pour approcher les "Multi-Way Cuts", un problème NP-complet.

Nous passons en revue alors les algorithmes existants de flot maximal/coupe minimale et discutons de leurs avantages. Pour comparaison, nous décrivons comment les coupes de graphe peuvent résoudre les problèmes de Contours et Régions Actifs Géodésiques.

Chapitre 4

Dans le chapitre 4, nous proposons une extension de la méthode des ensembles de niveaux : les ensembles de niveaux stochastiques (SLS), résultant d'une collaboration avec Gheorghe Postelnicu. Cela consiste dans l'ajout de

mouvements stochastiques [90, 91, 92] au cadre des ensembles de niveaux. Toutes les perturbations locales ne sont pas permises puisqu'elles doivent vérifier quelques contraintes venant de la méthode des ensembles de niveaux. Nous présentons quelques résultats théoriques nous permettant de fixer de manière définitive le cadre de calcul stochastique adéquat à une utilisation pour la méthode des ensembles de niveaux. Nous décrivons une implémentation pour le mouvement par courbure moyenne.

Les perturbations locales d'un ensemble de niveau permettent dans un cadre de recuit simulé de converger vers un meilleur minimum. Même si aucune preuve quant à l'obtention d'un optimum global n'est fournie, dans la pratique cette méthode a prouvé son utilité. Voir le chapitre 6 pour son application (Sec :6.1).

Chapitre 5

Dans le chapitre 5, nous proposons une extension d'un algorithme de coupes de graphe, les Coupes de Graphe Actives (issu d'un travail commun avec Yuri Boykov [86, 87]). Ce nouvel algorithme convient particulièrement bien pour des problèmes de vision par ordinateur. Il consiste en une nouvelle formulation symétrique du "Push-Relabel". Cependant, il vient également avec un groupe d'autres propriétés intéressantes telles que l'initialisation ou encore le fait qu'il produise une succession de la segmentation intermédiaire pendant l'optimisation (minima locaux).

Nous fournissons alors une preuve de convergence et quelques propriétés des coupes intermédiaires. Nous donnons également une autre implémentation plus rapide grâce à l'utilisation d'une structure arborescente dynamique.

Chapitre 6

Dans le chapitre 6, nous présentons quelques applications à la segmentation d'image des extensions précédemment exposées. En particulier, nous présentons les Contours Actifs Stochastiques [90, 91, 92], une application des ensembles de niveaux stochastiques pour la segmentation basée régions. Dans la pratique, il fournit de meilleurs résultats que les régions actives géodésiques standards. Nous discutons aussi de quelques autres avantages.

Puis nous nous intéressons au problème de segmentation par trois régions pour l'initialisation du "Digital Matting" [88, 89]. Le Digital Matting ou Alpha Matting consiste en l'extraction du premier plan, du fond et d'alpha un facteur de mélange à partir d'une image statique. La plupart des méthodes disponibles doivent être initialisées par une Trimap. Une Trimap consiste en une segmentation par trois régions : premier plan, fond et une région formée

des pixels mélangés. Une densité de probabilité spécifique a été conçue pour la région mélangée.

Ce problème est traité à la fois avec la méthode des ensembles de niveaux [89] et par coupes de graphe [88]. Nous discutons rapidement de la comparaison des deux approches.

Nous proposons aussi quelques applications de l'algorithme des coupes de graphe actives pour la segmentation d'image [87]. En particulier, nous montrons comment notre nouvel algorithme permet segmenter une image par une approche multi-échelle sans perdre le l'optimal global.

Chapitre 7

Dans le chapitre 7, nous présentons quelques applications de nos méthodes à la segmentation vidéo [87]. Nous présentons brièvement l'application évidente de l'algorithme des coupes de graphe actives à la segmentation vidéo. En effet, en employant le résultat de la segmentation à l'image précédente comme segmentation initiale, les coupes de graphe actives fournissent de très bonnes performances.

Puis nous proposons une nouvelle approche de segmentation par extraction de couches par le mouvement dans une vidéo [62, 63]. Bien plus que l'extraction des couches, nous les traquons également l'une derrière léautre. Ceci signifie que nous obtenons à la fois des couches visibles et cachées d'une scène vidéo.

Chapter 1

Active Contours

In this chapter, we present different standard approaches based on Active Contours. We first review existing approaches based on a contour formulation Sec:1.1. Indeed the first attempt for image segmentation relied on the following assumption. Object silhouette matches high image gradient. Based on this remark, some segmentation methods were proposed, mainly the Snake model (Sec:1.1.1) and the Geodesic Active Contour framework (Sec:1.1.2).

Historically, the next methods were based on region descriptions of objects (color, texture, motion,...) (Sec:1.2). This leads to a generalization of Snake (Sec:1.2.1) to regions and Geodesic Active Regions (Sec:1.2.3) a generalization of the Geodesic Active Contour formulation.

We then recall a result on Geodesic Active Regions where the descriptors depend on the segmentation (Sec:1.2.4). Indeed in that case, the energy gradient also depends on the descriptors gradients. This is particularly useful to incorporate shape prior into a segmentation process (Sec:1.3).

1.1 Contour based

Objects are usually defined by their boundaries or silhouette. It is commonly assumed that object boundaries have to coincide with high gradient magnitudes in the image. The problem can be addressed in an energy based framework in order to access to regular minimizing techniques such as gradient descent :

$$\arg\min_{\Gamma\in\mathcal{C}} E(\Gamma) = \arg\min_{\Gamma\in\mathcal{C}} -\int_0^1 \left|\nabla I\left(\Gamma\left(p\right)\right)\right|^2 dp \tag{1.1}$$

where $I : \Omega \in \mathbb{R}^2 \rightarrow \mathbb{R}^N$ is an image, $\Gamma : [0,1] \rightarrow \Omega$ is a parameterized contour and $\mathcal{C}$ is a given class of contours. Usually $\mathcal{C}$ corresponds to genus one closed contours but it could be more or less restrictive.

1.1.1 Snake Model

The *Snake* model makes use of internal and external forces to drive a contour (or snake) towards features of interest (usually object boundaries). Kass, Witkin and Terzopoulos gave its first formulation in [95] as an energy minimization problem. The snake functional energy is defined over the contour Γ as:

$$E(\Gamma) = \alpha \int_0^1 |\Gamma'(p)|^2 \, dp + \beta \int_0^1 |\Gamma''(p)|^2 \, dp - \lambda \int_0^1 |\nabla I(\Gamma(p))|^2 \, dp \quad (1.2)$$

The internal properties are embedded in the first two terms while the third one codes for external forces. In that case, it's assumed that object boundaries lie on high image gradients. The internal forces stand for a compromise between rigidity and elasticity of the curve while the external force attempts to lock the contour on object boundaries.

An obvious limitation of this model lies in the data term since one has to assume images have a constant contrast over the domain Ω. A lot of extensions and generalizations can be found in the literature to avoid this problem. By introducing in the data term a new decreasing function g from $\mathbb{R}^+$ into $\mathbb{R}^+$ with $\lim_{x \to +\infty} g(x) = 0$, it becomes possible to increase the robustness and/or be sensitive to other features:

$$E_{data}(\Gamma) = \lambda \int_0^1 g\left(|\nabla I(\Gamma(p))|\right)^2 dp \quad (1.3)$$

This formulation presents two main drawbacks: the functional depends on the curve parameterization and its topology cannot change over the evolution. If more than one object is present, it will be impossible to segment them with only one snake. Those problems have been widely discussed in the literature. However if the curve parameterization is not an insuperable difficulty and can be addressed by using different function basis like B-splines [50], the topology change is a more complex issue and cannot be achieved without a computationally expensive and complex implementation [122].

Another issue as to be considered, the minimization process was firstly addressed with a gradient descent method. Such a tool is strongly dependent on the initialization and the user has to provide an accurate initialization in order to retrieve a good minimum. Moreover, the energy in the equation (1.2) refers to the second derivative of the contour, which means that its *Euler-Lagrange* formulation will depend on the fourth derivative :

$$\frac{\partial \Gamma}{\partial t} = -\frac{\partial E}{\partial \Gamma} = -\alpha \frac{\partial^2 \Gamma}{\partial p^2} - \beta \frac{\partial^4 \Gamma}{\partial p^4} + \nabla E_{data} \quad (1.4)$$

Evaluating such a derivative on a discrete space always generates instabilities and inaccurate approximations. In order to avoid this problem, one can use B-splines approximation or some other minimization techniques like dynamic programming.

By considering only the internal forces, one could easily see that every curve tends to shrink and vanish. To fix this inclination, L. Cohen in [43] introduced a "*balloon force*" - a pressure force - as an antagonistic force. The *balloon force* could be seen as an inner area constraint.

$$E(\Gamma) = \underbrace{\alpha \int_0^1 |\Gamma'(p)|^2 \, dp + \beta \int_0^1 |\Gamma''(p)|^2 \, dp}_{\text{internal forces}} + \underbrace{\mu \int_{\Omega_{int}} dx}_{\text{balloon force}} + \underbrace{\lambda \int_0^1 g\left(|\nabla I\left(\Gamma(p)\right)|\right)^2 dp}_{\text{data term}} \tag{1.5}$$

Where Ω_{int} is the interior of the contour and the sign of μ allows to select between a shrink or an expand profile of the contour. Although this extension increases the robustness of snakes with respect to initialization, images need stronger features to overcome internal and balloon forces.

Moreover in this approach, only local aspects of the contour can be considered: curvature, length and local image structure underneath the curve. Due to those aspects, Snakes are very sensitive to local minima. The first attempt to avoid this problem was to use an image convoluted by Gaussian kernel. However if most of local minima are avoided, it results in a lost of precision of contour localization. This directly leads to a hierarchical formulation (convolution with smaller and smaller kernels).

Nevertheless if one has or can guess a good initial contour, Snake provides real time segmentation and can be successfully used (ex: for video tracking).

1.1.2 Geodesic Active Contour

Geodesic Active Contour was firstly designed to reconcile Snakes with geometric definition of contours. The main idea is to reformulate equations in an equivalent intrinsic way in order to escape the parameterization dependency. Caselles et al. in [26, 27] and Kichenassamy et al. in [98] proposed a similar formulation using different approaches.

Geodesic active contour model is a particular case of the snake model : by relaxing constraints on the contour - i.e. no rigidity constraint ($\beta = 0$) - the model becomes to :

$$E(\Gamma) = \underbrace{\alpha \int_0^1 |\Gamma'(p)|^2 \, dp}_{\text{smoothness}} + \underbrace{\lambda \int_0^1 g\left(|\nabla I\left(\Gamma(p)\right)|\right)^2 dp}_{\text{data term}} \tag{1.6}$$

This function aims at location a smooth curve following edges (high gradient spots or more generally g minimum spots).

In order to switch to an intrinsic framework, some hypothesis about the feature functional g are needed. Under the assumption that g is a strictly decreasing positive and asymptotically vanishing function $g(x) \to 0$ when $x \to \infty$, Caselles, Kimmel and Sapiro in [28] proved equation (1.6) minimization is equivalent to minimize the following equation :

$$E(\Gamma) = \int_0^1 g\left(\left|\nabla I\left(\Gamma\left(p\right)\right)\right|\right)\left|\Gamma'\left(p\right)\right| dp \tag{1.7}$$

where the two parts - smoothness and data term - can be easily recognized. By re-parameterizing the curve by its arc length, equation (1.7) becomes the so known ***Geodesic Active Contour*** formulation :

$$E(\Gamma) = \int_0^L g\left(\left|\nabla I\left(\Gamma\left(s\right)\right)\right|\right) ds \tag{1.8}$$

with the infinitesimal curve length $ds = \left|\Gamma'\left(p\right)\right| dp$ and L is the Euclidian length of the curve $\Gamma(s)$. According to equation (1.8), the segmentation problem comes down to a minimal path search weighted by a *metric* $g\left(\left|\nabla I\right|\right)$ given by the image.

This new functional is minimized using gradient descent methods. Thus a (local) minimum is attained by deforming, step by step, using the Euler-Lagrange equations (see [28, 99, 98] for details) :

$$\frac{\partial \Gamma}{\partial t} = g\left(\left|\nabla I\right|\right)\kappa\overrightarrow{\mathbf{n}} - \left(\nabla g\left(\left|\nabla I\right|\right)\cdot\overrightarrow{\mathbf{n}}\right)\overrightarrow{\mathbf{n}} \tag{1.9}$$

where $\overrightarrow{\mathbf{n}}$ is the inward Euclidian unit normal vector of the curve and $\kappa = div\left(\frac{\nabla\Gamma}{|\nabla\Gamma|}\right)$ its Euclidian curvature. The first term only adjusts the curve where the curvature is not null $\kappa \neq 0$ (non straight lines) or where the curve is not on top of edges $g\left(\left|\nabla I\right|\right) \neq 0$ and can be seen as an attracting force towards edges. The second term in only effective around image boundaries $\nabla g\left(\left|\nabla I\right|\right) \neq 0$ and can be seen as a refinement force as it centralizes the curve on boundaries and attempts to overcome the smoothing effect of the curvature. Indeed, an extreme case where $g \equiv 1$ leads to the so known ***Mean Curvature Motion*** equation :

$$\frac{\partial \Gamma}{\partial t} = \kappa\overrightarrow{\mathbf{n}} \tag{1.10}$$

In that special case, the curve is firstly deformed to a circle and then tends to shrink and vanish. The first part of the evolution reveals the smoothing

profile due to the curvature. The contraction and disappearance of the curve, as for the snake model, is due to the decrease of the perimeter of a circle with its radius.

The shrinking effect can be overcome by adding a *balloon force* as in Snake model so the evolving equation is easily transformed into :

$$\frac{\partial \Gamma}{\partial t} = g\left(|\nabla I|\right)(\kappa + \mu)\,\overrightarrow{\mathbf{n}} - \left(\nabla g\left(|\nabla I|\right) \cdot \overrightarrow{\mathbf{n}}\right)\overrightarrow{\mathbf{n}} \tag{1.11}$$

where μ is the balloon force. If $\mu < 0$, the balloon force counteracts the shrinking tendency while $\mu > 0$ boosts it.

Geodesic Active Contours are usually addressed in a Level Set framework (see Sections Sec:2 and Sec:2.5.2 for details).

1.2 Region based

The previous discussed Snake model and Geodesic Active Contour (see Sections Sec:1.1.1 and Sec:1.1.2) suffer both from some drawbacks such as *change of topology*. However their main drawback is due to their gist: those methods only focus on local information (boundary based) to extract contours. This drawback leads to two disadvantaging consequences: Firstly, the evolving interface gets stuck more easily on local minima. Secondly, as a consequence, the user may provide a *good* initial curve.

In order to take into account global information, one should no more consider a curve just as an interface but as an interface that separate two regions.

1.2.1 Region Snake

One of the first attempt to conciliate interface and region component in a unique framework was due to Chakraborty, Staib and Duncan in [30]. They proposed to minimize energy similar to:

$$E(\Gamma) = E_{shape}(\Gamma) + \alpha \int_0^1 |\nabla I(\Gamma(p))| dp + \beta \int_{\Omega_{in}} Reg(I(\omega))\, d\omega \tag{1.12}$$

where Ω_{in} stands for the interior region delimited by Γ. This region is often referred as the *object* region or just the *object*. The first term in (1.12) stands for Shape prior constraint. In fact, the evolving curve Γ_t is driven towards a (predefined) reference curve Γ_{prior} and should remain close to it. The second term is a classical high gradient (or boundary) attractor term. The third

term is an attempt to add interior region constraint: here, homogeneous intensity constraint.

Such an energy has some limitations and drawbacks. The region constraint is quite restrictive while it is not a very strong constraint. Moreover one could easily see that the model lacks of symmetry: there is no constraint for the outer part of the contour. Although this model is the first attempt to incorporate boundary and region constraints at the same time.

A more general idea based on Snake model was proposed by Ivins and Porrill in [82]. This new model takes advantage of colors and texture for partitioning the domain in two parts. Authors proposed to use a pressure force - like the balloon force (see Section Sec:1.1.1) - in order to drive the curve with regional information. In that case the pressure force is no more constant but depends on statistical properties of the inner part of the contour Ω_{in}.

$$E(\Gamma) = \underbrace{\int_0^1 |\Gamma'(p)|^2 dp + \alpha \int_0^1 |\Gamma''(p)|^2 dp}_{\text{Internal forces}} + \beta \underbrace{\int_{\Omega_{in}} g(I(\omega))\, d\omega}_{\text{Regional Term}} \tag{1.13}$$

One could easily recognize the internal forces formulation of Snake model in the two first terms of the above equation. If this model makes use of global information (using the functional g), it does it quite partially since there is no use of the outer part Ω_{out}. Moreover it does not make use of boundary information at all. To finish, in order to initialize, the user has to provide a consistent seed region as representative as possible of the whole object. This seed region will be then used to build the region functional g which will expand or shrink the contour when image pixels match or mismatch the seed's information.

Zhu and Yuille in [184] proposed an alternative to those models. This model is also inspired by the Snake model and can also deals with color and texture segmentation. Its energy can be written as follow:

$$E(\Gamma) = \int_0^1 |\Gamma'(p)|^2 dp + \alpha \left[\int_{\Omega_{in}} \log P_{in}(I(\omega))\, d\omega + \int_{\Omega_{out}} \log P_{out}(I(\omega))\, d\omega \right] \tag{1.14}$$

As one can see, this model does not take into account any boundary information. The first term stands for smoothness of the curve and the second codes for likelihood of the inner region with respect to the inner probability density function P_{in}. The third term is similar to the second but for outer region. If this model takes correctly into account both side of the contour, it neglects the boundary information.

Based on those work [30, 82, 184], Geodesic Active Region Sec:1.2.3 was proposed as an elegant alternative that makes use of both boundary and region information. As its name indicates, this model is also inspired by Geodesic Active Contour (see Section Sec:1.1.2). However before giving it a short description, one should be more familiar with Bayesian calculus.

1.2.2 Bayesian Formulation

This section describes an elegant formulation based on probability and Bayesian inference [139] to deal with regional information: intensity, color, texture, shape prior... One should be aware that in the whole section, we will make a misuse of language by considering equivalent a segmentation/partition and a contour which separates the domain in two parts. The main idea is to rewrite the posterior probability of the segmentation Γ knowing the image I using the Bayes rule:

$$P(\Gamma|I) = \frac{P(I|\Gamma)P(\Gamma)}{P(I)} \tag{1.15}$$

The left part of the equation could be interpreted as follow: In order to obtain the optimal contour (or segmentation), one should maximize the posterior probability with respect to Γ to extract the best interface for a given image. It can be remarked that for a given image I, the probability $P(I)$ is constant. So equation (1.15) could be rewritten as:

$$P(\Gamma|I) \propto P(I|\Gamma)P(\Gamma) \tag{1.16}$$

In the right part of equation (1.16), the first term $P(I|\Gamma)$ stands for the posterior probability of I given the segmentation Γ while the second term stands for the prior probability of the contour Γ. Maximizing the functional in (1.16) with respect to the contour Γ is completely equivalent to minimizing its loglikelihood:

$$-\log P(\Gamma|I) = -\log P(I|\Gamma) - \log P(\Gamma) \tag{1.17}$$

The first term on the right part *measures* how good the image distribution is given the contour Γ. This one allows inserting any kind of desired constraint into the model: intensity, color, texture... The second one allows inserting prior knowledge on the curve. Commonly this term stands for curve smoothness using the constraint $P(\Gamma) = exp(-\alpha Length(\Gamma))$. Although it can be a more advanced criterion such as Shape Similarity. See initiated work of Chakraborty, Staib and Duncan in [30] which makes use of shape constraint.

1.2.3 Geodesic Active Region

Based on encouraging work of [30, 82, 184], *Geodesic Active Regions* were firstly introduced by Paragios and Deriche in [134, 136] for texture segmentation, then for image segmentation [138] and tracking [137, 135]. As in *Geodesic Active Contour* (see Section Sec:1.1.2), it makes use of boundary information in an intrinsic framework and can be seen as an extension of *Geodesic Active Contour* that incorporates region information quite elegantly.

Boundary Term: This term is quite similar to *Geodesic Active Contour* and is rewritten here for simplicity:

$$E_B(\Gamma) = \int_0^L g(P_b(I(\Gamma(s))))|\Gamma'(s)|\, ds \tag{1.18}$$

where L is the Euclidean length of the contour Γ and Γ is parameterized by its arc length. P_b stands for boundary probability and g is a Gaussian function. This energy aims at finding the shortest closed curve according the "*metric*" $g(P_b(I(\Gamma(s))))$. Here the boundary probability should be design to extract desired boundary features. For more details, see Section Sec:1.1.2.

Region Term: This new term aims at finding a contour that best fits Ω_{in} and Ω_{out} with respect to two posterior probability of $I(\Omega_{in})$ and $I(\Omega_{out})$ given a contour Γ. Here the prior probability $P(\Gamma)$ in equation (1.16) is assumed to be constant and equal to $\frac{1}{Z}$ where Z is the total number of possible contours. This assumption means that each contour is equiprobable. This leads to the model:

$$P(\Gamma|I) \propto P(I|\Gamma) \propto P_{in}(I(\Omega_{in}))P_{out}(I(\Omega_{out})) \tag{1.19}$$

Then the regional term can be now written as follow:

$$E_R(\Gamma) = -\int_{\Omega_{in}} \log P_{in}(I(\omega))\, d\omega - \int_{\Omega_{out}} \log P_{out}(I(\omega))\, d\omega \tag{1.20}$$

For a better understanding of the influence of the region term, let consider its action on a pixel p. The classification rule depends on the ratio $r(p) = \frac{P_{in}}{P_{out}}(I(p))$. If $r(p) > 1$ then the pixel p tends to belong to the inner part of the contour. Similarly if $r(p) < 1$ then the pixel p tends to belong to the outer part. However if $r(p) = 1$, the two classes are equiprobable and then it is impossible to distinguish which class it should be assigned to.

Overall Energy: Then the overall energy is given by:

$$E = \alpha E_B + (1-\alpha)E_M = \alpha \overbrace{\int_0^L g(P_b(I(\Gamma(s))))|\Gamma'(s)|\,ds}^{\text{Boundary Term}} + (1-\alpha)\underbrace{\left[-\int_{\Omega_{in}} \log P_{in}(I(\omega))\,d\omega - \int_{\Omega_{out}} \log P_{out}(I(\omega))\,d\omega\right]}_{\text{Region Term}} \quad (1.21)$$

where the value $\alpha \in [0;1]$ adjusts influence between Boundary and Region terms.

Those two terms are antagonist since the boundary term aims at finding a smooth curve of minimal length according to the image structure and the region term tries to find two regions that fit the best the probabilities P_{in} and P_{out}.

The functional (1.21) is minimized using gradient descent method and one can obtain the following Euler-Lagrange formulation for a given point p of the curve Γ using the integration by parts formula:

$$\frac{\partial p}{\partial t} = \alpha(g(P_b(I(p)))\kappa(p) - \nabla g(P_b(I(p))) \cdot \overrightarrow{\mathbf{n}}(p))\overrightarrow{\mathbf{n}}(p) - (1-\alpha)(\log P_{in}(I(p))\overrightarrow{\mathbf{n}}_{in}(p) + \log P_{out}(I(p))\overrightarrow{\mathbf{n}}_{out}(p)) \quad (1.22)$$

However one should consider the normal orientation constraint $\overrightarrow{\mathbf{n}}(p) = \overrightarrow{\mathbf{n}}_{in}(p) = -\overrightarrow{\mathbf{n}}_{out}(p)$ and then the Euler-Lagrange formulation could be rewritten as:

$$\frac{\partial p}{\partial t} = \left[\alpha \underbrace{(g(P_b(I(p)))\kappa(p) + \nabla g(P_b(I(p))) \cdot \overrightarrow{\mathbf{n}}(p))}_{\text{Boundary component}} + (1-\alpha)\underbrace{\log\left(\frac{P_{out}(I(p))}{P_{in}(I(p))}\right)}_{\text{Region component}}\right]\overrightarrow{\mathbf{n}}(p) \quad (1.23)$$

One could easily recognize the previously defined ratio $r(p)$ inside the Region component. If the pixel underneath the considered point p belongs to the outer region, then $r(p) = \frac{P_{out}}{P_{in}}(I(p)) > 1$ and consequently $\log r(p) = \log \frac{P_{out}}{P_{in}}(I(p)) > 0$ which implies a region force in the inward normal direction. Similarly if the point p belongs to the inner region, then the region force will point in the opposite direction.

Caselles, Kimmel and Sapiro proposed in [27] a modified equation by adding $\lambda\kappa(p)\overrightarrow{\mathbf{n}}(p)$ in order to enforce more regularity to the curve:

$$\frac{\partial p}{\partial t} = \Bigg[\alpha \underbrace{\left(g(P_b(I(p)))\kappa(p) + \nabla g(P_b(I(p))) \cdot \overrightarrow{\mathbf{n}}(p)\right)}_{\text{Boundary component}} + (1-\alpha) \underbrace{\log\left(\frac{P_{out}(I(p))}{P_{in}(I(p))}\right)}_{\text{Region component}} + \lambda \underbrace{\kappa(p)}_{\text{Regularity}} \Bigg] \overrightarrow{\mathbf{n}}(p) \quad (1.24)$$

where $lambda \in [0;1]$. The role of mean curvature motion equation is to counterbalance the region term and avoid it to rip up the curve.

Since the Level Set method is more robust and combine some advantages, Geodesic Active Regions are usually implemented using this framework (see Sections Sec:2 and Sec:2.5.3).

1.2.4 DREAM^{2}S

The Paragios and Deriche model [134, 136] only considers fixed or *Static* descriptor functions. Indeed, while curve evolution, the descriptors functions should not be changed. This implies that we have to have access to descriptors that represent the object and the background prior to the segmentation (1.25).

$$E(\Gamma) = \int_{\Omega_{in}} k_{in}(\omega)\, d\omega + \int_{\Omega_{out}} k_{out}(\omega)\, d\omega + \int_{\Gamma} k_b(s)\, ds \quad (1.25)$$

where k_{in}, k_{out} and k_b are respectively the descriptors for the objects, the background and the boundaries.

However, this prior knowledge is often inaccessible. In that case, one usually consider dynamic functions for describing the regions based on the current segmentation like (1.26). Those descriptors are updated at each step of the evolution [24, 47].

$$E(\Gamma) = \int_{\Omega_{in}} k_{in}(\omega, \Omega_{in})\, d\omega + \int_{\Omega_{out}} k_{out}(\omega, \Omega_{out})\, d\omega + \int_{\Gamma} k_b(s)\, ds \quad (1.26)$$

In that typical case, Jehan-Besson, Barlaud and Aubert in [83] made the following relevant remark. If the descriptors depend on the segmentation, then the gradient of the energy should also depend on the derivative of those descriptor functions.

Since, derivatives with respect to domains cannot be computed, one introduces a dynamical scheme that links continuously an evolution parameter τ to each domain. Then (1.26) becomes:

$$E(\tau) = \int_{\Omega_{in}(\tau)} k_{in}(\omega, \tau)\, d\omega + \int_{\Omega_{out}(\tau)} k_{out}(\omega, \tau)\, d\omega + \int_{\Gamma(\tau)} k_b(s)\, ds \quad (1.27)$$

Thanks to this formulation, one could now compute the derivative of E with respect to τ and obtain the following result:

$$\frac{\partial E(\tau)}{\partial \tau} = F\left(k_{in}, k_{out}, k_b, \frac{\partial k_{in}}{\partial \tau}, \frac{\partial k_{out}}{\partial \tau}, \frac{\partial k_b}{\partial \tau}\right) \quad (1.28)$$

where F is a linear application.

Moreover by a complex process described in [83], the authors provide a complete formulation of the gradient. Actually, it amounts to adding a new correcting term to the gradient of Nikos and Deriche (1.23). Indeed this new term corresponds to the dependency term of the energy with respect to the derivative of the descriptors.

Now that the gradient is corrected, the minimizing process (gradient descent) leads to a real (local) minimum. Indeed using a corrupted gradient does not lead to a minimum of the considered energy. Another method is proposed in section Sec:4 that allows to minimize an energy based only on an estimation of the gradient.

However computing this additive term is not an easy process. Jehan-Besson, Barlaud and Aubert in [83] give a formulation when the descriptors depend continuously on the mean of the pixels intensities or on the determinant of the color covariance. In [72], they provide a similar result in case of shape inference.

1.2.5 The Mumford-Shah fonctional

The weak membrane model [126] introduced by Mumford and Shah, assumes that the image can be approximated by a piece-wise smooth image. In [126, 127], they proposed in order to extract the "closer" representation u of the image I by minimizing the following functional:

$$E(u, \Gamma) = \int_{\Omega} |u(\omega) - I(\omega)|^2 d\omega + \lambda \int_{\Omega \setminus \Gamma} |\nabla u(\omega)|^2 d\omega + \mu\, length(\Gamma) \quad (1.29)$$

This equation can be interpret as follow: the first term is the standard data term (mean square sum), while the third one corresponds to the classical smoothing term (the length of the interface between the regions). The second

term is the key point of the model. Indeed it codes for the smoothness of the function u everywhere excepts on the contours. If we assume u to be initialized to I, it will evolve in time toward a smoother version of I while it remains as close as possible to I. In fact, this evolution scheme corresponds to the projection of I in the piece-wise constant smooth images. One could easily link this approach to the total variation minimization functional [154].

A simpler approach proposed by Chan and Vese [33] based on [127] assumes that u should be piece-wise constant during the whole process. By adding this constraint, they could rewrite the previous equation of Mumford and Shah (1.29) as follow:

$$E(u) = \sum_{i=1}^{n} \int_{\Omega_i} |c_i - I(\omega)|^2 d\omega + \lambda \, length(\partial\Omega_i) \tag{1.30}$$

Where $\Omega_i, i = 1..n$ is a partition of the whole domain Ω such that $u(\omega) = c_i \in \mathbb{R}$ if $\omega \in \Omega_i$. The second term in (1.29) vanishes since the function u is piece-wise smooth by construction. One could easily see the relationship between this two formulation of the same problem.

In [32, 33, 31], the authors propose a Level Sets (see Sec:2 implementation of this evolution scheme. For more detail, we refer to those articles.

1.3 Conclusion and discussion

Recent developments in segmentation tend more and more to incorporate multiple region features [24, 47] such as: color histograms, textures, motion models, shapes...

Many recent works are focused on integration of shape constraint based on prior knowledge [35, 49, 83, 112, 141]. Indeed human visual system makes a large use of prior knowledges. Shape or structure constraints on objects are a strong information and allows segmenting objects when they are partially occluded or in presence of strong noise.

The next step is obviously to deal with multiple shapes (from a database). Indeed using a single shape constraint is too restrictive since it allows retrieving only one silhouette. This needs a very strong prior knowledge on the considered image. However, considering multiple shapes allows recognizing objects of different classes or of same class.

In order to deal with multiple shapes, standard approaches make use of shape statistics [34, 46, 47, 151] in a Bayesian Active Contours framework.

Another active domain on segmentation takes advantage of the following assumption: motion should be consistent inside a region. This leads

to the general framework of Motion Layer Extraction [6, 10, 11, 13, 48, 64, 97, 110, 129, 160, 172, 173]. Some new interesting results are presented in section Sec:7.2.

Chapter 2

Level Sets

This section broached the general problem of evolving a closed initial curve/surface Γ_0 driven by the movement β in the direction of its inward normal vector $\overrightarrow{\mathbf{n}}$:

$$\begin{cases} \frac{\partial \Gamma(t)}{\partial t} = \beta \overrightarrow{\mathbf{n}} \\ \Gamma(0) = \Gamma_0 \end{cases} \tag{2.1}$$

The normal part of movement alone counts and the tangential movement could be neglected since the evolution takes into account only curve correspondence and not point correspondences. In fact the point correspondences is lost along the evolution nevertheless point correspondences can be kept using [144].

The Level-set method [59, 60, 132] provides an alternative to the Snake model and deal elegantly with change of topology. Moreover it solves some main drawbacks of the Snake model: stability of numerical scheme, parameterization...

2.1 Principle

The basic idea is to consider a curve as a zero level of a level-set function u defined over the whole domain Ω. Then the evolution of a curve is replaced by a corresponding evolution of the level-set function. If this replacement introduces more computation by increasing the dimensionality of the problem, it also has some advantages. Indeed the curve embedded in a level-set function is represented in an implicit and intrinsic framework. One advantage of an implicit representation of a curve is that it no more depends on some parameterization. *Implicit representation:* The level-set function u has

to be Lipschitz on the domain.

$$\begin{array}{rccc} u: & \Omega \times \mathbb{R}^+ & \longrightarrow & \mathbb{R} \\ & (\omega, t) & \longmapsto & u(\omega, t) \end{array} \tag{2.2}$$

with $u(\Gamma_t, t) = 0$ and $\Gamma_t = \Gamma(t)$ and where the Lipschitz property on the domain Ω is independent of t:

$$\exists K,\, \forall t \in \mathbb{R}^+,\, \forall x, y \in \Omega \quad |u(x,t) - u(y,t)| \leqslant K \left\| x - y \right\| \tag{2.3}$$

According to this definition, the curve at time t could be retrieved by slicing at level zero of the function u at time t (with the simplification $u(\omega, t) = u_t(\omega)$):

$$\Gamma_t(\omega) = \{\omega \in \Omega,\ u(\omega, t) = 0\} = u_t^{-1}(\{0\}) \tag{2.4}$$

A common choice for the level-set function is the signed distance function to the curve (negative inside, positive outside). This choice is everything but innocent since it provides nice expressions of geometric properties of the curve.

$$u(\omega, t) = d(\omega, \Gamma_t)(\mathbb{1}(\Omega_{out}) - \mathbb{1}(\Omega_{in})) = \begin{cases} 0, & \omega \in \Gamma_t \\ d(\omega, \Gamma_t) > 0, & \omega \in \Omega_{out} \\ -d(\omega, \Gamma_t) < 0 & \omega \in \Omega_{in} \end{cases} \tag{2.5}$$

where $d(.,.)$ is the Euclidian distance. An example of signed distance is shown in Fig. (2.1). One should mention that the domain is partitioned in two parts using only one level set function and it can be shown [183, 169] that a n parts partition can be achieved using only $\log n$ level set functions.

Intrinsic geometric properties: Geometric properties of the curve could be extracted directly from the implicit function u by nice and easy expressions:

- The inward unit normal vector $\overrightarrow{\mathbf{n}} = \frac{\nabla u}{|\nabla u|}$
- The mean curvature is given by: $\kappa = -div\left(\frac{\nabla u}{|\nabla u|}\right)$

Expressing evolution in Level-Set framework: Now that intrinsic geometric properties of the curve are easily available, one could expressed the evolution formulation in an implicit framework. Indeed the main problem is: How should we evolve an implicit function u in time so that $u_t^{-1}(\{0\}) = \Gamma_t$ follows the evolution defined by (2.1)? The problem was widely discussed in [132].

The constraint contained in the question above could be written as:

$$u\left(\Gamma(t), t\right) = 0 \quad \forall t \geqslant 0 \tag{2.6}$$

Using the chain rule, we obtain the following result:

$$\frac{\partial u}{\partial \Gamma} \cdot \frac{\partial \Gamma}{\partial t} + \frac{\partial u}{\partial t} = 0 \tag{2.7}$$

$$\begin{aligned} \frac{\partial u}{\partial t} &= -\nabla u \cdot \frac{\partial \Gamma}{\partial t} \\ &= -\nabla u \cdot \beta \overrightarrow{\mathbf{n}} \\ &= -\beta \nabla u \cdot \overrightarrow{\mathbf{n}} \\ &= -\beta \nabla u \cdot \frac{\nabla u}{|\nabla u|} \\ \frac{\partial u}{\partial t} &= -\beta |\nabla u| \end{aligned} \tag{2.8}$$

Change of topology: Since the curve in embedded in the level-set function as its zero level and since it is possible to simulate the curve evolution (2.1) by evolving its implicit representation using (2.8), the change of topology of Γ_t do not require any special treatment. Along iterations, u remains a function as long as the speed β is continuous over space. However the zero level of u could change of topology during evolution: the curve could merge or split. See Fig. 2.1 for an illustration.

The Level-Set method provides other advantages such as:

- Numerical schemes are well studied, stable and can be chosen according to the needed precision.
- Extension to higher dimension is straightforward.
- This framework could also deal with tangential motion using a projection on the normal direction. $\frac{\partial \Gamma}{\partial t} = \overrightarrow{\beta}$ is simulated by $\frac{\partial \Gamma}{\partial t} = (\overrightarrow{\beta} \cdot \overrightarrow{\mathbf{n}})\overrightarrow{\mathbf{n}}$ or $\frac{\partial u}{\partial t} = -\overrightarrow{\beta} \cdot \nabla u$.

Recently in [145], Pons et al. provide an elegant method to deal with point correspondences in a Level Set evolution with a tangential component. Due to the projection on the normal direction, point correspondences on contours are corrupted. By tracking them along the evolution using the tangential component, Pons et al. are able to maintain good point correspondences. This is useful when the contour or surface carries some data like a texture. Authors apply this technique to morphing and brain unfolding.

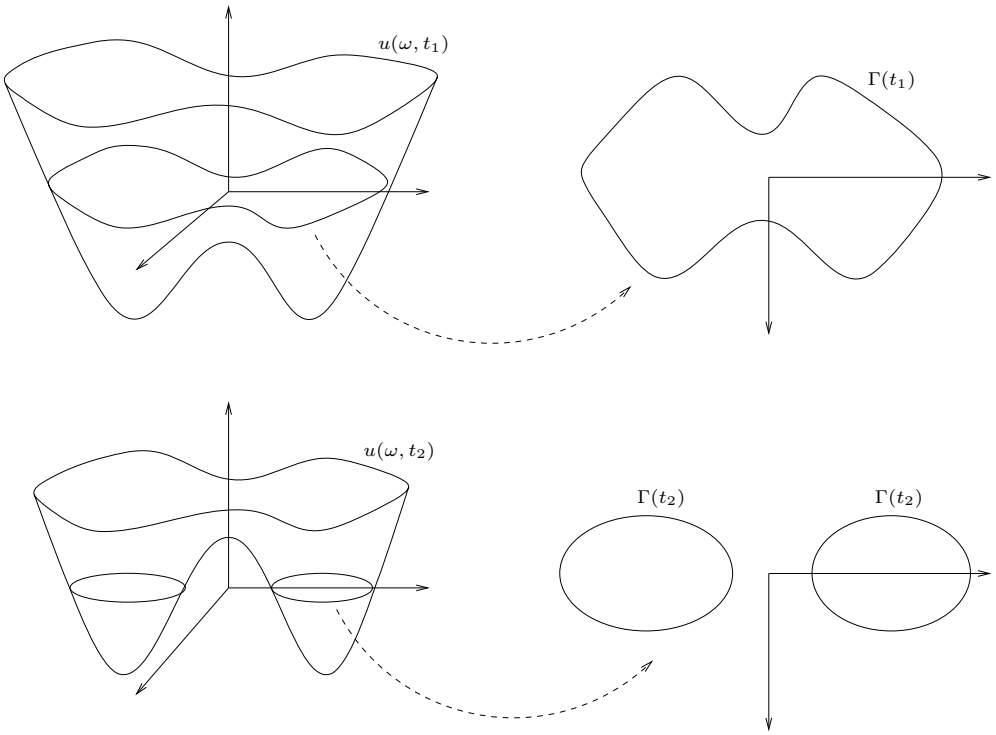


Figure 2.1: Illustration of a topology change. The zero level can, with any particular treatment, be split or merged.

2.2 Numerical Schemes

This section will explain and detail the implementation of first and second derivatives in order to compute the geometric properties of the interface: normal vector, curvature, gradient...

In [159], Sethian describes many numerical schemes and criteria. The reader could find in this book numerical schemes to approximate first and second order derivatives in case of convex *Hamiltonian* (noted H with $H(\nabla u) = \beta|\nabla u|$ in (2.8)). One could also find two numerical schemes in case of non-convex Hamiltonian.

The most common scheme will be described, based on [159], but let start by some notations et definitions. In 2D case, equation (2.8) could be rewritten as follow:

$$\frac{\partial u}{\partial t} + \beta|\nabla u| = \frac{\partial u}{\partial t} + H(u_x, u_y) = 0 \tag{2.9}$$

Let us define some operators:

$$\begin{array}{ll} D_{ij}^{x} u = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, & D_{ij}^{y} u = \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \\ D_{ij}^{+x} u = \frac{u_{i+1,j} - u_{i,j}}{\Delta x}, & D_{ij}^{+y} u = \frac{u_{i,j+1} - u_{i,j}}{\Delta y} \\ D_{ij}^{-x} u = \frac{u_{i,j} - u_{i-1,j}}{\Delta x}, & D_{ij}^{-y} u = \frac{u_{i,j} - u_{i,j-1}}{\Delta y} \end{array} \tag{2.10}$$

Convex Hamiltonian: For a convex Hamiltonian, the equation (2.9) can be approximated using the following functional:

$$u_{ij}^{t+1} = u_{ij}^{t} - \Delta t \left[\max(\beta_{ij}, 0)\nabla^{+} + \min(\beta_{ij}, 0)\nabla^{-}\right] \tag{2.11}$$

where approximations for ∇^+ and ∇^- are given by:

- In a first order approximation:

$$\nabla^{+} = \left[\max(D_{ij}^{-x}u, 0)^2 + \min(D_{ij}^{+x}u, 0)^2 + \max(D_{ij}^{-y}u, 0)^2 + \min(D_{ij}^{+y}u, 0)^2\right]^{\frac{1}{2}}$$
$$\nabla^{-} = \left[\max(D_{ij}^{+x}u, 0)^2 + \min(D_{ij}^{-x}u, 0)^2 + \max(D_{ij}^{+y}u, 0)^2 + \min(D_{ij}^{-y}u, 0)^2\right]^{\frac{1}{2}}$$

- In a second order approximation [161, 85, 159]:

$$\nabla^{+} = \left[\max(A, 0)^2 + \min(B, 0)^2 + \max(C, 0)^2 + \min(D, 0)^2\right]^{\frac{1}{2}}$$
$$\nabla^{-} = \left[\max(B, 0)^2 + \min(A, 0)^2 + \max(D, 0)^2 + \min(C, 0)^2\right]^{\frac{1}{2}}$$

with:

$$A = D_{ij}^{-x}u + \frac{\Delta x}{2}m\left(D_{ij}^{-x-x}u, D_{ij}^{+x-x}u\right)$$
$$B = D_{ij}^{+x}u - \frac{\Delta x}{2}m\left(D_{ij}^{+x+x}u, D_{ij}^{+x-x}u\right)$$
$$C = D_{ij}^{-y}u + \frac{\Delta y}{2}m\left(D_{ij}^{-y-y}u, D_{ij}^{+y-y}u\right)$$
$$D = D_{ij}^{+y}u - \frac{\Delta y}{2}m\left(D_{ij}^{+y+y}u, D_{ij}^{+y-y}u\right)$$

where m is the switching function:

$$m(x,y) = \begin{cases} 0, & \text{if } xy < 0 \\ x, & \text{if } xy \geqslant 0 \text{ and } |x| \leqslant |y| \\ y & \text{if } xy \geqslant 0 \text{ and } |x| > |y| \end{cases}$$

Non-convex Hamiltonian: For non-convex Hamiltonian, the reader is invited to look at [161, 159] for details.

Other common values: The curvature of a level could be directly approximate using centered derivative:

$$\kappa = -\frac{u_{xx}u_y^2 - 2u_x u_y u_{xy} + u_{yy}u_x^2}{\left(u_x^2 + u_y^2\right)^{\frac{3}{2}}}$$

When the normal vector is needed, it could be approximated by the mean of all the non-centered derivatives:

$$\begin{aligned} \tilde{\overrightarrow{\mathbf{n}}}_{ij} \;=\; & \frac{(D_{ij}^{+x}u, D_{ij}^{+y}u)}{\left\|(D_{ij}^{+x}u, D_{ij}^{+y}u)\right\|} + \frac{(D_{ij}^{+x}u, D_{ij}^{-y}u)}{\left\|(D_{ij}^{+x}u, D_{ij}^{-y}u)\right\|} \\ & + \frac{(D_{ij}^{-x}u, D_{ij}^{+y}u)}{\left\|(D_{ij}^{-x}u, D_{ij}^{+y}u)\right\|} + \frac{(D_{ij}^{-x}u, D_{ij}^{-y}u)}{\left\|(D_{ij}^{-x}u, D_{ij}^{-y}u)\right\|} \end{aligned}$$

where the potential null vector are forgotten in the previous equation. Then the vector is normalized to give: $\overrightarrow{\mathbf{n}}_{ij} = \frac{\tilde{\overrightarrow{\mathbf{n}}}_{ij}}{\left\|\tilde{\overrightarrow{\mathbf{n}}}_{ij}\right\|}$

2.3 Speeding it up

2.3.1 Narrowband

In [1], Adalsteinsson and Sethian propose a fast algorithm for the Level-Set method that do not update the whole implicit function but only a *band* that

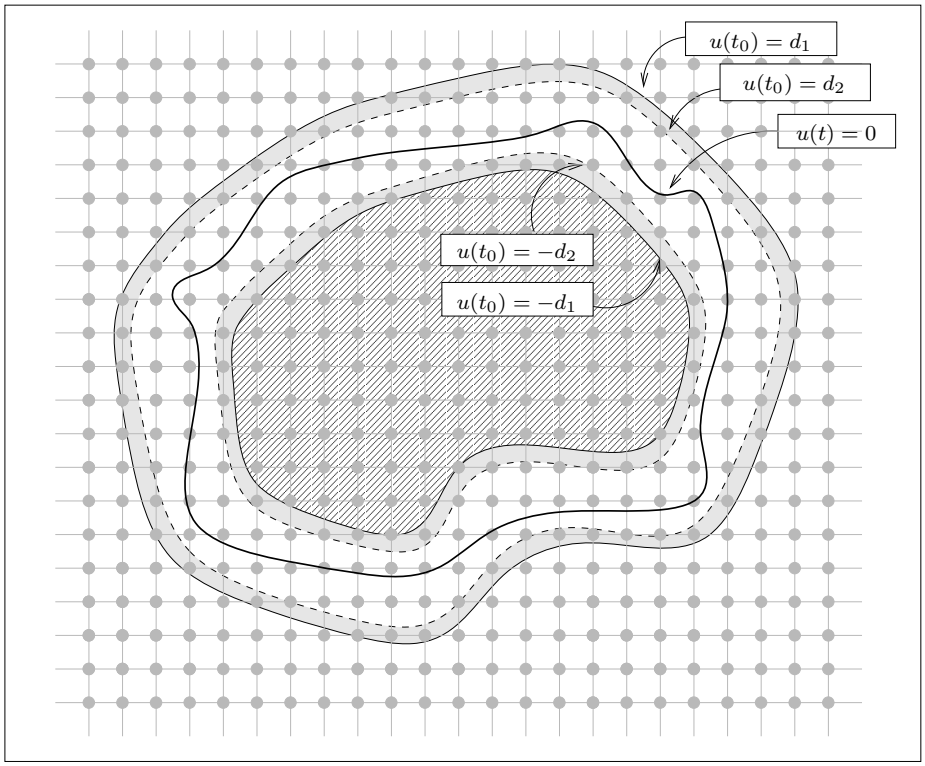


Figure 2.2: At time t_0, The interior and exterior band are initialized at distance d_1 et d_2. Then the zero-level evolves by updating the band only until it reaches the external band. In that case, the bands are reinitialized.

surrounds the zero-level (which is the region of all interest). This band or *narrowband* is in fact composed of two bands (B_1 and B_2 with $B_2 \subset B_1$). Those bands are defined according to the following equations (see Fig. 2.2 for details):

$$B_2 = u_0^{-1}([-d_2; d_2]) \quad \text{and} \quad B_1 = u_0^{-1}([-d_1; d_1]) \quad \text{with } d_2 < d_1$$

During evolution only points inside B_1 are updated until the current zero level $\Gamma_t = u_t^{-1}(\{0\})$ reaches the second band B_2. One can easily detect it by watching over sign changes within $B_1 \backslash B_2$. When that appends, the current zero level is considered as the new initial curve to redefine the Narrowband and the distance should be updated within it (see Section Sec:2.4.1).

This modified algorithm is widely used by the community since it greatly increases performances. Indeed implicit representation allows changes of topology, provides good numerical scheme and wipes out parameterization dependency but is also more computationally expensive. Without *Narrowband*, to evolve a curve (dimension 1) one should keep up to date a functional defined on a domain Ω (dimension 2). In fact, the Level Set method increases the dimensionality of the complexity by one. Nevertheless, the complexity is reduced back to the curve dimension by using the Narrowband method. One should notice that even if the complexity is reduced, the Level Set method with Narrowband is still more expensive than the Snake method (Section Sec:1.1.1).

2.3.2 Fast Marching

The *Fast Marching* method should also be mentioned since it is more efficient than Narrowband implementation. However, this method can only be applied for some class of PDEs. The motion speed **should be** of constant sign on the whole domain (i.e. β is always positive or always negative on Ω). This approach is based on *crossing time of a propagating front* Γ_t. Indeed the sign constraint on β means that the propagating front will cross only one time each point of the domain.

In fact, it can be shown that the crossing time T is linked to the motion speed β by the following PDE:

$$\beta |\nabla T| = 1 \quad \text{with} \quad T(\Gamma_t) = t \tag{2.12}$$

The above equation (2.12) means that crossing time is inverse proportional to motion speed. This method suits particularly well to distance updating/recomputing (see Section Sec:2.4.1). In [159], Sethian gives the following

upwind viscosity solution approximation of the gradient of T in (2.12):

$$\frac{1}{\beta_{ij}^2} = \max(D_{ij}^{-x}T, 0)^2 + \min(D_{ij}^{+x}T, 0)^2 + \max(D_{ij}^{-y}T, 0)^2 + \min(D_{ij}^{+y}T, 0)^2 \tag{2.13}$$

For operators Ds definition, see equations (2.10).

One naive solution would have been to remark that the equation (2.13) is quadratic in T in each point. However a direct approach using iterative scheme until convergence in each point is way to much expensive computationally speaking. This method is discussed in [152]. However the Fast Marching method is much more efficient.

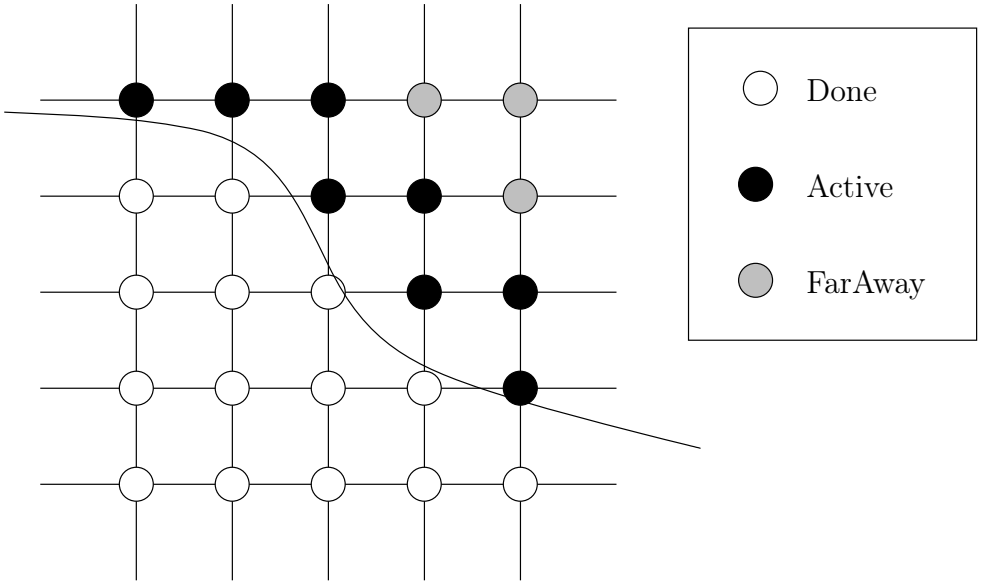


Figure 2.3: The Fast Marching method. In each point, we compute the crossing time of the front. Some of these points belong to a past time (Done), some others will be crossed in a close future (Active) and some others will not be crossed before a long time (Far Away).

The algorithm is based on a domain partition in three regions (see Fig. 2.3 for a graphical representation):

- *Done:* region of Ω where every point is behind the current front. Crossing time is known.
- *Active:* region of Ω where the front stands and acts. Crossing time is being estimated.

- *Far:* region of Ω where front does not have any influence. Crossing time is unknown and fixed to $+\infty$.

One important step is to initialize correctly the partition according to the initial curve Γ_0:

1. Every point (or node) is set to **Done** if the crossing time is negative ($T \leqslant 0$) which include also all points of Γ_0.
2. Every untagged node is marked to **Far** and its crossing time sets to $+\infty$.
3. Every node in the neighborhood of a **Done** node is set to **Active** and its crossing time is updated according to equation (2.13).

The algorithm can then be described by the following three steps:

1. *Remove:* Take the smallest estimate crossing time point ij in the **Active Set**. Tag it as **Done**.
2. *Update:* Tag every **Far** neighbor node to **Active**. Update the crossing time every **Active** neighbour node using equation (2.13).
3. *Loop:* Loop to step 1 until there is no more **Active** node.

The main difficulty is contained in step 1 of the algorithm. Maintaining a list of node and extracting the one with the smallest crossing time is not as easy as it seems. One should consider to make use of a *min-heap* data structure [157] for speed consideration. This structure is in fact a binary tree where each node contains a value and each child's value is greater than its parent's. Maintaining such a data structure has a cost. However this one is minimal for Fast Marching Method. Let assume the min-heap contains n nodes inside, the cost in mean of each atomic action can be reduced:

- *Extract the smallest node:* $O(1)$
- *Update the node (value and position):* $O(\log n)$ since the value could only decrease.
- *Remove the smallest node:* $O(\log n)$
- *Insert a node:* $O(\log n)$

For more details on Fast Marching Methods, the reader is referred to [7, 45, 163] for theoretical concerns and [100, 152] for more practical concerns.

2.4 Implementation

One of the main problems encountered during a time evolution of an implicit function u is that u does not remain a perfect distance function and then one should recompute the sign distance regularly. Moreover using the Narrowband method (see Section Sec:2.3.1) requires recomputing the distance function every time the contour leave the Narrowband (one could notice that the sign distance has to be computed only on the narrowband).

Another drawback is inherent to implicit representation. Indeed by considering this intrinsic representation, one should be able to compute β on the whole domain Ω. But sometimes this is not possible.

This section will discuss different strategies to deal with sign distances. Then we will present a method to manage the case where β can only be estimate on the curve.

2.4.1 Distance Function

In order to correct the distance corruption, one could use naive or intelligent approaches to restore the signed distance:

Naive method: Recomputing the distance from the zero level. In order to achieve good precision, one should extract a sub-pixel zero-level using marching squares (2D) or cubes (3D) [119] and then recompute the distance using a fast algorithm [52]. This method is quite expansive and becomes excessively expansive in 3D.

Correct the distance: A better way of doing it is to make use of PDEs. Indeed using the following PDE [165] on an implicit function

$$\frac{\partial u}{\partial t} = sign(u)(1 - |\nabla u|)$$

leads to the corrected distance function. However, the zero level is move by a small displacement.

Recompute using Fast Marching [158]: The best method is based on the Fast Marching method (see Section Sec:2.3.2) and consists on two evolution steps using $\frac{\partial \Gamma}{\partial t} = \overrightarrow{\mathbf{n}}$ and $\frac{\partial \Gamma}{\partial t} = -\overrightarrow{\mathbf{n}}$ respectively. The distance to the zero level is given by the crossing time of the propagating front. According to this propagation when a first pixel set to done has a value higher than t, the front reaches the level set values that corresponds to an absolute distance of t from the initial front position. This method combines speed and accuracy, does not move the zero level and suits well to a Narrowband implementation since it can be easily stopped at time t_1 (see Fig. 2.2).

2.4.2 Preserving Distance Function

One other way to deal with distance function is to try to preserve it over the evolution. Gomes in [77] proposed to consider the following evolution equation instead of (2.8):

$$\begin{cases} \frac{\partial u}{\partial t} = -\beta_0 |\nabla u| \\ \beta_0(\omega) = \beta(\omega_0) \quad \text{where} \quad \omega_0 = \omega - u(\omega)\nabla u(\omega) \end{cases} \tag{2.14}$$

This equation means that for every point $\omega \in \Omega$, we consider the force β_0 of the closest point ω_0 on the interface. Preserving sign distance map allows decreasing sensibly the frequency of correction. Even the Narrowband method could take advantage of this, since distance will only be recomputed when the zero-level reaches the boundary of the Narrowband. Another way to deal with this is discussed in [142].

2.4.3 Extending the speed value

It happens that the speed value β cannot be estimate on the whole domain. In that special case, the whole Level Set method breaks. However one could notice the previous equation (2.14) makes use of β on the curve only. This means that this stabilizing scheme (Section Sec:2.4.2) also deals with those special cases.

Another method was proposed in [142], the main idea is to propagate the value β on the curve along its normal. This is obtained by the following PDE evolution:

$$\frac{\partial \beta}{\partial t} = -sign(u)\nabla\beta \cdot \overrightarrow{\mathbf{n}} \tag{2.15}$$

This equation produce a β function on the whole domain that satisfies the following constraints:

$$\begin{cases} \beta(t\overrightarrow{\mathbf{n}}_0 + \omega_0) = const \quad \forall t \in \mathbb{R} \\ \beta(\omega_0) = \beta_0(\omega_0) \end{cases}$$

where β_0 is the reference speed value defined on the curve and ω_0 is a point on the curve.

These two methods seem to be quite similar however the second one is more robust in practice.

2.5 Examples

2.5.1 Mean Curvature Motion

The Mean Curvature Motion is one of the simplest evolutions that can be addressed in level set framework:

$$\frac{\partial u}{\partial t} = div\left(\frac{\nabla u}{|\nabla u|}\right)|\nabla u| \tag{2.16}$$

2.5.2 Geodesic Active Contour

Geodesic Active Contours are usually addressed in level set framework. The Euler-Lagrange formulation (1.9) of the equation (1.8) can be rewritten - considering the level set function u - as follow:

$$\frac{\partial u}{\partial t} = g(|\nabla I|)|\nabla u| div\left(\frac{\nabla u}{|\nabla u|}\right) + \nabla g(|\nabla I|) \cdot \nabla u \tag{2.17}$$

2.5.3 Geodesic Active Region

Level Set can also implement Geodesic Active Region. However, in that case the speed value cannot be compute on the whole area. Thus β should be expand over the whole domain using the method describe in Section Sec:2.4.3. The speed value on the curve is given by (see Section Sec:1.2.3 for more details):

$$\begin{aligned}\beta(\omega) = \alpha\left(g(P_b(I(\omega)))\kappa(\omega) + \nabla g(P_b(I(\omega))) \cdot \overrightarrow{\mathbf{n}}(\omega)\right) \\ + (1-\alpha)\log\left(\frac{P_{out}(I(\omega))}{P_{in}(I(\omega))}\right)\end{aligned} \tag{2.18}$$

After expanding the speed function over the domain, one simply evolves the curve using the easiest scheme:

$$\frac{\partial u}{\partial t} = -\beta|\nabla u|$$

Chapter 3

Graph Cuts

In this chapter we present a minimizing tool for Markov Random Fields (MRF). If this technique was introduced long time ago [69], its application to Computer Vision community is fairly recent [18]. This new approach receives recently a lot of attention since it provides a global optimum in relative small amount of running time. However it cannot deal with every Markov Random Fields. Indeed its applicability range is limited to binary MRFs and some other constraint. Currently the Computer Vision community spends a lot of work to formulates previous work into this framework in order to have access to a global optimum.

The chapter is organize as follow. The first section will describe briefly in which context this method lies and some other older approaches (Sec:3.1). Then we describe in more details the framework and which class of energies can be optimally and approximately minimize (Sec:3.2). Then we present some algorithms already available (Sec:3.3). Finally we propose an implementation of Geodesic Active Contours and Regions in a Graph Cuts framework (Sec:3.4).

3.1 Introduction

In this section we present the general framework of Graph Cuts. Indeed the Graph Cuts method comes directly from the Combinatorial community. And segmentation is now defined differently than in Level Set framework.

The following sections will first describe how segmentation is considered in this framework Sec:3.1.1. Then we present in more details the mathematical formulation (Markov Random Fields MRFs) Sec:3.1.2 and some available approaches in order to minimize MRFs Sec:3.1.3.

3.1.1 Labeling

Many problems in Computer Vision can be seen as labeling problems. Simply think about segmentation: segmenting an image in two regions comes down to assigning a unique label to each pixel (foreground or background, in or out).

A labeling problem is completely defined by a couple of sets: the site set and the label set. Sites correspond to features (usually pixels) carrying some properties which need to be estimated. All those properties lie in the label set.

Let respectively $\mathcal{P}$ and $\mathcal{L}$ be those sets:

$$\mathcal{P} = \{p_1, ..., p_n\}$$
$$\mathcal{L} = \{l_1, ..., l_k\}$$

with n sites and k different labels. Usually $\mathcal{P}$ is a set of pixels in an image or voxels in a volume but it can be any other feature such as edges, regions etc... Labels can be anything; they depend on and are designed for a specific application. For example, for segmentation it can be "*foreground*" and "*background*". For Stereo, labels represent disparity...

A labeling problem consists into finding a map M from $\mathcal{P}$ into $\mathcal{L}$ that assign for each site p a unique label $l = M(l_p)$. Let $\mathcal{M} = \mathcal{L}^n$ be the map space. This is a discrete optimization problem since the number of combinations is finite $|\mathcal{M}| = k^n$. However, the number of solutions is excessively big to search for the optimal solution in an exhaustive manner. Other methods had been developed by the Combinatorial community to solve some of those problems quite efficiently.

Many Computer Vision problems can be addressed using this general framework:

- *image segmentation:* $\mathcal{P} = \{pixels\}$ and $\mathcal{L} = \{0, 1\}$ (see []).
- *image restoration:* $\mathcal{P} = \{pixels\}$ and $\mathcal{L} = \{intensities\}$ (see []).
- *stereo reconstruction:* $\mathcal{P} = \{pixels\}$ and $\mathcal{L} = \{disparities\}$ (see []).
- *texture synthesis:* $\mathcal{P} = \{pixels\}$ and $\mathcal{L} = \{patches\}$ (see []).
- ...

3.1.2 Context

Markov Random Fields: this framework was introduced by Geman and Geman [73] and deals elegantly with site interactions. MRF need the addition of a neighborhood system $\mathcal{N}$ defined on $\mathcal{P}$. This neighborhood must satisfy some constraints. Let $p \in \mathcal{P}$ be a site and $\mathcal{N}_p$ its neighborhood such that:

$$\begin{array}{ll} (i) & p \notin \mathcal{N}_p \\ (ii) & \text{if} \quad p \in \mathcal{N}_q \quad \text{then} \quad q \in \mathcal{N}_p \end{array}$$

Let $M = (M_1, ..., M_n)$ be a set of random variables defined on $\mathcal{P}$, each taking its value into $\mathcal{L}$. A realization or configuration of the field M is noted $m = (m_1, ..., m_n)$. According to those definitions, M is a Markov Random Field iif:

$$\begin{array}{ll} (i) & P(M = m) > 0 \quad \forall\, m \in \mathcal{M} \\ (ii) & P(m_p | m_{\mathcal{P}-\{p\}}) = P(m_p | m_{\mathcal{N}_p}) \end{array}$$

The first point means that each configuration is probable. This assumption is needed in order to assert a well-defined joined probability (see [8] for details). The second point means that a label at a given site p only depends on its neighborhood.

One should now define *cliques*, a useful concept for sites interaction. Indeed, a clique C is a set of sites that verifies the following condition:

$$\forall p, q \in C \quad p \in \mathcal{N}_q$$

which signifies that each site within a clique is in the neighborhood of each other site. Let $\mathcal{C}$ be the set of all possible cliques. The relationship between sites is defined using potential functions on cliques. Let $V_C(m)$ be a potential function for the given clique C according to the configuration m. In fact, this potential function specifies labels relationship within the clique.

Gibbs distributions are defined by:

$$P(m) = P(M = m) = Z^{-1} exp\left(- \sum_{C \in \mathcal{C}} V_C(m) \right)$$

where Z is a normalizing constant. The Potential functions and the set of cliques define a Gibbs Random Field according to the Gibbs distribution $P(m)$.

The theorem of Hammersley-Clifford [8] proves the equivalence of Gibbs Random Fields and MRFs. This convenient result allows simulating a MRF using a Gibbs sampler for example.

In the following sections, mainly pair-wise cliques will be considered but this is not a major restriction since most of the problems can be addressed

using cliques of order two only. This restriction leads to the following Gibbs distribution:

$$P(m) = P(M = m) = Z^{-1} exp\left(- \sum_{p,q \in \mathcal{N}} V_{\{p,q\}}(m_p, m_q) \right)$$

A classical potential function is related to the Potts model:

$$V_{\{p,q\}}(m_p, m_q) = \mathbb{1}\left(m_p \neq m_q\right) = \begin{cases} 1 & \text{if } m_p \neq m_q \\ 0 & \text{otherwise} \end{cases}$$

This simple case enforces neighbor sites to have the same label. This model is widely use in many applications.

Maximum A Posteriori: The map M is in general not accessible, one could only estimate its realization m through an observation obs. The conditional probability $P(obs|m)$ is the link between the realization and the observation. A classical method to estimate the configuration m is to use Maximum A Posteriori estimation. This method was well studied by Geman and Geman [73] and aims at maximizing the posterior probability $P(m|obs)$. This one is related to the previous one by the Bayes rule:

$$P(m|obs) = \frac{P(obs|m)P(m)}{P(obs)} \tag{3.1}$$

Since the problem consists in maximizing the previous equation with respect to m, then $P(obs)$ does not act on it. Then the MAP problem is equivalent to:

$$\underset{m \in \mathcal{M}}{\arg\max} \quad P(obs|m)P(m) \tag{3.2}$$

$P(m)$ is given by the MRF model but one needs to estimate $P(obs|m)$. This term codes for the relationship between the model and its observation. In image restoration for example, this term controls the noise model (for example, a Gaussian noise). In order to obtain a convenient expression of $P(obs|m)$, let us assume that $P(obs|m) = \prod_{p \in \mathcal{P}} P(obs_p|m_p)$. This assumption holds when the noise is independent at each site p. Let us define also the data model D^{obs} which links observation and realization:

$$P(obs_p|m_p) = K \exp\left(-D_p^{obs}(m_p)\right) \tag{3.3}$$

where K is a normalizing constant. Unifying those assumptions, the complete estimation of $P(obs|m)$ is given by:

$$P(obs|m) \propto \exp\left(- \sum_{p \in \mathcal{P}} D_p^{obs}(m_p) \right) \tag{3.4}$$

The complete MAP-MRF problem can be rewritten as:

$$\arg\max_{m\in\mathcal{M}} \quad \exp\left(-\sum_{p\in\mathcal{P}} D_p^{obs}(m_p) - \sum_{p,q\in\mathcal{P}} V_{\{p,q\}}(m_p, m_q)\right) \tag{3.5}$$

which is equivalent, in an energy framework, to:

$$\arg\min_{m\in\mathcal{M}} \quad \sum_{p\in\mathcal{P}} D_p^{obs}(m_p) + \sum_{p,q\in\mathcal{P}} V_{\{p,q\}}(m_p, m_q) \tag{3.6}$$

The whole chapter will focus on such an energy and methods to minimize it.

3.1.3 Possible Approaches

For any energy like (3.6), a deterministic algorithm will have to explore every possible configuration in order to find a global optimum. This means that any deterministic and global minimizer will solve the problem in exponential time since the number of configurations $|\mathcal{M}| = k^n$ (where k is the number of labels and n the number of sites). Exhaustive search cannot be considered in that case, one needs to switch to different approaches.

A way of coming up to the generic problem is to switch to stochastic algorithms such as *Simulated annealing*. For deterministic methods, one should consider constraining the energy expression to reduce the search space.

Simulated Annealing: Simulated Annealing is the most used algorithm for minimizing such a general energy (3.6). It was proposed simultaneously by Cerny in [29] and Kirkpatrick et al. in [102] and subsequently introduced in Computer Vision community by Geman and Geman in [73]. The algorithm is presented here for Gibbs distributions but it can be used in a large panel of frameworks (see Section Sec:6.1 for an application to Level Set based segmentation).

Historically this algorithm was developed to imitate a thermodynamic procedure: a material annealing. Physical annealing consists in decreasing gradually temperature (cooling phase) of a metal while sometimes the metal is heated (annealing phase). The global process consists in an alternation of those phases. The energy that has to be minimized is inversely proportional to the regularity of its crystalline structure. Indeed the more regular the crystalline structure is, the more the metal is hard and resistant. If temperature is lowered according to a predetermined cooling function and the annealing phase is controlled, then this procedure reaches a global minimum of the energy.

Inspired by this process, simulated annealing makes use of a temperature which is decreased according to a certain scheme. Basically, a change of label is requested on a given site. Sites are taken one after each other based on a predefined order or randomly. If by this change, the global energy decreases, then the change is validated (cooling phase), otherwise it can be adopted according to a test on the energy variation (annealing phase). This test is based on a probability function depending on a control parameter: the temperature. The lower the temperature is, the less often the change is accepted. Basically, for high temperatures, simulated annealing performs a random walk and for low temperatures, it acts as a "gradient descent" and stops on a local minimum. With that kind of policy, the algorithm is convergent at least to a local minimum.

However, in some cases if the decreasing of the temperature is chosen carefully, simulated annealing is ensured to converge to a global minimum. Nevertheless, a convenient lowering speed is in practice too slow and is never followed. Actually, the main interest of such a method is not the global optimality but the capacity of escaping from local minima and reaching a lower minimum.

Dynamic Programming: For some particular energies, the global minimum can by achieved using Dynamic Programming (DP) [4]. The main advantage of DP is its speed and its main disadvantage is that it has difficulty to solve other energies than one-dimensional functions. However DP is effective in some widely used vision problems such as Snakes [166] (see Section Sec:1.1.1). Moreover, DP principle can be used to reduce the number of possible configurations to try in a Gibbs random fields [58].

Graph Cut: In the particular case where there are only two labels $\mathcal{L} = \{0, 1\}$, the global minimum can be found using the Graph Cut method for a certain type of energy even if the number of configurations is still 2^n. In [78] Greig et al. shows how to obtain an exact maximum for a binary MAP problem. However, an exact maximum is achieved only for binary functions.

That is why Ferrari et al. in [66] proposed a method based on Greig's one to obtain a strong local minimum for any number of labels. That is to say, a minimum contained within a factor of two of the global minimum. This method holds only for a small class of energies. In [67] Ferrari et al. describes an improved algorithm to obtain a global minimum for an even smaller sub-class of energies.

The next sections will describe in more details the Graph Cut framework, how and when it can be used.

3.2 Description

In this section we will first present the general Graph Cuts framework Sec:3.2.1. Then we will describe the class of binary energies that can be optimally minimize in this framework Sec:3.2.2. Then we present how Graph Cuts can be used to provide a strong minimum for the general n labels problem (Multiway Cuts) Sec:3.2.3.

3.2.1 Graph Basics

Let us consider a weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, $\mathcal{V}$ being the set of nodes and $\mathcal{E}$ directed edges connecting them. Two special *terminal* nodes are present: the *source* s and the *sink* t. Each edge connecting nodes p and q is assigned a non negative weight $w(p, q)$. Edges are broken in two groups: n-links and t-links. A n-link is an edge connecting two non-terminal nodes. A t-link connects a non-terminal node to a terminal node, s or t.

A st-cut $C \subset \mathcal{E}$, simply called "*a cut*" in the following, is a set of edges that satisfies the following properties:

- the resulting graph $\mathcal{G}(C) = \langle \mathcal{V}, \mathcal{E} - C \rangle$ separates the source node from the sink node such that there is no path linking the terminals.
- there is no subset of C that also separates the two terminals.

Such a cut defines a partition of the nodes of the graph into two disjoint subsets S et T such that the source $s \in S$ and the sink $t \in T$.

Its cost $|C|$ is the sum of the weights of all edges (p, q) such that $p \in S$ and $q \in T$. The reader should be careful and be aware that only edges leaving the S part are taken into account, no edge from T part to S is considered.

$$|C| = \sum_{\substack{p \in S \\ q \in T \\ (p,q) \in C}} w(p, q)$$

The minimum cut is the cut with minimal cost. One classical problem is to find the minimal cost cut in the graph. In order to minimize a given energy, one could set correctly edge weights on n- and t-links in such a way that the cost of the minimal cut corresponds to the global minimum of the energy.

Thanks to the Ford and Fulkerson theorem [69], it is proved that the minimal cut problem is equivalent to the max-flow problem. The max-flow problem aims at finding the maximal value of flow that could reach the sink from the source while considering edges as pipes and weights as pipes' capacities. In fact, the maximal flow value is equal to the minimal cut cost.

Given for a better understanding, a high-level proof of the Ford and Fulkerson's Max-Flow/Min-Cut theorem can be expressed as follow. Let us consider the set of all possible st-cuts that separate the source from the sink. One could see those cuts as a set of barriers that the flow has to cross to reach the sink. As the flow has to cross every barrier, one could easily see that the maximal flow is bounded by the minimal capacity of the barriers. Since the barrier of minimal capacity will be saturated first and separates the source from the sink, the barrier of minimal capacity is exactly the minimal cost cut.

There are already many algorithms to solve this problem (historically the first one was proposed by Ford and Fulkerson in their book) we refer the reader to [2] for more details on common algorithms. Some of these algorithms will be described briefly in the next sections.

3.2.2 What energy can be optimally minimized

Considering energies from the class (3.6), one should add some constraints on the second term $V_{\{p,q\}}$ in order to get access to global optimality. Let us rewrite this equation for easier reading:

$$E(m) = \sum_{p \in \mathcal{P}} D_p^{obs}(m_p) + \sum_{p,q \in \mathcal{P}} V_{\{p,q\}}(m_p, m_q) \tag{3.7}$$

The Maximal Flow/Minimal Cut method was firstly used in Computer Vision for binary restoration by Greig in [78]. In this application, the energy was designed as follow:

$$E(m) = \sum_{p \in \mathcal{P}} D_p^{obs}(m_p) + \lambda \underbrace{\sum_{p,q \in \mathcal{P}} \mathbb{1}\left(m_p, m_q\right)}_{\text{Potts model}} \tag{3.8}$$

The first term corresponds to a data consistency criterion while the second term aims at finding a region of shortest boundaries since the Potts model penalizes label discontinuities. The goal of this work was to compare simulated annealing and graph cut on a convenient energy. Authors aimed at proving that making use of simulated annealing with an incorrect and faster schedule of cooling leads to a local minimum that can be far from the optimum (provided by graph cut) even in this simple case. Maybe because on this very restrictive application, this work remained unknown at least for 10 years.

Later, based on the work of Ray and Cox [153], Ishikawa in [81] proposed a graph design to obtain an exact solution for a specific multi labeled MRF

(more than 2 labels). Indeed labels have to be linearly ordered which limits application of this technique to one dimension labels. For example it cannot make use of this method for optical flow. Moreover the interaction term should have the shape:

$$E(m) = \sum_{p \in \mathcal{P}} D_p^{obs}(m_p) + \sum_{p,q \in \mathcal{P}} \lambda_{pq} |m_p - m_q| \tag{3.9}$$

Here labels are supposed to be integers from $\{1, \ldots, n\}$.

We describe here the graph construction to minimize energy (3.9). At each site p are associated $k-1$ nodes $p_1, \ldots, p_{k-1}$. Edges from source to sink crossing nodes $p_1, \ldots, p_{k-1}$ are noted $e_1^p, \ldots, e_k^p$ and their capacities are respectively equal to $D_p^{obs}(m_1), \ldots, D_p^{obs}(m_k)$. Then edges connecting $p_j \rightarrow q_j$ are set to λ_{pq}. Figure 3.1 illustrates the graph for four labels. Any st-cut has to cut at least one edge e_j^p for each site p. To force the minimal cut to go through only one edge per site, reverse edges of infinite capacity are superposed (grey links in Figure 3.1). The optimal configuration of the MRF is obtained by assigning the label m_j to p if the edge e_j^p is cut.

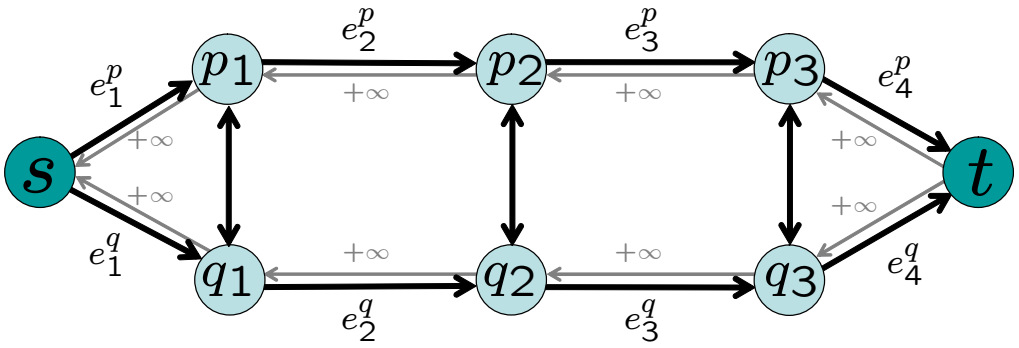


Figure 3.1: Ishikawa graph design for exact minimization of equation (3.9) for multi-label MRF. Here four labels are considered.

This case might be restrictive but Ishikawa generalized later in [80] this graph construction to any convex function of $|m_p - m_q|$. However the extension to convex potentials is relatively unuseful since such a potential tends to produce over-smoothed boundaries [156]. Moreover to extend to convex V_{pq} one has to introduce many edges into the graph described above which creates a much more complex graph. Nevertheless this is the only known case where a multi-label MRF can be solved optimally.

Getting back to binary MRFs, Kolmogorov and Zabih in [108, 109] settle a class of potential functions that can be minimized using Graph Cut. They

define for pair-wise interaction the notion of submodular functions. A binary pair-wise function f is submodular iif:

$$f(0,0) + f(1,1) \leqslant f(0,1) + f(1,0) \tag{3.10}$$

They prove the following result: any energy given by (3.7) where every V_{pq} is submodular can be optimally minimized using graph cuts. They also provide a similar result for ternary cliques that was used in [107]. Later, Freedman and Drineas generalized it to any clique [70].

Some interesting results provided by Kolmogorov and Boykov in [16], show how n-link weights could be set in order to approximate Riemannian metrics. This result will be more developed in a following section. The same idea reached its limit in [106] where the same authors proved that by using only n-links, the submodular constraint limits this approach to Riemannian metric plus Flux.

3.2.3 Multiway Cut

Multiway Cuts refer to the same problem but generalized to n terminals $\{t_1, \ldots, t_n\}$. The issue is to find the minimal cost cut that separates terminals from each other. However, it is known since [51], that this is a NP-complete problem as soon as there are more than two labels. The restriction to two labels corresponds to Graph Cuts. Multiway cuts address the problem of finding the optimal configuration of a multi-label MRF. In the previous section, we shown the graph construction proposed by Ishikawa to optimally solve some particular multi-label MRFs using a simple graph cut approach. Nevertheless, it cannot address more general MRFs. That is why a large community had widely studied this issue.

However since the problem is NP-complete and unless it is shown that $P = NP$, only approximating algorithms are considered. One simple algorithm proposed in [51] simply considers n two-way cut problems solved using graph cut. Each aims at finding the minimal cut that separate a chosen reference terminal from all others. This leads to n different cuts noted $C_1, \ldots, C_n$. A Multiway Cut is then constructed concatenating every cut except the one with the maximal cost:

$$C = \bigcup_{\substack{i=1\ldots n \\ i \neq i_{max}}} C_i \quad \text{where} \quad i_{max} = \arg\max_{i=1\ldots n} |C_i|$$

It can be proved that this methods leads to a good approximation of the minimal energy. It provides a minimum within a factor of $2 - \frac{2}{n}$. However

even if this method gives a close minimum with respect to the energy, the labeling could be far from the optimal one and in practice this is the case.

Based on those observations, two similar algorithms based on graph cuts were proposed by Boykov, Veksler and Zabih in [19, 20, 21]. Those two techniques provide a local minimum of the energy with respect to a given set of *transformations*. Those methods require the definition of a set of allowed transformations $\mathcal{T}_m$. A transformation refers to an allowed labeling change on a given configuration m of the considerer MRF. A local minimum is then defined by:

$$\forall T \in \mathcal{T}_m \quad E(T(m)) \geqslant E(m)$$

which means that considering any atomic allowed transformation on the labeling, none of them can decrease the energy. This framework suffers from a common drawback for optimization techniques. Since it needs an initialization and converges to a local minimum, the result strongly depends on the initial labeling.

Those techniques can be defined as follow:

- *α-expansion:* an α-expansion only considers one label (noted α) at a time. An α-expansion transformation T^α only modifies the current labeling such as any label can be changed to α or remains the same (noted $\bar{\alpha}$). Actually, α-expansion can be seen as a competition between the proposed new label α and the current label $\bar{\alpha}$ at each site. The problem can be solved as a binary optimization problem and summarized as follows: *which from α or $\bar{\alpha}$ should we keep in each site in order to optimally minimize the energy according to the allowed transformation?*

- *$\alpha\beta$-swap:* an $\alpha\beta$-swap considers a couple of labels (noted α and β) in the same time. An $\alpha\beta$-swap transformation $T^{\alpha\beta}$ changes the current labeling such as only labels α or β can be changed respectively to β or α. $\alpha\beta$-swap can be seen as a competition within the couple of labels. The problem can also be solved as a binary optimization problem and summarized as follows: *which label from α or β should we have in sites where those labels are present in the current labeling in order to optimally minimize the energy according to the allowed transformation?* The main advantage of $\alpha\beta$-swap on α-expansion, is that the resulting graph is smaller since one does not consider sites where the current label is different from α or β.

Graph cuts provide the best transformation T^α (respectively $T^{\alpha\beta}$) that minimizes the energy. However only MRFs with submodular interaction between labels can be considered here.

Starting from an initial configuration m, the procedure consists in the iterative optimization scheme defined above where all the labels are considered one after each other in a predefined or random order. Each iteration of this scheme does not produce a configuration of higher energy. For $\alpha\beta$-swap one should consider every possible couple of labels. Knowing that an α-expansion cycle is achieved after n iterations while $\alpha\beta$-swap cycle contains $\frac{n(n-1)}{2}$ iterations. However, those algorithms need several cycles to converge. In practice $\alpha\beta$-swap gives better results but needs more iterations and cycles to converge.

It can be proved that the local minimum is within a factor of two when considering the Potts model in a four connected neighborhood.

Figure 3.2 illustrates the graph design for both algorithms for a couple of neighbor sites $\{p, q\}$. Let us first consider α-expansion. $\bar{\alpha}_p$ and $\bar{\alpha}_q$ refer to

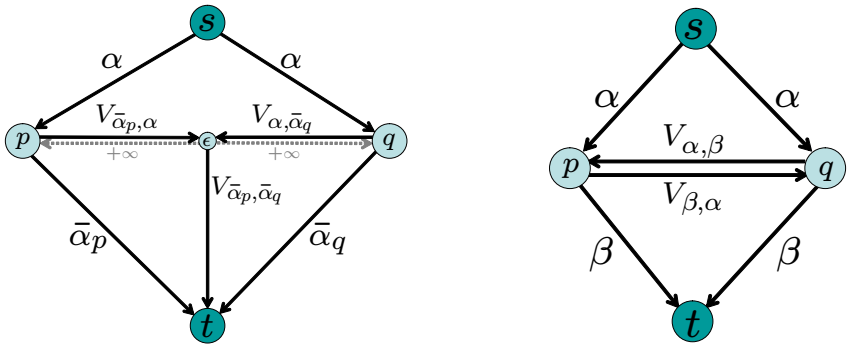


Figure 3.2: Graph design for α-expansion (Left) and $\alpha\beta$-swap (Right).

the current label of their respective site. For an easy understanding let us introduce an extra node (noted ϵ) in the center on the left part of the figure 3.2. For readability, let us assume that $V_{pq}(\alpha, \alpha) = 0$. Let us set the links according to the following table:

T-Links	Capacity
$s \rightarrow p$	$D_p^{obs}(\alpha)$
$p \rightarrow t$	$D_p^{obs}(\bar{\alpha}_p)$
$s \rightarrow q$	$D_q^{obs}(\alpha)$
$q \rightarrow t$	$D_q^{obs}(\bar{\alpha}_q)$
$\epsilon \rightarrow t$	$V_{\bar{\alpha}_p,\bar{\alpha}_q}$

N-Links	Capacity
$p \rightarrow \epsilon$	$V_{\bar{\alpha}_p,\alpha}$
$q \rightarrow \epsilon$	$V_{\alpha,\bar{\alpha}_q}$
$\epsilon \rightarrow p$	$+\infty$
$\epsilon \rightarrow q$	$+\infty$

Table 3.1: Link settings for the α-expansion algorithm. See left part of figure 3.2 for correspondences.

In table 3.1, the first t-link measures the likelihood of site p to have label α

based on data observation. The next three t-links are set similarly. The fifth t-link codes for interaction between p and q when those two sites are labeled $\bar{\alpha}_p$ and $\bar{\alpha}_q$ respectively. The two first n-links code for site interaction. The first one when p is assigned to $\bar{\alpha}_p$ and q to α and similarly for the second one. Finally the last two n-links ensure some consistency when $V_{\bar{\alpha}_p,\bar{\alpha}_q}$ is lower to $V_{\alpha,\bar{\alpha}_q}$ or $V_{\bar{\alpha}_p,\alpha}$. However note that we still need the submodular constraint on V_{pq} which means that $V_{\bar{\alpha}_p,\bar{\alpha}_q} \leqslant V_{\alpha,\bar{\alpha}_q} + V_{\bar{\alpha}_p,\alpha}$. According to this construction, one could see that the minimal cut will cross the t-links associated to the optimal label: cutting $s \rightarrow p$ means that p should be assigned with α while cutting $p \rightarrow t$ means that p should remain with its current label. However it is possible to free ourself from the extra node, the infinite links and the extra assumption, namely $V_{pq}(\alpha, \alpha) = 0$, and for more details on how to implement it the reader should refer to [109].

Let us now consider the case of $\alpha\beta$-swap. Let us consider two sites $\{p, q\}$ such that p and q in the current configuration are assigned to α or β. The graph construction is quite simpler to the one for α-expansion. Edge links are set according to the following table:

T-Links	Capacity
$s \rightarrow p$	$D_p^{obs}(\alpha)$
$p \rightarrow t$	$D_p^{obs}(\beta)$
$s \rightarrow q$	$D_q^{obs}(\alpha)$
$q \rightarrow t$	$D_q^{obs}(\beta)$

N-Links	Capacity
$p \rightarrow q$	$V_{\beta,\alpha}$
$q \rightarrow p$	$V_{\alpha,\beta}$

Table 3.2: Link setting for the $\alpha\beta$-swap algorithm. See right part of figure 3.2 for correspondences.

The graph can be interpreted similarly to the α-expansion case. For more details on those algorithms, the reader is invited to read [21].

3.3 Algorithms

During all the previous sections, we had assumed that we had access to an algorithm for solving the Minimal Cut problem. In this section, we will describe many available algorithms to solve it.

Finding the minimal cut is not straightforward and since the Min-Cut problem is known to be equivalent to the Max-Flow problem [69], all those methods solve this dual problem. This approach was briefly introduced in Section Sec:3.2.1. We will firstly describe the Max-Flow problem and then we will make a rough sketch of two classes of algorithms.

3.3.1 Maximal Flow Problem

This dual problem lies also on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V}$ is a set of vertices (or nodes) and $\mathcal{E}$ is a set of directed pipes (or edges) connecting them. Two special nodes are present in $\mathcal{V}$: the source s and the sink t (where 't' means target). The capacity of a pipe is a function c from $\mathcal{V}^2$ into $\mathbb{N}$ such that if $(p, q) \notin \mathcal{E}$ then $c(p, q) = 0$.

For a high-level image of the problem, the graph defines a network of pipes. Let us imagine that the special node '*source*' is connected to an infinite source of water. The capacity is the quantity of water that could reach the head of the pipe from the tail. The max-flow problem can be summarize as: How many water could reach the special node '*sink*'?

Note that the cost of the minimal cut is equal to the maximal flow that could reach the sink.

A second and dual problem is: where are the bottlenecks in the network? This can be interpret for a water distribution problem as: where should we increase pipes' capacity in order to be able to distribute more water to the sink? Those pipes of limiting capacity are located on the Minimal Cut. How convenient!

That kind of problem has many applications such as:

- Electric transportation on an electric line network.
- Data transfers on internet.
- Factory product to sellers.
- Water distribution.
- ...

Let us define the notion of *flow*. The flow is a function f from $\mathcal{V}^2$ into $\mathbb{Z}$ such that:

$$\begin{cases} \forall\ (p, q) \in \mathcal{V}^2 & f(p, q) \leqslant c(p, q) & \text{Capacity constraint} \\ \forall\ (p, q) \in \mathcal{V}^2 & f(p, q) = -f(q, p) & \text{Symmetry} \\ \forall\ p \in \mathcal{V} \setminus \{s, t\} & \sum_{q \in \mathcal{V}} f(p, q) = 0 & \text{Flow conservation} \end{cases} \tag{3.11}$$

Such a function is also called a *feasible flow*.

We could now define the *residual graph* $\mathcal{G}_f = \langle \mathcal{V}, \mathcal{E}_f \rangle$ where the capacity on this graph is given by the residual capacity $c_f(p, q) = c(p, q) - f(p, q)$. The residual graph can be seen as the graph of the remaining available capacity. Here the symmetry introduced in the flow takes it sense since it allows to

cancel send of flow through an edge by increasing the capacity of the reverse edge: if one unit of flow is sent through the pipe (p, q) then its residual capacity decreases by one while the residual capacity of the reverse edge increases by one allowing to send this unit flow back to p. Note that an edge of null residual capacity is called *saturated edge*.

Two classes of algorithms were proposed to solve this problem. One proposed by Ford and Fulkerson in [69] which preserves flow conservation during the whole process and another one proposed by Goldberg in [76] which breaks the flow conservation rule until convergence. Those methods are detailed in the following sections.

3.3.2 Flow Conservation Preserving

All the algorithms presented here are based on the same idea: *augmenting paths*. The augmenting path approach is based on the following remark: considering a path from s to t on the residual graph of connected nodes by non-saturated edges, such a path could transport at least one unit of flow from s to t. Such a path is called an *augmenting path* since it has the capacity of augmenting the flow from from s to t.

The first algorithm was proposed by Ford and Fulkerson in [69] and it was quite simple. It can be written as follow:

Algorithm 1 Ford-Fulkerson Algorithm

Require: $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ a graph
Ensure: A maximal flow function f and the residual graph.
  $f(p, q) \leftarrow 0 \quad \forall\, p, q \in \mathcal{V}$
  $\mathcal{G}_f = \langle \mathcal{V}, \mathcal{E}_f \rangle \leftarrow \mathcal{G}$
  while There exists an augmenting path in $\mathcal{G}_f$ from s to t **do**
    Take such a path.
    Send one flow unit over the path.
    Update the residual graph $\mathcal{G}_f$.
  end while
  Return f and $\mathcal{G}_f$.

Let us define $V = |\mathcal{V}|$ the number of nodes in the graph $\mathcal{G}$ and $E = |\mathcal{E}|$ the number of edges. Then the complexity of the Ford and Fulkerson algorithm (Alg:1) is at worst $O(E|C|)$ where $|C|$ is the cost of the cut. Note that when the algorithm stops there is no more any path from s to t and so s and t are separated in the residual graph. This means that the residual graph can be segmented in two parts: the S part and the T part. The boundary between those two parts is the minimal cost cut. Moreover the total flow sent to the

sink is equal to the cost of the minimal cut C in the graph. Indeed at each iteration of the algorithm (Alg:1), the amount of flow that reaches the sink, increases by one. In addition, the cost of the cut C **in the residual graph** decreases by one since every path from s to t crosses the cut C. One could remark that at convergence, the cost of the cut C in the residual graph is null since C cuts the residual graph in two disjoined parts.

Later Dinic in [61] and Edmonds and Karp [65] proposed independently a similar variation of this algorithm. Instead of using any path from s to t, Edmonds and Karp proposed to use the shortest one. This leads to a more efficient algorithm of complexity $O(VE^2)$. However in his paper [61], Dinic proposed some other improvements and reduced the complexity to $O(V^2E)$. The shortest path is found using a breath-first search or a depth-first search algorithm [44].

Another extension that could be found in the literature [], introduces the use at each step of the maximal augmenting path[1] instead of the shortest.

Recently, Boykov and Kolmogorov in [15, 18], brought a new very efficient implementation to the Computer Vision community based on a symmetric approach. They used two trees on the residual graph: the source and the sink trees. The two trees grow from their respective terminal. When a contact happens between them, then there is an augmenting path from s to t. The algorithm is summarized in Alg:2.

Algorithm 2 Boykov-Kolmogorov Algorithm

Require: $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ a graph.
Ensure: a maximal flow function f and the residual graph.
  Let $S - Tree$ and $T - Tree$ be respectively the source and sink tree.
  $S - Tree \leftarrow s$
  $T - Tree \leftarrow t$
  while A tree could grow **do**
    Grow alternatively one tree.
    while There is a contact between the two trees **do**
      Take the augmenting path.
      Send as much flow as possible over the path.
      Update the residual graph $\mathcal{G}_f$. At least one edge is saturated.
      Fix the trees.
    end while
  end while
  Return f and $\mathcal{G}_f$.

[1] the augmenting path that can bring the biggest quantity of flow from s to t

The critical point of this algorithm lies on the management of the trees (growing, fixing, ...) to obtain a each contact a quite short augmenting path. However, this algorithm can be placed between the one of Ford and Fulkerson (any path) and the one of Dinic (the shortest path) since it does not guarantee to use the shortest but makes use of heuristics to obtain a short one. This is a good compromise between efficiency provided by the shortest path and time spent to search for it. The complexity of this algorithm is given by $O(VE|C|)$.

The complexity criterion should be taken carefully since it does not reveal the real speed of an algorithm. For example, the algorithm of Boykov and Kolmogorov is in practice almost linear. Moreover, the choice of an algorithm must be done according to its use: in case of a complete graph - i.e. a graph with any possible edge - one should choose an algorithm of minimal complexity with respect to the number of edges E... In the Combinatorial community, the encountered graphs are usually very dense however in Computer Vision community, graphs are almost sparse. That is why we do not have the same "*efficiency criterion*".

3.3.3 Non Flow Conservation Preserving

On the other side, a class of methods are based on the idea of relaxing the *Flow Conservation* rule (3.11) during computing. The rule is relaxed to obtain the following rule:

$$\forall\, p \in \mathcal{V} \quad \sum_{q \in \mathcal{V}} f(p,q) \geqslant 0 \tag{3.12}$$

This positive value is called, for every node except the terminals $\{s,t\}$, the *excess* of a node since it represents an excess of flow at a given node. A node with a positive non-null excess is call *excess node*. Let us introduce some other notations that will be useful in this section. Let us call an *outgoing edge* of a node p, any edge on the residual graph leaving the node (its tail is the node p). Similarly an *incoming edge* of a node is an edge on the residual graph reaching the node. In the next section, an edge will be named indifferently edge or arc. According to those notations the excess of a given node p could be expressed as follow:

$$excess(p) = \sum_{\substack{e_i \text{ an incoming edge} \\ head(e_i)=p}} f(e_i) - \sum_{\substack{e_o \text{ an outgoing edge} \\ tail(e_o)=p}} f(e_o) \tag{3.13}$$

Let us now define the notion of *preflow*. A *preflow* is a function from $\mathcal{V}^2$

into $\mathbb{N}$ such that:

$$\begin{cases} \forall\ (p,q) \in \mathcal{V}^2 & f(p,q) \leqslant c(p,q) \\ \forall\ (p,q) \in \mathcal{V}^2 & f(p,q) = -f(q,p) \\ \forall\ p \in \mathcal{V} & \sum_{q \in \mathcal{V}} f(p,q) \geqslant 0 \end{cases} \tag{3.14}$$

The first algorithms based on this idea of relaxing the flow conservation were introduced independently by Cherkassky [38] and by Goldberg and Tarjan [74, 75]. The most known and accomplished algorithm is the one of Goldberg and Tarjan: the *Push-Relabel* algorithm. We will now give a short description of this algorithm since every other algorithm is similar or based on this one.

In this method, each node is assigned a label or distance or even height. The first version of Push-Relabel is quite simple and is formed of two different steps: *Push* and *Relabel*. In order to describe those procedures, one should introduce some notation. A *neighbor* node is a node that can be reached from or can reach a given node in the residual graph along a non-saturated edge. An *admissible edge* is a non saturated edge that reaches, from some node another node of lower height:

$$\begin{aligned} &neighbor(p) = \{q \in \mathcal{V} \ /\ \exists\ (p,q) \in \mathcal{E}_f,\ c_f(p,q) > 0 \text{ or } c_f(q,p) > 0\} \\ &reachable(p) = \{q \in \mathcal{V} \ /\ \exists\ (p,q) \in \mathcal{E}_f,\ c_f(p,q) > 0\} \\ &admissible(p) = \{(p,q) \in \mathcal{E}_f \ /\ q \in reachable(p),\ height(p) > height(q)\} \end{aligned}$$

The two atomic steps are described in the next two actions (Act:1-2):

Action 1 Push	
Applicability:	p an excess node and q such that (p,q) is an admissible edge for p.
Action:	Send $\min(excess(p), c_f(p,q))$ flow to the node q.

Action 2 Relabel	
Applicability:	p a non terminal excess node such that $admissible(p) = \emptyset$.
Action:	Relabel p according to: $height(p) \leftarrow \min_{q \in reachable(p)} height(q) + 1$

At the initialization, the preflow is set to be null everywhere excepted at the source terminal where an infinite excess is set. Such a preflow is in fact

a flow according to its definition (3.11). The labels are initialized according to the distance to the sink excepted for the source. The source is labeled $|\mathcal{V}|$ since it can easily be proved that it is an upper bounding value of the distance to the sink.

The Push-Relabel algorithm can be now written as follow:

Algorithm 3 Push-Relabel Algorithm

Require: $\mathcal{G}_f = \langle \mathcal{V}, \mathcal{E}_f \rangle$ a graph.
Ensure: a maximal flow function f and the residual graph.
  while There is an applicable action (push or relabel) **do**
    Do it.
  end while
  Return f and $\mathcal{G}_f$.

The algorithm is convergent since the maximal height in the graph is $2|\mathcal{V}| - 1 = 2V - 1$. At convergence all the excess flow is located at the source and at the sink and the preflow is corrected into a feasible flow. In fact, the overflow is pushed back to the source. Moreover, the amount of excess at the sink is equal to the maximal flow. Furtherover the graph can be segmented in two parts by thresholding the height at level $|\mathcal{V}| = V$. The part where nodes are upper than the source, corresponds the S-part and all others to the T-part.

The algorithm implemented according to the scheme defined above (Alg:3) has a complexity at worst equal to $O(EV^2)$. However, the Push-Relabel algorithm has a high degree of freedom such as: priority of actions, schedule of relabeling, choice of excess node etc... A lot of extensions take advantage of this freedom.

The first extension was proposed by Goldberg and Tarjan in [74, 76] and is also known as the *Relabel-to-front* algorithm. This modified algorithm only adds a new rule of action ordering. The basic idea is to never leave an excess node until there is no more excess in it. This new rule is implemented in the following procedure called *discharge*:

This variation is more efficient and has a complexity of $O(V^3)$ [74, 76]. This confers a very good performance in case of complete or dense graph.

The next natural improvement concerns the choice of a strategy for ordering the push step. The main idea is to define an ordering policy to select the current active excess node. Following this idea, two main strategies come out: First-In/First-Out (FIFO) or Highest Label (HL). Those two are discussed in a paper of Cherkassky and Goldberg [39]. The FIFO algorithm works at worst in $O(V^3)$ [75, 76] and the HL algorithm in $O(V^2\sqrt{E})$ [42].

Algorithm 4 Discharge procedure

Require: p be a excess node with $reachable(p) = \emptyset$.
  repeat
    while Push action is applicable **do**
      Push.
    end while
    if $excess(p) > 0$ and p is not a terminal **then**
      Relabel.
    end if
  until $excess(p) = 0$ or p is a terminal

The main drawback of the Push-Relabel algorithm, whatever implementation is used, happens when the minimal cut is saturated. At this moment every excess nodes that are located above, have to go back to the source. Since the source height is $|\mathcal{V}| = V$, excess nodes trapped above the minimal cut have to raise up the height above the source in each node of the S-part in order to be able to reach the source. However using the Relabel scheme to raise up a complete zone is inefficient. In order to improve performance, one can introduce some heuristics in the algorithm to deals with those cases.

The first and simplest one is called *Global Relabeling* and consists in updating each label according to the distance to the sink in the residual graph. This can be done in linear time by a backwards breadth-first search. Compared to the Relabel step, this heuristic is computationally expensive. Such a heuristic is performed periodically (after every n Relabel steps) and can improve the performance drastically.

The second and more tricky one is called *Gap Relabeling*. It was proposed independently by Cherkassky [38] and Derigs and Meier [56]. This heuristic is based on the following remark: if there is a gap in the label histogram, then there are two unconnected regions in the graph. Indeed let us consider a path joining the source to the sink in the graph. The labels along such a path are decreasing by one at each step. This proves that if a gap appears in the label histogram, then no path from the source to the sink exists in the graph. This means that the graph is partitioned in two unconnected parts.

Let us consider a label $0 < g < V$ such that no node in the graph is labeled g but there are some nodes carrying a higher label, then g forms a gap in the label histogram and then the sink is not reachable from those nodes. Then the labels of nodes with label higher than g may be increased by V. The performance improvement using this heuristic is more variable. If Gap Relabeling improves significantly the performance, Global Relabeling is usually more efficient. However, in some cases, Gap relabeling discovers

many gaps and provides better running times but this is usually not the case for typical graph designs in Computer Vision.

The best implementation of Push Relabel makes use of *Dynamic Trees* rather than labels. The more efficient implementations of Push Relabel are as follows. In [76] Goldberg and Tarjan proposed an implementation that runs in $O(VE\log(V^2/E))$ time. An algorithm of King at al. in [101] runs in $O(VE + V^{2+\epsilon})$. An algorithm of Cheriyan et al in [37] runs in $O(VE + (V\log V)^2)$ and an algorithm of Ahuja et al. in [3] runs in $O\left(VE\log\left(\frac{V}{E\sqrt{|C|}}+2\right)\right)$.

As explained in the previous section, one should consider complexity very carefully since it does not reflect the real speed of an algorithm. For a comparison of those two techniques for Computer Vision problems, one could read [18].

Actually, it is very difficult to extract rules that link performance to graph shape. However, it seems that sparse graphs are the domain of augmenting path algorithms while dense or complete graphs belong to Push-Relabel algorithms. Nevertheless, this cannot be taken a honest truth and one has to test on is own graphs.

3.4 Examples

In this section, one will found an implementation of '*Geodesic Active Contour*' (Sec:1.1.2) and '*Geodesic Active Region*' (Sec:1.2.3) in a graph cut framework.

3.4.1 Geodesic Active Contour

Let I be an image and consider the graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. Each pixel is assigned to a node. Let us consider a local neighborhood $\mathcal{N}$ around each node which defines the set of edges. Typically $\mathcal{N}$ corresponds to a 4-connected, 8-connected or even 16-connected neighborhood. Figure 3.3 illustrates those neighborhoods.

In [16], Boykov and Kolmogorov describe how to set the edge weights in order to minimize the Geodesic Active Contour energy (1.8). However, an Euclidian or Riemanian metric can be approximated only and this approximation is related to the neighborhood complexity. A 4-connected neighborhood corresponds to the Manhattan distance but using a 16-connected neighborhood produces a quasi-perfect Euclidian distance (See Fig:3.3).

Let us assume that the object contour does not intersect the border of the image. In that case the sink is connected to each node of the border with an

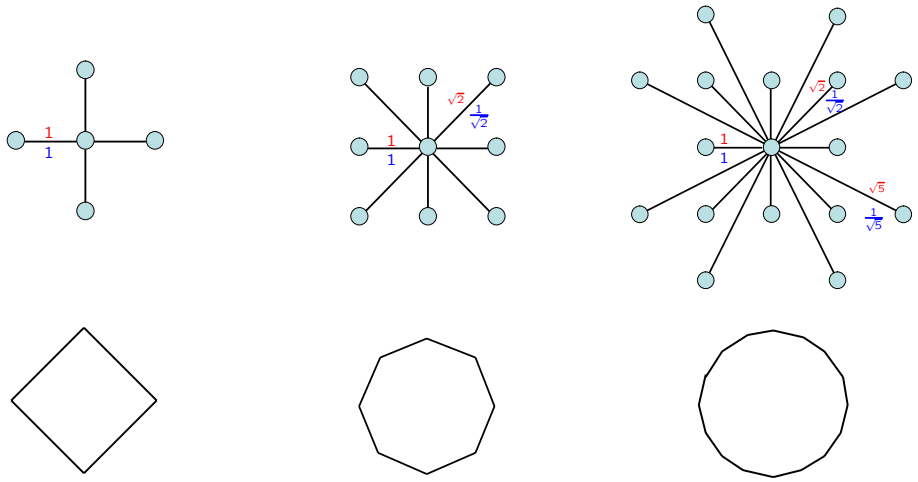


Figure 3.3: First Row: 4, 8 and 16-connex regular neighborhood. - Second Row: The unit sphere associated to their corresponding metric. In red, length of edges and in blue, weight of edges.

infinite link. The user has to give some tips by selecting some pixels inside the object (something like a seed). The corresponding nodes are connected to the source by infinite links. Now one should correctly set the n-links weights so that the minimal cut will separate the graph in two parts: object and background. According to [16], n-links have to be set using the following scheme:

$$\forall p, q \in \mathcal{V}, \quad (p, q) \in \mathcal{E}, \quad c(p, q) = \frac{g(|I_p - I_q|)}{length(p, q)} \tag{3.15}$$

After running a graph cut algorithm, the nodes connected to the source (S-part) are assigned to the object, the other ones to the background.

Using Graph Cuts to minimize such an energy provides some advantages:

- *Global minimum:* Obviously. This is a property of graph cuts.
- *Multi object segmentation:* If more than one object is present in the image, the user just has to provide more than one seed (at least per object one).
- *Interactive correction:* If the user, after segmentation, finds that the object is not correctly labeled, he could correct some labels by adding some infinite t-links in the residual graph (the source for the object, the sink for the background) and relaunch the algorithm on this modified residual graph. This approach allows to interactively correct the partition and to take advantage of the previous run to speed it up [14].

However, this methods needs to be initialized by hand (t-links) and the resulting partition depends on the initialization. The next section will show a way to avoid that.

3.4.2 Geodesic Active Region

The '*Geodesic Active Region*' can be solved in the same context. The graph design is the same as above concerning the n-link connection (links and weights). However the t-links are set differently and there is no need of the user intervention anymore. Geodesic Active Region differs from Geodesic Active Contour only by this additive term (1.20). Since the common part (1.8) is already encoded in n-links according to scheme (3.15) and since graphs are additive [108, 109], one should only implement the remaining part.

Equation (1.20) is encoded in the graph using n-links by applying the following rule at each node:

$$\forall p \in \mathcal{V}, \quad r(p) = \lambda \log\left(\frac{P_{out}(I(p))}{P_{in}(I(p))}\right),$$

$$\begin{cases} r(p) > 0 & \text{add a t-link from } p \text{ to the sink of weight } r(p). \\ r(p) < 0 & \text{add a t-link from the source to node } p \text{ of weight } r(p). \\ r(p) = 0 & \text{nothing.} \end{cases}$$

Similarly to the Geodesic Active contour case, after running a graph cut algorithm, nodes connected to source (S-part) are assigned to the object, others to the background.

This method does not make use of user initialization any more and always provides the global minimum of the energy.

Chapter 4

Stochastic Level Sets: an extension

This chapter presents a new extension of Level Set based on Stochastic Partial Derivative Equation (SPDE). Basically, it introduces the concept local random perturbations into the Level Set framework. This allows building a more generic framework: the Stochastic Level Sets.

This allows, for example, the use of more interesting minimizer than the straightforward gradient descent method. Combined with a Simulated Annealing minimization process, it allows obtaining a better minimum than with a standard gradient descent. Moreover it can also deal with the case where an accurate gradient is intractable or inaccessible (see section Sec:1.2.4).

This work stems from a join work with Gheorghes Postelnicu and is published in proceedings of the second VLSM workshop [90] and in a special issue of IJCV [92].

4.1 Introduction

4.1.1 Why adding noise?

Shape optimization techniques are a common framework to address various applications in Computer Vision, like segmentation, tracking, stereo vision etc. The objective of our approach is to improve these methods through the introduction of stochastic motion principles. These problems are most of the time stated as the minimization with respect to some hyper-surface Γ of $\mathbb{R}^N$ of some objective function $E(\Gamma)$. This is usually achieved using a gradient-descent method. Yet, in complex cases, E does not have any computable gradient with respect to Γ. In other cases, the minimization process gets

stuck into some local minimum, while no multi-resolution approach can be invoked[1]. To deal with those two frequent problems, one can naturally turn to a stochastic optimization approach. Even a simple Simulated Annealing method might be powerful enough to escape from local minima and to cope with an approximation of the shape gradient. Indeed, adding noise to the motion of a curve is a prerequisite to developing this idea.

4.1.2 Context

We are interested in letting $\Gamma(t)$ evolve according to the equation

$$\frac{\partial \Gamma}{\partial t}(t,p) = \beta(t,p)\overrightarrow{\mathbf{n}}(t,p) + \dot{\eta}(t,p)\overrightarrow{\mathbf{n}}(t,p) \tag{4.1}$$

where p is some parameterization of Γ, $\overrightarrow{\mathbf{n}}$ the normal to $\Gamma(t)$ at point $\Gamma(t,p)$ and where the normal velocity β depends on some stochastic perturbation $\dot{\eta}$ - here, the notation $\dot{\eta}$, standing for the "derivative" of the noise η w.r.t. time, will become clear further. The mean curvature motion $\beta = \kappa$ as well as many other problem oriented choices of β and their implementation with the Level Sets method [132] are well known. The novelty in our work is the implementation of the recently proposed stochastic flow (4.1) (see [171]) and its application to Computer Vision.

Stochastic dynamics of interfaces have been widely discussed in later years in the physics literature. The work in fields like front propagation or front transition is aimed at modeling and studying the properties of a moving frontier between two media that is subject to macroscopic constraints and random perturbations (which are due to the bulk). The natural translation of this dynamic in mathematical language is done through Stochastic Partial Differential Equations (SPDEs). These equations were introduced by Walsh in [171] and their mathematical properties were studied using mostly partial differential equations tools. Nevertheless, the problems researchers have to deal with are various and there is more than one way to add a stochastic perturbation to a PDE. An up to date survey of the existing models on stochastic motions by mean curvature can be found in [180]. It was only in recent years that the notion of viscosity solution for a SPDE was developed by Lions and Souganidis in a series of articles [113, 114, 116, 117]. Their notion of weak viscosity solution is very attractive for numerical applications, since they define the solution as a limit in a convenient space for a set of

[1] E.g., when using some statistical region model, a change in resolution may result in smoothing out the difference between the statistics of the interior and the ones of the exterior.

approximations. Since their pioneering work, related work has been done by Yip [179] and by Katsoulakis et al [96]. Another independent approach concerned with viscosity solutions of stochastic partial differential equations is due to Buckdahn and Ma [25]. Their approach is not well suited for Level Sets evolutions, though, since they do not allow certain functional dependencies that are common to all Level Sets evolutions.

4.2 Mathematics

4.2.1 Some Notions of Stochastic Calculus

This section is meant to offer the reader an intuition of the notion of Stochastic Calculus and of the supplementary challenges it poses. Focusing on the definition of the integral itself, we suppose the reader is familiar with the Brownian otion. We shall equally use the idea of a standard probability space, martingale, quadratic variation. Rigorous and complete introductions of Stochastic Calculus can be found in [94], [71] or [111]. Let $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$ be a standard probability space. We will consider that $W = (W_t^1, \ldots, W_t^m)_{t \geqslant 0}$ is a standard m-dimensional Brownian motion issued from 0. We are interested in finding an appropriate way of introducing the notion of stochastic differential with respect to the process W. To better understand the difficulty here, it is worth while mentioning that the paths of the Brownian motion are only $\frac{1}{2}$-Holder continuous, so they are nowhere differentiable. Hence, in order to give a meaning to what the term dW_t might mean, one can first define an integral with respect to W - the stochastic integral. Once this integral is defined, the differential is obtained using the integral defined. To keep the presentation as clear as possible, we suppose that $m = 1$ and all our processes are real-valued. The considered approach for the construction of such an integral is to define it as an isometry in the appropriate functional space. Indeed, consider a square integrable process $\Phi = (\Phi(t, \omega))_{t \geqslant 0}$. Trying to define the stochastic integral $\int_0^T \Phi(t, \omega) dW_t$, one would start with Riemann approximations

$$I_\Delta(\Phi)(T) = \sum_{i=1}^{n-1} \Phi(t_i, \omega) \left(W_{t_{i+1}} - W_{t_i} \right) \tag{4.2}$$

with $\Delta = \{0 = t_0 < t_1 < \cdots < t_n = T\}$ and hope to find a suitable space where the above sum would converge when $|\Delta| \to 0$. By proving some completeness results, one can show that every square integrable continuous process is integrable w.r.t. the Brownian motion and one can obtain

the **Ito stochastic integral** as the unique, square integrable martingale $I(\Phi) = (I(\Phi)(t))_{t\geqslant 0}$ which is the limit of $(I_\Delta(\Phi)(t))_{t\geqslant 0}$ when $|\Delta| \to 0$. The convergence happens in the pseudo-metric $\|X\| = \sum_{n=1}^{\infty} \frac{\min(1,\sqrt{\mathbb{E}(X_n^2)})}{2^n}$.

The price to pay for the convergence of the Riemann sums is that it happens in a process space, hence the limit, which is denoted as the stochastic integral $\int_0^T \Phi(s)dW_s$, does not hold a meaning path-wise, but only as a process. This means that the set of events $\omega \in \Omega$ where the Riemann sums mentioned above do not converge to the above introduced integral is of null measure. In the sequel, the convergence suggested above can be extended to an arbitrary dimension. Moreover, the extension can be taken with respect to local continuous martingales. For details, the reader is referred to one of the references above. Once this integral is defined, it is imperative one is given some chain rule formula. This is where the Ito lemma comes in. Consider a process $X = (X_t)_{t\geqslant 0}$ where:

$$X_t = X_0 + \underbrace{\int_0^t H_s^X \, dW_s}_{\text{Martingale}} + \underbrace{\int_0^t K_s^X \, ds}_{\text{Increasing Process}} \tag{4.3}$$

with $\int_0^T |H_s^2| dW_s < +\infty$ and $\int_0^T |K_s| d_s < +\infty$. The Ito lemma states that for all function $\alpha : \mathbb{R} \to \mathbb{R}$ of class $\mathcal{C}^2$ that $Y = (\alpha(X_t))_{t\geqslant 0}$ verifies the dynamics:

$$\alpha(X_t) = \alpha(X_0) + \int_0^t \alpha'(X_s) dX_s + \frac{1}{2}\int_0^t \alpha''(X_s) d\langle X, X\rangle_s \tag{4.4}$$

$$dY_t = \alpha'(X_t) dX_t + \frac{1}{2}\alpha''(X_t) d\langle X, X\rangle_t \tag{4.5}$$

where $dX_t = H_t^X \, dW_t + K_t^X \, dt$.

The main difference when compared to the regular chain rule is the appearance of an extra term in (4.5), also called Ito term, **or drift**, which involves the second derivative of α and the quadratic form $\langle X, X\rangle$, also known as the quadratic variation of the process X. The quadratic variation is not zero when X has some dependence upon a stochastic process and can be computed in the following manner. Suppose that X_t and Y_t are two Ito processes defined as in (4.3). Then

$$\langle X, Y\rangle_t = \int_0^t H_s^X H_s^Y \, ds$$

Note that $\langle X, X\rangle$, depends solely on the stochastic part of the dynamics and is independent of the bounded variation part. When $H^X \equiv 0$, then the classical chain rule is obtained in (4.5).

Now, using the Ito formula, a variation of the stochastic integral introduced above can be obtained so that the classical chain rule is satisfied. Consider a process X_t and Y_t two Ito processes defined as above. Then the **Stratonovich integral** of Y with respect to X is then given by the formula

$$\int_0^t Y_s \circ dX_s = \int_0^t Y_s \, dX_s + \frac{1}{2}\langle H^X, H^Y \rangle_t \tag{4.6}$$

Suppose now α is of class $\mathcal{C}^3$ and apply the Ito formula (4.4) to $\alpha'(X)$. Then

$$\alpha'(X_t) = \alpha'(X_0) + \int_0^t \alpha''(X_s)\, dX_s + \frac{1}{2}\int_0^1 \alpha^{(3)}(X_s)\, d < X, X >_s$$

and consequently $d\langle \alpha'(X), X\rangle_t = \alpha''(X_t) d\langle X, X\rangle_t$. Hence,

$$\alpha(X_t) = \alpha(X_0) + \int_0^t \alpha'(X_s) \circ dX_s \tag{4.7}$$

by noticing that the quadratic variation term that we obtained is equal to the Ito term in equation (4.4).

We conclude this section by the approximation of the stochastic integral mentioned above. For the Ito integral, we saw that, if $\Delta = \{0 = t_0 < t_1 < \cdots < t_n = T\}$, then

$$\lim_{|\Delta|\to 0} \sum_{i=0}^{n-1} Y_{t_i}(X_{t_{i+1}} - X_{t_i}) = \int_0^T Y_s \, dX_s$$

For the Stratonovich case, it can be proved that we have the two equally useful limits:

$$\lim_{|\Delta|\to 0} \sum_{i=0}^{n-1} \frac{Y_{t_i} + Y_{t_{i+1}}}{2}(X_{t_{i+1}} - X_{t_i}) = \int_0^T Y_s \circ dX_s \tag{4.8}$$

$$\lim_{|\Delta|\to 0} \sum_{i=0}^{n-1} Y_{\frac{t_i + t_{i+1}}{2}}(X_{t_{i+1}} - X_{t_i}) = \int_0^T Y_s \circ dX_s \tag{4.9}$$

One could remark that Ito integral is approached using a piecewise constant approximation of the process Y while Stratonovish integral by a piecewise linear approximation. As an example that is meant to emphasize the difference between the two integrals previously introduced, consider as before a Brownian motion W and $\lambda \in [0, 1]$. Let $\Delta = \{a = t_0 < t_1 < \cdots < t_n = b\}$,

$\Delta W_i = W_{t_{i+1}} - W_{t_i}$ and consider $\phi^\lambda(t_i) = (1-\lambda)W_{t_i} + \lambda W_{t_{i+1}}$. It can be proved that

$$\int_a^b W_s \circ dW_s = \lim_{|\Delta|\to 0} \phi^{\frac{1}{2}}(t_i)\Delta W_i = \frac{1}{2}\left[W_b^2 - W_a^2\right]$$
$$\int_a^b W_s dW_s = \lim_{|\Delta|\to 0} \phi^{0}(t_i)\Delta W_i = \frac{1}{2}\left[W_b^2 - W_a^2\right] - \frac{1}{2}(b-a)$$

Note also that one should be careful when simulating ΔW_i. A common mistake would be to implement it as $\Delta W_i = \Delta t \mathcal{N}_{(0,1)}$ giving the scheme:

$$u(t+\Delta t, \omega) = u(t,\omega) + \Delta t \mathcal{N}_{(0,1)}(t)$$

This would be incorrect, since the statistical properties of the evolution would then depend upon the discretization of the time grid. To see this, consider independent variables $x_i \sim \mathcal{N}_{(0,1)}$ and notice that the previous evolution at time T would amount to $\sum_{i=1}^n x_i \Delta t = \sum_{i=1}^n \frac{T}{n} x_i$ where $\Delta t = T/n$ is the discretization step. Given the independence of the x_i, the previous sum is a Gaussian variable $\mathcal{N}_{(0,T^2/n)}$, thus depending upon the discretization n of the time interval $[0,T]$. The correct scheme involves $\sqrt{\Delta t}$ instead of Δt:

$$u(t+\Delta t, \omega) = u(t,\omega) + \sqrt{\Delta t}\mathcal{N}_{(0,1)}(t) \tag{4.10}$$

Indeed ΔW_i should be equal to $\sqrt{\Delta t}\mathcal{N}_{(0,1)}$:

$$\Delta W_i \sim \mathcal{N}_{(0,t_{i+1}-t_i)} \sim \sqrt{t_{i+1}-t_i}\mathcal{N}_{(0,1)} \tag{4.11}$$

4.2.2 Proposed Model for the Stochastic Curve Evolution

In applications, the stochastic term will add to a deterministic force $F(D^2u, Du, x, t)$ (one of the simplest examples, analyzed in detail in [181], is concerned with the coupled evolution $du = \kappa|Du|dt + \dot{W}|Du|dt$). Hence, a naive way to write down the coupled evolution is

$$du = F|Du|dt + \dot{W}|Du|dt$$

The above equation will have a meaning if written as

$$du = F|Du|dt + |Du|dW_t \tag{4.12}$$

Concentrating on the stochastic part again, we remark that we made an implicit choice by considering the Ito integral in the above formula, but

we could have decided to go for the Stratonovich integral. So what is the difference between the two integrals from a Level Sets point of view? Let us consider the following invariance property that is required when working within a Level Sets framework: consider just a random evolution of the type

$$du = |Du| dW_t \text{ with } u(0,\cdot) = u_0 \tag{4.13}$$

where W is a one-dimensional Brownian motion. Then this evolution codes for the corresponding contour evolution

$$\frac{\partial \Gamma}{\partial t} = \dot{W}\mathbf{n} \text{ with } \Gamma(0) = \Gamma_0 \tag{4.14}$$

where $\dot{W}$ is Gaussian white noise and Γ_0 is the zero-level of u_0. The idea behind the Level Sets evolution framework is to have all the level sets of the implicit function given by (4.13) evolve according to the same dynamics (4.14). A smooth change of scale of a function satisfying (4.13) that leaves the zero-level unchanged should not influence the dynamics of the level sets contour - since the corresponding contour evolution (4.14) is not affected by this change of scale. Consider then a function $\alpha : \mathbb{R} \to \mathbb{R}$ such that $\alpha' > 0$ and $\alpha(0) = 0$ and the initial value problem

$$du = |Du| dW_t \quad \text{with} \quad u(0,\cdot) = u_0(\cdot) \tag{4.15}$$

If we consider u the solution of (4.15), then $v = \alpha(u)$ should verify the same dynamics, but with a different initial condition

$$dv = |Dv| dW_t \quad \text{with} \quad v(0,\cdot) = \alpha(u_0(\cdot))$$

as is the case in the deterministic framework. Nevertheless, one can apply the Ito rule to the dynamics (4.13) and see that

$$dv = d\alpha(u) = \alpha'(u)du = |Dv| dW_t + \frac{1}{2}\alpha''(u)|Du|^2 dt$$

and the assertion is not verified due to the additional Ito term. Hence, the problem (4.13) is ill-posed from a Level Sets point of view: **for a given initial curve $\Gamma(0)$, the choice of the initial implicit function u_0 modifies the solution of the equation!**. However, as observed by Lions and Souganidis, this invariance condition is verified if one replaces the Ito integral with the Stratonovich integral, since the latter does not include any additional term anymore. Hence, the *right way* to insert stochastic evolutions in the Level Sets framework is through the Stratonovich integral. We rewrite (4.13) accordingly

$$du = |Du| \circ dW_t \quad \text{with} \quad u(0,\cdot) = u_0(\cdot) \tag{4.16}$$

Then, if we consider $v = \alpha(u)$ (hence the corresponding initial condition will be $\alpha(u_0)$) the dynamics verified by v are

$$dv = \alpha'(u) \circ du = \alpha'(u)|Du| \circ dW_t = |Dv| \circ dW_t$$

and the invariance property is verified this time. Now, given the previous ingredient, the proposed random curve evolution model is given by

$$du = Fdt + |Du| \circ dW_t \tag{4.17}$$

Here, we used the Stratonovich integral, as opposed to (4.12).

A second example that suggests Stratonovich integration should be used when working with stochastic partial differential equations is concerned with the 1-dimensional perturbed heat equation $du = u_x dW_t + \lambda u_{xx} dt$. It can be shown that this equation reduces to a backward heat equation when considering Ito integration for $\lambda \in (0, \frac{1}{2})$ - hence ill-posed. Once again, the extra term Stratonovich integration solves this problem (for more details, see [115]).

What is the difference between the evolution (4.17) and a classical Level Sets evolution such as $du = Fdt$? Suppose the initial condition function is a signed distance function. Since the stochastic term only depends upon $|Du|$ (which equals 1 in this case) and the time parameter, all the points of the contour will have an extra random force which will be the same on the entire contour at each time step. This type of perturbation is indeed very important from a theoretical point of view, but we would like something more flexible in our applications. Typically, we would be interested in having white noise in both the time and spatial parameters. Nevertheless, white noise in space appears to add a lot of technical difficulties to the problem and the return on investment is quite small, since most of our models will evolve on discrete grid spaces. That is why we have opted for colored spatial noise, that is typically given by

$$W(t, x) = \sum_{i=1}^{m} \phi_i(x) W_t^i$$

where $\phi_i : \mathbb{R}^N \to \mathbb{R}$ are smooth functions with compact support. Note that other choices of colored spatial noise are possible. The final evolution model we propose is thus

$$du = F|Du|dt + |Du| \sum_{i=1}^{m} \phi_i(x) \circ dW_t^i \tag{4.18}$$

As a simplification, in practice we choose the functions with the same profile, but centered around a number of points x_i, that we call *noise sources*. Thus, our typical choice is

$$\phi_i(x) = \phi(x - x_i)$$

where ϕ is some convenient regular function.

4.2.3 Stochastic Viscosity Solutions

The theory developed earlier needs some sort of convergence results. As mentioned before, the proper type of solutions need to be used, so that the previous results from the Level Sets theory apply here. The notion of stochastic viscosity solution for fully nonlinear, second-order, possibly degenerate, stochastic partial differential equations such as the ones considered previously is put forward in a series of articles: [113], [114], [116] and [117]. Their theory is meant to apply precisely to equations such as (4.18), with $F = F(D^2u, Du, x, t)$. So far, a limit of their theory, which stands even today as an open question, is that they do not treat equations where the noise depends upon the space parameter (they only treat the case $\phi_i \equiv 1$, with our previous notation). However, experimental data suggests that their theory applies in cases like ours as well (see section Sec:4.3). Precisely, consider the equations

$$du = F(D^2u, Du, x, t)dt + \epsilon|Du| \circ dW_t \quad \text{with} \quad u(\cdot, 0) = u_0(\cdot) \tag{4.19}$$

$$du = F(D^2u, Du, x, t)dt + |Du|\dot{\xi}_\alpha(t) \quad \text{with} \quad u(\cdot, 0) = u_\alpha(\cdot) \tag{4.20}$$

where $\epsilon \geqslant 0$ and ξ_α is a family of smooth functions $\xi_\alpha : \mathbb{R}_+ \to \mathbb{R}$. Then we can cite the following theorem, summarizing their results:

Theorem 1 *The following hold a.s. in ω:*

1. *There exists a unique solution to* (4.19).

2. *Let $\{\xi_\alpha(t)\}_{t\geqslant 0}$ and $\{\eta_\beta(t)\}_{\beta>0}$ be two families of smooth functions such that as α and $\beta \to 0$, ξ_α and η_β converge to W uniformly on any compact in t and a.s. in ω. Let $\{u_\alpha\}_{\alpha>0}$ and $\{v_\beta\}_{\beta>0}$ in $BUC(\mathbb{R}_+ \times \mathbb{R}^N)$[2] be the solutions of* (4.20). *If $\lim_{\alpha,\beta\to 0} \|u_\alpha(\cdot, 0) - v_\beta(\cdot, 0)\|_{\mathcal{C}(\mathbb{R}^N)} = 0$, then, for all $T > 0$, $\lim_{\alpha,\beta\to 0} \|u_\alpha - v_\beta\|_{\mathcal{C}([0,T]\times\mathbb{R}^N)} = 0$. In particular, any smooth approximations of W produce solutions converging to the unique function stochastic viscosity solution of* (4.19).

3. *As $\epsilon \to 0$, the solution u^ϵ of* (4.19) *converges in $\mathcal{C}(\mathbb{R}_+ \times \mathbb{R}^N)$ to the solution of* (4.19) *with $\epsilon = 0$.*

[2] the space of **Bounded Uniformly Continuous** functions

Consequently, their result **allows us to simulate the solutions of such equations and be sure that the result of our computer simulation is what we expect it to be**. Furthermore, we mention that according to Lions, the convergence takes place in $C(\mathbb{R}_+ \times \mathbb{R}^N)$, which means that the numerical solutions we develop will be continuous and that they will be converge uniformly almost surely in $\omega \in \Omega$.

We end this theoretical part with an example by Souganidis on the explicit solution of the equation

$$du = |Du|\dot{\eta}dt \quad \text{with} \quad u(0,x) = |x| \tag{4.21}$$

where $\eta : \mathbb{R}_+ \to \mathbb{R}$ is a function of class $\mathcal{C}^1$ such that $\eta(0) = 0$. The explicit viscosity solution of this equation is given by

$$u(t,x) = \max\left[(|x| + \eta(t))_+ \,,\, \max_{s\in[0,t]}(\eta(s))_+\right]$$

where $(x)_+ = \max(0,x)$ (for a simple proof of the statement above, consider the case when $\eta(t) \equiv 1$ and compute the viscosity solution of the equation in that case); then, one can see that uniform convergence of $\eta \to W$ is sufficient to obtain the solution of the associated SPDE. Moreover, this simple case allows one to see that the random path η has a different effect on the solution that depends mainly on its sign. Indeed, as it can be observed from formula (4.21), there is a qualitative difference between the behavior of the solution depending on whether $\dot{\eta} > 0$ or $\dot{\eta} < 0$. This can be better understood watching a sample evolution in figure (4.1). Moreover, numerical artifacts will develop due to very frequent changes of sign of $\dot{\eta}$, since the use of η is only for heuristic purpose (the Brownian motion is nowhere differentiable). As a result, the regular reinitialization of the implicit function – a standard technique of the Level Set framework – is indispensable in the stochastic case.

4.2.4 Numerical scheme

The main problem when implementing Stratonovich evolutions is that they amount often to implementing implicit numerical schemes. Consider again the simple evolution $du = |Du| \circ dW(t)$. According to the approximating scheme (4.8), the direct way of simulating such a process is through the following implicit scheme:

$$u_{i+1} = u_i + \frac{1}{2}\left(|Du_i| + |Du_{i+1}|\right)\Delta W_i$$

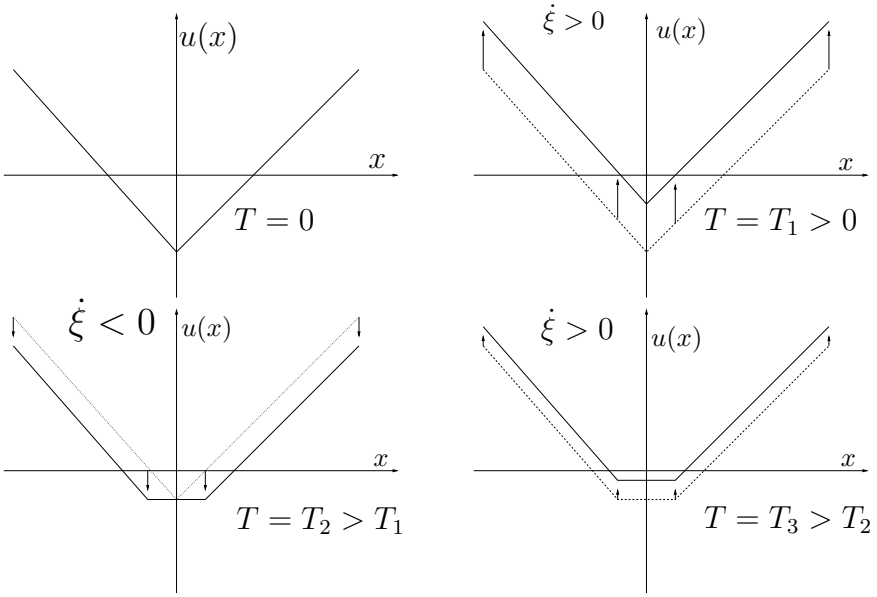


Figure 4.1: Examples of a typical evolution following the dynamics of equation (4.21). The extremely frequent changes of sign of the increments will alter the profile of the implicit function which does not remain a distance function. Hence, from an implementation point of view, some regular *reinitialization* of u is advisable.

To avoid working with an implicit scheme, notice that the schema presented for the simulation of the Ito integral is an explicit one and use the fact that the Stratonovich integral is equal to the Ito integral plus an additional drift. Consider the evolution

$$du = H(Du, x) \circ dW_t \tag{4.22}$$

where we have compacted the notation used previously. Here $H(p, x)$ is a function from $\mathbb{R}^N \times \mathbb{R}$ with real values. The typical example is $H(p, x) = |p|\phi(x)$, where, ϕ is some convenient regular function which is smooth enough. Such an evolution is equivalent, according to the definition of the Stratonovich integral, with the Ito evolution given by

$$du = H(Du, x)dW_t + \frac{1}{2}d\langle H(Du, x),\, W\rangle_t \tag{4.23}$$

To compute the drift, we start by rewriting the above dynamics in an integral form

$$u(t, x) = u_0(x) + \int_0^t H(Du(s, x),\, x) \circ dW_s$$

We can then take the derive with respect to the spatial parameter x and obtain

$$Du(t,x) = Du_0(x) + \int_0^t \left[D^2u(s,x)D_pH(Du(s,x),x) + D_xH(Du(s,x),x)\right] \circ dW_s$$

where D_pH (resp. D_xH) denotes the gradients of H w.r.t. p (resp. x) and D^2u denotes the spatial Hessian. Then, applying the Ito rule, we have

$$H(Du(t,x),x) = H(Du_0(x),x) + \int_0^t \left[D_pH \cdot (D^2uD_pH) + D_pH \cdot D_xH\right] \circ dW_s$$

Finally, if we consider the simplifying notation $A[u] = A(u,u)$, when A is some quadratic form, then the drift from equation (4.23) can be written as

$$\begin{aligned}\frac{1}{2}\langle H(Du,x),\, W\rangle_t = \frac{1}{2}\int_0^t (&D^2u(s,x)\,[D_pH(Du(s,x),x)] \\ &+ D_pH(Du(s,x),x) \cdot D_xH(Du(s,x),x))ds\end{aligned}$$

When $H = |p|\phi(x)$, the previous formula becomes

$$\langle H(Du,x),\, W\rangle_t = \int_0^t \left[\phi^2(x)D^2u(s,x)\left[\frac{Du(s,x)}{|Du(s,x)|}\right] + \phi(x)\, D\phi(x) \cdot Du(s,x)\right] ds$$

We can remark that the second order term in the above formula is a smoothing term. It can also be written

$$D^2u\left[\frac{Du}{|Du|}\right] = \Delta u - |Du| div\left(\frac{Du}{|Du|}\right) = \Delta u - |Du|\kappa$$

where κ denotes the mean curvature of the level set at point x. One can be alarmed by the presence of $-|Du|\kappa$. Nevertheless, the overall term is positive, since D^2u is a positive semi-definite matrix.

The above calculation remains valid if the dynamics depends on more than one Brownian motion. In conclusion, to simulate an evolution of the type

$$du = F|Du|\, dt + |Du| \sum_{i=1}^{m} \phi_i(x) \circ dW_t^i \tag{4.24}$$

we use

$$\begin{aligned}du =& F|Du|\, dt + |Du| \sum_{i=1}^{m} \phi_i(x) dW_t^i \\ &+ \frac{1}{2}\left((\sum_{i=1}^{m} \phi_i^2(x))D^2u\left[\frac{Du}{|Du|}\right] + (\sum_{i=1}^{m} \phi_i(x)D\phi_i(x)) \cdot Du\right) dt\end{aligned} \tag{4.25}$$

or, in the general case when the stochastic Hamiltonian is given by $H(p, x)$:

$$du = F|Du|\,dt + H(Du, x)dW_t + \frac{1}{2}\left(D^2u\,[D_pH] + D_pH \cdot D_xH\right)dt$$

4.3 Implementation

In this section, we test our scheme and investigate some simple geometrical properties the evolution that could guide the user toward a correct choice of noise.

4.3.1 One Gaussian noise simulation

Let us begin with the simple case of a Gaussian noise constant in space. We thus consider $du = |Du| \circ dW_t$ and implement:

$$du = |Du|dW_t + \frac{1}{2}D^2u(t,x)\left[\frac{Du(t,x)}{|Du(t,x)|}\right]$$

We use a standard WENO3 scheme [84] in space with step Δx and a first order explicit scheme in time with step Δt and verify the convergence of the approximation when the space step and/or the time step tend to zero. Again, please recall the use of $\sqrt{\Delta t}$:

$$u(t+\Delta t, x) = u(t,x) + |Du(t,x)|\sqrt{\Delta t}\mathcal{N}_{(0,1)}(t) + \frac{1}{2}D^2u(t,x)\left[\frac{Du(t,x)}{|Du(t,x)|}\right]$$

Because of the stochastic character of the evolution, one can only compare the different approximations through some statistical quantity[3]. For a given initial condition and a given final time T, *the variance of the area of the interior of the final curve* provides a simple and meaningful way to compare two approximations. The left part of figure 4.2 shows, for different values of Δx, the convergence of this variance when $1/\Delta t$ increases. As a reminder to avoid a naive mistake, we also implemented the evolution with Δt instead of $\sqrt{\Delta t}$ and verify that the variance of the area tends to zero!

As a test of the invariance of the Stratonovich differential, we compare, for a given initial curve $\Gamma(0)$, the mean of the area of the curve at a given final

[3]Actually, for a given time step Δt, we might fix the event ω and compare the approximations for different Δx but with the same Brownian. We also successfully used such a path-wise comparison when testing the invariance property of our scheme.

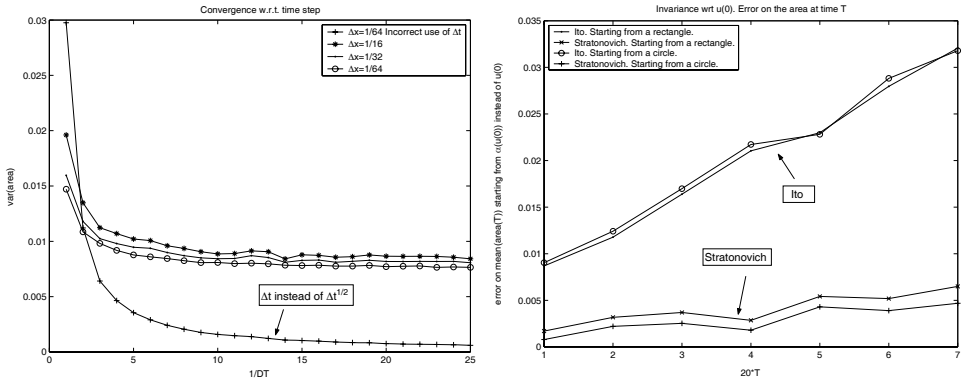


Figure 4.2: One Gaussian source. Left: convergence of the variance of the area at a given time T when Δt tends to 0 (plus the erroneous case when using Δt). Right: invariance of Stratonovich w.r.t. the choice of the initial implicit function.

time T for different choices of the initial implicit function $u(0)$ (namely the signed distance function $d_{\Gamma(0)}$ to $\Gamma(0)$ and $\alpha(d_{\Gamma(0)})$ with $\alpha(x) = e^x - 1$). The right part of figure 4.2 shows, for different values of T and different initial curves, the relative difference between the means of the final area for the initial conditions $d_{\Gamma(0)}$ and for $\alpha(d_{\Gamma(0)})$ in both the Ito and the Stratonovich cases . Note how the Stratonovich scheme is much more insensitive with respect to the choice of $u(0)$.

It could be proved [91, 92] that, for a given initial curve, the variance of the area of the curve at time T is a polynomial of degree N in T where N is the space dimension. In practice, for reasonable values of T, the relation between T and this variance is linear. As a final test, the left part of figure 4.3 shows, for $N = 1$ and $N = 2$, how this relation is respected by our scheme.

4.3.2 Several Gaussian noise sources simulation

Having the whole curve shrink or grow at the same time is not very useful. We will use a spatially dependent noise although the viscosity solution result is still an open question in this case. For a given number m of random sources, we implement the evolution (4.24) with $F = 0$ using the scheme (4.25). The m sources are equally distributed on a grid $\{x_i\}$ and $\phi_i(x) = \phi(x - x_i)$ where ϕ is such that $\phi_i(x_j) = \delta_{ij}$ and ϕ_i decreases smoothly from x_i to its neighbors. In practice, although not derivable in x_i, the classical multi-linear interpolation functions are sufficient. Note also that $\sum_{i=1}^{m} \phi_i(x) dW_t^i$ is no longer of variance 1 for all x, so that the stochastic motion would be weaker

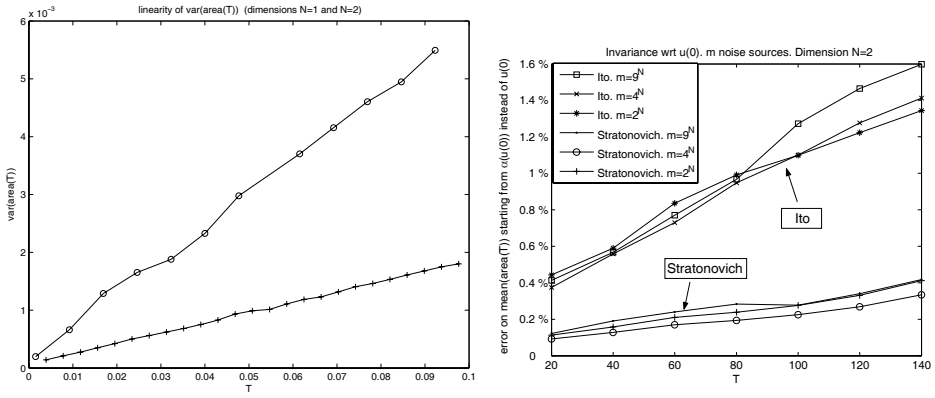


Figure 4.3: Left: Linear dependency between the final time and the variance of the area (one Gaussian). Right: invariance of Stratonovich wrt to the choice of the implicit function (several Gaussian sources).

between two sources. Using $\phi_i(x)/(\sum_{j=1}^m \phi_j^2(x))^{1/2}$ instead of $\phi_i(x)$ recovers a constant variance 1.

The drift will have a spatial derivative term (see (4.25)). Like figure 4.2 for one noise, the right part of figure 4.3 shows, for different values of m, how the Stratonovich scheme makes the evolution invariant with respect to the choice of $u(0)$.

With more than one source of noise, the points of the curve do not move at the same speed anymore, leading to the desired stochastic global deformation. As one should expect, with a large number of sources, the deformation is very noisy but the contributions of the sources tend to annihilate one each other. Thus, the curve does not move very far from its initial position. On the contrary, with a medium number of sources, the deformation is smoother but with ampler motions (see figure 4.4). Depending on his/her own application, the user might want to choose the optimal number of sources. As a first attempt to quantify the phenomenon, we measure how long it takes to the curve to move away from its initial position. For a given distance δ, we call the *expected exit time* the quantity $T(\delta) = \mathbb{E}(\inf\{t : \exists x \in \Gamma(t), d(x, \Gamma(0)) \geqslant \delta\})$ where $\mathbb{E}$ denotes the expectation. For a Brownian motion, the expected exit time from a ball is a quadratic function of the radius of the ball. In our case, such a result would be certainly hard to prove. Yet, our experiments show a similar relationship $T(\delta) \approx \xi(m)\delta^2$: (see the left part of figure 4.5). This useful relationship indicates clearly how long the user has to wait to see his/her curve getting away from its initial position. The right part of figure 4.5 plots ξ as a function of the number of sources. As expected, a large

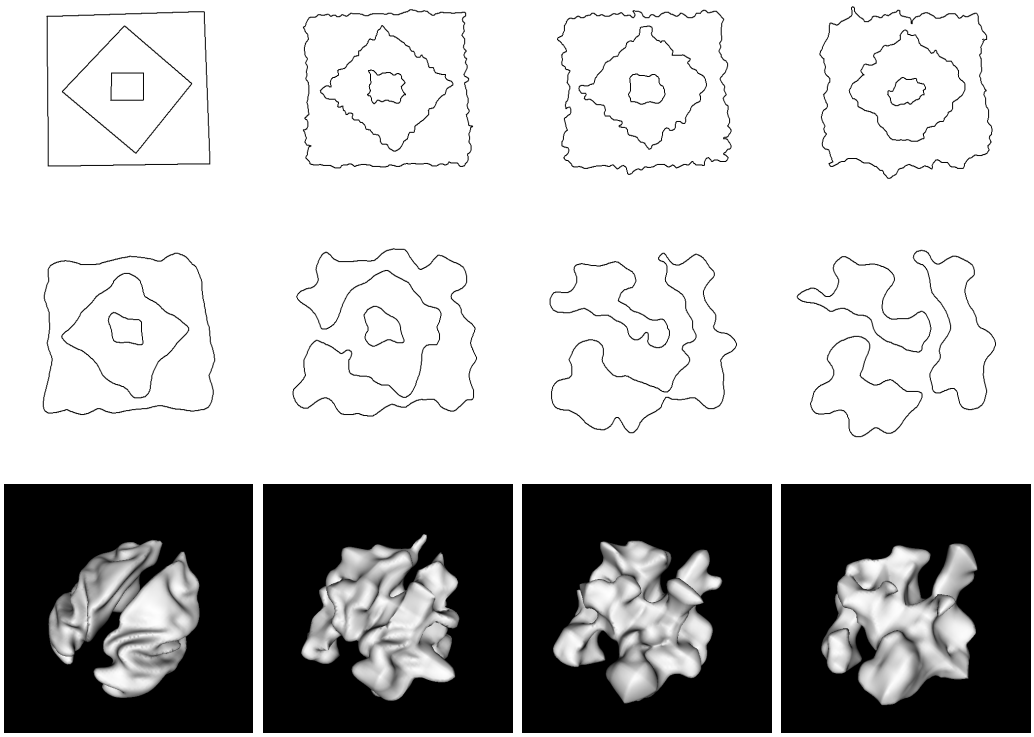


Figure 4.4: Different number of Gaussian noise sources. Top row: starting from the initial curve (top left), three time steps of the evolution with a large number of Gaussian sources. Middle row: from the same initial curve, four time steps of the evolution with a spatially smoother noise (small number of sources). Bottom row: a 3D example starting from the cortex of a monkey.

number of sources m induces a larger exit time, thus a larger ξ. Surprisingly, the smallest values of m give also a large ξ. We do not have any satisfactory explanation for this phenomenon... Anyway, these are only some very first step toward the understanding of the geometric properties of this kind of stochastic motion and many other quantities would be of great interest: the variations of the curvature, the time to get the curve split, etc.

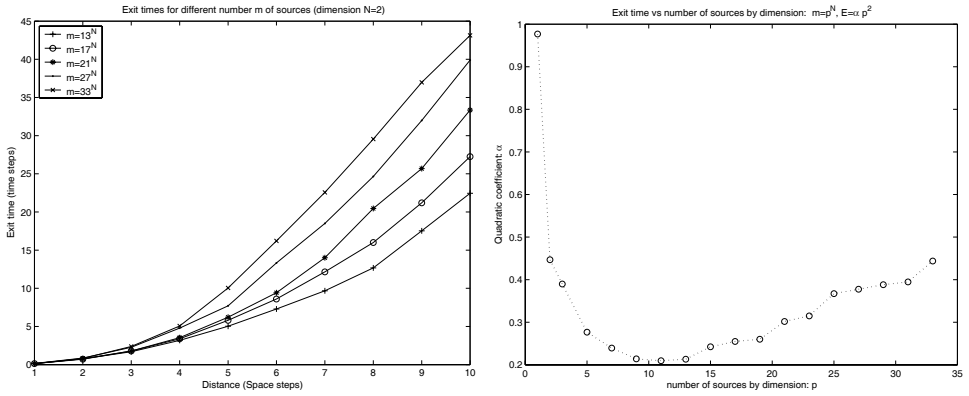


Figure 4.5: Several Gaussian sources. Left: Quadratic relation between the distance δ and the expected exit time from the band of thickness δ. Right: variation of the exit time w.r.t. the number of sources.

4.3.3 Stochastic Active Contour

Given some Computer Vision problem in a variational framework where we have to find the contour Γ that minimizes an energy $E(\Gamma) = E(u)$, we use the following simple Simulated Annealing decision scheme:

1. Start from some initial guess u_0
2. compute u_{n+1} from u_n using some dynamics, e.g. $du = |Du| \sum_{i=1}^{m} \phi_i(x) \circ dW_t^i$
3. compute the energy $E(u_{n+1})$
4. accept u_{n+1}:
   - if $E(u_{n+1}) < E(u_n)$
   - otherwise, accept u_{n+1} with probability $\exp\left(-\frac{E(u_{n+1})-E(u_n)}{T(n)}\right)$

5. loop back to step 2, until some stopping condition is fulfilled

Here, $T(n)$ is a time-dependent function that plays the same role as a decreasing temperature. Its choice is not obvious. If the temperature decreases too fast, the process may get stuck in a local minimum; on the contrary, decreasing too slowly, it may postpone convergence. A classical choice is $T(n) = T_0/\sqrt{n}$. The classical way to solve the previous minimization problem is to use a gradient descent method. The Euler-Lagrange equation is computed, leading to some evolution $\partial\Gamma/\partial t = \beta_c \overrightarrow{\mathbf{n}}$, or equivalently, in the Level Set framework, to $\partial u/\partial t = \beta_c|Du|$. We will actually use the classical motion as heuristics that drive the evolution faster toward a minimum, and replace the dynamics of step 2, by

$$du = \beta_c|Du|dt + |Du|\sum_{i=1}^{m}\phi_i(x) \circ dW_t^i \tag{4.26}$$

As often with genetic algorithms, the proof of the convergence of this algorithm toward a global minimum is still an open problem. However, we use numerical experiments to show how this algorithm performs better by avoiding some local minima that are problematic in the deterministic case. This is our main motivation, since local minima are the major problem of classical approaches. Note also, as already mentioned, that our framework can be used in cases when the shape gradient is too complex from a mathematical or computational point of view, or even impossible to compute.

Chapter 5

Active Cuts: a Graph Cuts extension

This work is a result of a join work with Yuri Boykov[1] and it is published in [87].

In this chapter, we add a number of novel concepts into global st-cut methods improving their efficiency and making them relevant for a wider class of applications in vision where algorithms should ideally run in real-time. The new *Active Cuts* method has three major properties, which are fairly unique for st-cuts algorithms.

- *Initial Cut:* Usually, min-cut/max-flow algorithms compute global optima solutions which do not depend on any initialization. This explains why standard combinatorial techniques do not really use a concept of initial cut. However, applications in vision often give some reasonable initial guess. Active Cuts is a new approach to solving max-flow/min-cut problems that can start from any initial cut. Its running time directly correlates with a Hausdorff distance between the initial cut and the globally optimal cut it computes.

- *Intermediate Cuts:* Standard combinatorial optimization algorithms generate only one good solution (one global optimum cut) computed at termination. In contrast, the proposed method outputs a sequence of cuts of decreasing cost as it converges to a global minimum. In computer vision, it is often beneficial to have a multitude of good solutions, which could be used for learning and robustness analysis. Intermediate cuts can be efficiently used in dynamic applications. They can also im-

[1]yuri@csd.uwo.ca, http://www.csd.uwo.ca/faculty/yuri, Department of Computer Science, University of Western Ontario, London, Ontario , Canada

prove interactivity and visual perception of the real-time performance of graph cuts on large data sets.

- *Symmetry:* Active Cuts approach adds symmetry into standard combinatorial optimization algorithms for st-cuts [75, 79]. We use *pseudoflow* [79] allowing nodes with positive flow *excesses* and negative *deficits*. However, our algorithm takes a symmetric approach complementary to [79]. Both excesses and deficits are actively pushed/pulled in the opposite directions away from a given initial cut towards the terminals s and t.

Recently, significant research efforts were devoted to efficiency of max-flow/min-cut methods in image segmentation [18, 105, 118]. The Active Cuts method will be tested in some basic applications of graph cuts like object/background extraction [14]. Despite polynomial complexity of graph cuts, computing globally optimal solutions for large images or volumes in real-time is still a challenge. Narrow bands can be used to improve efficiency [118, 178] by sacrificing global optimality. In contrast, we demonstrate a number of different ways where unique properties of Active Cuts allow achieving much better practical efficiency without loosing globally optimality.

Even in tests with poor initialization Active Cuts gave either comparable or better speed than the max-flow technique in [18] which is widely used in vision. If a good initial solution is provided then Active Cuts compute the globally optimal cut 2-10 times faster than [18], as an initial solution is of no help to previous st-cut algorithms. The theoretical worst case complexity of Active Cuts is similar to [18].

We also demonstrate applications of Active Cuts to other problems where an initial solution is naturally available: dynamic video segmentation, object tracking, and hierarchical segmentation of large images/volumes.

Earlier methods for accelerating graph cuts in video were reusing flows from previous frames [105]. They use [18] as a basic algorithm but "recycle" flow and other data structures from frame-to-frame. Instead of recycling flows, the Active Cuts algorithm can recycle cuts from previous frames. We show that Active Cuts running time is proportional to the amount of motion between two consecutive frames. In tests Active Cuts gave significantly better speed-up ratios with respect to [18] than the ratios reported in [105]. More importantly, in addition to recycling cuts, the Active Cuts algorithm can also recycle flows and other internal data structures, which will likely give even stronger speed-up, but this is left for conclusion. In other words, it is possible to combine the flow recycling of [105] and our cut recycling.

Active Cuts are also tested for hierarchical image segmentation, which is important for large data sets. An initial cut for Active Cuts at a fine scale

can be obtained from an optimal solution at a coarse scale. In contrast to earlier multilevel graph cut methods [118], we preserve global optimality.

The chapter is organized as follows. We first present our motivations in Section Sec:5.1. Our Active Cuts method was inspired by a number of recent ideas and Sections Sec:5.2 and Sec:5.3 explain how we combine them in a novel way based on our motivation to efficiently solve problems in computer vision. Sections Sec:5.4 and Sec:5.5 present in detail a first sketch and an advanced version of the Active Cuts algorithm and some theoretical results.

The next chapter 6 and Sec:7 present some results of our experimental evaluation of Active Cuts on a number of generic problems in static or hierarchical image segmentation and dynamic segmentation.

5.1 Motivations

In this section, we will present the motivations that lead us to the creation of this new algorithm. This work is based on the following observations.

First of all, augmenting path algorithms are known to be the most efficient algorithms for solving Max-Flow/Min-Cut problem. Dinic [61] is considered as the most efficient algorithm for Combinatorial community and Boykov and Kolomogorov [15, 18] is the fastest algorithm brought to the Computer Vision community. However recent studies shows that Push-Relabel approach is more powerful and has promising potential [5, 56, 57, 128, 40, 36]. This remark directs this work on Push-Relabel methods.

The second observation is more empirical and for the beauty's sake of the algorithm: **Symmetry**. Neither Goldberg and Tarjan and other Push-Relabel algorithms are symmetric, nor are Ford and Fulkerson, Dinic and other augmenting path algorithms, except the Boykov and Kolmogorov algorithm.

In this particular algorithm, one indisputable source of performance is the use of Dynamic Tree and its sharp management. Dynamic tree allows avoiding searching for the shortest path and provides along the tree short enough paths. However, an important milestone was passed by the use of two symmetric dynamic trees. This observation reinforces us in the idea that symmetry supplies real speed improvements and motivated the development of a symmetric Push-Relabel-like algorithm.

The last but not least, in Computer Vision many different algorithms **need** an initialization (Gradient descent methods) or **can take advantage** of an initialization to speed up the convergence. Ideally, we would like to be able to initialize a max-flow algorithm. This was already proposed by Kohli and Torr in [105] but in a restrictive approach. This will be discussed more

widely in the next sections.

5.2 Symmetrization

In order to definite a symmetric version of Push-Relabel, we need to introduce some new concepts. Let us track the dissymmetry in the Push-Relabel algorithm and propose new entities to fix it. First, let us describe in a high level what should be a Symmetric Push-Relabel-like algorithm.

In standard Push-Relabel, particles of positive excess flow are sent from the source towards the sink. Depending on the ordering policy (Sec:3.3.3) and the pushing policy (Standard or Discharging), excess is moving in front towards the sink. One could imagine dual particles moving in the graph from the sink towards the source. Drawing a parallel with particles is not innocent.

The Push-Relabel algorithm is dissymmetric in multiple points. It only considers positive excess nodes and flow is only moving from the source to the sink. As described above Push-Relabel only implements particles from the source while the first symmetrization should add particles from the sink. This may be done by the use of negative flow particles. The Relabel policy for those new particles should also be discussed.

5.2.1 Deficit node

According to the preflow definition (3.12), only positive value are allowed. The particles moving inside the graph are positive: they are *positrons*. As in particle physics, let introduce their twin particles: *electrons*. In this framework, electrons represent negative value of flow. That is to say lack of flow or even **deficit** of flow.

One could define the notion of *pseudoflow* as a relaxation of *flow* or even *preflow* functional. A *pseudoflow* functional is defined by:

$$\begin{cases} \forall\ (p,q) \in \mathcal{V}^2 & f(p,q) \leqslant c(p,q) \\ \forall\ (p,q) \in \mathcal{V}^2 & f(p,q) = -f(q,p) \\ \forall\ p \in \mathcal{V} & \sum_{q \in \mathcal{V}} f(p,q) \in \mathbb{Z} \end{cases} \tag{5.1}$$

This definition allows the existence of deficit nodes. When $\sum_{q \in \mathcal{V}} f(p,q)$ is positive, it is called excess else it is called deficit. Hochbaum in [79] already introduced those concepts. However, she stops here the symmetrization process and does not attempt to move the negative particles. In order to be

consistent with our goal of symmetry and since excesses are moving along the graph, one should also consider moving deficits.

Before giving a definition of deficit moves, let first discuss some possible interpretation of excesses and deficit. This is necessary and will help to understand those moves. So the main question is: *what are excesses?*

A first and trivial answer is *overflow of water*. However, excesses are also equivalent to t-links from the source with a capacity equal to the amount of excess flow on the node. Indeed every excess node can be converted back to a t-link. Left part of Figure 5.1 illustrates the conversion. Let consider such a t-link. It can be obviously convert to an excess node since a push step over one of this t-link will recreate it. To conclude with excess nodes, they can be replaced by t-link from the source.

Similarly deficit nodes can also be converted to t-link but t-link to the sink (see right part of Fig:5.1).

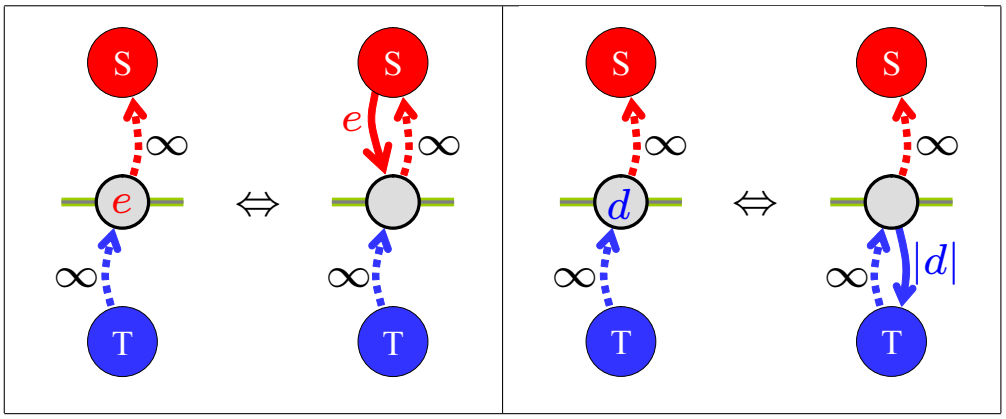


Figure 5.1: Left: Equivalence between excess and t-link from the source. - Right: Equivalence between deficit and t-link to the sink.

Nevertheless, what should be done when some excess reaches a deficit node (or vice versa)? **Annihilation**. Just like what happens when positrons meet electrons. We provide here an intuitive proof. When such a thing happens, the value of the flow at this node is equal to the sum of excess and deficit. Indeed, by assuming possible to have the two entities on the same node at the same time, one could give the following explanation. Figure 5.2 illustrates the process:

1. Let us consider such a node carrying both excess and deficit (left part of the figure).

2. Both excess and deficit are convert to t-links (middle part of the figure).

3. Then by considering the augmenting path *source* $\rightsquigarrow$ *node* $\rightsquigarrow$ *sink* of capacity $\delta = \min(|e|, |d|)$, we send δ units of flow from the source towards the sink (right part of the figure). Note that depending on the value δ, at least one t-link is saturated by the process.

4. Reconverting back the remaining t-link, we obtain:

   - If $e + d > 0$ then excess node with $e + d$ overflow.
   - If $e + d < 0$ then deficit node with $e + d$ lack of flow.
   - If $e + d = 0$ then an empty node.

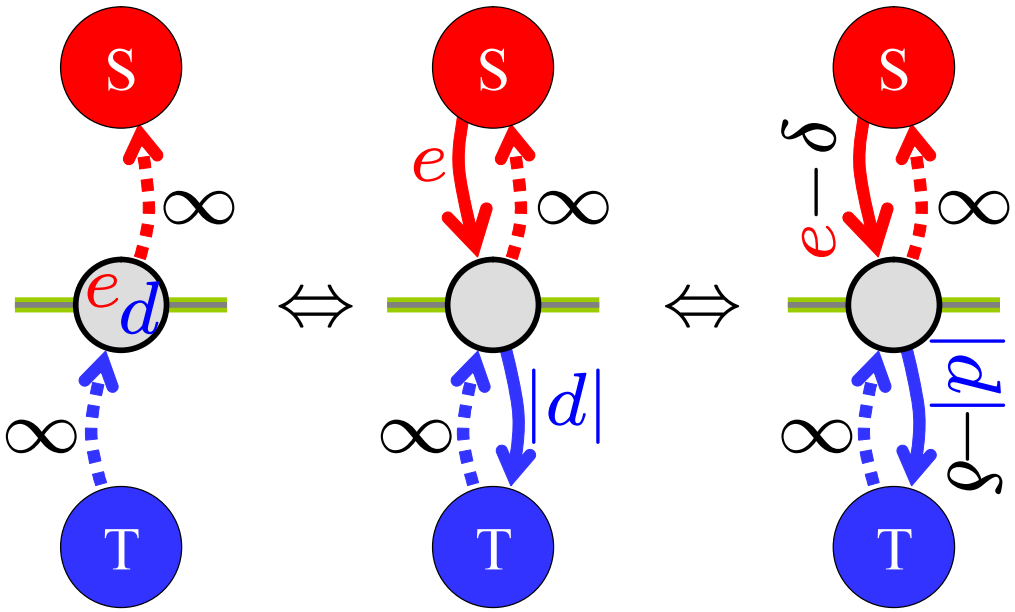


Figure 5.2: Left: Excess e and deficit d are present on the same node. - Middle: Excess and deficit are converted to t-links. Right: $\delta = \min(|e|, |d|)$ units of flow are sent from the source to sink generating a saturation: source and/or sink t-link depending on the value δ.

The sum rule of excess and deficit is quite simple but completely describe what should be done in that case.

One could remark that a contact between excess and deficit is similar to a contact between the two trees present in the Boykov and Kolmogorov algorithm Sec:3.3.2. Indeed when a contact occurs, by looking retrospectively at the path followed by the excess starting from the source and at the

path followed by the deficit, one obtains an equivalent augmenting path [2] of capacity equal to the minimum between excess and deficit flow. This is exactly the same when the two trees used in the Boykov and Kolmogorov algorithm meet. This observation reinforces our confidence in this proposition of a symmetric Push-Relabel.

Push-Relabel methods are very self-confident in the sense that they send some flow over the graph before knowing if it could reach the sink. The notion of deficit allows here a generalization by also grabbing preventively flow from the graph (deficit). In fact, deficit can be seen as an expectation of flow and simulates a quantity of flow already sent to the sink. Thus deficit should act and move from a node p to q like if the same quantity of excess had move from q to p. The standard *push* movement is illustrates on left part of figure Fig:5.3 and the new *pull* movement in the right side.

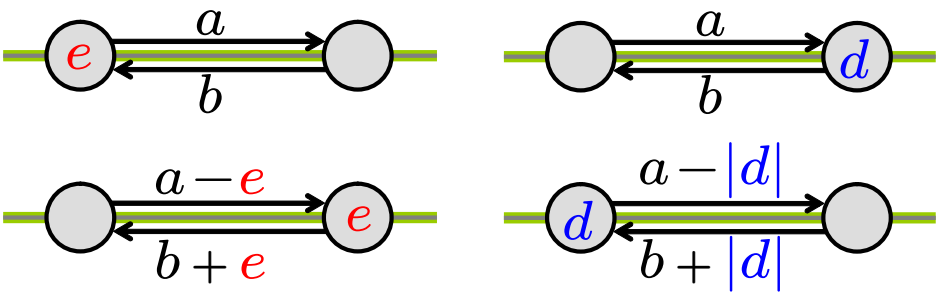


Figure 5.3: Moves between two nodes and their corresponding modifications of n-link weights. - Left Column: Push of excess from left to right node. - Right Column: Pull of deficit from right to left.

Now that nodes with negative flow are defined and their movement in the graph is described, one should focus on their label and as a consequence to their Relabel step.

5.2.2 Relabeling

In standard Push-Relabel algorithm, labels help to drive excess flow towards the sink and avoid them to be stuck into loops. With this aim in view, labels are initially defined as the distance to the sink in the graph. However the push or even the discharge procedure (see Sec:3.3.3 - Act:1 Alg:4) change the graph's topology by saturating edges in the graph.

[2] This path can be more complex than paths defined in Sec:3.3.2. In particular, they can have junctions.

The Relabeling procedure is a simple and local method to correct the labels in order to maintain a distance map to the sink. Ideally, one would keep labels equal to distance every time but this is too expensive. The Relabeling method corrects locally the distance (by above) and allows excess flow to move to another node. Nevertheless, if the sink is no more reachable from a part of the graph, Relabeling corrects the labels to match the distance to the source in this part.

For excesses, everything is fine with the standard definition of labels and Relabeling. But concerning deficits, this strategy does not suit. Deficits have to be attracted by the source. According to this observation, labels should be set as the distance to the source.

As no elegant combination of both distances (to the sink and to the source) could define a unique label that agrees the two particles' strategies, one should consider using two labels per node: one for excess $height^+ = h^+$ and one for deficit $height^- = h^-$.

However, this is not satisfying since the Relabeling of both labels cannot be treated simultaneously and two different Relabeling procedures have to be defined. In fact, if those labels are updated independently, it should not be the case since both of them are linked by the graph topology.

5.2.3 Algorithm

In this section, we present a basic symmetric Push-Pull-Relabels algorithm based on the two different labels as proposed above.

Initially those two labels (h^+ and h^-) are set according to the distance to their respective terminal: sink for h^+ and source for h^-. But with two exceptions: $h^+(source) \leftarrow |\mathcal{V}|$ and $h^-(sink) \leftarrow |\mathcal{V}|$ but this is very similar to the label initialization in standard Push-Relabel. Source and sink are also respectively set as infinite source of excess flow and deficit flow.

Before defining every procedures, let generalize some notations introduced in section Sec:3.3.3. First for excesses:

$$reachable^+(p) = \{q \in \mathcal{V} \ / \ \exists\ (p,q) \in \mathcal{E}_f,\ c_f(p,q) > 0\}$$
$$admissible^+(p) = \{(p,q) \in \mathcal{E}_f \ / \ q \in reachable^+(p),\ h^+(p) > h^+(q)\}$$

And for deficits:

$$reachable^-(p) = \{q \in \mathcal{V} \ / \ \exists\ (p,q) \in \mathcal{E}_f,\ c_f(q,p) > 0\}$$
$$admissible^-(p) = \{(p,q) \in \mathcal{E}_f \ / \ q \in reachable^-(p),\ h^-(p) > h^-(q)\}$$

The *Push-Pull* step, or moving step, for both excess and deficit is described in the algorithm (Alg:5) based on the standard version. A modified

Algorithm 5 Push-Pull

Require: p an excess node and q such that $(p,q) \in admissible^{+}(p)$.
Require: or p a deficit node and q such that $(p,q) \in admissible^{-}(p)$.
  if p is an excess node **then**
    Send $\min(excess(p), c_f(p,q))$ flow to the node q.
  else
    Grab $\min(|deficit(p)|, c_f(q,p))$ flow from the node q.
  end if

Algorithm 6 Relabels

Require: p a non terminal excess node such that $admissible^{+}(p) = \emptyset$.
Require: or p a non terminal deficit node such that $admissible^{-}(p) = \emptyset$.
  if p is an excess node **then**
    Relabel p according to: $h^{+}(p) \leftarrow \min_{q \in reachable^{+}(p)} h^{+}(q) + 1$
  else
    Relabel p according to: $h^{-}(p) \leftarrow \min_{q \in reachable^{-}(p)} h^{-}(q) + 1$
  end if

version of relabeling for both excess and deficit nodes is given in the algorithm (Alg:6).

A symmetric version of the standard Push-Relabel is straightforward. One just has to replace Push step by *Push-Pull* step and Relabel by *Relabels*. We present here a symmetric version of the discharge algorithm since its is more efficient (Alg:7).

Algorithm 7 Symmetric Discharge procedure

Require: p be an excess with $reachable^{+}(p) = \emptyset$.
Require: or p is a deficit node with $reachable^{-}(p) = \emptyset$.
  repeat
    while Push-Pull action is applicable **do**
      Push-Pull.
    end while
    if $excess(p) \neq 0$ and p is not a terminal **then**
      Relabels.
    end if
  until $excess(p) = 0$ or p is a terminal

After convergence, every node with label h^{+} higher than $|\mathcal{V}|$ is assigned to source and every node with label h^{-} higher than $|\mathcal{V}|$ is assigned to sink.

None of them could be assigned to the two terminals. But some of them could be orphans, in that case two different cuts of same cost are extracted C^+ and C^- (see Fig:5.4).

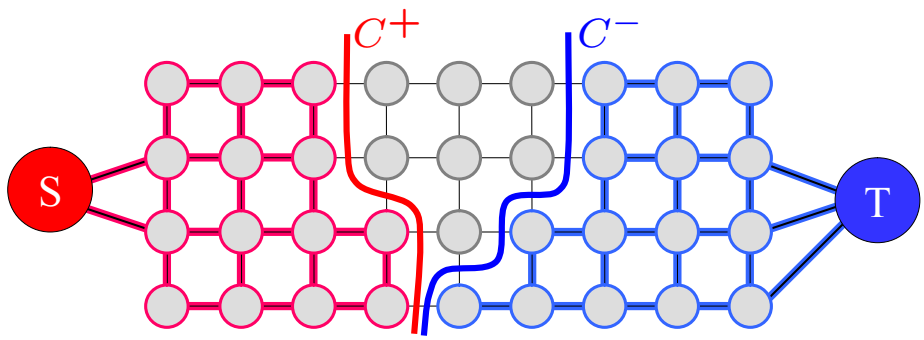


Figure 5.4: Illustration of a result given by the proposed symmetric Push-Pull-Relabels algorithm (Alg:7). Here C^+ and C^- are respectively the cut from the source and from the sink. They have exactly the same cost. The gray part is separated from the the two terminals. This illustrates the case where more than one optimal cut are available. If only one is available, C^+ and C^- are equal.

However no proof of this algorithm is provided, but it can be achieved using the standard Push-Relabel proof as a template to drive it.

This algorithm works perfectly but is not satisfying since it needs to introduce two independent labels per node. The next section will provide a way to initialize the graph and in the same time remove elegantly this drawback.

5.3 Initialization

A lot of optimization techniques needs to be initialized: Gradient descent, Dynamic programming, Simulated annealing, ... However initializing an algorithm usually comes with some drawbacks. The main one: *dependency of the result with respect to the initialization*. However initialization provides an interesting property since it can also be considered as a prior knowledge on the solution (the solution will remain close to the initialization).

Nevertheless using initialization on a global optimizer is meaningless considering this property. Indeed the solution is a global minimum and does not depend on the initialization. However one could expect to speed up the convergence if the initialization is close to the solution.

In this section, we present a method of Kohli and Torr [105] and then we propose a more general method of initialization.

5.3.1 Flow initialization

The first idea that comes out to anyone accustomed to Max-Flow/Min-Cut algorithms is to initialize the graph with a given flow functional. Indeed any max-flow algorithm is initialized with a null flow function, but one could easily use any flow function as soon as it verify the definition (3.11). This idea is developed in [105].

The main application is for video segmentation since you should possess a flow function that is consistent with the graph. However enforcing a given flow to a graph may cause problems. Indeed a flow stemmed from a graph, may not verify the flow definition for another graph. Basically, let consider two nodes p and q, the flow $f(p,q)$ crossing the edge (p,q) may overflow the edge capacity on the new graph: $f(p,q) > c(p,q)$.

In that case, the flow should be corrected. Kohli and Torr provide some tricks to do it. The first one allows removing every edge overflow in the graph. Let consider two nodes p,q such as the given $f(p,q)$ is overflooding the capacity of the link $c(p,q)$: $f(p,q) > c(p,q)$. In fact there is too much flow inside the graph and a correcting method should decrease this flow.

In order to correct this violation of the capacity constraint (3.11), one should consider the latent infinite t-links. Indeed adding infinite t-links (from the sink to a node and from a node to the source) does not change the problem definition since a cut is oriented from the source to the sink (see left part of Fig:5.5).

Considering the path P defined as $sink \rightsquigarrow q \rightsquigarrow p \rightsquigarrow source$, we obtain a decreasing path of capacity $c_f(q,p) = c(p,q) + f(p,q)$ (see middle part of Fig:5.5). This path allows to decrease the flow in the graph which is what we need. Let define $\delta = f(p,q) - c(p,q)$ the overflow inside the pipe (p,q). Thus to correct the overflow, one have to send back only δ unit of flow along the decreasing path P. Note that the resulting edge (p,q) is saturated.

This method allows to correct the overflow on edges and create two corresponding t-links of capacity δ (see right part of Fig:5.5).

This method is very powerful. The more the flow comes from a similar graph, the more it will saturate edges and the more the algorithm is fast. But since a coherent flow function must be available, this method is only applicable to a limited number of cases.

Note that Push-Relabel based methods provide a more suitable framework due to a more flexible notion: preflow. In fact using Push-Relabel

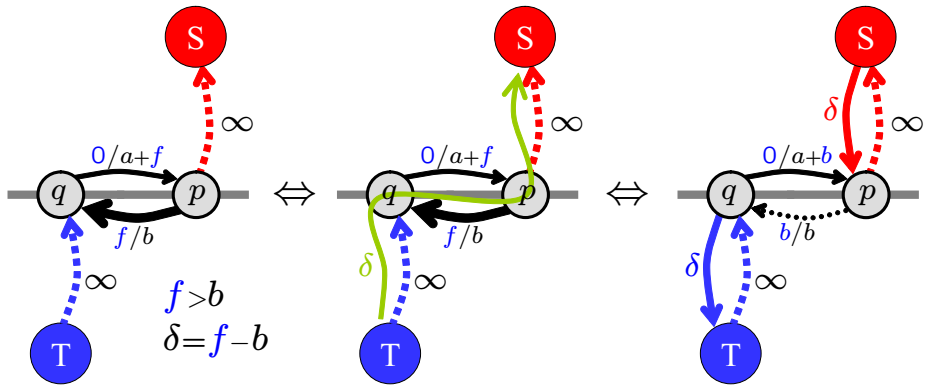


Figure 5.5: Fixing tricks. From left to right, this figure illustrates how one could send back some flow from the sink to the source in order to satisfy the capacity constraint. This results into the creation of two t-links of equal capacity δ. Thick edges stand for over-flooded edges. Dotted edges stand for saturated edges.

framework, the correction is even more simple. The correction comes down to remove the overflow by creating an excess node (see Fig:5.6)

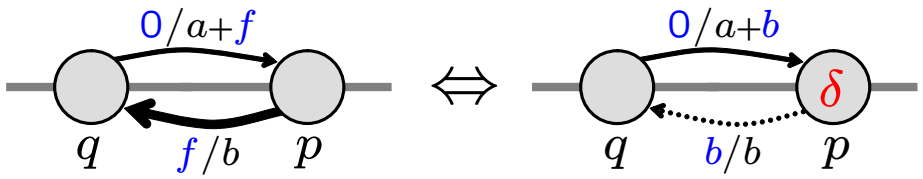


Figure 5.6: This figure illustrates the fixing tricks (Fig:5.5) in a preflow formulation like in a Push-Relabel algorithm. In this formulation, one does not have to add some new t-links. Thick edges stand for over-flooded edges. Dotted edges stand for saturated edges.

In next section, we will provide another less restrictive method of initialization.

5.3.2 Cut initialization

The main idea presented here is based on the following observation. Given a graph and a maximal flow function, one could easily find the minimal cut. By applying the flow on the graph, the residual graph is separated in two parts by a cut of null cost. This is minimal cost cut of the graph. **But** given a minimal cost cut, finding a flow functional is not trivial.

We deduce from this observation that a flow function provides more information that a simple cut. Yet initializing the graph using a cut is less restrictive. This initial cut can be the result of any segmentation algorithm or prior information.

One may want to initialize the algorithm with any arbitrary cut. This cut C_0 could be stemmed from a previous frame in case of video segmentation, or from a coarse level for hierarchical segmentation, or even from the user. Following the definition of a cut, we have to saturate every edges going from the source part to the sink part. Those edges are drawn in green in the figure Fig:5.7.

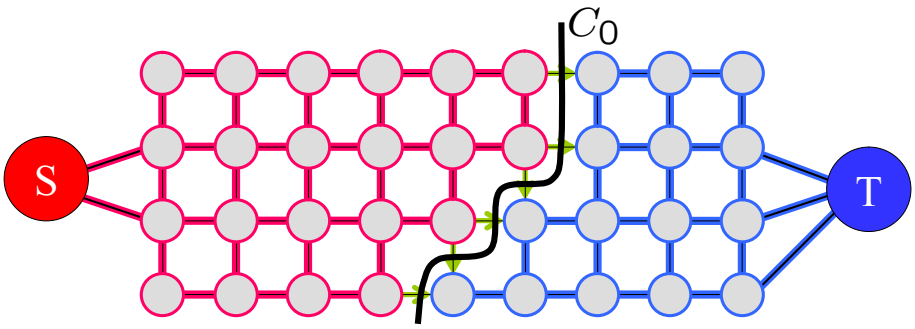


Figure 5.7: Illustration of initialization. C_0 corresponds to the initial cut provides to the algorithm. The red and blue parts correspond respectively to the Source and Sink part of the tree after cutting. The green arrows correspond to links that have to be saturated in order to produce a cut in C_0.

In order to saturate those edges, we use the same trick used by Kohli and Torr in [105] for correcting the overflow (see Sec:5.3.1). However as we will lie in a symmetric Push-Pull-Relabel framework, the notion of pseudoflow is available and we could transform the new t-links into excess and deficit node (see Fig:5.8).

Now we have obtain a graph split in two parts: the source where lie the deficit nodes and the sink part where lie the excess nodes (see Fig:5.9). This is very convenient since now deficits and excesses do not lie on the same graph. This allows us to remove the redundant label and use only one per node. Note that if the cut C_0 is the minimal cut then every excess and deficit nodes will be able to make its trip to its respective terminal.

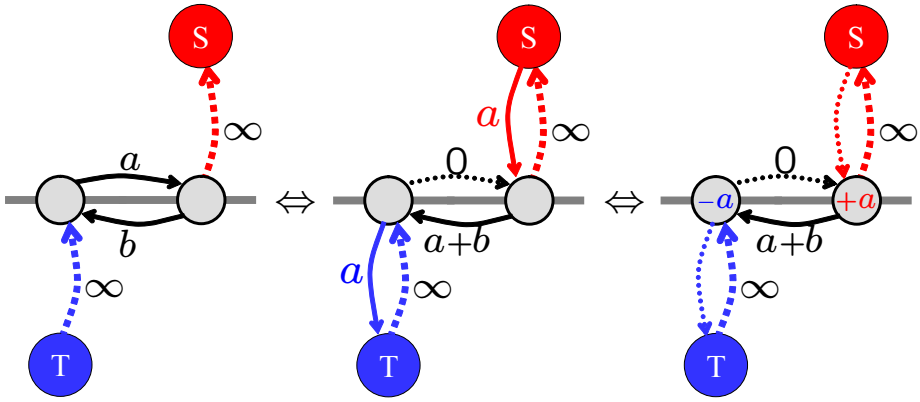


Figure 5.8: We illustrate here how we could saturate a given edge of any capacity by using the tricks described in the figure Fig:5.5. In the middle, we present the result in a feasible flow framework and in the right side, in a pseudoflow formulation. Dotted edges stand for saturated edges.

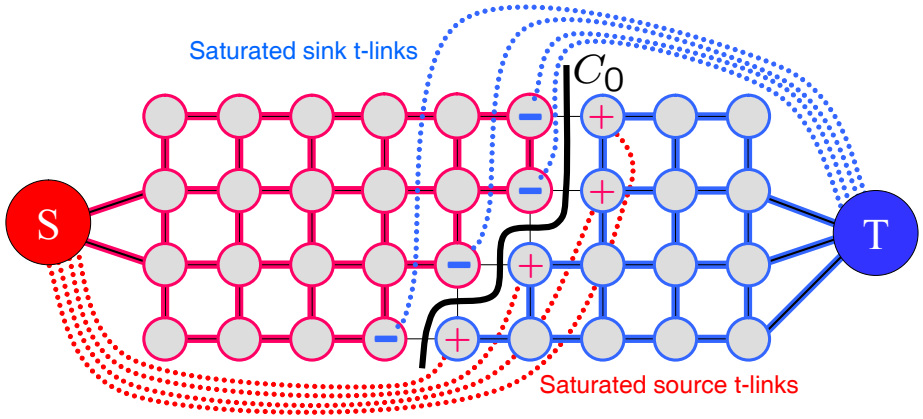


Figure 5.9: We show here the resulting graph after saturating green edges (Fig:5.7) in a pseudoflow formulation. Dotted edges stand for saturated edges.

5.4 Active Cuts algorithm

5.4.1 Algorithm outlines

In this section, we will draw a rough sketch of the proposed algorithm. Given a weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ and a given initial cut C_0, the algorithm presented here makes use of a pseudoflow formulation and returns a minimal cut.

The graph is first initialized using the method described above (see Sec:5.3.2). This results in a graph formed of two parts (source and sink part) only connected by edges oriented from sink to source. This very convenient since excesses lie in the sink part and are attracted by the sink terminal and symmetrically for the deficit. Even if the graph is not exactly separated in two distinct graphs, we can consider it for some times.

Then the standard Push-Relabel is launched on the sink-part. A Pull-Relabel algorithm is considered for the other part based on the Push-Pull-Relabels algorithm (see Alg:7). Those two algorithms are very similar. Indeed the Pull-Relabel algorithm works exactly like a Push-Relabel algorithm on a *reverse graph*. A reverse graph is a graph where every edge is turn over and deficit transformed into excess. Moreover, each t-link is changed of terminal: source t-link becomes sink t-link...

At convergence, some excesses (respectively deficits) may be stuck in the sink part (respectively source part) of the graph. In fact by running these two algorithms, we obtain two minimal cuts C^+ and C^-. C^+ is the minimal cut of the sink part and similarly for the C^- min-cut. Figure Fig:5.10 illustrates such a case.

We give here a first result related to the cost of cuts C^+ and C^-.

Theorem 1 *The cost of C^+ is equal to the cost of C_0 minus the stuck excess flow. Similarly the cost of C^- is equal to the cost of C_0 minus the stuck deficit flow.*

This result is almost trivial.

Proof : Let describe the proof for C^+ since it can be transcribed identically for C^-.

Before running the Push-Relabel algorithm on the sink part, there were exactly $|C_0|$ units of excess flow in this part of the graph. By definition of C^+, only $|C^+|$ units of flow can reach the sink. Thus $|C_0| - |C^+|$ units of excess flow remain stuck between C_0 and C^+ after convergence of the algorithm. □

We give here a first result related to position of the minimal cut of the

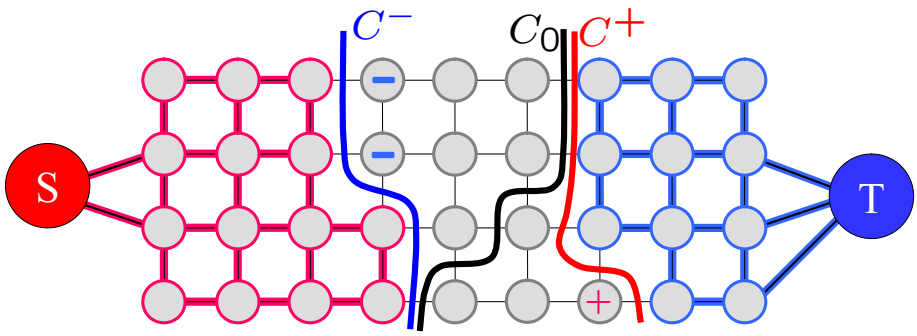


Figure 5.10: In this figure, we illustrate a possible resulting graph after convergence of both Push-Relabel and Pull-Relabel algorithms on their respective part. In this case, both excesses and deficits remain stuck. However this algorithm already provides two new cuts C^+ and C^- of lower cost than C_0.

whole graph with respect to the number of deficit and excess stuck by the two minimal cuts C^+ and C^-.

Theorem 2 *At convergence, four different cases emerge:*

- *If no excess nor deficit remains, then the initial cut C_0 is the minimal one. If neither excess nor deficit remains, then it means that no cut in the source nor the sink part has a lower cost than the initial cut. Thus, it can be easily shown that the initial cut is the minimal cut.*
- *If only excesses are stuck and all the deficits reach the source, then the minimal cut C^+ of the sink part is the minimal cost cut of the whole graph. The result comes from the same argument. One could note that the cost of the minimal cut is equal to the cost of the initial cut minus the total excess flow remaining.*
- *Similarly if only deficits get stuck and all the excesses reach the sink, then the minimal cut C^- of the source part is the minimal cost cut of the whole graph.*
- *If both excesses and deficits get stuck, then this is the more complex case where the minimal cost cut is on both part.*

We provide here the proof of the first case. The same argument is applicable to the other cases (the last case except) and its adaptation is almost straightforward. This following proof is done by reductio ad absurdum.

Proof :

Let consider the initial cut C_0 and the minimal cut C_{min}. Four different cases are feasible; we will prove that three of them are absurd:

- The minimal cut completely lies on the source part, and $|C_{min}| < |C_0|$. Since there is $|C_0|$ deficits and that they have to cross C_{min} to reach the source and only $|C_{min}|$ can cross it, it results that at least $|C_0| - |C_{min}|$ deficit should get stuck. This is in contradiction with our assumption.
- The same argument applies if the minimal cut lies on the sink part.
- The minimal cut lies on both parts and $|C_{min}| < |C_0|$. Reader should refer to the figure Fig:5.11 for an illustration. The assumption $|C_{min}| < |C_0|$ can be rewritten with the notation of the figure as $c + d < a + b$. Let now consider the cut formed of the part c and b. Since no deficit remains, we have $c + b \geqslant a + b$ and by symmetry, $a + d \geqslant a + b$. This gives the contradiction $c + d \geqslant a + b$.
- The minimal cut is C_0.

The study of all those cases can be summarized as follow. Either the minimal cut is C_0 or C_0 has the same cost as the minimal cut. Thus C_0 is a minimal cut. □

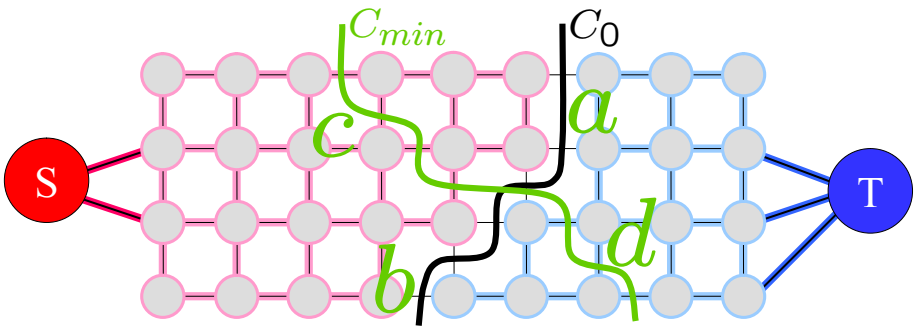


Figure 5.11: This figure illustrates a graph with an initial cut C_0 in black and a minimal cut C_{min} in green. It is assumed that these cuts cross each others. This leads to a partition of the cuts. $(a, b) \rightsquigarrow C_0$ and $(c, d) \rightsquigarrow C_{min}$. For readability, the red part or source part is in pink, and the sink part in light blue.

Concerning the first three cases, the proposed algorithm clearly solve the min-cut problem. The special case where some deficit and excess remain after convergence is more problematic and is widely discussed in the next section.

5.4.2 Special case

We now focus on the special case where some excesses and deficits remain stuck in the graph after convergence.

Let first define some notations before providing a theoretical result on the position of the minimal cut and then we settle the issue and give a complete algorithm.

The reader is invited to refer to the figure Fig:5.12 for an easier understanding. Let call *right side* the part of the graph on the right side of the initial cut C_0. Similarly, we also define the *left side*. Let call C^+ the minimal cut of the right side and C^- the minimal cut of the left side. Let also consider the minimal cut C_{min} of the whole graph.

On figure Fig:5.12, the cut C_0 is formed of four parts a, b, c and d. Similarly C^+ is formed of a, f and g, C^- of e, c and d and C_{min} of h, i and j.

The red art is in this section called the *source part* since this part of the graph is still connected to the source terminal and symmetrically for the blue part. Furthermore, the gray part is the part where lie the remaining excesses and deficit.

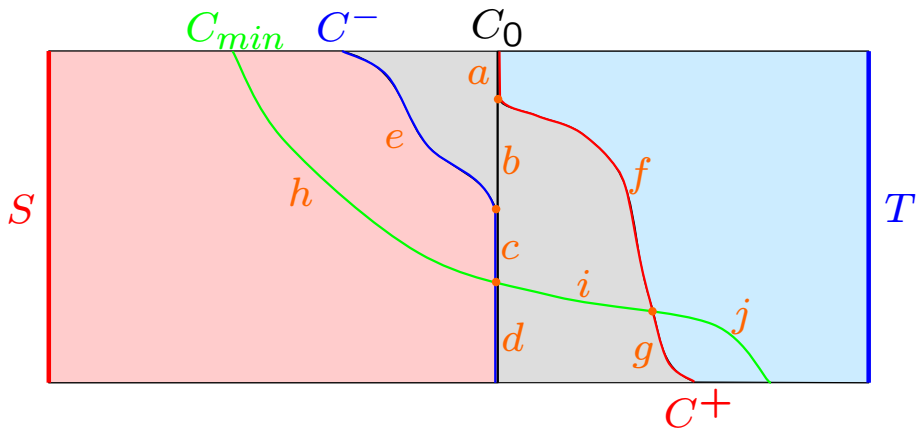


Figure 5.12: Similarly to Fig:5.11, we illustrate here a more complex case. The cuts positions are chosen in order to represent some typical cases. On top of C_{min} and C_0, we add the two cuts C^+ and C^- as illustrate in Fig:5.10. This leads to the following partition of cuts. $(a, b, c, d) \rightsquigarrow C_0$, $(h, i, j) \rightsquigarrow C_{min}$, $(a, f, g) \rightsquigarrow C^+$ and $(e, c, d) \rightsquigarrow C^-$.

Now we can express the result as follow.

Theorem 3 *When both excess and deficit remain after convergence then a minimal cut of the graph lies in the gray part.*

Proof : Let assume that the minimal cut (in green on the figure Fig:5.12) does not lie on the gray part. We will prove that this is impossible. First, let move apart some basic cases.

- C_{min} *is not completely on the left side:* Indeed, this contradicts the definition of C^-.
- C_{min} *is not completely on the right side:* Indeed, this contradicts the definition of C^+.

Thus the only possibility is that the minimal cut lies on both sides. There are three different cases that completely describe the position of C_{min}.

- One side, at least, of C_{min} completely lies in red or blue part. This case is illustrated in the left side of figure Fig:5.12.
- One side, at least, of C_{min} lies in the gray part and cross the minimal cut C^+ or C^- to reach the red or blue part. This case is illustrated in the right side of figure Fig:5.12.
- C_{min} lies in the gray zone.

We will now prove that the two first cases are impossible unless a minimal cut lies in the gray part.

Let consider the first case. Let consider the cut $C^{-\prime}$ defined by the curves h and d in the figure Fig:5.12. According to the definition of C^- we have $|C^{-\prime}| \geqslant |C^-|$ which means that $h + d \geqslant e + c + d$. This is equivalent to $|C_{min}| = h + i + j \geqslant e + c + i + j$ but this is impossible due to the definition of C_{min} unless $|C^{-\prime}| = |C^-|$. In that case the cut composed of the curves e, c, i, j is also a minimal cut of the graph and its left part lies in the gray part.

The second case is quite similar. Let consider the cut $C^{+\prime}$ defined by the curves a, f, j in the figure Fig:5.12. According to the definition of C^+ we have $|C^{+\prime}| \geqslant |C^+|$ which means that $a + f + j \geqslant a + f + g$. This is equivalent to $|C_{min}| = h + i + j \geqslant h + i + g$ but this is impossible due to the definition of C_{min} unless $|C^{+\prime}| = |C^+|$. In that case the cut composed of the curves h, i, g is also a minimal cut of the graph and its right part lies in the gray part.

Combining those results allows us to conclude. □

Now let summarize the situation. We know that the minimal cut is inside the gray part. Moreover, every remaining excesses and deficits are inside the gray part. Both excesses and deficits are stuck in the gray part since boundaries are saturated cuts. This indicates that all the remaining work will takes place in the gray part and that the next step of the algorithm will

focus only on the gray part. Note that left and right side of the gray zone are not disconnected since it exists n-links from right to left part (reverse n-links are saturated by construction of the initialization). By converting back every excess and deficits into t-links, one could clearly see that source and sink are not separated yet.

The next step the algorithm consists into alternatively converting back every excesses or every deficits into t-links and relaunching the algorithm until convergence (see Fig:5.13). At convergence, the boundary between red and blue part is the minimal cost cut C_{min} of the graph.

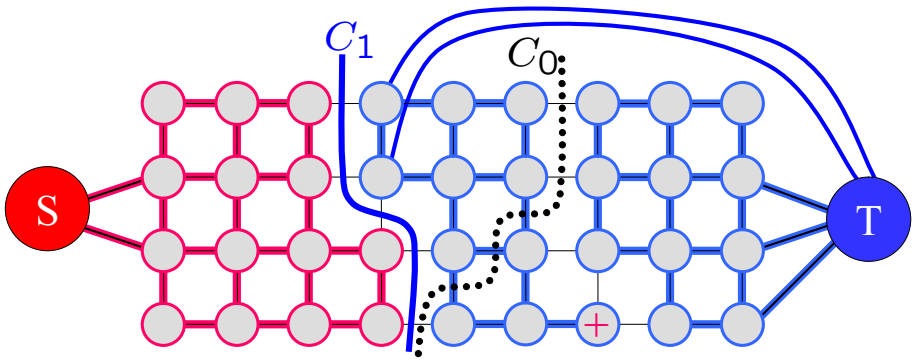


Figure 5.13: This figure shows how we should get rid of remaining excesses and deficits. In that case, more deficits that excesses were remaining. So deficits are convert back to t-links (blue links) and the gray zone is repainted in blue. The initial cut C_0 is dotted this it does not correspond anymore to the current cut between the source and sink part . C_1, the old C^-, is the new current cut since in that case we have $|C^-| < |C^+| < |C_0|$.

In fact a better strategy can be used by counting the number excess flow and deficit flow. Indeed based on the sign of $\delta = \sum_{p \in Gray} excess(p) + deficit(p)$ (remember that $deficit(p) \leqslant 0$), one could define the more efficient strategy on conversion.

In fact if $\delta < 0$ then there are more deficits than excesses and converting back deficits to t-links will lead to less moving particles (excesses in that case). Moreover, this choice is also motivated by the fact that it will connect the gray zone to the sink and paint the gray zone in blue. Therefore in that case, $|C^-| < |C^+| \leqslant |C_0|$ and the new cut separating the red part from the blue part is the minimal one. This choice sets the minimal cost cut between C^+ and C^- as the new cut C_1 (see Fig:5.13). By doing so, one obtain a series of decreasing cost cuts C_0, C_1, C_{min} (see Fig:5.14).

And similarly if $delta > 0$.

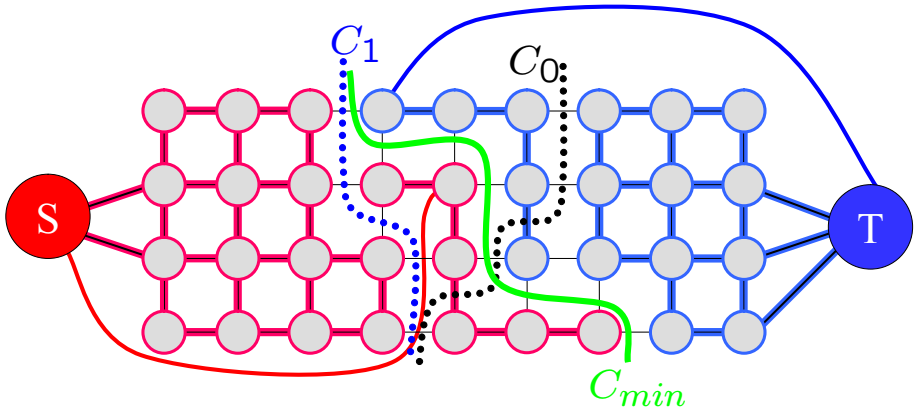


Figure 5.14: We illustrate here a resulting graph at convergence. C_0 and C_1 are the past cuts (in dots). The minimal and last cut C_{min} separate the graph in two unconnected parts (red and blue). Some t-links remain (red and blue links).

Property 1 *The Active Cuts algorithm provides a series of decreasing cuts where the last one is the minimal cost cut of the graph. Moreover, the second cut is a local minimum a string way since it is a global minimum on half the graph.*

This leads to a complete definition of the algorithm.

5.4.3 Algorithm

We give here the complete description of the algorithm, namely Active Cuts algorithm (see Alg:8).

We could discuss a little bit the complexity of this algorithm. Let assume that the initial cut splits the graph in two parts of same number of nodes. Then let note $C_x(|\mathcal{V}|, |\mathcal{E}|)$ the complexity of the Push-Relabel algorithm used in Active Cuts. In that case the complexity of the first part of the algorithm is $2C_x\left(\frac{|\mathcal{V}|}{2}, \frac{|\mathcal{E}|}{2}\right)$. Concerning the second part of the algorithm, at worst the complexity is equal to $C_x(|\mathcal{V}|, |\mathcal{E}|)$.

Thus in that case, the total complexity at worst is equal to $2C_x\left(\frac{|\mathcal{V}|}{2}, \frac{|\mathcal{E}|}{2}\right) + C_x(|\mathcal{V}|, |\mathcal{E}|)$. One could easily see that this worst that the standard Push-Relabel algorithm. But if the Active Cuts algorithm is more complex, it also

Algorithm 8 Active Cuts

Require: $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ a weighted graph.
Require: An initial cut C_0.
Initialize according the initial cut C_0 (see Sec:5.3.2 for details).
Run the Push-Relabel algorithm on sink part.
Run the Pull-Relabel algorithm on source part.
Paint in *gray* nodes with height higher than $|\mathcal{V}|$ on the whole graph.
$\delta \leftarrow \sum_{p \in Gray} excess(p) + deficit(p)$.
if $\delta < 0$ **then**
C^- is renamed as C_1.
Set the *working* part as the gray part.
Convert back deficits into sink t-links (see Fig:5.1 for details).
Paint in *blue* every *gray* node.
Run the Push-Relabel algorithm on the *working* part.
Paint in *red* nodes with height higher than $|\mathcal{V}|$ on the *working* part.
else
C^+ is renamed as C_1.
Set the *working* part as the gray part.
Convert back excesses into source t-links (see Fig:5.1 for details).
Paint in *red* every *gray* node.
Run the Pull-Relabel algorithm on the *working* part.
Paint in *blue* nodes with height higher than $|\mathcal{V}|$ on the *working* part.
end if
Every red node is assigned to the source.
Every blue node is assigned to the sink.
Minimal cut C_{min} is the boundary between red and blue nodes.
return C_0, C_1 and C_{min}.

provides more information: series of cuts. Moreover this complexity is for the worst case, that is to say when the gray part of the gray is the complete graph, which is highly improbable. The complexity of this algorithm is strongly correlated to the size of the gray zone.

By assuming a worthwhile case where the gray zone is small, one could approximate the complexity of the algorithm by $2C_x\left(\frac{|\mathcal{V}|}{2}, \frac{|\mathcal{E}|}{2}\right)$. If we make use of the *Discharge* version of Push-Relabel (see Alg:4 in Sec:3.3.3), we obtain a complexity of $\frac{1}{4}C_x(|\mathcal{V}|, |\mathcal{E}|)$.

5.5 Implementation

When it comes to performance, standard implementations of Push-Relabel, even the *Discharge* version, are not fast enough.

The critical part of the algorithm lies in the management of connectivity inside the graph. Indeed if labels are reliable at the beginning of the algorithm, this is no more the case after some times. Some basic improvements like Global Relabel or Gap Relabel [38, 56], allows some speed improvement.

However the real performances of the algorithm are revealed when we make use of *Dynamic Tree*. The best implementations of Push Relabel relie on *Dynamic Trees* instead of labels to drive excesses in the graph [76, 101, 37, 3].

In the next section, we will first describe our dynamic tree design. Then we propose a modified version the Active Cuts algorithm. And, in the end, we will focus on some interesting properties of this new algorithm.

5.5.1 Dynamic Tree

Based on work of Goldberg and Trajan [76] and Hochbaum [79], we proposed here a new Dynamic Tree structure in order to get access to higher performance.

We make use of two dynamic trees like in the Boykov and Kolmogorov algorithm [18]: a source tree (red) directs deficits to the source and a sinks tree (blue) directs excesses to the sink. Those two trees are illustrated in the figure Fig:5.15. Note that the source tree connects nodes to the source via edges (red) that are non-saturated in the direction from the terminal towards the leaf nodes. The sink tree connects nodes to the sink via edges (blue) that are non-saturated in the direction from leafs towards the terminal. Due to saturated edges on the initial cut C_0, the source and the sink trees can never overlap.

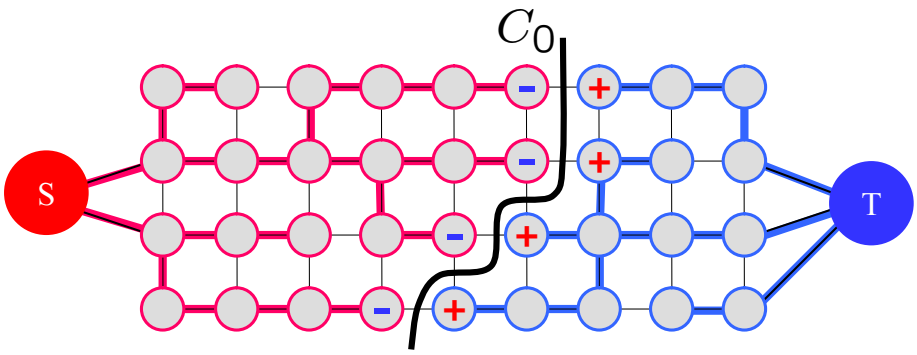


Figure 5.15: This figure illustrates the initialization step sauch as in Fig:5.9. The initial cut C_0 is shown in black. However we also show the tree structure underneath. Thick red and blue edges represent connectivity along the two trees: source tree (red) and sink tree (blue).

As excesses and deficits move along the edges on the trees, we maintain short paths on both trees dynamically mainly based on ideas described in [18]. Both trees maintain the following properties:

- A tree is rooted at a terminal.
- A tree spans all nodes 'reachable' from a root.
- Each node in the tree has only one parent.
- All edges included in a tree are non-saturated.

Let describe now more precisely how to fix the tree. A tree becomes corrupted when some flow moves along the tree and saturates one edge. Let consider a node p inside the sink tree. This node carries some excess flow and by push some flow to its parent q, the edge linking them (p, q) get saturated. Figure Fig:5.16 illustrates it.

We call this fixing action *Fast Adoption* if the node p may find a new reachable parent in its neighborhood that is not one of its descendant (like shown in the right side of figure Fig:5.16). This case is quite simple and very fast.

However, it may happen that no other node can be become its new parent. This case is more complex and two different situations may happen. Whether it cannot be found a connection from the subtree of p to the sink tree and in that case the subtree is isolate from the sink terminal, whether the subtree of p can be connected to the sink tree.

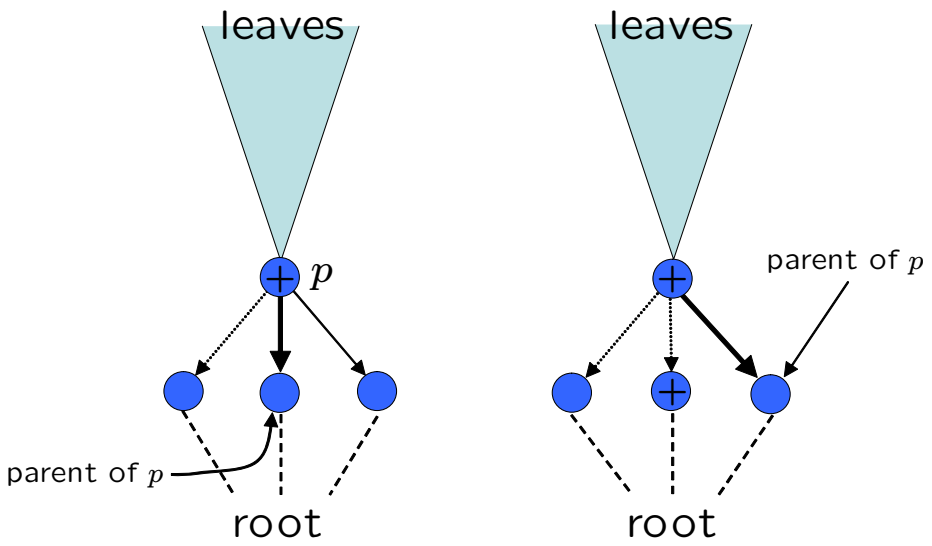


Figure 5.16: We describe here a Fast Adoption step. The light blue cone stands for the descendants of the node p in the Dynamic Tree. Thick edges represent parental connectivity while dotted edges represent saturated edges. - Left Side: some excess is present on node p. By pushing excess to its parent, the parent edge is saturated. - Right Side: the parental connectivity of p is restored since another node is reachable from p.

In the first situation, the subtree of p is isolated from the terminal. That is to say there is no non-saturated edge from a node of the subtree to a node of the sink tree. This means that we have a new cut separating the subtree from the sink. The subtree has to be painted in *gray*. This action is called *Orphanize* since we cannot find a new parent for the subtree.

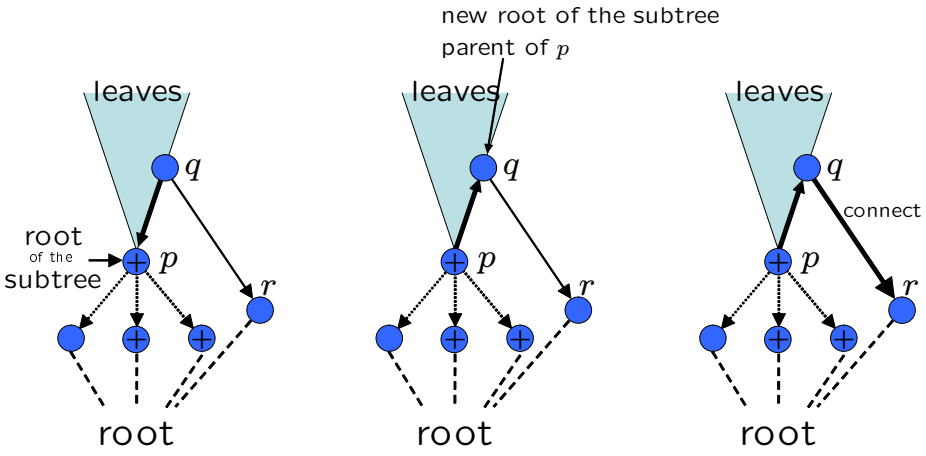


Figure 5.17: We describe here a Slow Adoption step. The light blue cone stands for the descendants of the node p in the Dynamic Tree. Thick edges represent parental connectivity while dotted edges represent saturated edges. - Left Side: some excess is present on node p and by pushing excess to its parent, the parental edge is saturated. However no node is directly reachable from p - Middle Side: a node q is found inside the children, that could reach a node r outside the cone. This node q is set to be the new parent of the whole cone. - Right Side: the r is set as the new parent of q and the Dynamic Tree is restored. - Two possible problems can occurr: First, there is no node like q available, in that case we have a new cut since the cone cannot reach another node. Second, setting q as the root is possible only if there is no saturated reverse edges from $p \rightsquigarrow q$, in that case, more than one slow adoption step may be necessary.

An illustration for a second situation is provided by the figure Fig:5.17. This fixing action is called *Slow Adoption* and consists into three steps:

1. Find a descendant q of p with a non-saturated edge that links to a node r of the sink tree outside of the subtree of p (left part of Fig:5.17).

2. Set q as the new root of the subtree by reversing parent/child relationship on the path $q \rightsquigarrow p$ (middle part of Fig:5.17).

3. Set r as the new parent of q (right part of Fig:5.17).

If the first step does not succeed, then we are in fact in the first situation. Nevertheless in the second step, by reverting the relationship on the path $q \rightsquigarrow p$, we assume that we do not use any saturated edge. This is not all the time true. If such a case happens, then the subtree is split into two subtrees and the fixing process is re-launched on the unfixed subtree.

5.5.2 Algorithm

We will now present a modified version of the Active Cuts algorithm (Alg:8) that takes advantage of the dynamic tree structure.

Before discussing an implementation on the dynamic tree defined in section Sec:5.5.1, one should define some notations. Let call *orphan node* any node that belongs to a subtree isolated from its corresponding terminal. Indeed this is the case when the tree root of the considered node is not a terminal.

The fixing operation described in the second part of the section Sec:5.5.1 is the corresponding implementation of Relabel step on a dynamic tree. This operation is called *adoption* and could succeed (*fast-adoption* or *slow-adoption*) or failed (*orphanize*).

Let now adapt the push step to the dynamic tree. Indeed, it is replaced by the following implementation. We also give the pull step implementation to fix minds.

Algorithm 9 Push on Dynamic Tree

Require: p a non-orphan excess node.
  $q \leftarrow parent(p)$.
  Send $\min(excess(p), c_f(p,q))$ flow to the node q.
  return $c_f(p,q) == 0$.

Algorithm 10 Pull on Dynamic Tree

Require: p a non-orphan deficit node.
  $q \leftarrow parent(p)$.
  Grab $\min(deficit(p), c_f(q,p))$ flow from the node q.
  return $c_f(q,p) == 0$.

Let now define a Push-Pull step based on the two previous (see Alg:11). We could know define the complete Active Cuts algorithm on dynamic trees.

The proposed algorithm is able to take advantage of the dynamic tree in order to speed up the algorithm but also to improve its result pertinence.

Algorithm 11 Push-Pull on Dynamic Tree

Require: p a non-orphan excess or deficit node.
  if p is a excess node **then**
    $saturated \leftarrow push(p)$.
  else
    $saturated \leftarrow pull(p)$.
  end if
  $state \leftarrow succeed$.
  if $satured == true$ **then**
    $state \leftarrow adoption(p)$.
  end if
  if $state == succeed$ **then**
    return DONE.
  else
    return STUCK.
  end if

This improvement is based on the following observation. Since we can detect easily when some part of the graph is isolated from the targeted terminal (*Orphanize* proceduce), we can detect excesses and deficits that cannot reach their terminal before convergence of the algorithm.

Thus we do not wait for convergence to convert back stuck excesses and deficits to t-links. This allows increasing the number of intermediate cuts returned by the algorithm. Yet the number of returned cuts depends on the graph but also on the strategy for active node selection. Note that the returned series is a non strictly decreasing cuts series. Figure Fig:5.18 illustrates a cycle of the Active Cuts algorithm on Dynamic Trees on a simple graph.

Complexity of such an algorithm also depends on the selecting order for push-pull as discuss in [39, 75, 76, 42]. This will be discussed later.

The next section will provide some interesting properties of the proposed algorithm.

5.5.3 Properties

The first properties discussed here is the decreasing cost of cuts in the returned series of the algorithm. Next, we will provide a result related to the cost and the relative position of two successive cuts. But first, let give a trivial lemma.

Lemma 1 *It cannot happen that two successive cuts intersect each other.*

Algorithm 12 Active Cuts

Require: $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ a weighted graph.
Require: An initial cut C_0.
  $n \leftarrow 0$.
  Initialize according the initial cut C_0 (see Sec:5.3.2 for details).
  Paint every node in *Gray*.
  repeat
    $n \leftarrow n + 1$.
    Grow the trees into *Gray* nodes.
    repeat
      Push-Pull().
    until STUCK is returned
    Paint the isolated subtree in *Gray*.
    $\delta \leftarrow \sum_{p \in Gray} excess(p) + deficit(p)$.
    if $\delta < 0$ **then**
      Let note C_n the boundary between red nodes and other nodes.
      Convert back deficits of the Gray zone into sink t-links (see Fig:5.1 for details).
    else
      Let note C_n the boundary between blue nodes and other nodes.
      Convert back excesses of the Gray zone into source t-links (see Fig:5.1 for details).
    end if
  until there is no more excess and deficit nodes
  Every red node is assigned to the source.
  Every blue node is assigned to the sink.
  Minimal cost cut of the graph is $C_n = C_{min}$.
  return $(C_0, C_1, \cdots, C_{min})$.

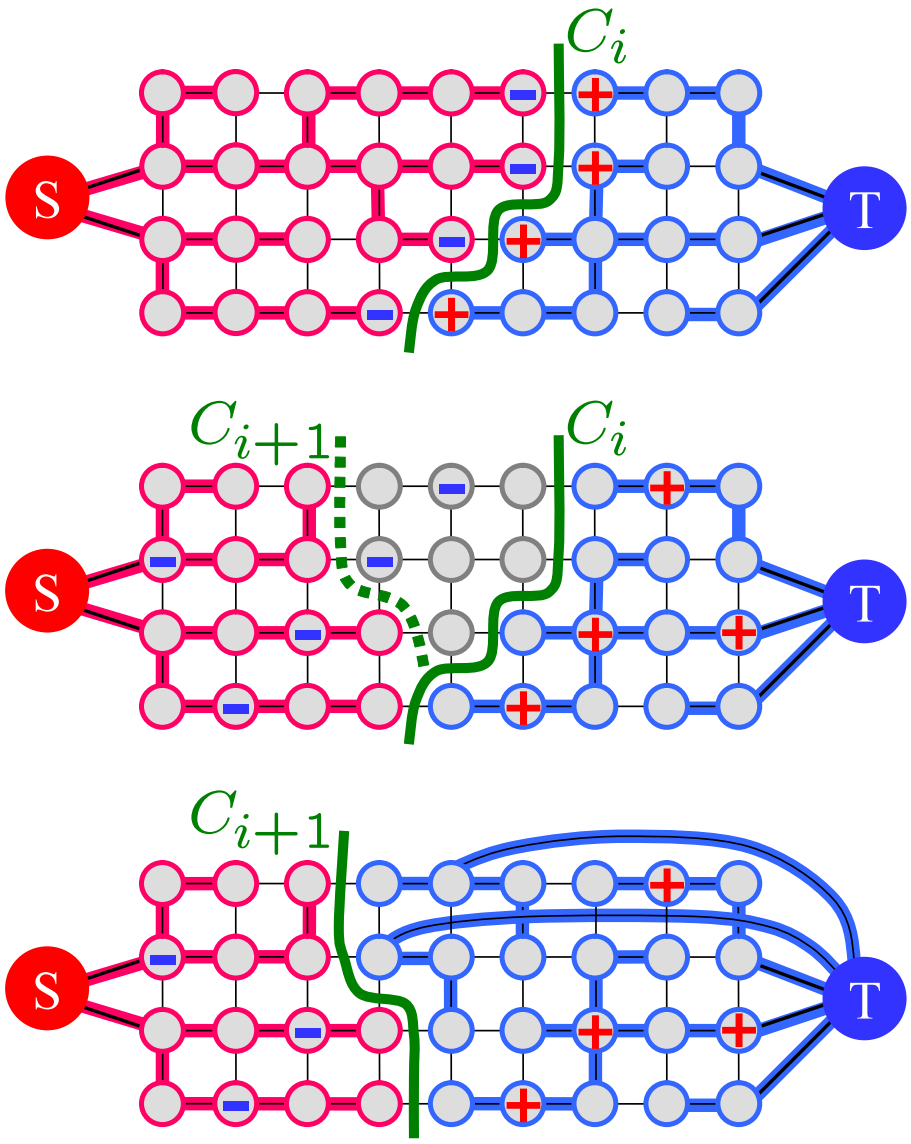


Figure 5.18: This figure illustrates a cycle of the Active Cuts algorithm on Dynamic Trees on a simple graph. - Top: C_i is the current cut. - Middle: Some excesses or deficits (here deficits) get stuck, a new cut appears C_{i+1}. - Bottom: The stuck excesses or deficits are converted back into t-links.

Proof : This is due to the construction of the algorithm. The new cut lies on the boundary of the isolated subtree. Since we only move one node (excess or deficit) at a time and since the two trees are unconnected, the subtree belonged to one side of the graph only. These two observations lead to the result. □

Property 2 *The proposed algorithm, namely Active Cuts on Dynamic Trees, produce a series of cuts of decreasing cost.*

Proof : Let assume that it exists $i \in \{0 \cdots n\}$ such that the cost of two successive cuts is increasing $|C_i| < |C_{i+1}|$. Let define I_i the part of the graph between C_i and C_{i+1} (in fact the gray part).

Due to the orientation of the cut, we have to distinguish the two possible relative positions of the cut C_{i+1} with respect to C_i. C_{i+1} could be on the source side or on the sink side of C_i. And that's all since two successive cuts do not cross each other (see Lem:1).

Using divergence theorem on I_i we obtain the following result:

$$\begin{aligned} \int_{I_i} \nabla \cdot flow &= \int_{\partial I_i} flow \cdot \overrightarrow{n} \\ &= \begin{cases} |C_{i+1}| - |C_i| & \text{if } C_{i+1} \text{ is in the source side.} \\ |C_i| - |C_{i+1}| & \text{if } C_{i+1} \text{ is in the sink side.} \end{cases} \end{aligned}$$

This tells us that if C_{i+1} is in the source side, then $\int_{I_i} \nabla \cdot flow > 0$ which means that some excesses are stuck in I_i and may be already converted into t-links. Thus I_i is painted in red and then even if C_{i+1} is saturated, it does not separate the source from the sink. Thus C_{i+1} is a not a valid cut for the algorithm. The same argument can be used if C_{i+1} is in the sink side. □

We only provide the following proposition without any proof but it can be achieve using a similar approach.

Property 3 *If the cut jumps from C_i to C_{i+1}, then every cut C between C_i and C_{i+1} is such that:*

$$|C_{i+1}| < |C|$$

Proof : This comes from the fact that if there is a cut C such that $|C| \leqslant |C_{i+1}|$ in the area delimited by C_i and C_{i+1}, then it should have be saturated before C_{i+1}. □

5.5.4 Ordering active nodes

In this section, we compare two standard orders used to process active nodes (node where the flow is non-null). The First-In/First-Out (FIFO) and Highest Label (HL) strategies are described in [39, 75, 76, 42]. The table Tab:5.5.4 gives some counting of atomic operations on two 2D segmentations.

We also introduce two variants of the Active Cuts algorithm:

- *Active Cuts with Adoption After All (AC + AA):* this variant consists into postponing *Adoption* until no excess nor deficit can move.
- *Active Cuts with Fast Adoption and Adoption After All (AC + FA+AA):* this variant consists into postponing *Slow Adoption* and *Orphanize* until no excess nor deficit can move.

		AC		AC + FA+AA		AC + AA	
	Ordering	FIFO	HL	FIFO	HL	FIFO	HL
Face	Move	9987	9601	11797	10241	11243	9546
	Adoption	6468	6487	7221	6837	5570	5785
	Fast rate	51	49	44	48	42	40
	Cuts	280	326	4	4	11	11
Lung	Move	329864	147789	240117	157389	255599	x
	Adoption	589966	797692	982476	446172	495188	x
	Fast rate	11	7	3	10	6	x
	Cuts	298	291	18	15	23	x

Table 5.1: This table shows the counting statistics on two image segmentations (face and lung) for every variant of the algorithm. We count: Move (push and pull), Adoptions, Fast rate corresponds to percentage of Fast adoption with respect of all adoptions and number of generated intermediate cuts.

It turns out that we obtain a result comparable to standard Push-Relabel. If counting is better for HL strategy, HL is more difficult to implement and takes more time to maintain than a FIFO list. The best compromise for speed is to use standard Active Cuts with FIFO strategy.

Nevertheless, we introduce another trick to improve performance. In order to push or grab flow by only one-step, we push or pull the flow until an edge is saturated. Moreover in order to use a *Discharge* implementation (see Sec:3.3.3), after a moving step if some flow (or deficit) remains, the node is not put at the end of the list but at the front. This trick allows the best performance so far.

5.6 Conclusion

We present a new Active Cuts approach for maxflow/min-cut problems that significantly improves practical efficiency (2-6 times) and applicability of graph cuts to many problems in image analysis (see sections Sec:6.3 and Sec:7.1). The method can effectively use initial solutions that are often available in dynamic and hierarchical set-ups in vision. Such an initial solutions are also available in iterative multilabel optimization techniques like α-expansions [21].

Before converging to a global minima, Active Cuts outputs a multitude of intermediate solutions that, for example, can be used be accelerate iterative learning-based methods, e.g. [148]. It can also be combined with other methods for accelerating graph cuts in dynamic [105] and hierarchical [118] fashion.

However C++ code of [105] was not available at that time and those results should be carefully considered until a complete comparison is done. Nevertheless one can note that techniques describe in [105] can also be integrated to the Active Cuts algorithm.

Chapter 6

Applications to image segmentation

In this chapter, we present some results based on the Stochastic Level Set technique described in section Sec:4. This work stems from a join work with Gheorghes Postelnicu and is published as a conference paper at VLSM'03 [90] and as a journal paper in a special issue of IJCV [92].

Then we proposed a new segmentation procedure to initialize Digital Matting methods [162, 155, 41, 174, 143, 164, 12]. This work is broached using both Level Set (see section Sec:2) and Graph Cuts (see section Sec:3) formulations. This allows us to illustrate a case of obvious superiority of Graph Cuts in comparison to Level Set approach. This work is published as a technical report [89] (for the Level Set approach) and as a conference paper at VLSM'05 [88].

Finally, we present some applications of The Active Cuts algorithm to image segmentation. We show here some interesting aspects of Active Cuts in comparison to standard Graph Cuts on both standard and hierarchical segmentation approaches. This work stems from a join work with Yuri Boykov and will be published as a conference paper at CVPR'06 [87] and is also available as a technical report [86].

6.1 Stochastic Active Contours

Stochastic Active contour (Sec:4.3.3) could be used in the Geodesic Active Contours framework [27] where segmentation is based upon gradient intensity variations. Yet, a multiscale approach is often used successfully in that context to overcome the local minimum problem. Other segmentation schemes [140] use a region model (eg. texture, statistics) that is less

adapted to multiscale. We will first focus on one such a case, namely the single Gaussian statistics model in [150].

6.1.1 Single Gaussian model

In their unsupervised segmentation framework [150], the authors model each region of a gray-valued or color image I by a single Gaussian distribution of unknown mean μ_i and variance Σ_i. The case of two regions segmentation turns into minimizing the following energy:

$$E(\Gamma, \mu_1, \Sigma_1, \mu_2, \Sigma_2) = \int_{\Omega_1} e_1(x) + \int_{\Omega_2} e_2(x) + \nu \text{length}(\Gamma)$$

where Ω_1 is the region inside Γ, Γ_2 the outside, and $e_i(x) = -\log p_{\mu_i \Sigma_i}(I(x))$ with $p_{\mu_i \Sigma_i}(I(x)) = C|\Sigma_i|^{-1/2} e^{-(I(x)-\mu_i)^T \Sigma_i^{-1}(I(x-\mu_i)/2}$ being the conditional probability density function of a given value $I(x)$ with respect to the hypothesis (μ_i, Σ_i). The parameters (μ_i, Σ_i), estimated from the pixel actually inside and outside Γ, are functions of Γ. Thus, the energy is a function of Γ only: $E(\Gamma, \mu_1, \Sigma_1, \mu_2, \Sigma_2) = E(\Gamma)$. Its Euler-Lagrange equation is not obvious, but finally simplifies into the minimization dynamics

$$\beta_c = e_2(x) - e_1(x) + \nu \, div\left(\frac{du}{|du|}\right)$$

The authors successfully segment two regions, even when they have the same mean but only different variances. However, the evolution could easily be stuck into some local minimum while a multiscale approach might modify the statistics so that no segmentation would be possible anymore. As demonstrated figure 6.1, a simple Simulated Annealing scheme with dynamics (4.26) overcomes this problem. Figure 6.2 shows the same phenomenon on a real image. Note that this image was successfully segmented by the authors of [140]. Yet, they used an adapted model of texture. Here, the Stochastic Active Contours framework succeeds in making a simple unsupervised single Gaussian model recover the correct regions.

6.1.2 Gaussian mixtures

As an illustration of the case when the Euler-Lagrange equation cannot be computed, we extend the previous method to region statistics modeled by a mixture of Gaussian distributions of parameters $\Theta_i = (\pi_i^1, \mu_i^1, \Sigma_i^1, ..., \pi_i^{n_i}, \mu_i^{n_i}, \Sigma_i^{n_i})$. with $\sum_j \pi_i^j = 1$. The conditional probability density function of a given value

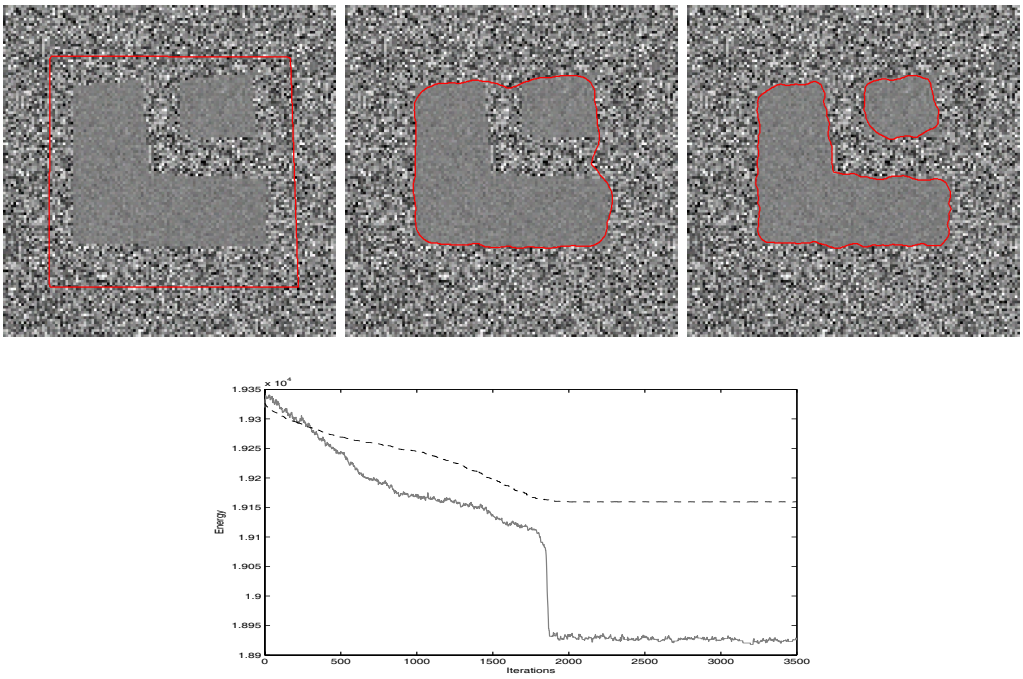


Figure 6.1: Segmentation of two regions modeled by two unknown Gaussian distributions (same mean, different variances). Top row: the initial curve, the final state of the classical approach stuck in a local minimum, and the final state of this method. Bottom row: evolution of the energy (dashed: deterministic method, solid: this method)

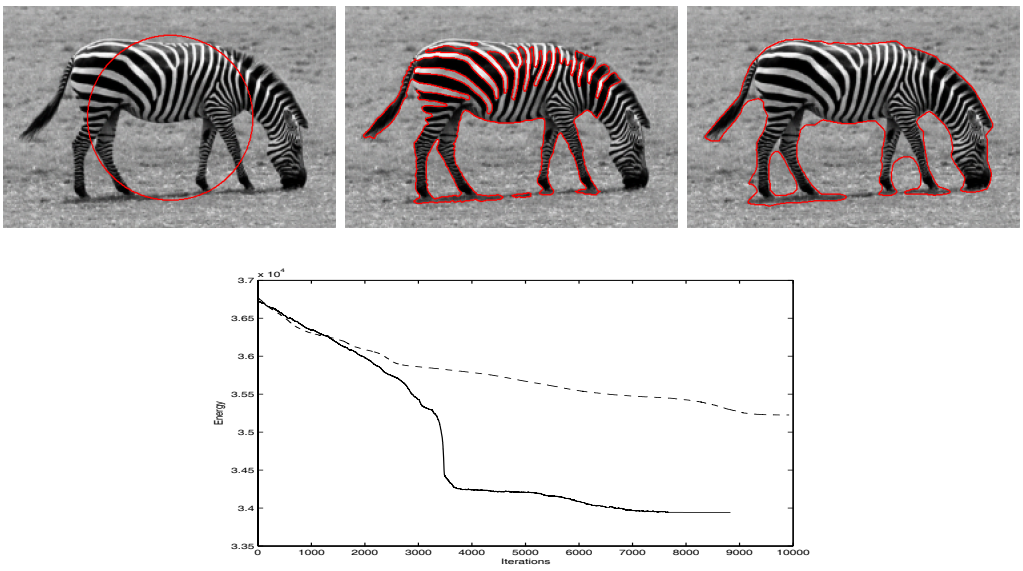


Figure 6.2: Segmentation of two regions modeled by two unknown Gaussian distributions. Top row: the initial curve, the final time step of the classical method, again stuck in a local minimum and the final step of this method. Bottom row: evolution of the energy (dashed: deterministic method, solid: stochastic method)

$I(x)$ becomes:

$$p_{\Theta_i}(I(x)) = \sum_{j=1}^{n_i} \pi_j p_{\mu_i^j \Sigma_i^j}(I(x))$$

The number of Gaussian distributions can be given, estimated at the initial time step, or dynamically evaluated using a Minimum Description Length criterion[146] or the Minimal Message Length method [170]. A large literature is dedicated to the problem of estimating Θ_i from input samples. We have used the original K-Means algorithm pioneered by MacQueen [120], although we have tested extensions like the Fuzzy-K-Means [9, 53], the K-Harmonic-Means [182], and the Expectation-Maximization algorithm (EM), first proposed in [55]. The latter solves iteratively

$$\widehat{\Theta}_i = \arg\max_{\Theta_i} \int_{x \in \Omega_i} \log p_{\Theta_i}\left(I\left(x\right)\right) \, dx$$

(Please refer to appendix for details and references).

The segmentation problem still consists in minimizing the same energy, with now $e_i(x) = -\log p_{\Theta_i}(I(x))$. Unfortunately, we now have to deal with a complex dependency of Θ_i with respect to Γ. In fact, the learning algorithm acts as a "black box" implementing $\Gamma \rightarrow \Theta_i(\Gamma)$. As a consequence, the Euler-Lagrange equation of the energy $E(\Gamma, \Theta_1(\Gamma), \Theta_2(\Gamma)) = E(\Gamma)$ cannot be computed. A deterministic contour evolution driven by $\beta_c = e_2 - e_1 + \nu\kappa$ may get stuck just because $\beta_c \overrightarrow{\mathbf{n}}$ is not the exact gradient. Yet, the Stochastic Active Contours can still be used, with β_c as heuristics. As a simple illustration of this, let us consider the synthetic example of figure 6.3. The region to segment is a square. The square and the background are each modeled by a mixture of two equally weighted Gaussian distributions: $\Theta_i = (\frac{1}{2}, \mu_i^1, \Sigma, \frac{1}{2}, \mu_i^2, \Sigma)$. As an initial guess, we shift the square toward the bottom-right corner. Although a bit of the background is in Ω_1, Θ_1 is correctly estimated. Yet, for some reason, the K-Means algorithm estimates Θ_2 approximatively by $(1-\epsilon, \frac{\mu_2^1+\mu_2^2}{2}, \Sigma', \epsilon, \frac{\mu_1^1+\mu_1^2}{2}, \Sigma'')$. During its convergence, the deterministic method keeps such an incorrect Θ_2 and finally gets stuck at a roughly correct place but with an incorrect model, leaving some interior pixels outside (especially in the corners, because of the smoothing term of the energy). The colored hexagons below the images indicate the means and variances of the mixtures components and their respective weights. See also how the energy increases in the end! On the contrary, this method does not rely completely on the incorrect gradient only and finally *"discovers"* the correct model, leading to a somehow better fit. Notice the energy level drop-down when the K-Means algorithm ejects the interior pixels as negligible and

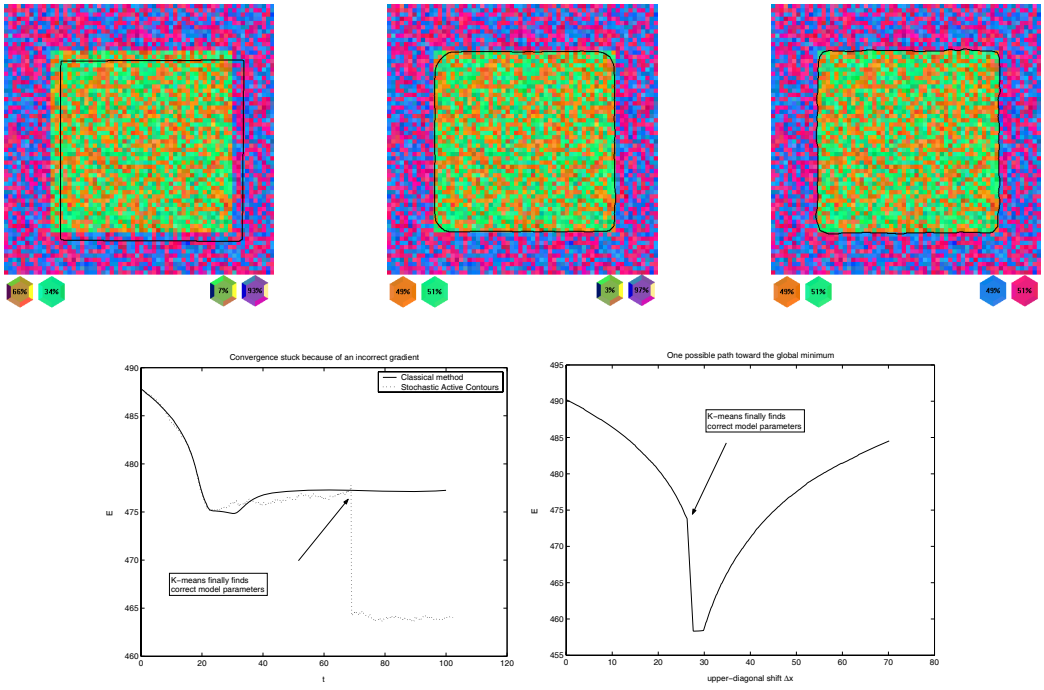


Figure 6.3: A case where the gradient is not correct (see text). Top row, from left to right: initial position, final position with the classical method (the model is not correctly recovered - see percentages in the hexagons), leading to rounded corners), final position with this method (the model is correctly recovered). Bottom left: evolution of the energy in both cases. Bottom right: energy for a translation of the curve that goes through the correct segmentation.

shifts to the correct model. The last graph of figure 6.3 is a plot of the energy when the initial square is *manually* translated from the bottom-right corner to the upper-left one, going through the correct position. It clearly shows that the heuristic gradient by itself gets stuck in a local minimum, whereas this method comes much closer to the desired minimum.

6.1.3 Results

Even when the deterministic scheme converge more or less, this method shows a better ability to overcome local minima: figure 6.4 shows how Γ can be stuck leading to a dramatic evolution toward completely false regions. Finally, figure 6.5 shows some more examples on other real images. Animations corresponding to all the presented examples can be seen on internet website located at `http://cermics.enpc.fr/~juan/IJCV`.

6.1.4 Conclusion

Based on recent work on Stochastic Partial Differential Equations by Lions and Souganidis, we have a simple and well-founded method to implement the stochastic motion of a surface in a Level Set framework. This method is used as the key point of a stochastic extension to standard shape optimization methods in Computer Vision. In the particular case of segmentation, we introduced the *Stochastic Active Contours*, a natural extension of the well-known active contours. Our method overcomes the local minima problem and can also be used when the Euler-Lagrange equation of the energy is out of reach. This extension is not time consuming: the only computational effort is computing the energy . Convincing results are presented with the segmentation of regions modeled by unknown statistics, namely single Gaussian distributions or mixtures of Gaussian distributions.

6.2 Trimap Segmentation

Given an image, digital matting consists in extracting a foreground element from the background. Standard methods are initialized with a *trimap*, a partition of the image into three regions: a definite foreground, a definite background, and a *blended region* where pixels are considered as a mixture of foreground and background colors. Recovering these colors and the proportion of mixture between both is an under-constrained inverse problem, sensitive to its initialization: one has to specify an accurate trimap, leaving undetermined as few pixels as possible.

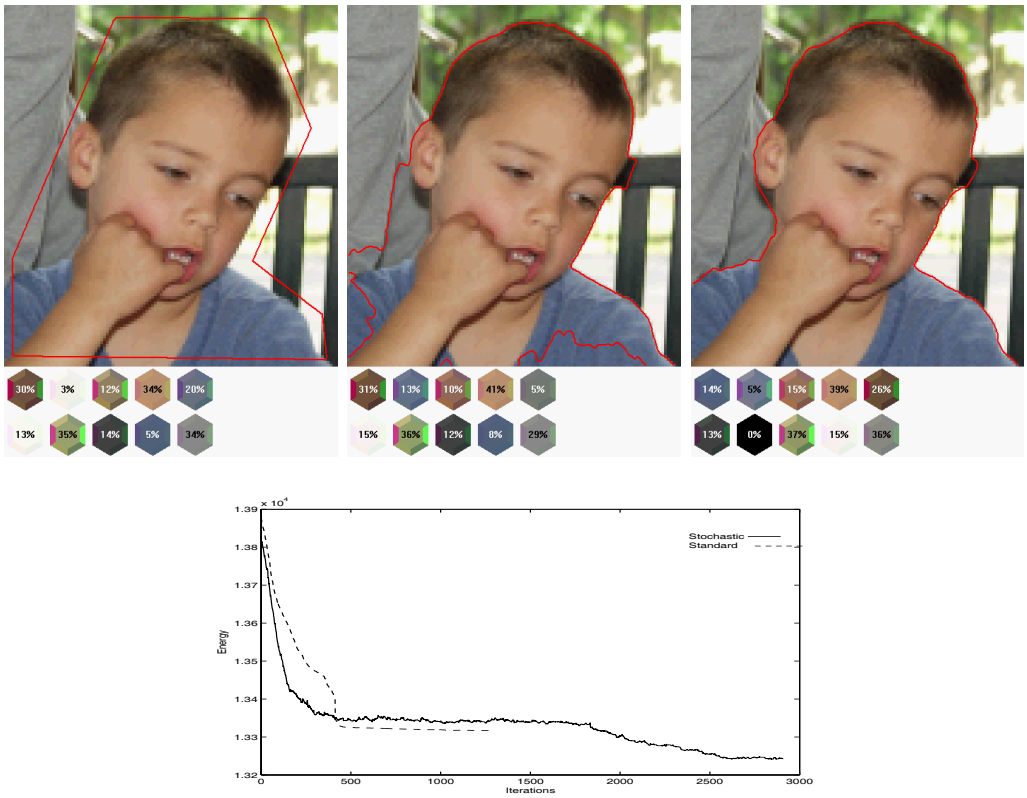


Figure 6.4: Segmentation of two regions modeled by two unknown Gaussian mixtures. Top row: the initial curve, the final state of the deterministic method, stuck in a local minimum and the final state of this method. Bottom row: evolution of the energy.

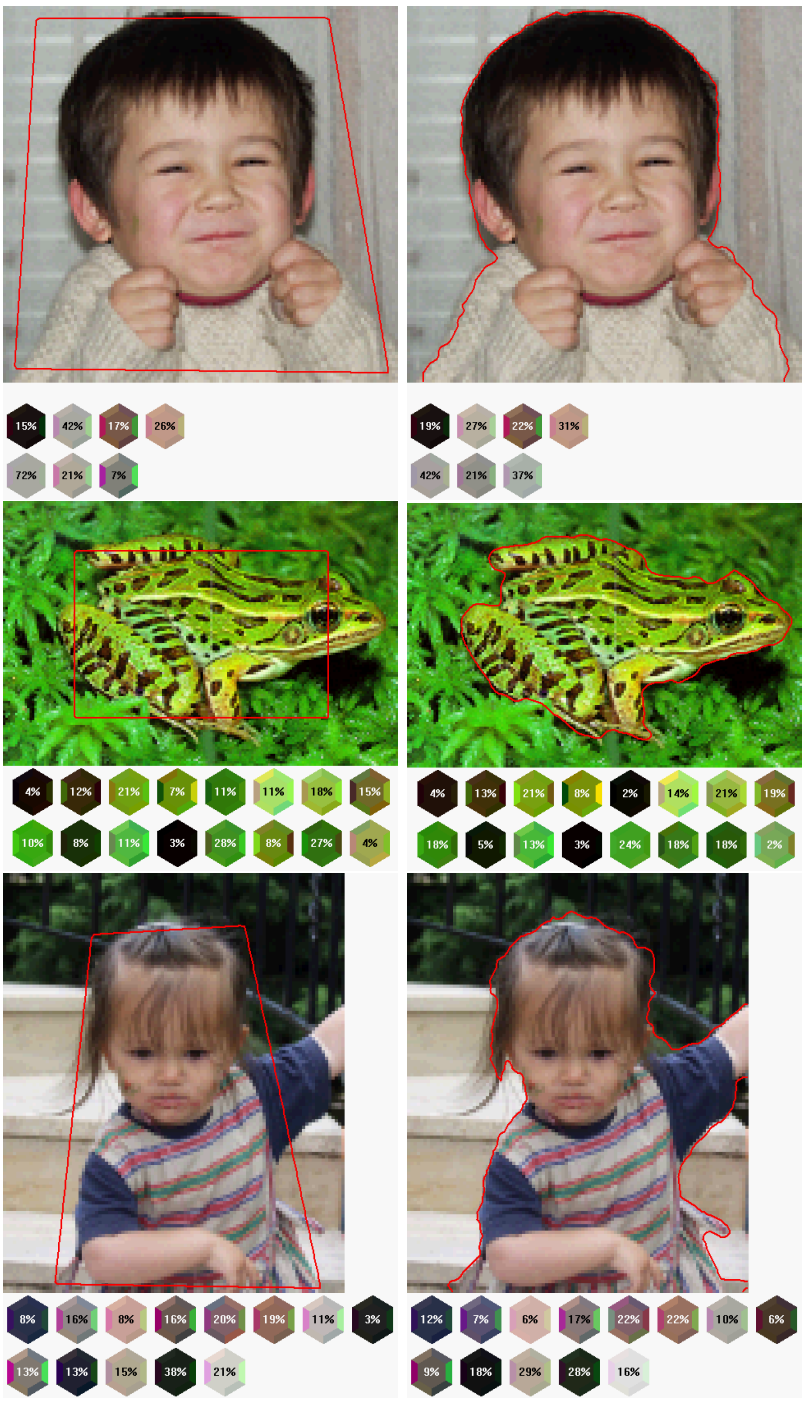


Figure 6.5: Segmentation of two regions modeled by two unknown Gaussian mixtures. Left column: the initial states. Right column: the corresponding final states of this method.

First, we propose a new segmentation scheme to extract an accurate trimap from just a coarse indication of some background and/or foreground pixels. Standard statistical models are used for the foreground and the background, while a specific one is designed for the blended region. The segmentation of the three regions is addressed for comparison by both Level Set method and by an iterative Graph Cut based optimization scheme. This user-friendly trimap is similar to carefully hand specified ones.

As a second step, we take advantage of our blended region model to design an improved matting method coherent. Based on global statistics rather than on local ones, our method is much faster than standard Bayesian matting, without quality loss, and also usable with manual trimaps.

6.2.1 Introduction

The commonly used model of digital or alpha matting is the following. An image I is considered as a mixture between a foreground I_F and a background I_B, mixture quantified by an *alpha mask* $\alpha \in [0, 1]$. For each pixel x, this writes

$$I(x) = \alpha(x)I_F(x) + (1 - \alpha(x))I_B(x) \tag{6.1}$$

Such a blending has multiple reasons: transparent objects, aliasing, blur or motion blur. The problem is to recover I_F, I_B and α from I.

This inverse problem is under-constrained and can not be solved without priors. Historically, a solution was proposed in the case of a known constant background, e.g. a blue screen [162]. Recently, inspired by computer vision techniques, methods based on a model of the foreground and of the background were proposed that greatly improve the matte quality, even without a blue screen. Since the pioneering work of Ruzon and Tomasi [155], several methods have been proposed [41, 174, 143, 164].

As a prerequisite of any method, the user has to specify a so-called *trimap*, partitioning the image into three regions: a set Ω_F of definitely foreground pixels (where α will always be 1), a set Ω_B of definitely background pixels ($\alpha = 0$), and a blended region Ω_M where α, I_F and I_B are unknown. Ω_M has to be an intermediate region, separating Ω_F from Ω_B. Matting methods suffer from sensitivity to this initial condition and one has to specify it accurately, leaving undetermined as few pixels as possible. Moreover, when too small Ω_F and Ω_B are given, the matting process generally does not work at all.

Digital matting was primitively developed for movie production. For a specialist, carefully specifying a trimap is a long but feasible process (actually faster and easier than alpha masking). Today, extracting a subject from a picture for editing purpose becomes a standard in a non professional context.

Speed becomes also an issue, particularly with the ever increasing resolution of digital cameras.

This paper addresses both user-friendly trimap design and speed. First, we propose a trimap segmentation scheme from just a small subset of the background and/or foreground, that can be for instance specified by the user with a brush-like tool. Standard Gaussian Mixture Models (GMMs) are used for foreground and background modeling, while a specific statistical model is proposed for the blended region. To save the user from specifying some obscure number of components, the GMMs parameters are determined with a coupled Expectation Maximization (EM) / Minimum Description Length (MDL) scheme. For the sake of speed, the segmentation of the three regions is conducted simultaneously by an iterative Graph Cut based optimization. The resulting trimap proves to be similar to carefully hand specified ones.

As a second step, we take advantage of our blended region model to design an improved matting method. Based on global statistics rather than on local ones, our method is much faster than the original Bayesian matting, although without quality loss. It can also be used with manually designed trimaps.

6.2.2 Related work

The original work of Ruzon and Tomasi [155] laid the foundations of most of the actual methods, for which the key point consist in modeling the background and the foreground with some statistical model. In their famous *Bayesian Matting*, Chuang et al. [41] improved both the statistical model and the way to use it to recover the alpha mask and the original background and foreground colors. Since then, Rother et al. proposed *GrabCut* [12, 147], a method inspired by Boykov and Jolly work [14], where the image is actually segmented into two regions using an iterated Graph Cuts [109] scheme. A smooth alpha mask is then modeled as a ramp of variable width to be estimated. As a result, it is unadapted to non smooth objects like hairs or trees. The GrabCut method does not need a trimap. It can be seen more as a two regions segmentation with a smooth transition between the two regions, than as a strictly speaking digital matting method. However, as another member of iterated Graph Cuts methods [14], our trimap segmentation has similarities with the segmentation step of GrabCut.

In their *Poisson Matting*, Sun et al. propose another prior on α, based on its gradient and Poisson equations, already used in image editing [143]. Their method supply different modes, refinements and filters, manually invoked by the user. Again, priors on α or its gradient can be questionable as the blending might have different origins and the blended objects different scales with respect to pixels size. Moreover, manual decisions might limit

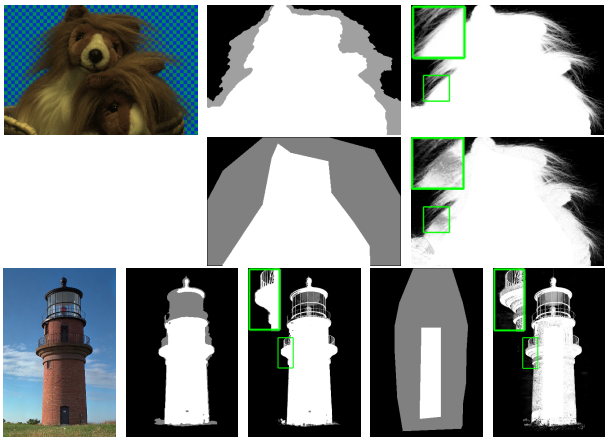


Figure 6.6: Sensitivity of Bayesian matting [41] to the trimap. First row: the original image, an accurate trimap and its corresponding alpha. Second row: a coarse trimap and its alpha. Third row: same as rows 1 and 2 for another image.

the usability of this technique for non specialists. In conclusion, Bayesian matting can be considered as the less *ad-hoc* method so far. Its weak points are the need of an accurate trimap (a problem common with other matting techniques) and its slowness due to many local statistics estimations. Figure 6.6 demonstrates how a coarse trimap affects digital matting.

To our knowledge, the only works addressing trimap design are video oriented. Following the original work by Mitsunaga et al. [125], one can specify trimaps for some *key frames* and interpolate them in the intermediate frames. In their recent work, Xiao and Shah [176] proposed an occlusion based trimap extraction couple with motion layer segmentation. However, it is unusable not only for still pictures but also in real film production where motion is often fast and/or heavily blurred.

This section is organized as follows. First, section Sec:6.2.3 exposes the background-foreground model and our parameters estimation method. Then, section Sec:6.2.4 introduces the trimap segmentation, details its implementation, and compares it with manual segmentations. Finally, section Sec:6.2.5 proposes an improved fast, global and accurate matting method, and shows results.

6.2.3 Two Regions Segmentation

As a first step toward our trimap segmentation, we first focus on segmenting an image into two regions, each of them having its own characteristics, a-priori unknown. This often called *Unsupervised Segmentation* has recently

received a lot of attention from the Computer Vision community. Many approaches have been proposed, among which some Level Set [132, 131] based methods (e.g. [24, 93]). More recently, using the Graph Cuts framework, Boykov and Jolly initiated an iterated method [14], further developed by Rother et al. in [12] and in their GrabCut scheme [147].

In this section, we briefly describe the segmentation part of the GrabCut scheme. Already known to the GrabCut aware reader, the content of this section introduces definitions and notations. The slight difference with the original work is that we plead for a more sophisticated parameter estimation method, EM + MDL based, mathematically more justified, more user-friendly, and yielding somehow better results.

Let I be a color image defined over a domain Ω. For all $x \in \Omega$, $I(x)$ is a pixel defined in a color space (e.g. RGB or CieLab). Let Ω_U be a part of Ω specified by the user. Our goal here is to segment Ω into two *"coherent"* regions that we will abusively still call the background and the foreground, respectively still denoted by Ω_B Ω_F, such that $\Omega_U \subset \Omega_B$.

Region Modeling

Following previous work and using a statistical approach, each region Ω_X ($X = F$ or B) is modeled by a Probability Density Function (PDF) approximated by a Gaussian Mixture Model (GMM):

$$p_X(I) = \sum_{i=1}^{N_X} \pi_i^X G_{\mu_i^X, \Sigma_i^X}(I) \text{ with } \sum_{i=1}^{N_X} \pi_i^X = 1 \text{ and } \pi_i^X \in [0,1]$$

Each component is represented by a Gaussian of mean μ_i^X and covariance Σ_i^X: $G_{\mu,\Sigma}(I) = \frac{|\Sigma|^{1/2}}{(2\pi)^{3/2}} e^{-(I-\mu)^T \Sigma^{-1}(I-\mu)/2}$ and π_i^X is the prior of the i^{th} component with respect to all components, i.e. its proportion in the mixture.

Estimating the parameters $\Theta_X = \{N_X, (\pi_i^X, \mu_i^X, \Sigma_i^X)_{i=1..N_X}\}$ is a widely studied problem. For a given N_X, one can use the K-Means algorithm (see [121]), a fast but approximate method. This method is widely sensitive to its initialization. Moreover it does not provide a likelihood maximum, which is not appropriate for a segmentation based on likelihood maximization. Indeed the K-Means just solves:

$$(\mu_i^X, \Sigma_i^X) = \arg\min_{(\mu_i, \Sigma_i)} \sum_{x \in \Omega_X} \left\| I(x - \mu_{k(I(x))} \right\|^2_{\Sigma_{k(I(x))}}$$

with $k(I) = \arg\min_k \|I - \mu_k\|^2_{\Sigma_k}$, $\|I - \mu\|_\Sigma$ being the Mahalanobis distance between I and μ with respect to Σ. Note that [12] suggests [175] as a variant and that [41] uses the method in [130]. We prefer the EM algorithm [124, 123].

It is much more robust with respect to the initial parameters and provides a likelihood maximum, solving:

$$(\pi_i^X, \mu_i^X, \Sigma_i^X) = \arg\min_{(\pi_i,\mu_i,\Sigma_i)} \sum_{x\in\Omega_X} p(I(x))$$

Finally, we combine the EM algorithm with a MDL [68] estimation of N^X, saving the user from manually adjusting the number of Gaussian components. Note that we have also tested more recent algorithms like Split and Merge EM [167], without any significant improvement.

Energy Design

Let γ be the partition function of Ω into Ω_F and Ω_B: $\gamma(x) = F$ if $x \in \Omega_F$, $\gamma(x) = B$ otherwise. Under the hypothesis that regions are independent with respect to their color distribution, it is natural to use the posterior probability of the pixels as a segmentation criterion, thus stating the problem as minimizing an energy:

$$E_{data}(\gamma) = \int_\Omega -log\ p_{\gamma(x)}(I(x))\ dx \tag{6.2}$$

An extra control term should be added to constrain the smoothness of the solution. However since this term depends on the chosen framework, we will give them in the next sections.

Level Set

Unsupervised Segmentation of an image has recently received a lot of attention from the Computer Vision community. Many approaches have been proposed, the majority of them are Level Set [132, 131] based (e.g. [24, 93]). We will follow this track and refer the reader to [24] for details.

Modeling the pixel colors in each region by some statistical model, the unsupervised segmentation problem can be formulated as the minimization of the energy:

$$E = E_{data} + \nu\, length(\Gamma)$$

If we assume the PDFs fixed and E depending only on Γ (contour of γ), it can be shown that the *shape gradient* (see [54]) leads to the following steepest gradient descent minimization scheme:

$$\begin{aligned} \Gamma(p,0) &= \Gamma_0(p) \\ \frac{d\Gamma(p,t)}{dt} &= [p_2(C(\Gamma)) - p_1(C(\Gamma)) + \nu\kappa]\,\overrightarrow{\mathbf{n}} \end{aligned}$$

where $\Gamma_0(p)$ is some initial guess, $\Gamma(p,t)$ the contours evolving toward the minimum of E, κ their curvature and $\mathbf{n}$ their unit normal.

The resulting segmentation is said *unsupervised* because the PDFs are not fixed but considered unknown and thus re-estimated at each time step using the current position of the contour. In some cases (depending on how the PDFs are estimated), the previous shape gradient is exact, even when taking into account the dependency of the PDFs upon Γ.

Graph Cut

When minimizing E either with a Level Sets Method approach [149, 92] or with a Graph Cuts one [14], one should be aware of the dependency of the PDFs upon γ. This leads to an iterated process that is usual in the Level Sets gradient descent, but is not in the case of Graph Cuts. As we do not need sub-pixel accuracy, we opt for a Graph Cuts approach, mainly for speed reasons. Using EM instead of K-means is theoretically important: the algorithm consists in alternately updating the PDFs according to the segmentation and in segmenting according to the PDFs. At each step, the energy decreases:

- Updating the PDF using EM ensures that E_{data} decreases, E_{smooth} being fixed.
- The Graph Cuts step ensures that E decreases.

The smoothing term in a graph cut framework is often addressed as a local smoothness constrain: neighbor pixels should belong to the same region. This yields an additional smoothness energy:

$$E_{smooth}(\gamma) = \int_{\Omega} \left(\int_{y \in \mathcal{N}(x)} \mathcal{V}(x,y)\ dy \right)\ dx$$

where $\mathcal{N}(x)$ is a local neighborhood of x and $\mathcal{V}(x,y) = \mathcal{V}^0(x,y)$ if $\gamma(x) \neq \gamma(y)$ with $\mathcal{V}(x,y) = 0$ otherwise.

Under the assumption that the frontier between the two regions corresponds to high image gradients, a frequent choice is $\mathcal{V}^0(x,y) = \lambda \exp(-\frac{\|I(x)-I(y)\|^2}{2\sigma^2})$ where λ is some positive constant controlling the degree of smoothness and σ is set as in [14]. The global energy to minimize ends to:

$$E(\gamma) = E_{data}(\gamma) + E_{smooth}(\gamma)$$

Let us just recall useful notations [22]. We consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ that is a set of nodes $\mathcal{V}$ and directed edges $\mathcal{E}$ connecting them. Two special

terminal nodes are present: the source s and the sink t. Each edge (p, q) connecting a node p to a node q is assigned a weight $t_{p,q}$. Edges are broken in two groups: n-links and t-links. A n-link is an edge connecting two non-terminal nodes. A t-link connects a non-terminal node to a terminal node. A cut C is a partitioning of the nodes of the graph into two disjoint subsets S et T such that the source $s \in S$ and the sink $t \in T$. Its cost is the sum of the weights of all edges (p, q) such that $p \in S$ and $q \in T$. A minimum cut is a cut with minimal cost and one minimum cut can be determined in polynomial time with a max-flow extraction algorithm.

Here, each pixel of the image is associated to a node and to edges for each of its neighbors. Each node is also connected to the sink and the source. The weights on the t-links deal with data constrain and those on the n-links account for smoothness. For a pixel x associated to node p, let D_X be the negative logarithm of the probability density function associated to region Ω_X: $D_X(p) = -\log\ p_X(I(x))$. The Graph is built according to table 6.2.3. After the cut, the nodes that are still connected to the source, are assigned to Ω_F, the others to Ω_B. Figure 6.7 shows the result of this segmentation process on some test image using both the method in [12] and a method using an EM/MDL estimation. Note that some details misclassified by the original method are correctly handled by the EM/MDL approach. Yet, these improvements are not decisive. More important is the fact that the MDL based estimation of N_X proves to be reliable and masks one annoying parameter from the user.

link	weight	for
$t_{s,p}$	0	$p \in \Omega_U$
$t_{p,t}$	∞	$p \in \Omega_U$
$t_{s,p}$	$D_B(p)$	$p \notin \Omega_U$
$t_{p,t}$	$D_F(p)$	$p \notin \Omega_U$
$t_{p,q}$	$\mathcal{V}(p, q)$	$q \in \mathcal{N}(p)$

Table 6.1: Weights associated to a node p in the graph

6.2.4 Trimap

Assuming that the blended region will also be modeled by a PDF $p_M(I)$, still to be modeled, the data driven part of the energy is unchanged and given by equation (6.2) with a new partition function that reflects the 3 regions.

However, keeping the same smoothing term is a nonsense. A high image gradient does not indicate a frontier between two regions anymore. Instead,

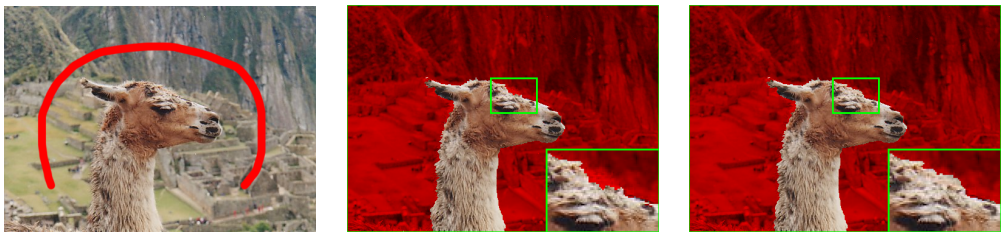


Figure 6.7: Importance of then EM estimation and reliability of the MDL criterion. Original image with background specification in red (left) and the corresponding segmentations using the method in [12] with fixed $N_F = N_B = 5$ (middle) and our EM/MDL approach (right)

we use the length of the frontiers separating the regions as a smoothing energy. Classical in the Level Set framework and inducing mean curvature motion, this can also be handled rigorously in an Markov Random Field framework (see [17]). Here, we will restrict ourselves to an approximation, just replacing the previous $\mathcal{V}^0(x, y)$ to a decreasing function of the distance between x and y (e.g. $\mathcal{V}^0(x, y) = \lambda/(1 + d(x, y))$).

Keeping the same GMM models for p_B and p_F, we still have to design a model for p_M in order to define the energy to minimize.

PDF

A straightforward solution would be to take a third GMM for p_M and to estimate its parameters $\Theta_M = \{N_M, (\pi_i^M, \mu_i^M, \Sigma_i^M)_{i=1..N_M}\}$ via the same EM/MDL scheme as for p_B and p_F. It would be a mistake. Indeed, p_M is not independent from p_B and p_F: in Ω_M, I, I_F and I_B are related by equation (6.1). Despite this, one could willingly ignore this dependency and try to segment $(\Omega_B, \Omega_M, \Omega_F)$ as three regions with each one its own independent GMM. Unfortunately, it is not obvious that the resulting iterated minimizing process will converge to the desired regions without a very accurate initialization, specifying pixels of the three regions. On the contrary, making p_M depend on p_B and p_F will turn out to be sufficient to keep a coarse initialization Ω_U.

Let us examine equation (6.1). We will assume for simplicity that both I_F and I_B come from one single Gaussian of the respective GMMs p_F and p_B. In their Bayesian estimation of layers from multiple images, Wexler et al. [174] assume that α follows a Beta law. Yet, they choose the parameters of the Beta distribution by estimating them on some reference image. Thus, although Kitamoto gives in [103, 104] a Gaussian approximation of a mixture of two Gaussian distributions when the mixture coefficient follows a Beta

Law, we prefer to simply consider that α follows a uniform law. In that case, if I_F comes from $G_{\mu_i^F,\Sigma_i^F}$ and I_B from $G_{\mu_j^B,\Sigma_j^B}$, the distribution of I can be approximated by another Gaussian $G_{\mu_{ij}^M,\Sigma_{ij}^M}$, given also by Kitamoto in [103, 104] as:

$$\mu_{ij}^M = \frac{\mu_i^F + \mu_j^B}{2} \quad \text{and} \quad \Sigma_{ij}^M = \frac{1}{3}\left(\Sigma_i^F + \Sigma_j^B\right) + \frac{1}{12}\left(\mu_j^B - \mu_i^F\right)\left(\mu_j^B - \mu_i^F\right)^T \tag{6.3}$$

Note that this is, again, an approximation and that more sophisticated models could be investigated. Actually, our simple assumption of a uniform α, and of a Gaussian approximation for I, will turn out to give good results. With this choice, it is natural to model p_M with another GMM, whose $N_M = N_F N_B$ components are now fixed and dependent on p_F and p_B:

$$p_M(I) = \sum_{i=1}^{N_F}\sum_{j=1}^{N_B} \pi_{ij}^M G_{\mu_{ij}^M,\Sigma_{ij}^M}(I)$$

where $\sum_{ij}\pi_{ij}^M = 1$ and where the $(\mu_{ij}^M, \Sigma_{ij}^M)$ are given by equation (6.3). The only free parameters are the (π_{ij}^M), and we estimate them with an EM algorithm on Ω_M.

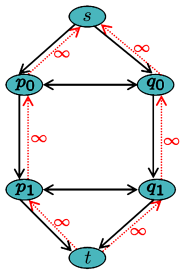
Implementation

Level Set: For a level set implementation the reader is refereed to Sec:2 and Sec:6.2.3.

Graph Cut: As we assume that the blended region Ω_M separates Ω_F from Ω_B, we can use the Graph Cuts implementation described in [80] which is simpler than the usual α-expansion based algorithm and provides a global minimum. Each pixel x is represented by two nodes p_0 and p_1. The graph is built according to figure 6.8. After the cut, each node is labeled according to the following rule:

- If the link between $\{s, p_0\}$ is cut, the node is assigned to the foreground.
- If the link between $\{p_0, p_1\}$ is cut, the node is assigned to the blended region.
- If the link between $\{p_1, t\}$ is cut, the node is assigned to the background.

Here we use the method described by Kolmogorov and Zabih in [109] to force the algorithm to cut one and only one of the three links $\{s, p_0\}$, $\{p_0, p_1\}$ and $\{p_1, t\}$. It consists in adding infinite reverse edges on the graph (see red



link	weight	for
t_{s,p_0}	∞	$p \in \Omega_U$
t_{p_0,p_1}	∞	$p \in \Omega_U$
$t_{p_1,t}$	0	$p \in \Omega_U$
t_{s,p_0}	$D_F(p)$	$p \notin \Omega_U$
t_{p_0,p_1}	$D_M(p)$	$p \notin \Omega_U$
$t_{p_1,t}$	$D_B(p)$	$p \notin \Omega_U$
t_{p_i,q_i}	$\mathcal{V}(p,q)$	$q \in \mathcal{N}(p)$

Figure 6.8: Trimap segmentation. Graph representation for two nodes p and q and associated weights.

links on figure 6.8). Like in the two regions case, we use an iterative scheme. However we found that using the two regions segmentation as a first step gives good initial estimates for p_B and p_F and speeds up the convergence.

First results

Level Set: As we want the image to be segmented in three regions, we use two level sets, following the work in [168]. The energy for this segmentation is similar to the two regions segmentation one:

$$E = E_{data} + \nu \left(length\left(\Gamma_{FM}\right) + length\left(\Gamma_{MB}\right)\right)$$

where the PDF for the three regions (Foreground, Mixed and Background region) are three GMM such as in two regions segmentation. Here we do not make use of the new designed PDF.

We see from the Figure 6.9 on the right, that optimizing the contour using standard unsupervised segmentation is not appropriate as the contour penetrate too much in the Mixed region (motion blur wing of the Butterfly).

In fact by using GMM for the Mixed region, we have assume all the regions to be independent. However that's not the case for the Mixed region and the new model must be used to describe this one.

Graph Cut: Figure 6.10 shows the trimap obtained for the reference image in [41] from just a coarse indication of the background. It is similar to the hand designed one used in the original work. For comparison purposes, we show also the poor trimap obtained when naively modeling the blended region with an independent GMM, even when starting from a more accurate initialization. This confirms the result obtained with levelset implementation and shows also that the problem detected using level set methods is not based on the local optimality of the provide solution. Indeed even a global optimum (a teach iteration) by using graph cut leads to a poor segmentation.

Figure 6.9: Unsupervised three regions segmentation - Upper left: initialization (Foreground in white, Background in gray) - Lower left: final segmentation - Right: Zoom on the wing segmentation

Figure 6.15**a** and 6.15**b** in next section show many other automatic trimaps. Table 6.2.4 gives the running times of the trimap extraction (and of the first step of two regions segmentation) for some of our test images, on a standard 2.4GHz PC without any specific optimization. These are the times for a complete convergence and the process might be stopped before. A multi-scale approach would also improve speed significantly. Anyway, these are to be compared with the times needed for a cautious manual segmentation, depending on the user's ability and/or equipment. Note that the more complex a manual segmentation would be, the more the automatic segmentation seems to require time to converge (see images on figures 6.15**a** and 6.15**b**).

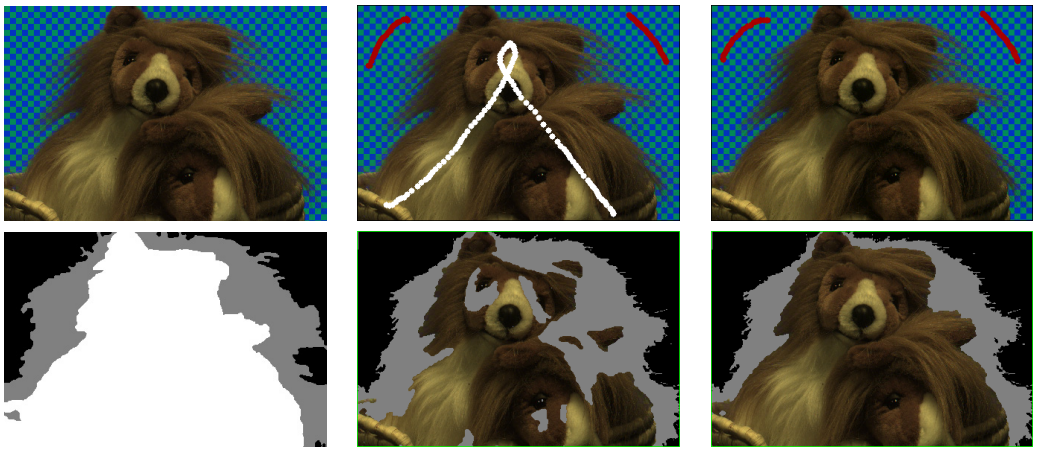


Figure 6.10: Automatic trimaps. First column: the original image and the hand designed trimap used in [41]. Second column: background/foreground initialization (in red/white) and the obtained trimap, naively considering p_M as an independent GMM. Third column: background only initialization (in red) and the trimap obtained with our method

Image	First step	Total time
Teddy Bear	36s	94s
Butterfly	14s	28s
Light	48s	133s

Table 6.2: Running times for trimap segmentation on some test images.

6.2.5 New Matting Algorithm

In this section, we propose a new matting algorithm taking advantage of our blended region model. Based on global statistics rather than on local ones,

it is faster than the original Bayesian matting, although without quality loss.

Chuang et al.'s Bayesian matting algorithm is based on estimating local statistics of the foreground and of the background. For each pixel in the blended region, a neighborhood is considered, where the foreground and the background are respectively modeled by two Gaussian distributions $G_{\mu^F_{loc},\Sigma^F_{loc}}$ and $G_{\mu^B_{loc},\Sigma^B_{loc}}$.

Estimating a local mean and covariance for each pixel is inefficient from a computational point of view. Moreover, limiting the distribution of the neighborhood of a pixel to one single Gaussian may sometimes be a too coarse approximation. We propose to take advantage of our global GMM analysis of the foreground and the background carried out during the segmentation process. Keeping the assumption that I_F and I_B come from one Gaussian each, we choose these two Gaussian distributions respectively among the components of p_F and p_B. We use $\pi^M_{ij} G_{\mu^M_{ij},\Sigma^M_{ij}}(I)$ to measure which Gaussian distributions most probably explain I. Thus, we simply:

1. choose the pair (i_0, j_0) that maximizes $\pi^M_{ij} G_{\mu^M_{ij},\Sigma^M_{ij}}(I)$
2. use Chuang et al.'s solving scheme with $G_{\mu^F_{i_0},\Sigma^F_{i_0}}$ and $G_{\mu^B_{j_0},\Sigma^B_{j_0}}$ as priors for I_F and I_B instead of the local estimations $G_{\mu^F_{loc},\Sigma^F_{loc}}$ and $G_{\mu^B_{loc},\Sigma^B_{loc}}$.

The resulting process turns out to be faster than the original method and the results are similar. Note that it is essential that the GMMs have enough components to explain all the colors/textures locally present in the image. Our EM/MDL estimation ensures this.

6.2.6 Results

Level Set

On synthetic images (Figure 6.11), we show that our algorithm converge toward a good estimation of the real trimap although the initial segmentation is quite far from the solution. Moreover when adding additive Gaussian noise, the algorithm still converges to a trimap not close to the reel one and the Mixed region does not vanish because noisy data.

We used here a bench image given by Chuang et al. On the first row of the figure 6.12, we see the initial image and the initial segmentation where the Foreground is displayed in white and the Background in gray. On the second row, on the left we see the user defined trimap used by Chuang et al, on the right the reconstruct trimap from the initial segmentation using our algorithm. On the third row, on the left we see the true Alpha Matte and on the right the alpha estimation using Chuang algorithm with a bad initialized

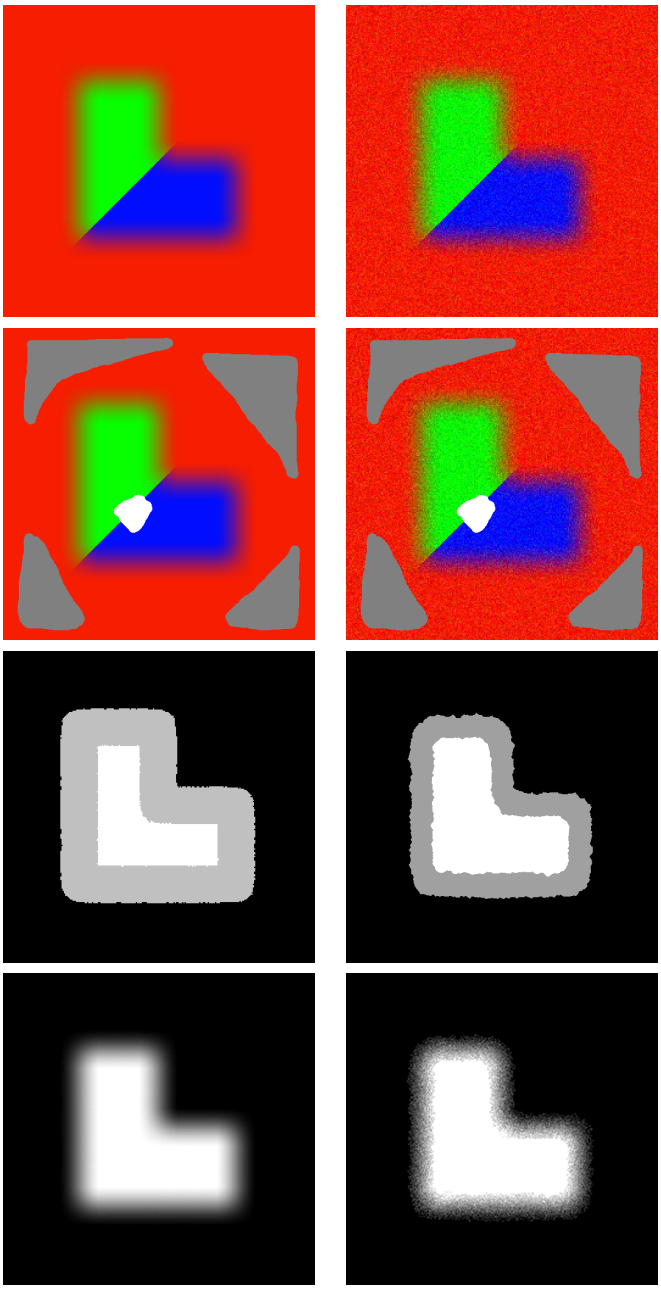


Figure 6.11: First row: initial image, noisy initial image (gaussian blur $\sigma = 20$) - Second row: initial segmentation (foreground in white, background in gray) - Third row: final trimap segmentation (foreground in white, Mixed region in gray, background in black) - Fourth row: Alpha estimation

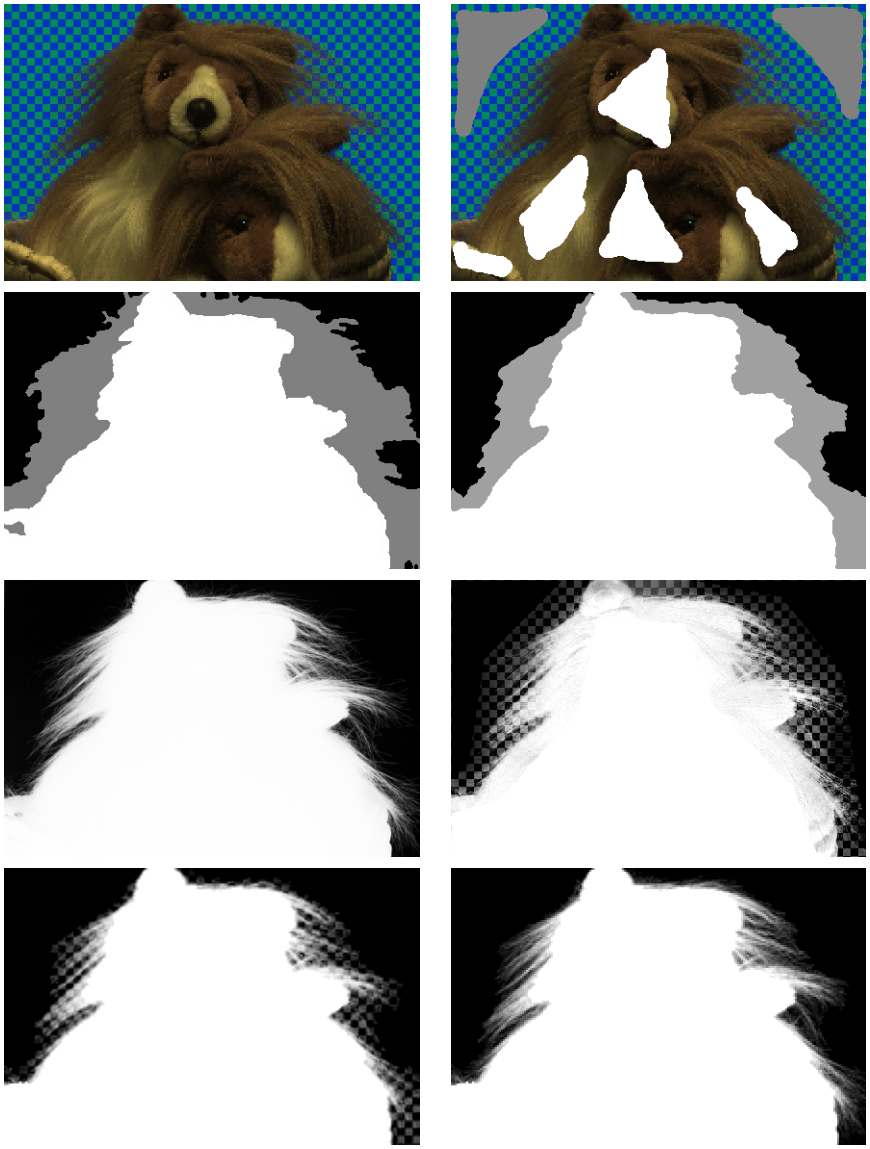


Figure 6.12: Teddy Bear bench - First row: initial image & initial segmentation - Second row: High quality user maid trimap by Chuang et al & final trimap obtained by our algorithm - Third row: True Alpha Matte & Alpha Estimation using a rough trimap and Chuang et al algorithm - Fourth row: Alpha Estimation using our final trimap and Chuang et al algorithm & Alpha Estimation using our final trimap and our Alpha Matting algorithm

trimap. As this technique uses a local estimation of the foreground and background statistics, the quality of the alpha estimation is directly linked to the quality of the initialization. In fact, on the fourth row, on the left, we see the Chuang's alpha estimation using our generated trimap and the result is clearly better than with a rough initialization. On the right, we see our alpha estimator using our trimap. The quality is obviously improved by using our method however our trimap evolution algorithm penetrate too much in the Mixed Region.

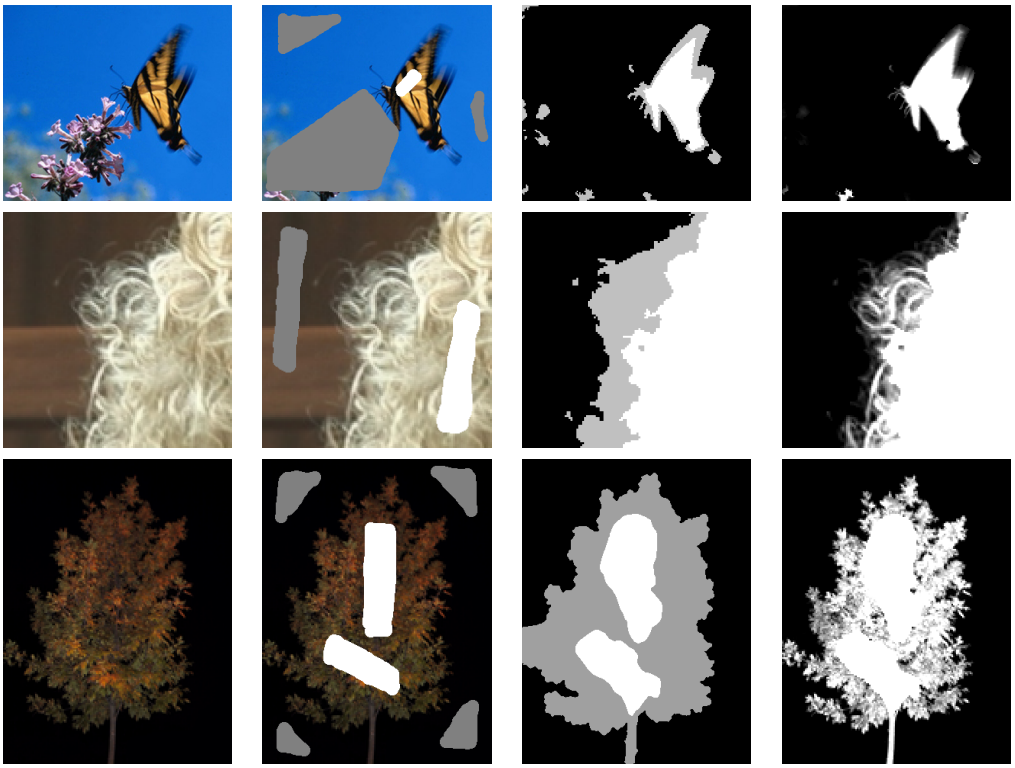


Figure 6.13: Real Image Samples - First column: image - Second Column: Initial segmentation - Third Column: final trimap segmentation - Fourth Column: Alpha Estimation using Chuang Algorithm

If we look attentively to the real images on figure 6.13, we can see that the antenna of the butterfly is not correctly segmented by our algorithm. This is due to the smoothing term of our energy. Such a term is quite unavoidable when using a level set implementation. This should be solved by using the Graph-Cut implementation.

Graph Cut

In their original work on Bayesian matting, Chuang et al. proposed a real bench image, supplying a ground truth for the alpha mask (see [41]). Figure 6.12 shows this *true* mask, compared those obtained with their and our matting algorithm, using our automatic trimap in both cases. The result are similar and the relative errors, in L^2 norm in region Ω_M, respectively gives 1.5% and 1.4% errors.

As expected, the main advantage of our method is its computational efficiency. Table 6.3 gives the running times of both methods for some of our test images, under the same conditions as previously (standard 2.4GHz PC, no specific optimization). We observed a speedup of about 100. Please note that this would also stand when starting from a manual trimap. The only overhead for our matting would be to estimate the global statistics from this trimap before running, which is actually negligible with respect to the matting process.

Finally, figures 6.15**a** and 6.15**b** show the complete process of our method on several test images: the original images (figure 6.15**a** only), the user's initialization, the segmented trimap, the mask obtained with Bayesian matting (fig. 6.15**a** only), the one obtained with our method, and a recompositing from our (α, I_F, I_B) estimation. It demonstrates how a simple initialization without any additional parameter (e.g. number of Gaussian distributions) is enough to get accurate trimaps, and how our fast matting method gives results similar to the ones obtained with the original but slower Bayesian matting.

Image	Bayesian matting	Our matting
Teddy Bear	47s	0.36s
Butterfly	2.7s	0.027s
Light	37s	0.27s

Table 6.3: Running times for the standard Bayesian matting and for our method on some test Trimap/Images.

6.2.7 Conclusion

In this section, we proposed a segmentation method aimed at extracting an accurate Trimap for the digital matting problem. A statistical model is specifically designed for the blended region and both Level Set and iterative Graph Cut based optimization schemes are used. This allows retrieving a

Figure 6.14: From top to bottom, left to right: three alpha masks (ground truth, Bayesian matting using our trimap, our method using our trimap), a recompositing using our mask and foreground estimations

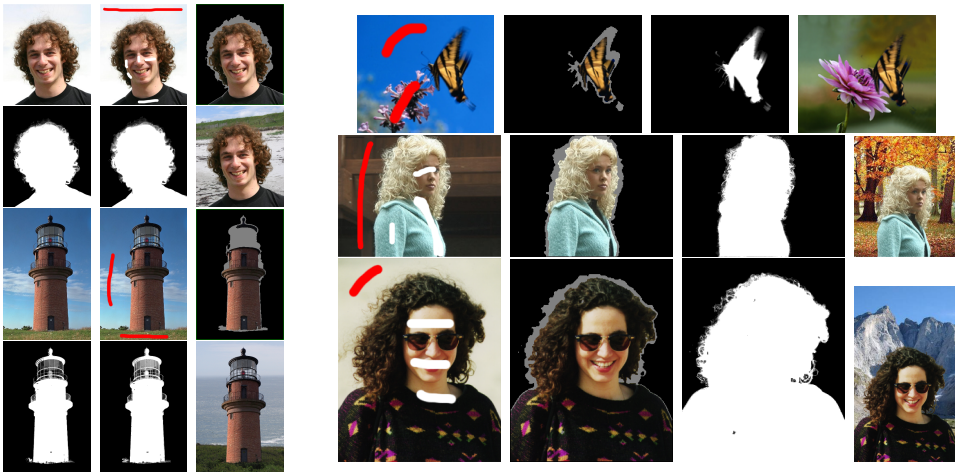


Figure 6.15: **a.** For each image, in reading order: original image, user's initialization, automatic trimap, Bayesian matting, our matting, recompositing. **b.** On each line, from left to right: user's initialization, automatic trimap, our matting, recompositing.

Trimap segmentation from just a coarse specification of some background and/or foreground pixels. It also proved in this case the superiority of the Graph Cuts based method with respect to the Level Set implementation. This is a nice result since Graph Cut approach runs sightly faster than Level Set approach.

The resulting Trimap is similar with those obtained by a meticulous hand drawing. Finally, taking advantage of this blended region model, we describe a improved digital matting method, based on global statistics, much faster than the original Bayesian matting, although without quality loss. This method is also usable starting from a manual Trimap.

6.3 Active Cuts for Image Segmentation

6.3.1 Static Segmentation

Figure Fig:6.19 clearly shows that the Hausdorff distance between initial and optimal segmentation is strongly correlated to the running time of the algorithm. We therefore expect that a "closer" initial cut will yield greater performance. Our experiments show, however, that for static segmentation an initial cut based on a Voronoï partition still gives good results. The more closely the initial cut corresponds to the optimal cut, the faster the convergence.

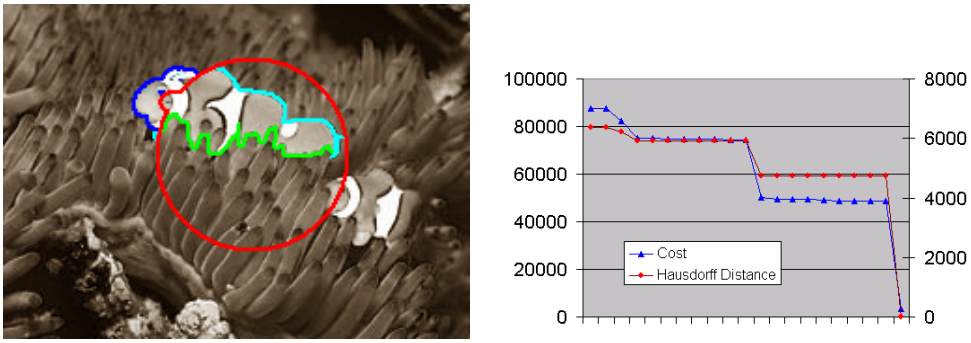
For image segmentation, one should provide initialization for our algorithm. This could be Voronoï partitioning (like in Figure Fig:6.18), a circular area provided by the user (like in Figure Fig:6.16 and Figure Fig:6.17) or some other prior (shape, color, texture...). However this initialization should be carefully chosen as the speed of our algorithm is correlated to the Hausdorff distance between initial and optimal segmentation as shown in the graph of Figure Fig:6.19.

We remarked that successive cuts often appear to **carve** the initial segmentation towards the global minima. In that case, successive cuts near the global optima can be viewed as good local minima and may be used in third party algorithm as highly confident cuts. These cuts may prove useful as shape priors or for updating statistical properties of regions on the fly in applications such as GrabCuts [148].

The speed of our algorithm is typically twice as fast as [18]. However on some "difficult" segmentations, these two algorithms run in comparable time.

(a) Active Cuts (b) Cost of intermediate cuts

Figure 6.16: Active cuts in image segmentation (a). Initial cut is shown in red color. Intermediate cuts (displayed in different colors) gradually "carve" out the global minima solution that accurately follows object boundary. The cost of intermediate cuts and their Hausdorff Distance to the global optima decrease in time (b).



(a) Active Cuts (b) Cost of intermediate cuts

Figure 6.17: Clown fish segmentation over structured background. (a). Initial cut, shown in red color, is a round area given by the user (b). The final cut (global minima, shown in green) accurately follows the fish external contours without becoming stuck on local minima in the clown disguise. The cost of cuts (blue plot) and their Hausdorff distance (red plot) from the global optima decrease in time. Active cuts converge to a globally minimal cut almost twice as fast (4.4ms) as the state of the art max-flow/min-cut algorithm in [18] (7.7ms).

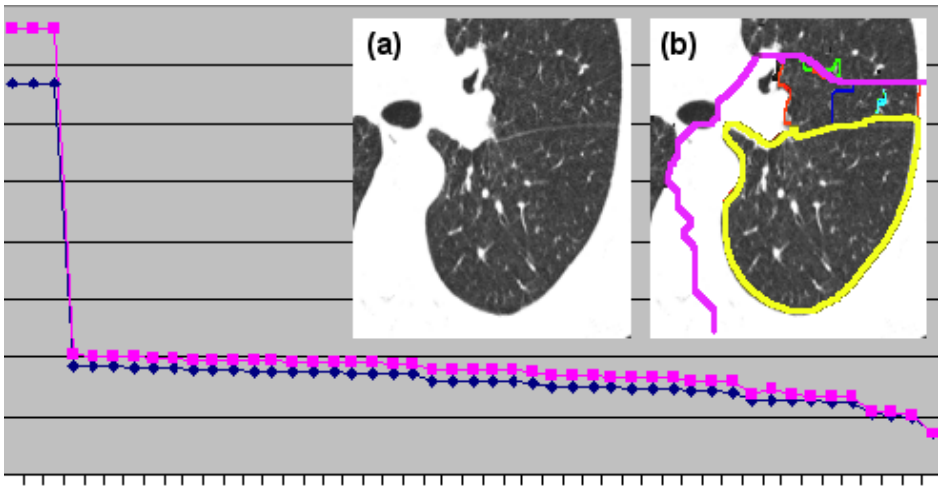
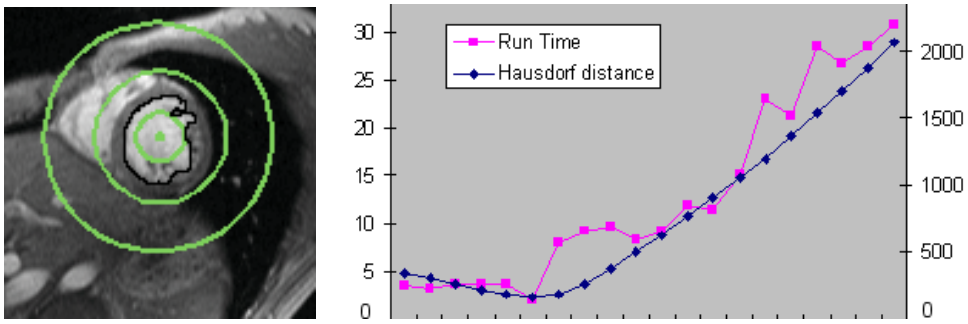
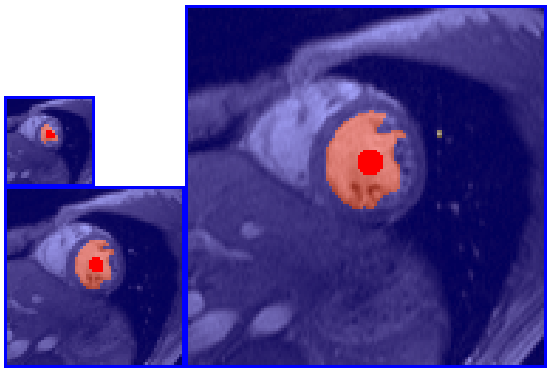


Figure 6.18: Lung lobe segmentation is difficult due significant image clutter (a) that generates a large number of strong local minima. Initial cut is shown in purple color (b). The final cut (global minima, shown in yellow) accurately follows a faint fissure boundary between two lobes. The cost of cuts (blue plot) and their Hausdorff distance (red plot) from the global optima decrease with time. Starting from a remote initial solution Active Cuts converges to a globally minimal cut twice as fast (33ms) as the state of the art max-flow/min-cut algorithm in [18] (69ms).



(a) Initial cuts (b) Run time

Figure 6.19: Active cuts can start from any initial cut (green circles) in (a). Plots (b) show that the running time until convergence to a global minima strongly correlates with a Hausdorff distance between initial cut and the actual minimum cut. The horizontal axis in (b) shows the radius of initial solution.



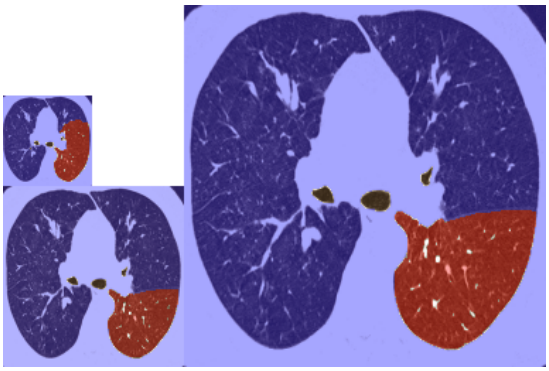
Algorithm	Time
MaxFlow (BK)	18.15ms
ActiveCut	18.52ms
Hierarchical ActiveCut	Level 2 : 0.70ms Level 1 : 0.61ms Level 0 : 8.59ms Total : 9.90ms

Figure 6.20: Hierarchical segmentation using 16-Neighborhood. For each level, initial cut is set to an optimal cut/segmentation from the previous level, the deepest level is initialized using some image partition (like Voronoï partition). Speed of our algorithm is achieved when good initializations are provided. Hierarchical segmentation is an elegant way to estimate a minimum st-cut. Table gives timing comparison for the Boykov-Kolmogorov maxflow/mincut algorithm and our method using standard initialization (Voronoï partition) and Hierarchical initialization with only 2 levels of decimation. One can see a speed improvement around an order of 2.

6.3.2 Hierarcical Segmentation

For static segmentation, ideally one should use prior knowledge to estimate the segmentation (shape, color, texture...). However, if no prior is available, a hierarchical approach provides an elegant way to initialize our algorithm by using segmentation of sub-sampled data. Such "coarse cuts" have recently been used to accelerate graph cuts, but the narrow band technique of [118] sacrifices global optimality by limiting the scope of the solution. In contrast, we simply use the coarse cut as our initial cut on the full graph, thereby retaining global optimality. Figure Fig:6.20 shows an example of hierarchical initialization on large neighborhood using only two levels of decimation. In this example, speed is improved by an order of two even though our hierarchical implementation itself is not optimized.

Our hierarchical algorithm does, on occasion, provide no significant speed up. Figure Fig:6.21 shows a case were a Voronoï initial cut gives convergence as quickly as an initial cut generated by our hierarchical approach. By looking at the table, one can see that most time is spent in the *"Level 0"* step. The cluttered patterns inside the lung lobe makes segmentation more difficult because there are many strong local minima in the problem and the algorithm must explore all of them to find the global optima.



Algorithm	Time
MaxFlow (BK)	26.47ms
ActiveCut	19.98ms
Hierarchical ActiveCut	Level 2 : 0.45ms Level 1 : 2.14ms Level 0 : 16.95ms Total : 19.54ms

Figure 6.21: Hierarchical segmentation using 4-Neighborhood and directed edges (dark object prior). Optimal segmentation from coarse scale is to initialize finer scale. Speed of the Hierarchical approach is better but deceiving in this case. However the internal structure on the lung lobe makes it difficult to segment and the timing table confirms that the more torn level ("*Level 0*") is main time-consumer.

Chapter 7

Applications to video segmentation

In this chapter, we present an application of Active Cuts for Video segmentation. We briefly compare it to Dynamic Graph Cut [105]. This work we be published as a conference paper at CVPR'06 [87].

Then we present a recent development on Motion Layer Extraction. We propose a new method based on a Graph Cut formulation that allows segmenting objects on a video sequence and tracking them even when they are partially occluded. Indeed this technique provides both visible and hidden layers. This work stems from a join work with Romain Dupont and was submitted at ICPR'06 [62] and is also available as a technical report [63].

7.1 Active Cuts for Video Segmentation

Given that our algorithm can take advantage of a close initial cut, one natural application is that of video segmentation (or dynamic segmentation). When there is enough spatio-temporal consistency between frames of a video, one can assume that the optimal segmentations will be also spatially similar and therefore the cut from the previous frame will make a good initial cut for the current frame. This kind of consistency assumption is widely used in the vision community and was recently applied to graph cuts [105].

Figure Fig:7.1 shows an example of video segmentation for hand tracking. The graph shows that our algorithm is in an average of 5 times faster than the standard maxflow algorithm of [18]. Moreover this graph confirms the observation in Figure 2 that the time of convergence is correlated to the change between two consecutive segmentations.

It is interesting to note that we could also integrate the flow and tree re-

cycling proposed by Kohli and Torr in [105] into our method in order squeeze more performance out of dynamic segmentation.

We were not able to do a complete and careful comparison of [105] since their code was not available at that time. But this should be done in a close future.

7.2 Motion Layers

We consider the extraction of the layers composing a video sequence, each of them being approximated by a planar set of objects having the same parametric motion. This well studied representation (see [6, 10, 11, 13, 64, 97, 110, 129, 160, 172, 173]) offers a good trade-off between low- and high-level of information for numerous applications, such as robust motion segmentation, efficient video compression, 3D reconstruction of urban scenes, etc. The main issues addressed in this context are the estimation of the motion of the layers, the outliers and occlusion detection, the determination of the number of layers, the choice of regularization criteria and the accuracy and robustness of the segmentation.

In [177], Xiao and Shah present a method based on temporal constraints between a frame and its successors ($1 \mapsto 2$, $1 \mapsto 3$, $1 \mapsto 4$, ...) that takes into account what they call occlusions (actually, point modeling two distinct phenomenons: (i) objects becoming hidden and (ii) noisy point with impossible tracking). Their method does not intrinsically give smooth segmentations from one frame to the other as frames are processed independently.

On the contrary, our method takes advantage of temporal information for the whole sequence. Indeed, it simultaneously processes all the sequence considering temporal constraints between successive frames $1 \mapsto 2 \mapsto 3 \mapsto 4 \mapsto ...$, guaranteeing a smooth labeling. Furthermore, it explicitly recovers the hidden parts of the layers, that can disappear behind an another one and re-appear a few frames later: *a disappearing point is not only detected like in [177] but also tracked while being hidden until it re-appears*! Finally, tracking both visible and hidden parts of layers reduces segmentation ambiguities, namely the number of *undefined* points (see further).

Hidden layers. For each pixel, we consider its corresponding visible layer and all hidden layers if any. Given n, the number of layers, we associate each pixel $\mathbf{x}$ with its label $l_{\mathbf{x}} = (v_{\mathbf{x}}, \mathbf{h}_{\mathbf{x}}) \in \mathcal{L}$, with $\mathcal{L} = (\mathcal{V} \times \mathcal{H}) \setminus \mathcal{F}$, where $\mathcal{V} = [1, n] \cup \{\emptyset_{\mathcal{V}}\}$ is the visible space, $\mathcal{H} = \{\mathbf{false}, \mathbf{true}\}^n$ is the hidden one and $\mathcal{F}$ refers to forbidden combinations (see further). The special label $\emptyset_{\mathcal{V}}$ corresponds to an indetermination on the visible layer choice (*undefined pixels* or *"outliers"*). The i^{th} coordinate $\mathbf{h}_{\mathbf{x}}^i$ of vector $\mathbf{h}_{\mathbf{x}}$ indicates the hidden

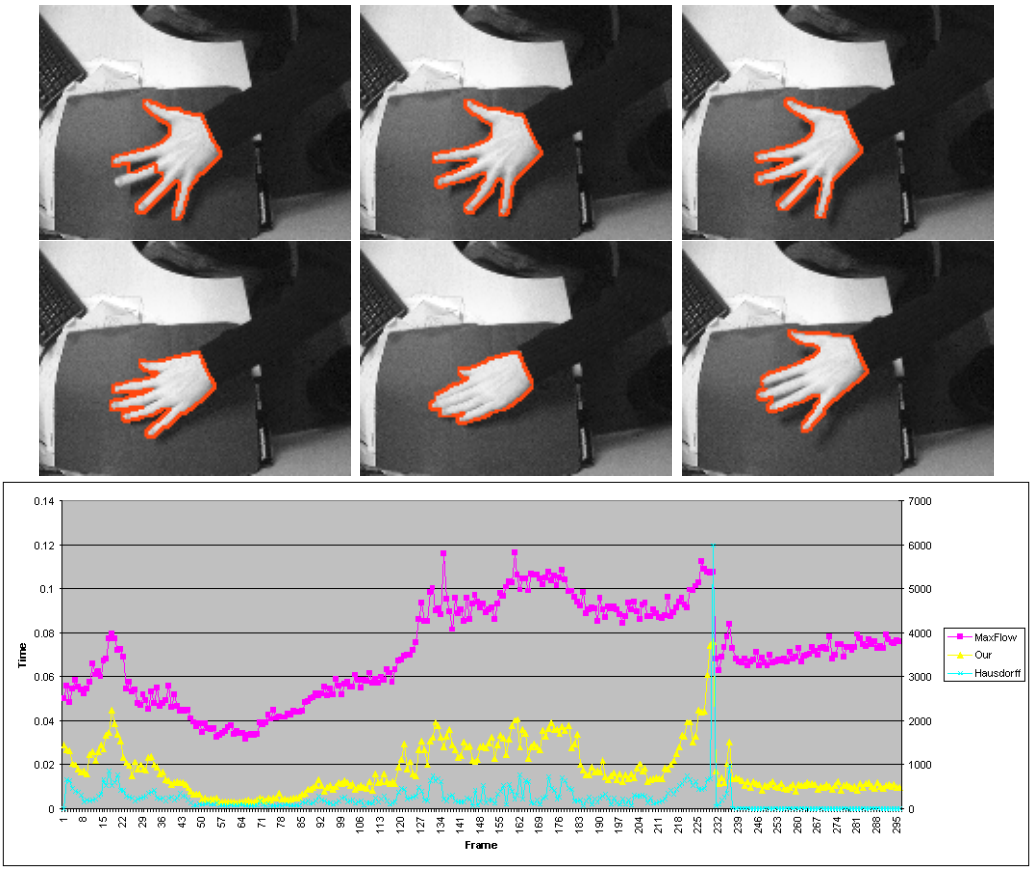


Figure 7.1: Dynamic segmentation of a video sequence. The Active cuts algorithm (yellow) runs 2-6 times faster than the state-of-the-art max-flow algorithm in [18] (red). In each new frame initial cut for our algorithm is set to an optimal cut/segmentation from the previous frame. The speed of our algorithm almost linearly proportional to the magnitude of motion shown by the plot of the Hausdorff distance between the segments in consecutive frames (blue). Note that active cuts can be further accelerated in dynamic applications by "recycling" flow computed in the previous frame [105].

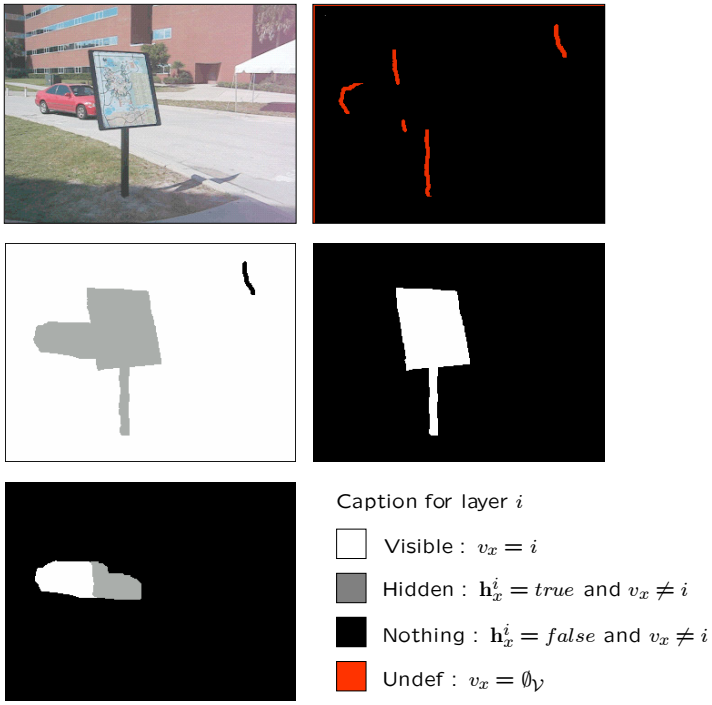


Figure 7.2: Example of labeling. Note that these images **are not the results** obtained by our algorithm but an example of what could be a reasonable segmentation.

state of the i^{th} layer (**true** if hidden, **false** if visible or non present). For a given pixel, a layer cannot be both visible and hidden, i.e. $\mathbf{h}_{\mathbf{x}}^{v_{\mathbf{x}}} \neq$ **true**: $\mathcal{F}$ is the set of such forbidden cases. Figure 7.2 illustrates such a labeling.

The reminder of this paper is organized in the following way. section 2 presents the energy used for classification. section 3 provides some important information about the implementation and shows results on both synthetic and real data. The last section gives some conclusion and future directions.

Motion model. We note $\mathcal{T}_v^t$ the parametric motion of layer v between frames t and $t+1$. No motion is associated to layer $\emptyset_{\mathcal{V}}$. Our experiments use classical projective motions, thus approximates the scene by three-dimensional plane objects, although any other model could be used (e.g. affine). Motion estimation follows our previous work [64] and will not be detailed here, though any other equivalent method could be used.

Initialization. Our method is initialized with n pre-computed layers (accurate or not), obtained through pre-existent methods like the ones in [64, 177]. When the correspondences between the layers of successive frames is not explicitly given by this initial segmentation, we recover it easily, associating

a layer v at time t to the one at time $t+1$ that most overlaps its image through $\mathcal{T}_v^t$.

Overall process. Our method consists in alternating, until some stabilization: (i) layer segmentation and (ii) refinement of the motion parameters from the visible part of the layers (which, again, will not be detailed here).

7.2.1 Classification

Given T frames, n layers, and $\mathcal{T}_v^t (v \in [1,n], t \in [1,T])$ their motion models [1], we consider the labeling problem consisting in determining a function $L : (\mathbf{x}, t) \mapsto l_{\mathbf{x}}^t = (v_{\mathbf{x}}^t, \mathbf{h}_{\mathbf{x}}^t) \in \mathcal{L}$. We plug the problem into a variational framework and will design in the sequel an energy that L should minimize. Note that we consider a constant number of layers throughout the sequence. Such a limitation could be relaxed through appropriate methods.

Motion energy

The motion energy is based on visible parts of the layers and is indeed related to the images ("*data term*"). The *forward* motion residual $r_v(\mathbf{x})$ for the pixel $\mathbf{x}$ under motion $\mathcal{T}_v$ is defined by:

$$r_v^t(\mathbf{x}) = \left\| I^t(\mathbf{x}) - I^{t+1}(\mathcal{T}_v^t(\mathbf{x})) \right\| \tag{7.1}$$

where I^t is the image at time t. To reduce the influence of high motion residuals, we apply a smoothed Heaviside operator ψ (Fig. 7.3) given by:

$$\psi(r_v) = \tan^{-1}\left(r_v^2 - \tau\right) + \pi/2 \tag{7.2}$$

We define a labeling cost function d_I by:

$$d_I(l_{\mathbf{x}}, \mathbf{x}) = \begin{cases} \psi\left(r_{v_{\mathbf{x}}}(\mathbf{x})\right) & \text{if } v_{\mathbf{x}} \in [1,n] \\ \psi_{undef} & \text{if } v_{\mathbf{x}} = \emptyset_V \end{cases} \tag{7.3}$$

where the parameter ψ_{undef} adjusts the classification of pixels as undefined. The *forward* motion energy E_{FM}^t is then, for a given frame t:

$$E_{FM}^t(L) = \int_\Omega d_I(l_{\mathbf{x}}^t, \mathbf{x}) d\mathbf{x} \tag{7.4}$$

where Ω is the image domain. To increase robustness, we also consider the *backward* motion residual (as in [129]) and its associated energy noted $E_{BM}^t(L)$. It is defined similarly, considering frame $t-1$ instead of frame

[1] when explicitly needed, the frame number t will be indicated by a superscript

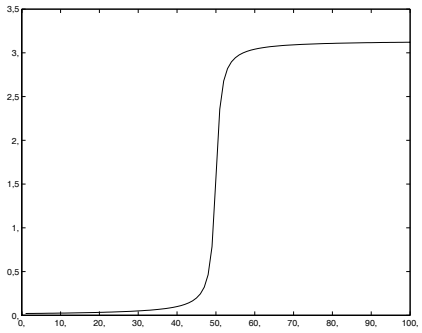


Figure 7.3: Smoothed Heaviside operator ψ shape (with $\tau = 50$).

$t+1$ and the reverse motion $(\mathcal{T}_v^t)^{-1}$ instead of $\mathcal{T}_v^t$. Moreover, we embed this criterion into a temporal-multiscale framework, considering also the motion residuals between frame t and frames $t+1, t+2, t+3, \cdots, t-1, t-2, t-3, \cdots$ to handle small motion cases.

Spatial regularization

As in every noisy and under-constrained problem, spatial regularization has to be introduced. Both visible and hidden parts of the layers are regularized through the following energy:

$$E_S^t(L) = \iint_{\Omega^2} \phi(||\mathbf{x} - \mathbf{y}||) d_S^t \left(l_\mathbf{x}^t, l_\mathbf{y}^t\right) d\mathbf{y} d\mathbf{x} \tag{7.5}$$

where ϕ is some kernel (e.g Gaussian) and $d_S^t(.,.)$ is a dissimilarity measure between two labels. Discontinuous labels for both visible and hidden layers must be penalized. We encourage also the frontier of the layer to belong to pixels with high image gradient. This gives the following function:

$$\begin{aligned} d_S^t(l_\mathbf{x}, l_\mathbf{y}) \; &= \mu_V \mathcal{I}\left(v_\mathbf{x} \neq v_\mathbf{y}\right) \exp\left(-\frac{\|I^t(\mathbf{x}) - I^t(\mathbf{y})\|^2}{2\sigma^2}\right) \\ &+ \mu_H \sum_{i=1}^{n} \mathcal{I}\left(\mathbf{h}_\mathbf{x}^i \neq \mathbf{h}_\mathbf{y}^i\right) \end{aligned} \tag{7.6}$$

where $\mathcal{I}(i)$ equals 1 if i is true, 0 otherwise, σ is the standard deviation of the norm of the gradient of the images, and (μ_V, μ_H) some constants adjusting spatial regularization with respect to the other energy terms.

Temporal constraints

Temporal constraints are designed for both temporal smoothness and temporal consistency between visible and hidden layers. To this end, using motion

information, we penalize discontinuous labeling between frames. To simplify notations, we note $\mathbf{x}_i = \mathcal{T}_i^t(\mathbf{x})$ the image of $\mathbf{x}$ in frame $t+1$ through the motion of layer i at time t. Our *forward* temporal energy is written as follows:

$$E_{FT}^t(L) = \int_\Omega \Big[\mathcal{I}(v_\mathbf{x} \neq \emptyset_\mathcal{V}) d_V\left(l_\mathbf{x}^t, l_{\mathbf{x}_{v_\mathbf{x}}}^{t+1}\right) + \sum_{i=1}^n \mathcal{I}(\mathbf{h}_\mathbf{x}^i = \mathbf{true}) d_H^i\left(l_\mathbf{x}^t, l_{\mathbf{x}_i}^{t+1}\right)\Big] d\mathbf{x} \quad (7.7)$$

where $d_V(.,.)$ and $d_H^i(.,.)$ are dissimilarity measures given by:

$$d_V(l_\mathbf{x}, l_\mathbf{y}) = \begin{cases} 0 & \text{if } v_\mathbf{x} = v_\mathbf{y} \\ \lambda_H & \text{if } \mathbf{h}_\mathbf{y}^{v_\mathbf{x}} = \mathbf{true} \\ \lambda_D & \text{otherwise} \end{cases} \quad (7.8)$$

and:

$$d_H^i(l_\mathbf{x}, l_\mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{h}_\mathbf{y}^i = \mathbf{h}_\mathbf{x}^i \\ \lambda_V & \text{if } v_\mathbf{y} = i \\ \lambda_D & \text{otherwise} \end{cases} \quad (7.9)$$

where λ_H, λ_V and λ_D respectively penalize the following events: hiding, reappearing, and completely disappearing. It can be shown that λ_D has to be chosen greater than λ_V and λ_H (see section Sec7.2.2) and that the following inequality $\lambda_H + \lambda_V \leqslant \lambda_D$ must be respected.

As in the data term, we also consider *backward* constraints, leading to a symmetric temporal energy E_{BT}^t. Moreover, similarly as for the motion residual, we also embed these temporal constraints into a temporal multi-scale framework to increase robustness (especially in cases of slow motions) considering also constraints between frame t and frames $t+1, t+2, t+3, t-1, t-2, t-3, \cdots$ and so on.

Overall energy

Our overall energy to extract the optimal partition of the T images is finally:

$$E(L) = \sum_{t=1}^T \underbrace{E_{FM}^t(L) + E_{BM}^t(L)}_{\text{data term (motion)}} + \underbrace{E_S^t(L)}_{\text{spatial regularization}} + \underbrace{E_{FT}^t(L) + E_{BT}^t(L)}_{\text{temporal constraints}} \quad (7.10)$$

Next section will describe the optimization process used to minimize this global energy.

7.2.2 Energy minimization

We plug our spatially continuous energy minimization problem into a discrete Markov Random Field framework [73]. The global energy (EQ. 7.10) is discretized considering a 4- or 8- neighborhood for the spatial constraints. Due to its efficiency, we use the *alpha*-expansion algorithm [23, 108] provided that distance function d_S is sub-modular (easy to verify) and that temporal constraints fit also submodularity requirement.

About the submodularities of the temporal constraints

First, we remind what a submodular function is.

Definition 1 *A sub-modular function $D(.,.)$ verifies $D(l_x, l_y) + D(l_\alpha, l_\alpha) \leqslant D(l_x, l_\alpha) + D(l_\alpha, l_y)$ for two given pixels $\mathbf{x}$ and $\mathbf{y}$ (see [108] for more details).*

To demonstrate that the temporal constraints fit the submodularity requirement, we introduce these two following functions V and H (which depend on d_V and d_H) :

$$V_{\mathbf{x},\mathbf{y}}(l_\mathbf{x}, l_\mathbf{y}) = \mathcal{I}\left(\mathbf{y} = T_{v_\mathbf{x}}(\mathbf{x}) \wedge v_\mathbf{x} \neq \emptyset_\mathcal{V}\right) \cdot d_V(l_\mathbf{x}, l_\mathbf{y}) \quad (7.11)$$

$$H^i_{\mathbf{x},\mathbf{y}}(l_\mathbf{x}, l_\mathbf{y}) = \mathcal{I}\left(\mathbf{y} = \mathcal{T}_i(\mathbf{x}) \wedge \mathbf{h}^i_\mathbf{x} = \mathbf{true}\right) \cdot d^i_H(l_\mathbf{x}, l_\mathbf{y}) \quad (7.12)$$

Theorem 1 *The function $(V + \sum_i H^i)$ is submodular if λ_D is greater than λ_V and λ_H.*

Proof :

Summary of the proof: we will show that functions D and H^i are submodular providing $\lambda_V = \lambda_H = \lambda_D$. However, considering some particular cases, we will also show that the function $(D + \sum_i H^i)$ is submodular providing $\lambda_V \leqslant \lambda_D$ and $\lambda_H \leqslant \lambda_D$. For the other cases, we use the fact that the sum of two submodular functions is submodular.

First, we consider the function $D()$: the table 7.1 shows all the cases which give information about the constraints between λ_H and λ_D. Cases V5 and V8 are impossible as a change of visible labeling to v_α implies a change of projected pixel $\mathcal{T}_{v_\alpha}$ to consider: as a consequence, the requirement $\mathbf{y} = \mathcal{T}_{v_\alpha}$ will not then be satisfied anymore except if $\alpha = v_\mathbf{x}$ [2]. Valid cases V3 and V6 show that the following equality $\lambda_D = \lambda_H$ must be respected. And similarly for H^i as shown in table 7.2 : valid cases H3 and H6 constrain the following equality $\lambda_D = \lambda_V$.

[2] We consider here that all motion models have different parameters.

case	$V(l_{\mathbf{x}}, l_{\mathbf{y}}) \leqslant V(l_{\mathbf{x}}, l_\alpha) + V(l_\alpha, l_{\mathbf{y}})$	obtained if	state
V1	$\lambda_H \leqslant 0 + 0$	$\{v_{\mathbf{x}} \neq v_{\mathbf{y}}\} \wedge \{v_{\mathbf{x}} = v_\alpha = v_{\mathbf{y}}\}$	$\Rightarrow$ impossible
V2	$\lambda_D \leqslant 0 + 0$	equiv. to previous case	$\Rightarrow$ impossible
V3	$\lambda_D \leqslant \lambda_H + 0$	$\{v_{\mathbf{x}} \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_{\mathbf{x}}} = \mathbf{false}\}$ $\wedge\{v_{\mathbf{x}} \neq v_\alpha \wedge \mathbf{h}_\alpha^{v_{\mathbf{x}}} = \mathbf{true}\}$ $\wedge\{v_\alpha = v_{\mathbf{y}}\}$	$\Rightarrow$ possible !
V4	$\lambda_D \leqslant 0 + \lambda_H$	$\{v_{\mathbf{x}} \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_{\mathbf{x}}} = \mathbf{false}\}$ $\wedge\{v_{\mathbf{x}} = v_\alpha\}$ $\wedge\{v_\alpha \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_\alpha} = \mathbf{true}\}$	$\Rightarrow$ impossible
V5	$\lambda_D \leqslant \lambda_H + \lambda_H$	$\{v_{\mathbf{x}} \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_{\mathbf{x}}} = \mathbf{false}\}$ $\wedge\{v_{\mathbf{x}} \neq v_\alpha \wedge \mathbf{h}_\alpha^{v_{\mathbf{x}}} = \mathbf{true}\}$ $\wedge\{v_\alpha \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_\alpha} = \mathbf{true}\}$ $\wedge\{v_\alpha = v_{\mathbf{x}}\}$	$\Rightarrow$ impossible
V6	$\lambda_H \leqslant \lambda_D + 0$	$\{v_{\mathbf{x}} \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_{\mathbf{x}}} = \mathbf{true}\}$ $\wedge\{v_{\mathbf{x}} \neq v_\alpha \wedge \mathbf{h}_\alpha^{v_{\mathbf{x}}} = \mathbf{false}\}$ $\wedge\{v_\alpha = v_{\mathbf{y}}\}$	$\Rightarrow$ possible !
V7	$\lambda_H \leqslant 0 + \lambda_D$	$\{v_{\mathbf{x}} \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_{\mathbf{x}}} = \mathbf{true}\}$ $\wedge\{v_{\mathbf{x}} = v_\alpha\}$ $\wedge\{v_\alpha \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_\alpha} = \mathbf{false}\}$	$\Rightarrow$ impossible
V8	$\lambda_H \leqslant \lambda_D + \lambda_D$	$\{v_{\mathbf{x}} \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_{\mathbf{x}}} = \mathbf{true}\}$ $\wedge\{v_{\mathbf{x}} \neq v_\alpha \wedge \mathbf{h}_\alpha^{v_{\mathbf{x}}} = \mathbf{false}\}$ $\wedge\{v_\alpha \neq v_{\mathbf{y}} \wedge \mathbf{h}_{\mathbf{y}}^{v_\alpha} = \mathbf{false}\}$ $\wedge\{v_\alpha = v_{\mathbf{x}}\}$	$\Rightarrow$ impossible

Table 7.1: Cases considered for the submodularity of $D()$.

case	$H^i(x,y) \leqslant H^i(x,\alpha) + H^i(\alpha,y)$	obtained if,	state
H1	$\lambda_V \leqslant 0+0$	$\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\mathbf{y}}\} \wedge \{\mathbf{h}^i_{\mathbf{x}} = \mathbf{h}^i_{\alpha} = \mathbf{h}^i_{\mathbf{y}}\}$	$\Rightarrow$ impossible
H2	$\lambda_D \leqslant 0+0$	equiv. to previous case	$\Rightarrow$ impossible
H3	$\lambda_D \leqslant \lambda_V + 0$	$\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} \neq i\}$ $\wedge\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\alpha} \wedge v_{\alpha} = i\}$ $\wedge\{\mathbf{h}^i_{\alpha} = \mathbf{h}^i_{\mathbf{y}}\}$	$\Rightarrow$ possible !
H4	$\lambda_D \leqslant 0 + \lambda_V$	$\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} \neq i\}$ $\wedge\{\mathbf{h}^i_{\mathbf{x}} = \mathbf{h}^i_{\alpha}\}$ $\wedge\{\mathbf{h}^i_{\alpha} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} = i\}$	$\Rightarrow$ impossible
H5	$\lambda_D \leqslant \lambda_V + \lambda_V$	$\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} \neq i\}$ $\wedge\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\alpha} \wedge v_{\alpha} = i\}$ $\wedge\{\mathbf{h}^i_{\alpha} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} = i\}$ $\wedge\{\mathbf{h}^i_{\alpha} = \mathbf{h}^i_{\mathbf{x}}\}$	$\Rightarrow$ impossible
H6	$\lambda_V \leqslant \lambda_D + 0$	$\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} = i\}$ $\wedge\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\alpha} \wedge v_{\alpha} \neq i\}$ $\wedge\{\mathbf{h}^i_{\alpha} = \mathbf{h}^i_{\mathbf{y}}\}$	$\Rightarrow$ possible !
H7	$\lambda_V \leqslant 0 + \lambda_D$	$\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} = i\}$ $\wedge\{\mathbf{h}^i_{\mathbf{x}} = \mathbf{h}^i_{\alpha}\}$ $\wedge\{\mathbf{h}^i_{\alpha} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} \neq i\}$	$\Rightarrow$ impossible
H8	$\lambda_V \leqslant \lambda_D + \lambda_D$	$\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} = i\}$ $\wedge\{\mathbf{h}^i_{\mathbf{x}} \neq \mathbf{h}^i_{\alpha} \wedge v_{\alpha} \neq i\}$ $\wedge\{\mathbf{h}^i_{\alpha} \neq \mathbf{h}^i_{\mathbf{y}} \wedge v_{\mathbf{y}} \neq i\}$ $\wedge\{\mathbf{h}^i_{\alpha} = \mathbf{h}^i_{\mathbf{x}}\}$	$\Rightarrow$ impossible

Table 7.2: Cases considered for the submodularity of $H()$.

However, for the cases V3 and H3 (which force λ_H and λ_V to be greater than λ_D), one can see that the function $(D + \sum_i H^i)$ is actually submodular without any constraints on λ_H, λ_V and λ_D (as shown in figure 7.4).

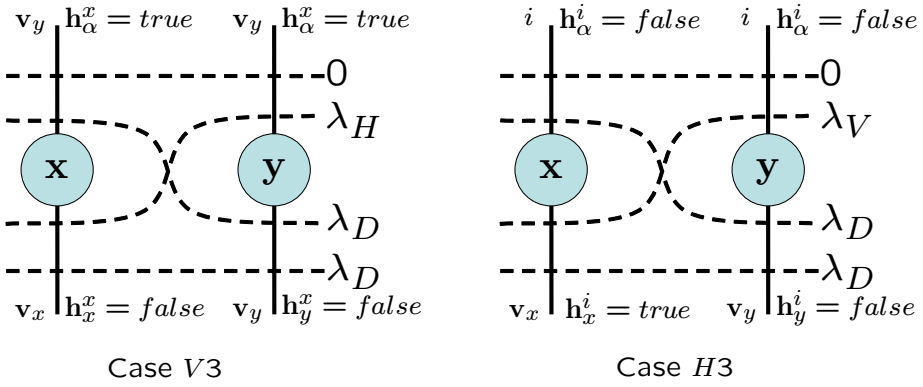


Figure 7.4: Cases V3 and H3 (with resp. $\mathbf{y} = \mathcal{T}_{v_\mathbf{x}}(\mathbf{x})$ and $\mathbf{y} = \mathcal{T}_i(\mathbf{x})$) . Both cases are *graph-representable* as the inequalities $\lambda_D \leqslant \lambda_D + \lambda_H$ for case V3 and $\lambda_D \leqslant \lambda_D + \lambda_V$ for case H3 are respected $\forall \lambda_D, \lambda_H$ and $\lambda_V \geqslant 0$ (see tables 7.1 and 7.2 for details) .

For the other valid cases, $D()$ and $H^i()$ (and so $D + \sum_i H^i$) are submodular providing that $\lambda_V \leqslant \lambda_D$ and $\lambda_H \leqslant \lambda_D$.

□

Furthermore, one can see that the following inequality $\lambda_H + \lambda_V \leqslant \lambda_D$ must be respected if we want hidden parts of the layers to be recovered. Indeed, if not, the cost of a disparition to a hidden layer (cost : λ_H) followed by an apparition to a visible layer (cost : λ_V) would be coster than a disparition to '*nothing*', i.e. to any hidden layer, which would only cost λ_D (indeed, in such a case, there is no apparition constraint, so no apparition cost).

Minimization process

Even then, labeling cannot be achieved in reasonable time using a straightforward *alpha*-expansion since the number of possible labels $(v, \mathbf{h})$ increases dramatically with the number of layers: $(n + 2)2^{n-1}$ possible expansions! However the problem could be circumvented limiting *alpha*-expansions to a sub-space of $\mathcal{L}$.

First minimization method:

One can consider only a change of the visible layer and one hidden layer, i.e. $(v, \mathbf{h}^i)$-expansions for successive choices of i. Using this approach, we

reduce the number of optimization steps to $2n^2$: for each visible layer j (so n iterations), we process $2n$ $(v_j, \mathbf{h}^i)$-expansions, testing in same time if the j-th layer is visible and if the i-th layer is hidden or not (with $i \neq j$).

However, some labelings are impossible to obtain. Consider the following example (figure 7.5): the optimal solution should be $v_{\mathbf{x}} = 0, v_{\mathbf{x}'} = 1, v_{\mathbf{x}'} = 2$ and $\mathbf{h}^0_{\mathbf{x}'} = \mathbf{h}^0_{\mathbf{x}''} = \mathbf{true}$ (all other hidden layers set to $false$). If we consider initial labelings such as $v_{\mathbf{x}} = 0, v_{\mathbf{x}'} = 1, v_{\mathbf{x}'} = 2$ but $\mathbf{h}^0_{\mathbf{x}'} = \mathbf{h}^0_{\mathbf{x}''} = \mathbf{false}$, there is any $(v_j, \mathbf{h}^0)$-expansion which could give the optimal solution. Indeed, neither the $(v_1, \mathbf{h}^0 = \mathbf{true})$-expansion, nor the $(v_2, \mathbf{h}^0 = \mathbf{true})$-expansion could change the labels of $\mathbf{x}'$ and $\mathbf{x}''$. Note that such a limitation is also encountered even if we change not only one hidden layer but also all the other ones at same time.

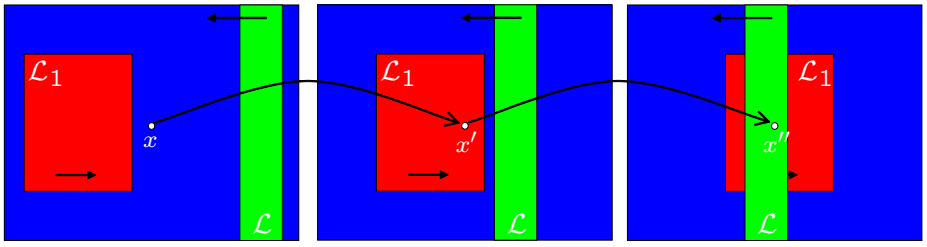


Figure 7.5: Example of sequence where optimal solution could not be obtained through $(v_j, \mathbf{h}^i)$-expansions. Here, there are three layers $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2$ (resp. in blue, red and green), the white pixel $\mathbf{x}$ and the projected ones $\mathbf{x}' = \mathcal{T}_0(\mathbf{x})$ and $\mathbf{x}'' = \mathcal{T}_0(\mathbf{x}')$.

Only a change of hidden labeling without modifying any visible labeling could handle such a case. Hence, we propose a second minimization process to solve the problem.

Second minimization method:

One can consider alternatively

- only a change of the visible layer without modifying the hidden layer states (except for the corresponding hidden layer $\mathbf{h}^v$ which is set to $\mathbf{false}$) $\Rightarrow (v, \mathbf{h}^v = \mathbf{false})$-expansions *for successive choices of* v.
- and only a change of one hidden layer, without modifying visible layer $\Rightarrow (\mathbf{h}^i = \mathbf{false}/\mathbf{true})$-expansions *for successive choices of* i.

Using this approach, we reduce the number of optimization steps to $3n$: we process each visible layer (so n iterations) and $2n$ $(\mathbf{h}^i)$-expansions, testing if

the i-th layer is hidden or not. This yields in practice to acceptable minimization times, without modifying noticeably the segmentation.

But such an approach has also some drawbacks: some labelings can be unreachable. For example, if a pixel $\mathbf{x}$ is currently labeled as $(v_{\mathbf{x}} = 1, \mathbf{h}^0_{\mathbf{x}} = \mathbf{h}^1_{\mathbf{x}} = \mathbf{false})$ and if the optimal label is $(v_{\mathbf{x}} = 0, \mathbf{h}^1_{\mathbf{x}} = \mathbf{true})$, it is not yet guaranteed that a $(v = 0)$-expansion will decrease the overall energy, changing the label of $\mathbf{x}$ to $(v_{\mathbf{x}} = 0, \mathbf{h}^1_{\mathbf{x}} = \mathbf{false})$.

The corresponding graph is a three-dimensional one, the third dimension being time. The data and spatial regularization terms of the energy are standard in the graph-cut framework. During a $v-$ or $\mathbf{h}^i-$expansion, the *backward* and *forward* spatial constraints yield links between each pixel $\mathbf{x}$ at time t and $2(2 + n)$ other pixels: $\mathbf{x}_v$ or $\mathbf{x}_{\mathbf{h}}$, $\mathbf{x}_{v_{\mathbf{x}}}$ and $\mathbf{x}_{\mathbf{h}^i_{\mathbf{x}}}$ $(i \in [1, n])$ at time $t + 1$ and similarly at time $t - 1$ (see figure 7.6).

7.2.3 Results

Synthetic sequence

Figure 7.7 shows the results obtained on a synthetic sequence ($n = 3$). Throughout the sequence, the proportion of misclassified visible pixels is 0.06% and the proportion of pixels where the complete label l (visible *and* hidden parts) is incorrect is also 0.06%: for each pixel, classification fails or succeeds globally. Note that in this particular sequence, no pixel is classified as undefined. Indeed, only noise or aliasing could generate such pixels. Because hidden parts are modelized, the undefined label do not account anymore for points that become hidden like in [177].

Real sequences

As a first step[3] toward comparing our results to state of the art method like [64, 177], we show the results obtained for a real sequence (fig. 7.8). One can see that the segmentation of the visible layers is comparable to the usually obtained segmentation. Note that the wheels of the car are sometime classified as undefined because the number n of layers is fixed too small (the wheels have their own motion). A splitting/merging approach could be used to choose n dynamically. We are in the process of implementing this.

Moreover, our goal was to extract the hidden parts of the layers and this is correctly done. Continuous labeling between frames is obtained, providing non-disrupted segmentation throughout the sequences. Again, note that the number of undefined pixels is rather small: unlike in [177] where these pixels

[3]No ground truth is provided here!

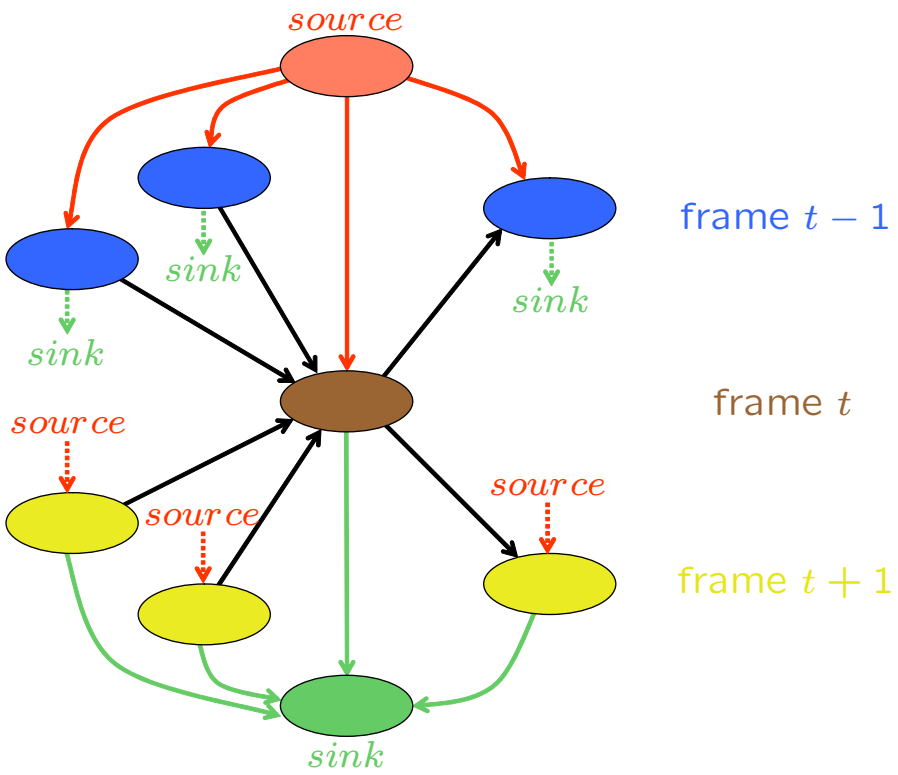


Figure 7.6: Graph construction: Source t-links are shown in red, sink t-links in green. Considering a $(V, \mathbf{h}^V = \textbf{false})$-expansion (or $\mathbf{h}^H$-expansion), temporal n-links are shown in black and link the pixel $\mathbf{x}$ (frame t) to pixels $\mathbf{x}_{v_\mathbf{x}}, \mathbf{x}_{\mathbf{h}^i_\mathbf{x}}, \mathbf{x}_V$ (or $\mathbf{x}_H$) of frames $t-1$ and $t+1$. *Note*: for clarity, only the links relative to the i-th hidden layer are shown.

code also for points that are going to be hidden, in our method $v_\mathbf{x} = \emptyset_V$ only stands for a lack of image information (e.g. too much noise).

7.2.4 Conclusion and discussion

We have presented a novel global optimization process for motion layer segmentation in a video sequence. Considering the hidden parts of the layers, we achieve a continuous labeling, even is case of occlusion: when hidden, a point is tracked until reappearance. Ongoing work includes dealing with (i) processing longer sequences through shifting windows, (ii) more robustness thanks to multi-scale analysis in time and (iii) coping with a robust determination of a variable number of layers.

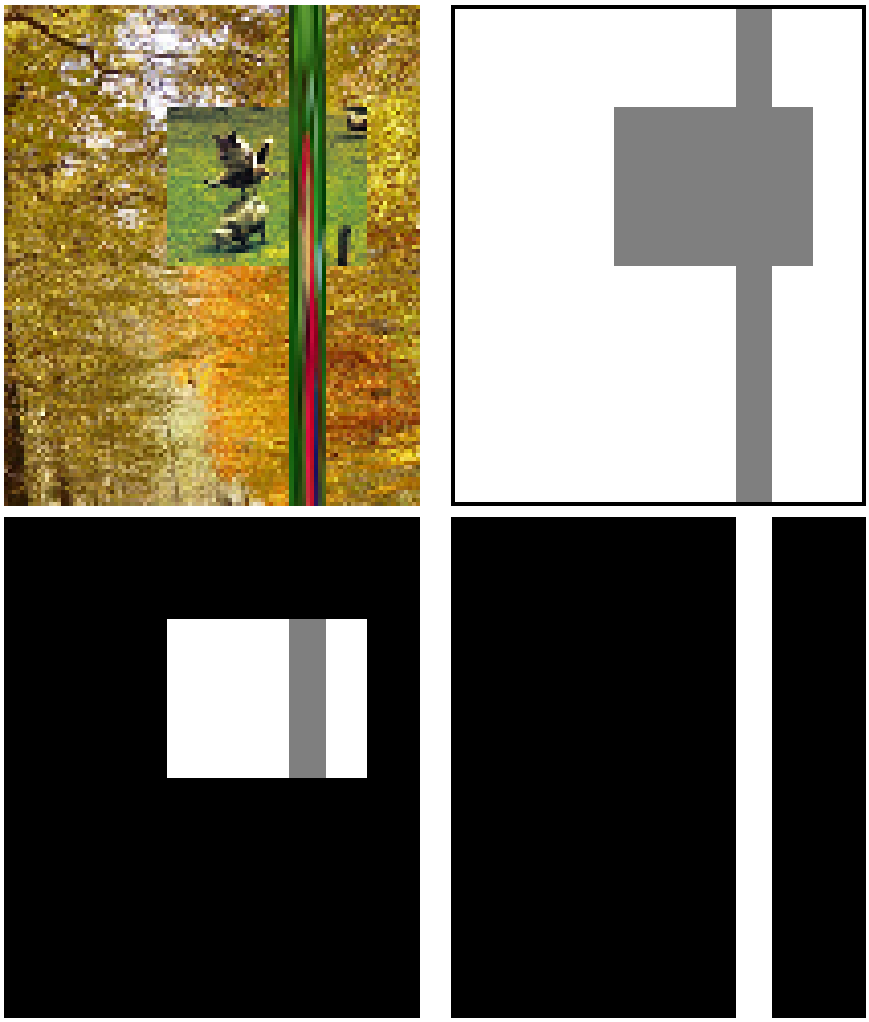


Figure 7.7: A synthetic sequence. From top to bottom, left to right: original sequence, layers 1, 2 and 3 (white=visible, grey=hidden) (Note: on this particular image of the sequence, no pixel is classified as undefined)

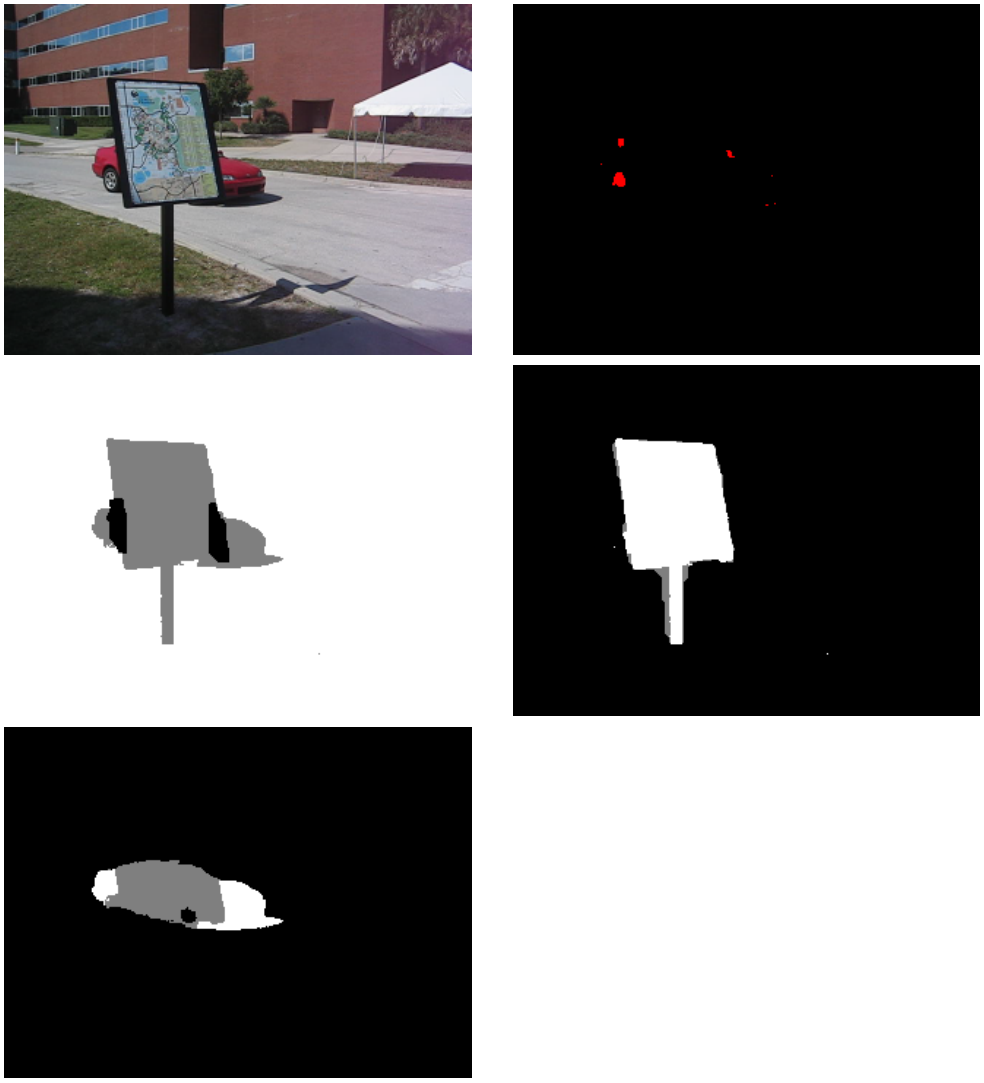


Figure 7.8: Carmap sequence. From top to bottom, left to right: original sequence, undefined pixels (in red), layers 1, 2 and 3 (white=visible, grey=hidden).

Conclusion

Born of a join work with Gheorghes Postelnicu, we presented the Stochastic Level Sets, a general framework to introduce random perturbations in the Level Set method. This could be applied to a minimization process that performs better minimization by avoiding local minima: the Stochastic Active Contours. Moreover, it also provides a minimizing technique in Level Set formulation for energies with inaccessible or intractable gradient.

We successfully apply it to unsupervised image segmentation based on region information. Even if no theoretic result about global optimum convergence is provided, in practice this method greatly improves the results.

In the future, we would be interested into bring some theoretic results about the convergence of the Stochastic Active Contours. Since the Stochastic Active Contours are only one application of the Stochastic Level Sets (SLS), providing new applications of SLS to Computer Vision problems is a promising research domain.

We also provide a new user-friendly segmentation technique for Digital Matting. Indeed Digital Matting's first application is special effects for movies. Having an easy and almost automatic tool is necessary since special effects are omnipresent in movies nowadays. Most recent Digital Matting methods need to be initialized by a Trimap segmentation. By automatizing this process or in that case, making it as easy as possible, one could reduce sensitively post-production time. Our segmenting method proved to be robust, fast and easy to use.

Graph Cut clearly shows here its superiority in comparison to Level Set. However, one should note that Level Set can minimize a wider class of energies than Graph Cut.

We would like to bring those results to video in order to produce more applicable result for post-production.

We then proposed a new max-flow/min-cut algorithm based on a symmetric formulation: the Active Cuts. This led us to a more efficient algorithm with interesting properties especially for Computer Vision community: Initialization by a cut and providing a succession of intermediate decreasing

cuts during optimization.

We provided some applications of Active Cuts for segmentation. In particular, Active Cuts showed to be applicable to hierarchical segmentation without loosing global optimality. It also provided great performance for video segmentation. Not all its possible applications and properties have been prospected yet and it should be studied more (for example usefulness of intermediate cuts).

It would be good to continue the Dynamic Tree exploration for Active Cuts since it was a critical source of performance for our algorithm. Hochbaum [79] and Goldberg and Tarjan [75, 76] use more complex Dynamic Trees and are able to reduce the algorithm complexity.

It would also be nice to spend some time on a complexity evaluation of the Active Cuts algorithm. Indeed complexity of the state of the art algorithm for Computer Vision community [18] is not even polynomial. It may be possible to obtain a polynomial complexity for the Active Cuts algorithm since it is very similar to the Push-Relabel algorithm. However, the key for this problem might be hidden in a different Dynamic Tree structure.

Then it would be greatly appreciate to make a careful comparison with other techniques such as Level Set. Indeed a lot of Level Set users do not clearly understand and often underestimate Graph Cut techniques. On video segmentation, one should carefully compare the Active Cuts algorithm with Dynamic Cuts [105]. One can also imagine merging them in order to obtain an Active Dynamic Cuts with even higher performance on video segmentation.

We then proposed a robust layer extraction method based on a Graph Cut formulation that provide both Visible and Hidden layers in the same time by tracking then along the video sequence. This leads directly to a future work: integrating Digital Matting in this application like in [176]. Another direction would be to incorporate colors and texture to the object tracking in order to be more robust to motion estimation.

Conclusion (version française)

Issu d'une collaboration avec Gheorghes Postelnicu, nous avons présenté les "Ensembles de Niveaux Stochastiques", un cadre générique qui permet d'introduire élégamment le concept de perturbations aléatoires à la méthode dites des "Ensembles de Niveaux". Cette nouvelle approche a été appliquée avec succès au problème des "Contours Actifs". Les ensembles de niveaux stochastiques permettent l'utilisation de méthodes de minimisation plus robustes telles que le recuit simulé. En les associant, nous avons obtenu une méthode de contours actifs plus robuste et moins sensible aux minima locaux : les "Contours Actifs Stochastiques". De plus, les contours actifs stochastiques ne nécessitent qu'une estimation du gradient de l'énergie à minimiser : ce qui permet son utilisation même lorsque le gradient ne peut pas être calculé de manière exacte car trop complexe ou inaccessible.

Nous avons par la suite appliqué avec succès les contours actifs stochastiques à la segmentation région non supervisé d'images. Même si aucun résultat théorique concernant une convergence vers un minimum global n'est fourni, on peut observer qu'en pratique cette méthode améliore de manière significative les résultats.

Deux directions pour compléter ces recherches peuvent être entreprises. Tout d'abord, d'un point de vu théorique, obtenir des résultats de convergence vers un optimum global. Ensuite le cadre des ensembles de niveaux stochastiques est suffisamment vaste pour laisser espérer d'autres applications que les contours actifs stochastiques.

Nous avons aussi décrit une nouvelle technique rapide de "Digital Matting" simple d'utilisation. En effet sa première et principale application concerne les effets spéciaux de films. Les effets spéciaux sont omniprésents dans les productions cinématographiques contemporaines et l'utilisation d'outils simple et quasi-automatique est devenue nécessaire afin de réduire les temps de post-traitement. La plus part des méthodes existantes font appel à une initialisation par une "Trimap" ou segmentation trois régions. En automatisant la génération de cette trimap - ou en la simplifiant autant que possible - il est possible de réduire significativement le temps et le coût de post-production.

Notre algorithme est robuste, rapide et facile d'utilisation.

Les "Coupes de Graphe" ont ici montré leur supériorité par rapport aux méthodes des ensembles de niveaux bien que ces dernières permettent de minimiser une plus grande classe d'énergies que les coupes de graphe.

Nous aurions voulu appliquer ca la vidéo pour fournir un outil plus apte à être utiliser en post-production. Cela reste à faire.

Nous avons exposé un nouvel algorithme de flot maximal/coupe minimale basé sur une formulation symétrique : les "Coupes de Graphe Actives". Cet algorithme est plus efficace et fourni de nouvelles propriétés particulièrement intéressantes pour la communauté de Vision par Ordinateur : initialisation par une coupe et retourne une succession de coupes intermédiaires de coût décroissant au cours de l'optimisation.

Nous avons exploré quelques applications des coupes de graphe actives pour la segmentation. Cet algorithme convient tout particulièrement aux approches multi-résolutions et cela sans perdre l'optimum global. Il permet aussi d'obtenir des performances très intéressantes pour la segmentation vidéo. Si toutes les applications possibles n'ont pas encore été explorées, il apparaît clairement que les coupes de graphe actives doivent encore être étudiées : par exemple, quelle utilité pour les coupes successives ?

Un aspect des coupes de graphes actives qui mérite une attention particulière concerne le choix de la structure d'arbre dynamique utilisé dans notre implémentation. En effet, Hochbaum [79], ainsi que Goldberg et Tarjan [75, 76] utilisent des arbres dynamiques plus complexes qui permettent de réduire la complexité de leur algorithme.

De plus, l'évaluation de la complexité de l'algorithme des coupes de graphe actives n'a pas été réalisée. Cependant les similarités avec l'algorithme dit de "Push-Relabel" nous permettent d'espérer une complexité polynomiale (peut-être au prix de quelques petits changements dans l'algorithme). Ce problème apparaît être clairement relié au précédent.

Une comparaison attentive avec d'autres techniques comme celle des ensembles de niveaux, peut s'avérer très utile. En effet beaucoup d'utilisateurs de la méthode des ensembles de niveaux n'entrevoient pas toujours le potentiel et sous-estiment souvent les coupes de graphe. Cependant ces deux techniques de minimisation d'énergies n'ont pas les mêmes domaines d'applicabilité et fournissent des résultats très différents. Le choix de l'une de ces techniques par rapport à une autre doit se faire en connaissance de cause et en fonction de l'application souhaitée.

Concernant la segmentation vidéo, une comparaison des coupes de graphe actives avec l'algorithme des "Coupes de Graphe Dynamiques" [105] doit être effectué. On peut aussi très bien imaginer de les associer pour obtenir un algorithme de "Coupes de graph Actives et Dynamiques" avec de plus

grande performance encore pour la segmentation vidéo.

Nous avons aussi proposé une méthode robuste d'estimation de couches basé sur une formulation par coupes de graphe qui permet d'extraire les couches visibles ainsi que les couches cachées en même temps en les traquant au cours de la séquence vidéo. On peut clairement imaginer intégrer un algorithme de "Digital Matting" dans cet application à la manière de Xiao et Shah [176]. Une autre direction pourrait être d'incorporer l'information de couleur et de texture pour amélioré le suivi d'objets et obtenir une estimation du mouvement plus robuste.

Bibliography

[1] Adalsteinsson, D., and Sethian, J. A fast level set method for propagating interfaces. *Journal Of Computational Physics 120* (1995), 269–277.

[2] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[3] Ahuja, R. K., Orlin, J. B., and Tarjan, R. E. Improved time bounds for the maximum flow problem. *SIAM J. Comput. 18*, 5 (1989), 939–954.

[4] Amini, A. A., Weymouth, T. E., and Jain, R. C. Using dynamic programming for solving variational problems in vision. *IEEE Transaction on Pattern Analysis and Machine Intelligence 12*, 9 (1990), 855–867.

[5] Anderson, R. J., and Setubal, J. C. Goldberg's algorithm for the maximum flow in perspective: a computational study. In *Network Flows and Matching: First DIMACS Implementation Challenge* (1993), D. S. Johnson and C. C. McGeoch, Eds., AMS, pp. 1–18.

[6] Ayer, S., and Sawhney, H. S. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In *ICCV* (1995), p. 777.

[7] Barles, G., and Souganidis, P. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic Analysis 4* (1991), 271–283.

[8] Besag, J. Spatial interaction and the statistical analysis of lattice systems. *Journal of Royal Statistical Society 2* (1974), 192–236.

[9] Bezdek, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981.

[10] Black, M. J., and Anandan, P. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Comput. Vis. Image Underst. 63*, 1 (1996), 75–104.

[11] Black, M. J., and Jepson, A. D. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Trans. Pattern Anal. Mach. Intell. 18*, 10 (1996), 972–986.

[12] Blake, A., Rother, C., Brown, M., Perez, P., and Torr, P. Interactive image segmentation using an adaptive gmmrf model. In *ECCV* (2004).

[13] Bouthemy, P., and Francois, E. Motion segmentation and qualitative dynamic scene analysis from an image sequence. *Int. J. Comput. Vision 10*, 2 (1993), 157–182.

[14] Boykov, Y., and Jolly, M.-P. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV* (2001), pp. 105–112.

[15] Boykov, Y., and Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *Energy Minimization Methods in Computer Vision and Pattern Recognition* (2001), pp. 359–374.

[16] Boykov, Y., and Kolmogorov, V. Computing geodesics and minimal surfaces via graph cuts. In *Proceedings of the Ninth IEEE International Conference on Computer Vision* (Washington, DC, USA, 2003), vol. 1, IEEE Computer Society, pp. 26–33.

[17] Boykov, Y., and Kolmogorov, V. Computing geodesics and minimal surfaces via graph cuts. In *ICCV* (2003).

[18] Boykov, Y., and Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 9 (2004), 1124–1137.

[19] Boykov, Y., Veksler, O., and Zabih, R. Markov random fields with efficient approximations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 1998), IEEE Computer Society, pp. 648–655.

[20] Boykov, Y., Veksler, O., and Zabih, R. A new algorithm for energy minimization with discontinuities. In *Proceedings of Energy Minimization Methods in Computer Vision and Pattern Recognition* (1999), pp. 205–220.

[21] Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Transaction on Pattern Analysis and Machine Intelligence 23*, 11 (2001), 1222–1239.

[22] Boykov, Y., Veksler, O., and Zabih, R. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence 23* (2001), 1222–1239.

[23] Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (2001), 1222–1239.

[24] Brox, T., Rousson, M., Deriche, R., and Weickert, J. Unsupervised segmentation incorporating colour, texture, and motion. In *10th International Computer Analysis of Images and Patterns* (Aug. 2003), no. 2756 in Lecture Notes on Computer Science., Springer Verlag, pp. 353–360.

[25] Buckdahn, R., and Ma, J. Stochastic viscosity solutions for nonlinear stochastic partial differential equations. *Stochastic Processes and their Applications 93* (2001), 181–228.

[26] Caselles, V., Kimmel, R., and Sapiro, G. Geodesic active contours. In *Proceedings of the Fifth International Conference on Computer Vision* (1995), pp. 694–699.

[27] Caselles, V., Kimmel, R., and Sapiro, G. Geodesic active contours. *The International Journal of Computer Vision 22*, 1 (1997), 61–79.

[28] Caselles, V., Kimmel, R., Sapiro, G., and Sbert, C. Minimal surfaces based object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 4 (1997), 394–398.

[29] Cerny, V. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications 45* (1985), 41–51.

[30] Chakraborty, A., Staib, H., and Duncan, J. Deformable Boundary Finding in Medical Images by Integrating Gradient and Region Information. *IEEE trans.Medical Imaging 15*, 6 (1996), 859–870.

[31] Chan, T., Sandberg, B., and Vese, L. Active contours without edges for vector-valued images. *Journal of Visual Communication and Image Representation 11* (2000), 130–141.

[32] Chan, T., and Vese, L. Active contours without edges. Tech. Rep. 98-53, UCLA CAM Report, 1999.

[33] Chan, T., and Vese, L. Active contours without edges. *IEEE Transactions on Image Processing 10*, 2 (Feb. 2001), 266–277.

[34] Charpiat, G., Faugeras, O., and Keriven, R. Approximations of shape metrics and application to shape warping and empirical shape statistics. *Foundations of Computational Mathematics 5*, 1 (2005), 1–58.

[35] Charpiat, G., Keriven, R., Pons, J.-P., and Faugeras, O. Designing spatially coherent minimizing flows for variational problems based on active contours. In *Proceedings of the 10th IEEE International Conference on Computer Vision* (2005), pp. 1403–1408.

[36] Chekuri, C. S., Goldberg, A. V., Karger, D. R., Levine, M. S., and Stein, C. Experimental study of minimum cut algorithms. In *SODA '97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 1997), Society for Industrial and Applied Mathematics, pp. 324–333.

[37] Cheriyan, J., Hagerup, T., and Mehlhorn, K. Can a maximum flow be computed on o(nm) time? In *Automata, Languages and Programming, 17th International Colloquium* (Warwick University, England, 1990), M. S. Paterson, Ed., vol. 443 of *Lecture Notes in Computer Science*, Springer, pp. 235–248.

[38] Cherkassky, B. V. A fast algorithm for computing maximum flow in a network. In *Collected Papers: Combinatorial Methods for Flow Problems*, A. V. Karzanov, Ed., vol. 3. The Institute for Systems Studies, 1979, pp. 90–96. In Russian. English translation appears as 'A fast algorithm for constructing a maximum flow through a network' in Amer. Math. Soc. Trans. Ser. 2, Vol. 158, pp. 23–30, 1994.

[39] Cherkassky, B. V., and Goldberg, A. V. On implementing the push-relabel method for the maximum flow problem. *Algorithmica 19*, 4 (1997), 390–410.

[40] Cherkassky, B. V., Goldberg, A. V., Martin, P., Setubal, J. C., and Stolfi, J. Augment or push: a computational study of bipartite matching and unit-capacity flow algorithms. *J. Exp. Algorithmics 3* (1998), 8.

[41] Chuang, Y.-Y., Curless, B., Salesin, D. H., and Szeliski, R. A bayesian approach to digital matting. In *CVPR* (December 2001), vol. 2, pp. 264–271.

[42] Cohen, E., and Megiddo, N. Strongly polynomial-time and nc algorithms for detecting cycles in periodic graphs. *J. ACM 40*, 4 (1993), 791–830.

[43] Cohen, L. On active contour models and balloons. *CVGIP: Image Understanding 53* (1991), 211–218.

[44] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[45] Crandall, M., Evans, L., and Lions, P. Some properties of viscosity solutions of Hamilton–Jacobi equations. *Trans. AMS 282* (1984), 487–502.

[46] Cremers, D., Kohlberger, T., and Schnörr, C. Shape statistics in kernel space for variational image segmentation. *Pattern Recognition 36*, 9 (2003), 1929–1943.

[47] Cremers, D., Rousson, M., and Deriche, R. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *International Journal of Computer Vision* (2006).

[48] Cremers, D., and Soatto, S. Motion competition: A variational framework for piecewise parametric motion segmentation. *International Journal of Computer Vision 62*, 3 (May 2005), 249–265.

[49] Cremers, D., Tischhäuser, F., Weickert, J., and Schnörr, C. Diffusion snakes: Introducing statistical shape knowledge into the mumford–shah functional. *International Journal of Computer Vision 50*, 3 (2002), 295–313.

[50] Curwen, R., and Blake, A. Dynamic contours : Real-time active splines. In *Active Vision*, A. Blake and A. Yuille, Eds. The MIT Press, 1993, ch. III, pp. 39–58.

[51] Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., and Yannakakis, M. The complexity of multiterminal cuts. *SIAM Journal on Computing 23*, 4 (1994), 864–894.

[52] Danielsson, P.-E. Euclidean Distance Mapping. *Computer Graphics and Image Processing 14*, 3 (Nov. 1980), 227– 248.

[53] deGruijter, J., and McBratney, A. A modified fuzzy k means for predictive classification. In *Bock,H.H.(ed) Classification and Related Methods of Data Analysis*. Elsevier Science, Amsterdam., 1988, pp. 97–104.

[54] Delfour, M., and Zolésio, J.-P. *Shapes and geometries*. Advances in Design and Control. Siam, 2001.

[55] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society 39* (1977), 1–38.

[56] Derigs, U., and Meier, W. Implementing goldberg's max-flow algorithm - a computational investigation. *ZOR-Methods and Models of Operations Research 33* (1989), 383–403.

[57] Derigs, U., and Meier, W. An evaluation of algorithmic refinements and proper data-structures for the preflow-push approach for maximum flow. *ASI Series on Computer and System Sciences 8* (1992), 209–223.

[58] Derin, H., and Elliott, H. Modelling and segmentation of noisy and textured images using gibbs random fields. *IEEE Trans. PAMI 9* (1987), 39–55.

[59] Dervieux, A., and Thomasset, F. A finite element method for the simulation of Rayleigh-Taylor instability. *Lecture Notes in Mathematics 771* (1979), 145–159.

[60] Dervieux, A., and Thomasset, F. Multifluid incompressible flows by a finite element method. In *Seventh International Conference on Numerical Methods in Fluid Dynamics* (June 1980), W. Reynolds and R. MacCormack, Eds., vol. 141 of *Lecture Notes in Physics*, pp. 158–163.

[61] Dinic, E. A. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics Doklady 11* (1970), 1277–1280.

[62] Dupont, R., Juan, O., and Keriven., R. Robust segmentation of hidden layers in video sequences. In *Proceedings of the 18th International Conference on Pattern Recognition* (2006), p. submitted.

[63] Dupont, R., Juan, O., and Keriven., R. Robust segmentation of hidden layers in video sequences. Tech. Rep. 06-21, CERTIS, 2006.

[64] Dupont, R., Paragios, N., Keriven, R., and Fuchs, P. Extraction of layers of similar motion through combinatorial techniques. In *EMMCVPR* (2005).

[65] Edmonds, J., and Karp, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of ACM 19*, 2 (1972), 248–264.

[66] Ferrari, P., Frigessi, A., and de Sa, P. G. Fast approximate map restoration of multicolor images. *Journal of the Royal Statistical Society 57*, 3 (1995), 485–500.

[67] Ferrari, P. A., Gubitoso, M. D., and Neves, E. J. Reconstruction of gray-scale images. *Methodology and Computing in Applied Probability 3* (1997), 255–270.

[68] Figueiredo, M., Leitao, J. M. N., and Jain, A. K. On fitting mixture models. In *Energy Minimization Methods in Computer Vision and Pattern Recognition* (1999), pp. 54–69.

[69] Ford, L., and Fulkerson, D. *Flows in Networks*. Princeton University Press, Princeton, 1962.

[70] Freedman, D., and Drineas, P. Energy minimization via graph cuts: Settling what is possible. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2005), vol. 2, IEEE Computer Society, pp. 939–946.

[71] Gard, T. *Introduction to Stochastic Differential Equations*. Marcel Dekker, New York and Basel, 1988.

[72] Gastaud, M., Barlaud, M., and Aubert, G. Combining shape prior and statistical features for active contour segmentation. *IEEE*

Transactions on Circuits and Systems for Video Technology 14, 5 (2004), 726–734.

[73] Geman, S., and Geman, D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 6 (1984), 721–741.

[74] Goldberg, A. V. *Efficient Graph Algorithms for Sequential and Parallel Computers*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1987. (Also available as a Technical Report TR-374, Lab. for Computer Science, M.I.T., 1987).

[75] Goldberg, A. V., and Tarjan, R. E. A new approach to the maximum flow problem. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing* (1986), ACM, pp. 136–146.

[76] Goldberg, A. V., and Tarjan, R. E. A new approach to the maximum-flow problem. *Journal of ACM 35*, 4 (1988), 921–940.

[77] Gomes, J., and Faugeras, O. Reconciling distance functions and level sets. *Journal of Visual Communication and Image Representation 11* (2000), 209–223.

[78] Greig, D., Porteous, B., and Seheult, A. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society: Series B 51* (1989), 271–279.

[79] Hochbaum, D. S. The pseudoflow algorithm and the pseudoflow-based simplex for the maximum flow problem. *Lecture Notes in Computer Science 1412* (1998), 325–337.

[80] Ishikawa, H. Exact optimization for markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 10 (October 2003), 1333–1336.

[81] Ishikawa, H., and Geiger, D. Occlusions, discontinuities, and epipolar lines in stereo. In *Proceedings of the Fifth European Conference on Computer Vision* (1998), vol. 1406, pp. 232–248.

[82] Ivins, J., and Porrill, J. Active region models for segmenting colours and textures. *Image and Vision Computing 13*, 5 (1995), 431–438.

[83] Jehan-Besson, S., Barlaud, M., and Aubert, G. Dream2s: Deformable regions driven by an eulerian accurate minimization method for image and video segmentation. *ijcv 53*, 1 (June 2003), 45–70.

[84] Jiang, G.-S., and Peng, D. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM Journal of Scientific Computing 21*, 6 (2000), 2126–2143.

[85] Jiang, G.-S., and Shu, C.-W. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics 126* (1996), 202–228.

[86] Juan, O., and Boykov, Y. Active cuts for real-time graph partitioning in vision. Tech. Rep. ♯655, University of Western Ontario, CS Dept, 2005.

[87] Juan, O., and Boykov, Y. Active graph cuts. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2006), p. to appear.

[88] Juan, O., and Keriven, R. Trimap segmentation for fast and user-friendly alpha matting. In *Prooceedings of 3rd International Workshop on Variational, Geometric, and Level Set Methods in Computer Vision* (2005), vol. 1, pp. 186–197.

[89] Juan, O., and Keriven, R. Unsupervised segmentation for digital matting. Tech. Rep. 05-04, CERTIS, 2005.

[90] Juan, O., Keriven, R., and Postelnicu, G. Stochastic mean curvature motion in computer vision: Stochastic active contours. In *Proceedings of the 2nd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision* (2003), pp. xx–xx.

[91] Juan, O., Keriven, R., and Postelnicu, G. Stochastic motion and the level set method in computer vision: Stochastic active contours. Tech. Rep. 04-01, CERTIS, 2004.

[92] Juan, O., Keriven, R., and Postelnicu, G. Stochastic motion and the level set method in computer vision: Stochastic active contours. *International Journal of Computer Vision* (2006), in press.

[93] Kadir, T., and Brady, M. Unsupervised non-parametric region segmentation using level sets. In *Proceedings of ICCV 2003* (October 2003).

[94] Karatzas, I., and Shreve, S. *Brownian motion and stochastic calculus*. Springer-Verlag, New York, 1991.

[95] Kass, M., Witkin, A., and Terzopoulos, D. Snakes: Active contour models. In *First International Conference on Computer Vision* (London, June 1987), pp. 259–268.

[96] Katsoulakis, M., and Kho, A. Stochastic curvature flows: Asymptotic derivation, level set formulation and numerical experiments. *Journal of Interfaces and Free Boundaries 3* (2001), 265–290.

[97] Ke, Q., and Kanade, T. A robust subspace approach to layer extraction. In *MOTION '02: Proceedings of the Workshop on Motion and Video Computing* (2002), p. 37.

[98] Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A., and Yezzi, A. Gradient flows and geometric active contour models. In *Proceedings of the Fifth International Conference on Computer Vision* (1995), pp. 810–815.

[99] Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A., and Yezzi, A. Conformal curvature flows: from phase transitions to active vision. *Archive for Rational Mechanics and Analysis 134* (1996), 275–301.

[100] Kimmel, R., Siddiqi, K., Kimia, B., and Bruckstein, A. Shape from shading: Level set propagation and viscosity solutions. *IJCV 16* (1995), 107–133.

[101] King, V., Rao, S., and Tarjan, R. A faster deterministic maximum flow algorithm. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 1992), Society for Industrial and Applied Mathematics, pp. 157–164.

[102] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983 220, 4598* (1983), 671–680.

[103] Kitamoto, A. The moments of the mixel distribution and its application to statistical image classification. In *Advances in Pattern Recognition (SPR'00), LNCS 1876* (8 2000), A. Amin, F. Ferri, J. Inesta, and P. Pudil, Eds., pp. 521–531.

[104] Kitamoto, A., and Takagi, M. Area proportion distribution – relationship with the internal structure of mixels and its application to image classification. *Systems and Computers in Japan 31*, 5 (5 2000), 57–76.

[105] Kohli, P., and Torr, P. H. S. Effciently solving dynamic markov random fields using graph cuts. In *Proceedings of the 10th IEEE International Conference on Computer Vision* (2005), IEEE Computer Society, pp. 922–929.

[106] Kolmogorov, V., and Boykov, Y. What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. In *Proceedings of the Tenth IEEE International Conference on Computer Vision* (Washington, DC, USA, 2005), vol. 1, IEEE Computer Society, pp. 564–571.

[107] Kolmogorov, V., Criminisi, A., Blake, A., Cross, G., and Rother, C. Bi-layer segmentation of binocular stereo video. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2005), vol. 2, IEEE Computer Society, pp. 407–414.

[108] Kolmogorov, V., and Zabih, R. What energy functions can be minimized via graph cuts? In *Proceedings of the 7th European Conference on Computer Vision-Part III* (London, UK, 2002), Springer-Verlag, pp. 65–81.

[109] Kolmogorov, V., and Zabih, R. What energy functions can be minimized via graph cuts? *IEEE Transaction on Pattern Analysis and Machine Intelligence 26*, 2 (2004), 147–159.

[110] Kumar, M. P., Torr, P. H. S., and Zisserman, A. Learning layered motion segmentations of video. In *ICCV* (2005).

[111] Kunita, H. *Stochastic Flows and Stochastic Differential Equations*. Cambridge University Press, 1990.

[112] Leventon, M., Grimson, E., and Faugeras, O. Statistical Shape Influence in Geodesic Active Contours. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (Hilton Head Island, South Carolina, June 2000), IEEE Computer Society, pp. 316–323.

[113] Lions, P., and Souganidis, P. Fully nonlinear stochastic partial differential equations. *C.R. Acad. Sci. Paris Ser. I Math 326* (1998), 1085–1092.

[114] Lions, P., and Souganidis, P. Fully nonlinear stochastic partial differential equations: nonsmooth equations and applications. *C.R. Acad. Sci. Paris Ser. I Math 327* (1998), 735–741.

[115] Lions, P., and Souganidis, P. Equations aux derivees partielles stochastiques non lineaires et solutions de viscosite. Seminaire X-EDP (Unpublished Lecture), 1999.

[116] Lions, P., and Souganidis, P. Fully nonlinear stochastic partial differential equations with semilinear stochastic dependence. *C.R. Acad. Sci. Paris Ser. I Math 331* (2000), 617–624.

[117] Lions, P., and Souganidis, P. Uniqueness of weak solutions of fully nonlinear stochastic partial differential equations. *C.R. Acad. Sci. Paris Ser. I Math 331* (2000), 783–790.

[118] Lombaert, H., Sun, Y., Grady, L., and Xu, C. A multilevel banded graph cuts method for fast image segmentation. In *Proceedings of the Tenth IEEE International Conference on Computer Vision* (2005), pp. 259–265.

[119] Lorensen, W. E., and Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the SIGGRAPH* (Anaheim, CA, July 1987), M. Stone, Ed., pp. 163–169. in Computer Graphics, Volume 21, Number 4.

[120] MacQueen, J. Some methods for classification and analysis of multivariate. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), pp. 281–297.

[121] MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), L. M. L. Cam and J. Neyman, Eds., vol. 1, pp. 281–297.

[122] McInerney, T., and Terzopoulos, D. T-snakes: Topology adaptive snakes. *Medical Image Analysis 4* (2000), 73–91.

[123] McLachlan, G., and David, P. *Finite Mixture Models*. Wiley, New York, 2000.

[124] McLachlan, G., and Krishnan, T. *The EM algorithm and extensions*. Wiley, New York, 1997.

[125] Mitsunaga, T., Yokoyama, T., and Totsuka, T. Autokey: Human assisted key extraction. In *SIGGRAPH* (1995).

[126] Mumford, D., and Shah, J. Boundary detection by minimizing functionals. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (San Francisco, CA, June 1985), IEEE, pp. 22–26.

[127] Mumford, D., and Shah, J. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics 42* (1989), 577–684.

[128] Nguyen, Q. C., and Venkateswaran, V. Implementations of goldberg-tarjan maximum flow algorithm. In *Network Flows and Matching: First DIMACS Implementation Challenge* (1993), D. S. Johnson and C. C. McGeoch, Eds., AMS, pp. 19–42.

[129] Odobez, J.-M., and Bouthemy, P. Direct incremental model-based image motion segmentation for video analysis. *Signal Processing 66*, 2 (1998), 143–155.

[130] Orchard, M. T., and Bouman, C. A. Color Quantization of Images. *IEEE Trans. on Signal Processing 39*, 12 (1991), 2677–2690.

[131] Osher, S., and Paragios, N., Eds. *Geometric Level Set Methods in Imaging, Vision and Graphics*. Springer Verlag, 2003.

[132] Osher, S., and Sethian, J. Fronts propagating with curvature dependent speed: algorithms based on the Hamilton–Jacobi formulation. *Journal of Computational Physics 79* (1988), 12–49.

[133] Paragios, N., Chen, Y., and Faugeras, O. *Handbook of Mathematical Models in Computer Vision*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[134] Paragios, N., and Deriche, R. Geodesic active regions for texture segmentation. Tech. Rep. 3440, INRIA, France, June 1998.

[135] Paragios, N., and Deriche, R. Geodesic active regions for motion estimation and tracking. Tech. Rep. 3631, INRIA, France, Mar. 1999.

[136] Paragios, N., and Deriche, R. Geodesic active regions for supervised texture segmentation. In *Proceedings of the 7th International Conference on Computer Vision* (Kerkyra, Greece, Sept. 1999), IEEE Computer Society, IEEE Computer Society Press.

[137] Paragios, N., and Deriche, R. Unifying boundary and region-based information for geodesic active tracking. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (Fort Collins, Colorado, June 1999), IEEE Computer Society.

[138] Paragios, N., and Deriche, R. Coupled geodesic active regions for image segmentation : a level set approach. In *Proceedings of the 6th European Conference on Computer Vision* (Dublin, Ireland, June 2000), vol. II, pp. 224–240.

[139] Paragios, N., and Deriche, R. Geodesic active regions: a new paradigm to deal with frame partition problems in computer vision. *Journal of Visual Communication and Image Representation, Special Issue on Partial Differential Equations in Image Processing, Computer Vision and Computer Graphics 13*, 1/2 (march/june 2002), 249–268.

[140] Paragios, N., and Deriche, R. Geodesic active regions and level set methods for supervised texture segmentation. *The International Journal of Computer Vision 46*, 3 (2002), 223–247.

[141] Paragios, N., Rousson, M., and Ramesh, V. Matching distance functions: A shape-to-area variational approach for global-to-local registration. In *Proceedings of the 7th European Conference on Computer Vision* (May 2002), vol. 2, pp. 775–789.

[142] Peng, D., Merriman, B., Osher, S., Zhao, H., and Kang, M. A PDE-based fast local level set method. *Journal on Computational Physics 155*, 2 (1999), 410–438.

[143] Perez, P., Gangnet, M., and Blake, A. Poisson image editing. In *SIGGRAPH* (2003).

[144] Pons, J.-P., Hermosillo, G., Keriven, R., and Faugeras, O. How to deal with point correspondences and tangential velocities in the level set framework. In *Proceedings of the 9th International Conference on Computer Vision* (Nice, France, 2003), IEEE Computer Society, IEEE Computer Society Press, pp. 894–899.

[145] Pons, J.-P., Hermosillo, G., Keriven, R., and Faugeras, O. How to deal with point correspondences and tangential velocities in the level set framework. In *Proceedings of the 9th IEEE International Conference on Computer Vision* (2003), vol. 2, pp. 894–899.

[146] Rissanen, J. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Company, 1989.

[147] Rother, C., Kolmogorov, V., and A., B. Grabcut - interactive foreground extraction using iterated graph cuts. In *SIGGRAPH* (2004).

[148] Rother, C., Kolmogorov, V., and Blake, A. Grabcut: interactive foreground extraction using iterated graph cuts. *ACM Transaction on Graphics (SIGGRAPH) 23*, 3 (2004), 309–314.

[149] Rousson, M., Brox, T., and Deriche, R. Active unsupervised texture segmentation on a diffusion based space. In *International Conference on Computer Vision and Pattern Recognition* (Madison, Wisconsin, USA, June 2003), vol. 2, pp. 699–704.

[150] Rousson, M., and Deriche, R. A variational framework for active and adaptative segmentation of vector valued images. In *Proc. IEEE Workshop on Motion and Video Computing* (Orlando, Florida, Dec. 2002), pp. 56–62.

[151] Rousson, M., and Paragios, N. Shape priors for level set representations. In *Proceedings of the 7th European Conference on Computer Vision* (London, UK, May 2002), vol. 2, Springer-Verlag, pp. 78–92.

[152] Rouy, E., and Tourin, A. A Viscosity Solutions Approach to Shape-from-Shading. *SIAM Journal of Numerical Analysis 29*, 3 (June 1992), 867–884.

[153] Roy, S., and Cox, I. J. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Proceedings of the Sixth International Conference on Computer Vision* (1998), pp. 492–502.

[154] Rudin, L., Osher, S., and Fatemi, E. Nonlinear total variation based noise removal algorithms. *Physica D 60* (1992), 259–268.

[155] Ruzon, M., and Tomasi, C. Alpha estimation in natural images. In *CVPR* (2000).

[156] Scharstein, D., and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision 47*, 1-3 (2002), 7–42.

[157] Sedgewick, R. *Algorithms*. Addison-Wesley, Reading, MA, 1988.

[158] Sethian, J. Curvature flow and entropy conditions applied to grid generation. *Journal Of Computational Physics 115* (1994), 440–454.

[159] Sethian, J. *Level Set Methods*. Cambridge University Press, 1996.

[160] Shanon X. Ju, Michael J. Black, A. D. J. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *CVPR* (1996), p. 307.

[161] Shu, C. W., and Osher, S. Efficient implementation of essentially non-oscillatory shock-capturing schemes, ii. *Journal of Computational Physics 83* (1989), 32–78.

[162] Smith, A. R., and Blinn, J. F. Blue screen matting. In *23rd annual conference on Computer graphics and interactive techniques* (1996), ACM Press, pp. 259–268.

[163] Souganidis, P. Approximation schemes for viscosity solutions of Hamilton–Jacobi equations. *Journal of Differential Equations 59* (1985), 1–43.

[164] Sun, J., Jia, J., Tang, C., and shum, H. Poisson matting. In *SIGGRAPH* (2004).

[165] Sussman, M., Smereka, P., and Osher, S. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *J. Computational Physics 114* (1994), 146–159.

[166] Terzopoulos, D., Witkin, A., and Kass, M. Constraints on Deformable Models: Recovering 3D shape and Nonrigid Motion. *Artificial Intelligence 36*, 1 (1988), 91–123.

[167] Ueda, N., Nakano, R., Ghahramani, Z., and Hinton, G. E. Smem algorithm for mixture models. *Neural Computation 12*, 9 (2000), 2109–2128.

[168] Vese, L., and Chan, T. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision 50* (2002), 271–293.

[169] Vese, L. A., and Chan, T. F. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision 50*, 3 (2002), 271–293.

[170] Wallace, C., and Boulton, D. An information measure for classification. *Computer Journal 11*, 2 (1968), 185–194.

[171] Walsh, J. An introduction to stochastic partial differential equations. In *Ecole d'Été de Probabilités de Saint-Flour*, vol. XIV-1180 of *Lecture Notes in Math.* Springer, 1994.

[172] Wang, J., and Adelson, E. Layered representation for motion analysis. In *CVPR93* (1993), pp. 361–366.

[173] Weiss, Y. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR* (1997).

[174] Wexler, Y., Fitzgibbon, A., and Zisserman, A. Bayesian estimation of layers from multiple images. In *ECCV* (2002).

[175] X. Descombes, M. Sigelle, F. P. Estimating gaussian markov random field parameters in a nonstationary framework: Application to remote sensing imaging. *IEEE Trans. on Image Processing 8*, 4 (April 1999), 490–503.

[176] Xiao, J., and Shah, M. Accurate motion layer segmentation and matting. In *CVPR* (2005).

[177] Xiao, J., and Shah, M. Accurate motion layer segmentation and matting. In *CVPR* (2005), pp. 698–703.

[178] Xu, N., Bansal, R., and Ahuja, N. Object segmentation using graph cuts based active contours. In *IEEE Conference on Computer Vision and Pattern Recognition* (2003), vol. II, pp. 46–53.

[179] Yip, N. Stochastic motion by mean curvature. *Arch. Rational Mech. Anal. 144* (1998), 331–355.

[180] Yip, N. Stochastic curvature driven flows. In *Stochastic Partial Differential Equations and Applications*, G. Da Prato and L. Tubaro, Eds., vol. 227 of *Lecture Notes in Pure and Applied Mathematics*. Springer, 2002, pp. 443–460.

[181] Yip, N., and Souganidis, P. Uniqueness of motion by mean curvature perturbed by stochastic noise. *Annales de l'Institut Henri Poincare - analyse non lineaire*, 21 (2004), 1–23.

[182] Zhang, B., and Hsu, M. K-harmonic means – a data clustering algorithm, 1999.

[183] Zhao, H., Chan, T., Merriman, B., and Osher, S. A variational level set approach to multiphase motion. *Journal of Computational Physics 127*, 0167 (1996), 179–195.

[184] Zhu, S., and Yuille, A. Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 9 (Sept. 1996), 884–900.