Tracking and Describing Deformable Objects Using Active Contour Models

by

Frédéric F. Leymarie

B.Eng., Polytechnic School of Montreal, Canada, 1986

A dissertation submitted in partial fulfillment of the requirements for the
Degree of Master of Engineering
in the Department of Electrical Engineering
at McGill University

MONTREAL, QUEBEC, CANADA

February 1990

Also published as Technical Report CIM-90-9, McGill Research Center for Intelligent Machines. This version is a reprint produced in February 2003, while at Brown University. It contains a few updates and corrections.

Abstract

In this thesis we consider a number of issues in developing techniques and algorithms to automate the visual tracking of deformable objects in the plane. We have applied these techniques in cell locomotion and tracking studies. We examine two classes of computer vision problems. First, we consider the segmentation of a noisy intensity image and the tracking of a nonrigid object. Second, we consider the shape analysis of an amorphous object. In evaluating these problems, we explore a new technique based on an active contour model commonly called a "snake." The snake permits us to simultaneously solve, in constrained cases, both the segmentation and tracking problems. We present a detailed analysis of the snake model, emphasizing its limitations and shortcomings, and propose various improvements to the original description of the model. Then, we study the two complementary types of shape descriptors: boundary- and region-based. We propose to combine these within the context of the grassfire transform. Two new algorithms are described. First, we present a contour segmentation technique using mathematical morphology on the curvature function. Accurate localization for different scales of curvature features is achieved. Second, the snake model is used to simulate the grassfire transform using the previously extracted contour features. This permits us to produce a multiscale skeleton representation of shape which is based on the Euclidean distance metric. New significance criteria for our shape descriptors, such as the "region-support" of curvature extrema and the "ridge-support" of skeleton branches are also proposed. Finally, numerous implementation details are discussed; for example, the description of an efficient sequential Euclidean distance transform.

Sommaire

Dans cette thèse nous considérons différents algorithmes et techniques afin d'automatiser le suivi visuel d'objets déformables dans le plan; nous avons appliqué ces techniques à l'étude du mouvement de la cellule. Pour ce faire, nous examinons deux classes de problèmes dans le cadre de la vision par ordinateur. Premièrement, nous considérons la segmentation d'image et le suivi visuel d'un objet informe. En second lieu, nous considérons l'analyse morphologique d'objets "informes." Afin de résoudre ces problèmes, nous explorons une nouvelle technique basée sur le concept d'un "modèle de contour actif," communément appelé *snake*. Cette technique nous permet de solutionner simultanément la segmentation et le suivi visuel. Nous présentons une analyse détaillée du *snake*, mettant l'accent sur ses limites d'application et ses défauts, et nous proposons plusieurs améliorations relatives à la description du modèle original. Par la suite, nous étudions deux types complémentaires de descripteurs de forme, basés respectivement sur les notions de contour de l'objet et de région intérieure à ce contour. Nous proposons de combiner ces descripteurs par l'intermédiaire de la transformée dite du *grassfire*. Deux nouveaux algorithmes sont alors décrits. Nous présentons d'abord une technique de segmentation de contour utilisant la morphologie mathématique appliquée à la fonction de courbure. Nous obtenons ainsi une localisation précise de certaines caractéristiques de la fonction de courbure, tels ses extrema, et ce, pour différentes échelles. Ensuite, le modèle du *snake* est utilisé afin de simuler la transformée du *grassfire* tout en utilisant les résultats obtenus lors de la segmentation du contour. Ceci nous permet de produire une représentation de forme par son "squelette" à différentes échelles, tout en utilisant une métrique de distance euclidienne. De plus, de nouveaux critères de sélection sont proposés pour nos descripteurs de forme. Finalement, plusieurs problèmes de mise en application sont discutés, entre autres la description d'un algorithme de tranformée de distance euclidienne, optimisé au niveau de sa complexité numérique.

Acknowledgments

My first thanks are for my thesis director, Professor Martin D. Levine, whose constant support, patience and enthusiasm were of invaluable help. I extend my thanks: to David Gauthier and Professor Peter B. Noble for providing the initial data and assistance in understanding the issues involved in cell tracking; to David G., Frank DiGiuseppe and Benoit Dubuc for many hours of proof-reading and discussions; to Benjie Kimia, Guy Godin, Chantal David, and many others in the lab for their thoughts and critics; to Dr. Demetri Terzopoulos for assistance with the numerical analysis.

A special thanks goes to my friend Daniel Kornitzer who first introduced me to the field of computer vision and then to the lab at McGill. A special thanks also goes to Professor Paul Cohen of "Ecole Polytechnique de Montréal" who was my first mentor in this field and who gave me the first opportunity to spend a summer doing research in computer vision. Many friends and relatives made this journey possible and enjoyable. Warmest regards to them: my parents, my grandparents, Theresa Hinz, Geneviève Robert, Daniel K., Christian Consol, François Lagueux, Benjie K.

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada in the form of a postgraduate scholarship. It was also partially supported by the Medical Research Council of Canada under Grant no. MT-3236.

Contents

Chapter 1

Introduction

This thesis addresses a number of issues in developing techniques and algorithms to permit the computerized automation of the visual tracking of deformable objects in the plane. We seek to give the computer the capacity to recognize, identify and describe nonrigid objects as they deform and move. More specifically we are interested in the visual tracking of living cells, such as human fibroblasts [115] as they move on a planar surface such as glass. We think of such living cells as constituting an ideal *amorphous* object for testing our algorithms since their boundary can deform simultaneously at different subparts to take virtually any form (Figure 1.1).

Our interest in the study of cell movement arises from a long term research project to characterize the dynamic behavior of those living human cells which use *pseudopods*, that is, protrusions of their cellular membrane, as a mean for locomotion.[1] Cell movement is a fundamental process of importance to cell biology [115, 83]. Comprehension of aspects of cell biology as diverse as migrations of cells in embryological development and host defense mechanisms is directly dependent on the study of cell movement [161]. Such a study may be performed by first recording sequences of images of cells obtained with a

[1]This project has emerged from a collaboration between the faculties of Dentistry, Medicine and Engineering at McGill University. The following reference list constitutes a good summary of the research work which has been accomplished in our group during the last thirteen years [34, 58, 89, 60, 161, 88, 115, 52].

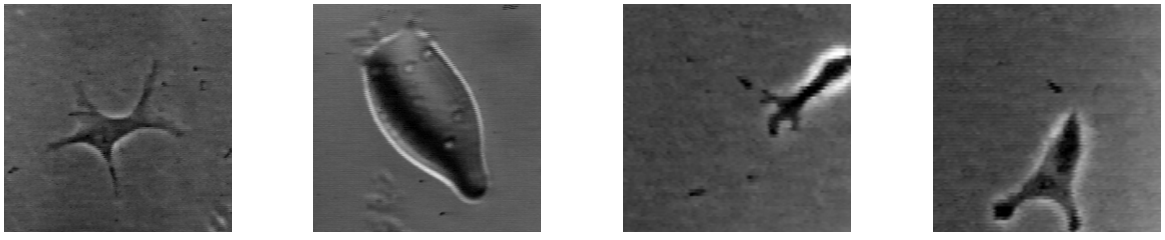


Figure 1.1: Example of various forms taken by a living cell.

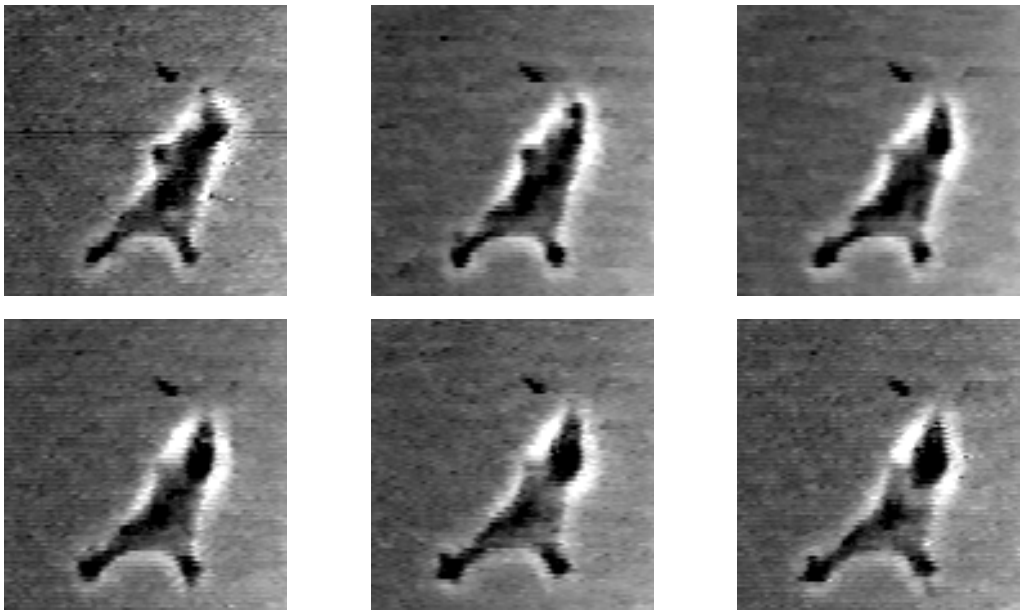


Figure 1.2: Example of an image sequence of a deforming and moving cell.

digitizing camera mounted on a microscope. Cell tracking is then achieved, and the cells are followed from frame to frame (Figure 1.2), in order to detect changes in position and variation in shape properties, that is, shape attributes of the cell's body and subparts or pseudopods. Cell dynamics can then be characterized by these variations in time of the cells' shape features.

1.1 The Cell Tracking Problem

Imagine being a technician or researcher in a physiology laboratory, performing visual analyses for experiments on cell motion. Below follows a brief summary of a typical day in "your" life:

> Here you are again. Fixing your eyes to the eyepiece of the microscope. For hours you have been meticulously noticing any change in the cell's form, seeking any special event that will reveal why and how it moves, sleeps and deforms, or reacts to the presence of an intruder or a companion, or catastrophically divides itself giving birth to two new sisters.
>
> The work is painful. The images are sometimes clear, sometimes blurred or corrupted by extraneous moving objects and noise. The experiments become

worthwhile when tracking many of the "companion" cells. How can one characterize the shape of these amorphous living beings?

It is late at night now. But you must continue as the results of your work may be considerably impoverished if you lose contact with one of the cells and start to mistakenly track another. You cannot use a radioactive isotope to mark the cells and help your sleepy visual system since it would corrupt the cell's behavior.

You start thinking that you still have to repeat such an experiment tens of times and for tens of different cells in different chemical environments. Analyzing the data is the essence of your work, but it seems now to be a long term goal. You close your eyes.

The engineering problems we are facing can be situated at two levels. First is the need for automation. It should now be evident that the visual analysis of cell motion requires its automation on a computer to perform numerous tedious computations, such a job being impractical for a human to realize efficiently. This automation system should provide the human experimenter with reliable qualitative as well as quantitative informations, such as shape deformation dynamics, average motion or multicellular interactions. In a sense, we can think of this system as a tool to extract and summarize, from the raw data, the essential information of the tracking for the benefit of the experimenter.

At a second and more profound level, is the need to develop "qualitatively accurate" as well as "numerically efficient" techniques. Such methods should permit us to solve a number of essential computer vision problems. By qualitatively accurate we mean to say that we are looking for techniques and processes giving us meaningful qualitative information, such as the position of relatively significant corners, bumps and parts of a cell's shape. Furthermore, such information should be robustly computed in the presence of noise. Numerical efficiency refers to our need to provide algorithmic implementations which are optimized for speed, since the amount of data to be processed is enormous. In our particular case we will restrict ourselves to algorithms for sequential computers. Within the objectives of this thesis, we will try to address two classes of computer vision problems summarized in the following two questions:

1. How to segment an image and track significant events of a nonrigid object as it deforms and moves?

2. How to identify and describe the shape of an amorphous object in both the static and dynamic cases?

1.2 Approach and Motivations

Until recently there has not existed a method for quantifying the observable changes in the shape of a cell membrane that occur during locomotion [115, 119]. Changes in cellular

membrane shape activity, such as the formation of pseudopods, are the first morphological events visible as the cells respond to chemotactic agents. Moreover, many important types of cells, such as polynuclear leukocytes (PMN) and lymphocytes are known to use their locomotory organs, the pseudopods, to locomote. Little is known of the pseudopod kinetics of these cells during locomotion [115]. Many attempts to study and characterize locomotory paths, considering cells as points moving on a planar surface,[2] have lead to advances in the understanding of *chemotaxis*[3] and *chemokinesis*.[4] However, it is only recently that an image interpretation system capable of describing and quantifying, not only the properties of locomotion of a moving cell, but also its shape and structural dynamics, has been designed and implemented on a computerized environment [60, 161, 88, 115, 52].

It is on the basis of the results of this research that we have decided to approach the problems of "tracking and describing deformable objects" such as living human cells. We have decided to focus our attention on the essential first processing steps of image segmentation and shape description and not consider higher level tasks, such as the feedback monitoring and the control of a complete image understanding system (for examples of such systems, see [60, 161]). Image segmentation and shape description are two basic, but complex, domains of research in computer vision; and even if many attempts at providing solutions to them have been proposed in the past, no definitive answers have yet been given [87].

Segmentation of an image to extract cell boundaries or contours is the first task required to identify what is a cell and what is not in a digitized image. Many techniques are actually available. Techniques based on "classical" segmentation algorithms, such as region growing, edge detection and relaxation labeling, usually require too much computation to obtain accurate results [67]. Therefore, we have explored a new segmentation technique based on the work of Kass *et al.* [78] which introduces the concept of an *active contour model*, also commonly called a *snake*. We are interested in this new concept, because the snake permits us to simultaneously solve both the segmentation and tracking problems. Snakes can be represented as energy-minimizing splines guided by external constraint forces and image forces such as lines, edges, subjective contours and region homogeneities found in the image. Furthermore, internal spline forces impose smoothness constraints on the modeled contours. By combining and integrating various types of information found in an image, snakes can lead to results at least comparable to other image segmentation techniques. But it is the dynamic behavior of the snakes which is mainly of interest here. From frame to frame, a snake will stick to the cell contour by following any small deformations which

[2] Looking at cell movement without regard to its membrane activity constitutes the large majority of the research done so far; for example, see [129, 34, 17, 89, 163, 50].

[3] *Chemotaxis*: Reaction by which the direction of locomotion of cells is determined by substances in their environment [115].

[4] *Chemokinesis*: Reaction by which the direction of locomotion of cells and/or the frequency of turning of cells, moving at random, is determined by substances in their environment [115].

may occur when a cell moves. However, an essential assumption is required: a cell can move only a small distance between frames. This is needed because the snake is in some sense relatively "blind" in its search for the optimal contour. The snake is able to track only small deformations on its own, that is, without the help of higher level or more complex processes. Consequently, most of the processing time required by the snake to converge to the cell boundary is consumed in the initial frame. In addition, some amount of user interaction to provide an initial position for the snake is required in the first analyzed frame. Then, snakes implicitly take care of tracking the cell by dynamically sticking to boundaries.

Shape description is considered once the cell boundaries are available, that is, when they are segmented from the digitized image. Following the research of our predecessors [52], we have tried to optimize shape description algorithms based on the notion of the *skeleton* of a shape, that is, on its representation by "idealized thin lines" that retain the connectivity or topology of the original shape. The skeleton is believed to be the most powerful representation available at present for characterizing the shape deformations and evolution of shape subparts of nonrigid natural forms [115, 52]. The *skeletonization* of a shape, that is, the process by which the skeleton is obtained, permits us to explicitly relate significant boundary features to the internal structure of an object. In particular, we have used, as a skeletonization process, the *grassfire* transform where an object's boundary is taken as an initial fire front which propagates within the object's interior region. Points where the fire front folds or interacts with itself are retained as indicators of shape features such as symmetries, subparts, protrusions and depressions. To simulate the fire propagation we again use the snake model and show its advantages over previous approaches for shape skeletonization [94]. However, our shape skeletonization method requires some initial processing of the object boundary to identify those curvature extrema where the fire front collapses as soon as we "ignite" the fire. To do so we have designed a new method for extracting contour features on the basis of *morphological operators* applied to the curvature function of the boundary [90].

1.3 Overview and Contributions

Our goals are twofold. First we wish to provide and implement robust techniques for the segmentation and tracking of nonrigid objects such as cells, as they deform and move. Second we must consider the shape description problem. Since the keystone of our approach to these problems relies on the notion of an active contour model, we first discuss this topic in Chapter 2. There we give a detailed analysis of the model in both the continuous and discrete domains. We have included this material because of the lack of such a detailed description in the existing literature. These insights lead us to propose various improvements to the original model. In particular, we demonstrate the need for normalizing the forces acting on the snake so that it remains stable under adverse conditions. We also propose

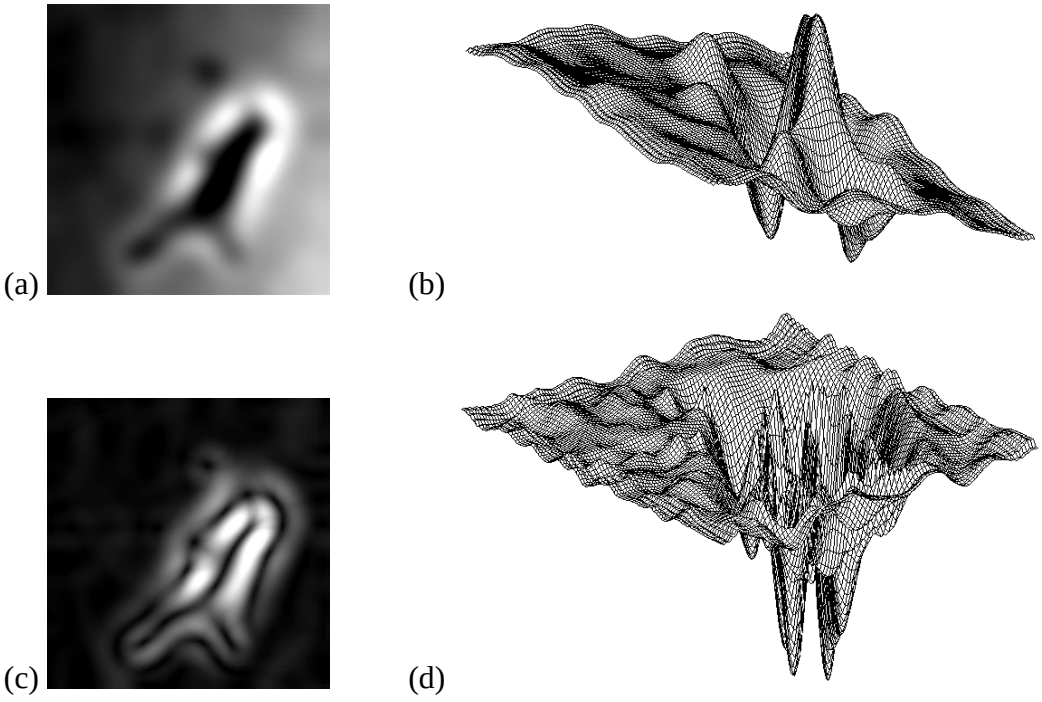


Figure 1.3: Intensity image seen as a topographic map. This three-dimensional (3-D) surface representation is produced by taking the intensity values as heights. White is considered the zero level height, while black is the highest level. In (a) is shown the image of a cell blurred using a Gaussian filter. In (b) is shown its representation as a 3-D surface. In (c) is shown the gradient of the image in (a), and in (d) is shown its representation as a 3-D surface.

a new criterion for the optimization of the search of the image contours which is based on topographic concerns, where the intensity image is seen as a kind of topographic map (Figure 1.3). Finally, we emphasize the limitations of the active contour model which requires powerful initialization processes and which arbitrarily imposes certain smoothness constraints on the raw data.

We then consider in Chapter 3 noisy intensity images and use the snake model to perform image segmentation. The first step consists of producing a *potential surface* or topographic map representation of the image (Figure 1.3). This surface is filtered to emphasize those features we wish the snake to be sensitive to (*e.g.*, edges, lines). Therefore, in Chapter 3, we first consider the problem of efficient image filtering. We study a certain class of algorithms known as *Hierarchical Discrete Correlation* [38] and propose some improvements especially suited to the snake model. Once the image is appropriately filtered, we impose the snake on its potential surface representation to perform the segmentation of the

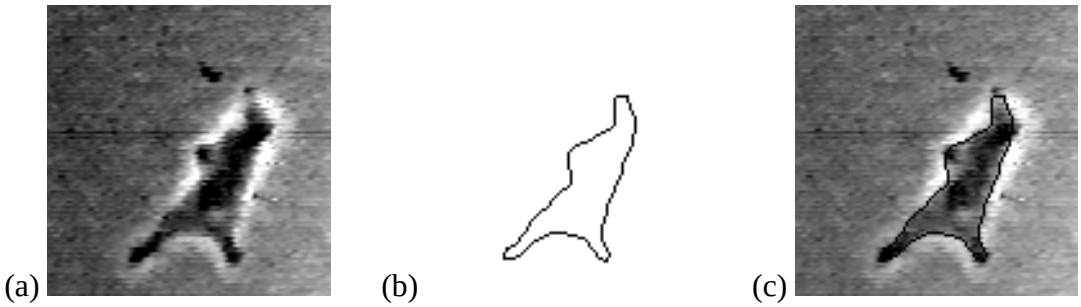


Figure 1.4: Image segmentation using the snake. In (a) is shown the original intensity image. In (b) is shown the line drawing obtained from the optimal snake position. In (c), the snake in (b) is superimposed on the image in (a).

object's boundary in the intensity image (Figure 1.4). Although our results for the image segmentation of a nonrigid object are promising, we do demonstrate that the snake model fails in certain particular cases due to its lack of ability to accurately sense the *local shape* of the potential surface.

We then apply the snake model to the tracking of cells as they deform and move (Figure 1.5).[5] In general, the snake model provides satisfying results, but fails in certain adverse cases. In particular, a cell may locally and temporarily leave the focal plane (*e.g.*, a pseudopod grows above the glass surface). When the cell returns to the focal plane, the snake may be unable to track what corresponds to large deformations.

After image segmentation and tracking, we consider the shape description problem. In Chapter 4, we first present a new method for extracting contour features such as extrema of curvature and straight line segments (Figure 1.6). Our new method is an application of mathematical *morphology to functions* [135] used to process the curvature function of an object contour. The main contribution of this approach is a new scale-space representation, obtained with the use of nonlinear morphological operators, which possess a number of advantages over the more traditional representations based solely on Gaussian smoothing.

Finally, in Chapter 5, we combine the extracted contour features with a new implementation of the grassfire transform, simulated using the snake model (Figure 1.7).[6] This permits us to explicitly relate boundary and region information, a departure from most shape description techniques found in the literature. We also make other more practical contributions. Specifically, we propose the new concept of a *dynamic skeleton* (*i.e.*, a deformable skeleton), we address the subject of *branch significance* of a skeleton and propose the concept of *ridge support* to characterize it. We also consider a number of implementational

[5] Addendum (February 2003): A journal article based on Chapters 2 and 3 was published in IEEE-PAMI in 1993 [98].

[6] Addendum (February 2003): A journal article based on Chapter 5 was published in IEEE-PAMI in 1992 [97].

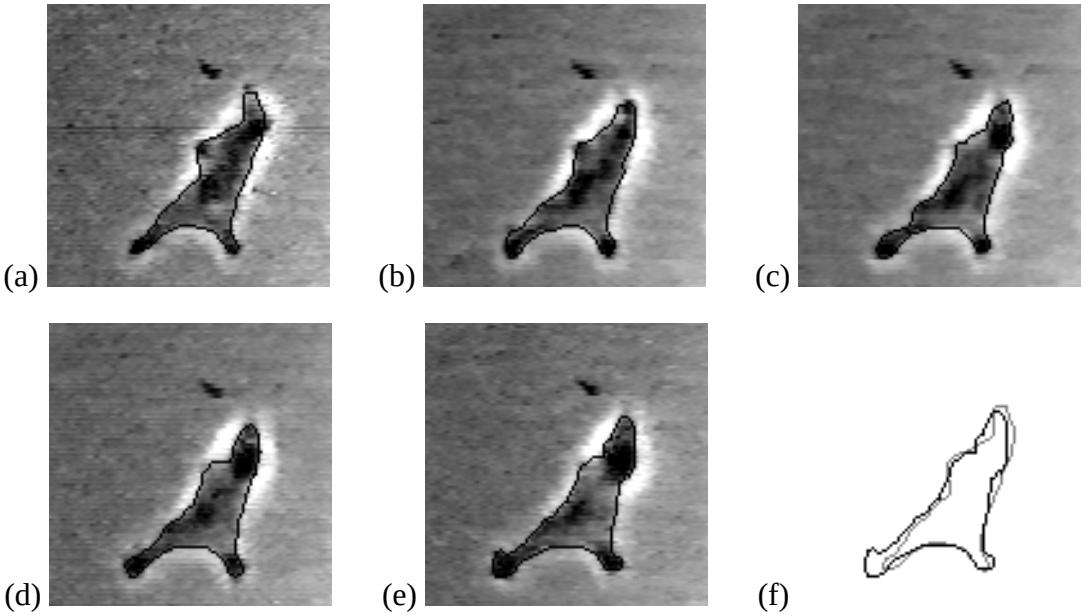


Figure 1.5: Cell tracking using the snake model. All frames have their optimal snakes superimposed to show the extracted contour. In (f) are shown the extracted contours in the first frame (grey dotted contour) and the last one (dark connected contour) to illustrate the total deformation that was recovered from the observed sequence.

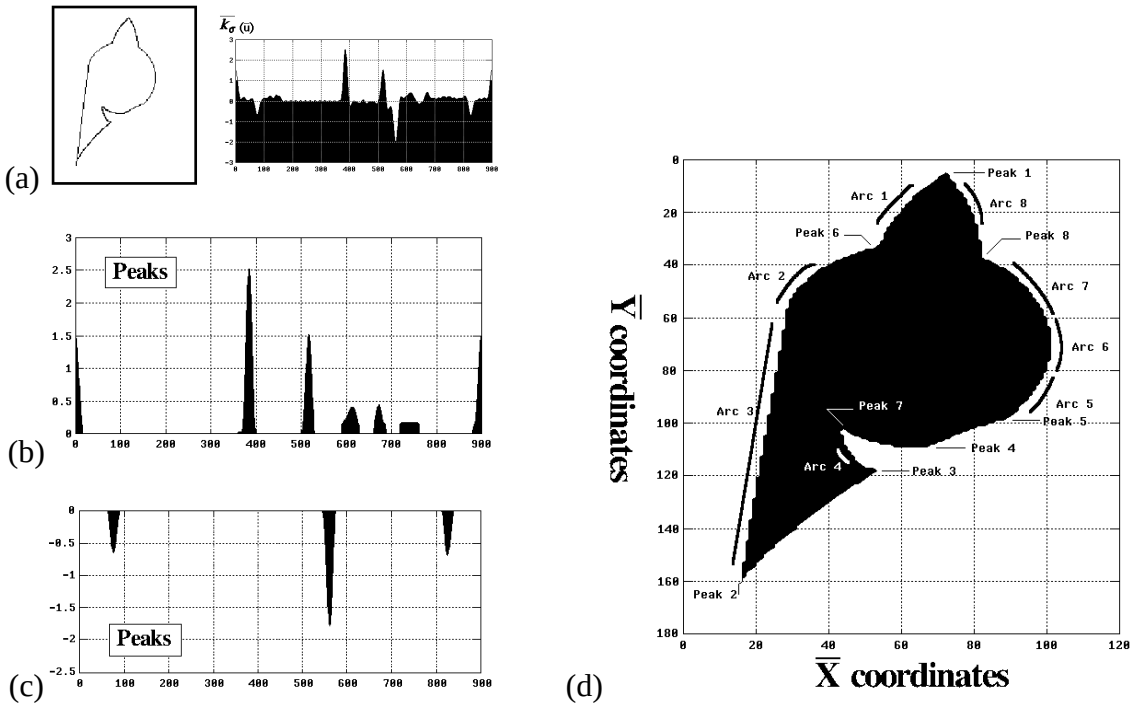


Figure 1.6: Segmentation of a contour. We use mathematical morphology operators on the curvature function to help us extract contour features. In (a) are shown the initial object outline on the left-hand-side and its smoothed discrete curvature function $\overline{k}_\sigma(\overline{u})$ on the right-hand-side. In (b) are shown the extracted peaks of the positive part of $\overline{k}_\sigma(\overline{u})$. These will correspond to significant convexities along the boundary. In (c) are shown the extracted peaks of the negative part of $\overline{k}_\sigma(\overline{u})$. These will correspond to significant concavities along the boundary. Finally, in (d) is shown the resulting segmentation of the object outline in terms of arcs and curvature extrema.

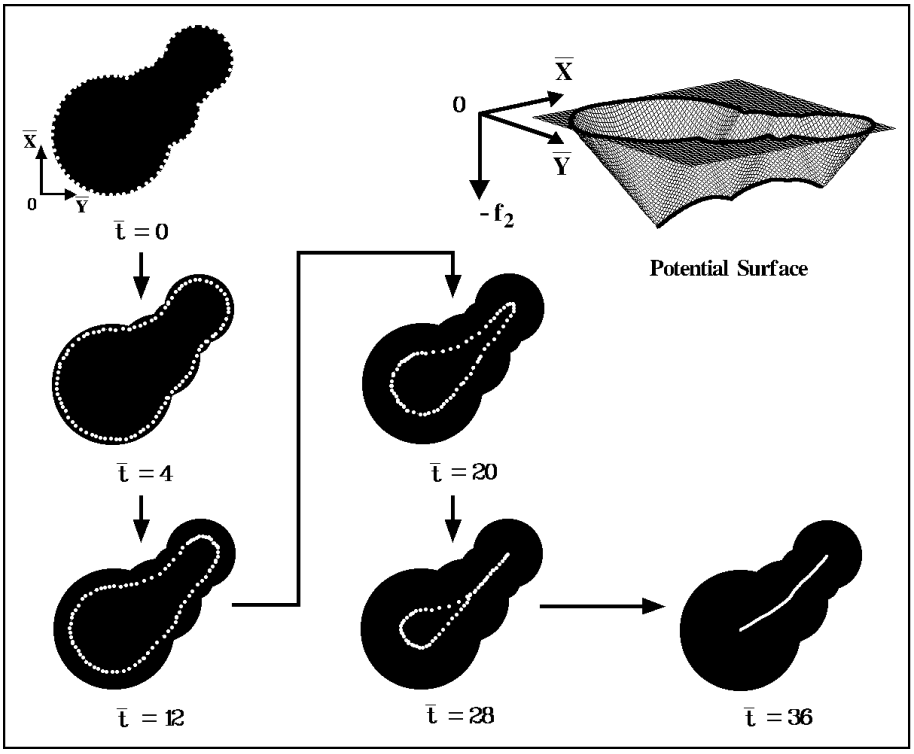


Figure 1.7: Simulating the grassfire transform (*i.e.*, shape skeletonization) using the snake model. The potential surface is shown at the top right corner. The fire is ignited at time $\overline{t} = 0$ where time corresponds to iteration. The stable state occurs at $\overline{t} = 36$ iterations. This last picture shows the final result when the number of snake elements is increased so that the snake is spatially connected, that is, without any gaps.

issues. In particular, we provide details of an optimized implementation of an Euclidean distance mapping algorithm, which gives us a suitable potential surface representation for the snake model when simulating the grassfire transform (Appendix D).[7]

[7] Addendum (February 2003): A journal article based on Appendix D was published in CVGIP-IU in 1992 [96].

Chapter 2

The Snake Model

The snake model arose from the regularization paradigm applied to computer vision in which vision is seen as a problem of inverse optics [142], therefore being in general underconstrained. Within this regularization framework, the problem is further constrained or regularized by imposing smoothness constraints on an object fitting the data. A measure of the discrepancy between the data and the fitted object is also used to relate the model to this data in an attempt to recover the embedded features. Since we are mainly interested in the problem of reconstructing or recovering contours of cells in images, we have only considered one-dimensional (1-D) objects. Such an object is the active contour, also called "snake," first proposed by Kass *et al.* [78] as a possible tool for solving the segmentation problem. It consists of an energy-minimizing spline guided by external constraint forces and influenced by potential field forces that maneuver it toward desirable features. Furthermore, internal constraints can be used to enforce smoothness or to introduce a discontinuity at a point on the snake. The snake model was originally intended to be applied to graph of functions when searching for local minima of these graph of functions. Specifically, an image can be seen as a 2-D function (see the following section for more details), and, by an appropriate filtering scheme, significant image features such as edges and contours will correspond to local minima of the graph of this function.

The snake is made active by always minimizing its energy function. If it is positioned "close enough" to significant contours in the image, that is, close enough so that potential field forces exist in relation to these contours, the snake will move and ultimately reach a local energy minimum corresponding to that image contour. Therefore, the snake relies on external mechanisms to place it close to salient contours. Although this approach does not completely solve the image segmentation problem, the snake's dynamic behavior and implicit connectivity prove to be key factors in the decision to use it for cell tracking, or more generally for the tracking of deformable shapes.

The problem of selecting an initial position for the snake becomes almost negligible when one considers a sequence of images; for example, when attempting to perform a mo-

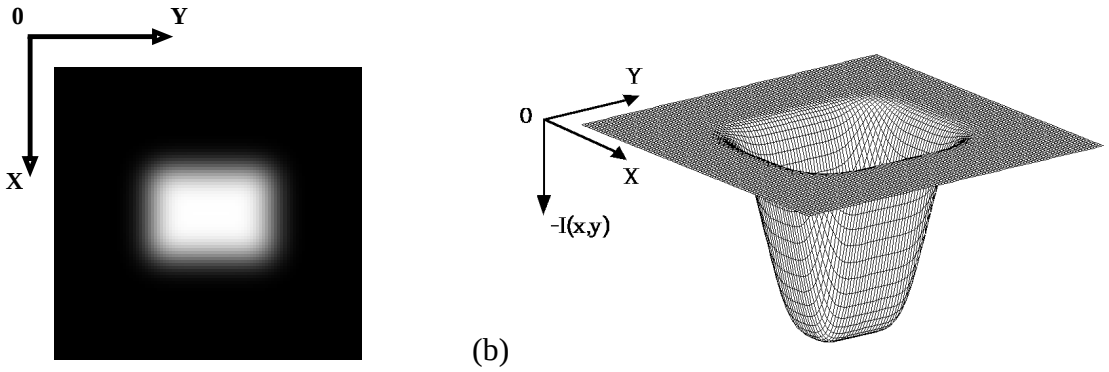
tion analysis of deformable shapes. In such a case, only the very first frame or image causes a real initialization problem, which can always be solved by permitting user interaction. For the following frames, the "optimal" snake position found in the preceding frame may be used as a good initial position for the analysis of the current frame. This technique will work quite satisfactorily if one assumes that "large deformations" will not occur from frame to frame. By "'large deformations" we mean to say deformations for which potential field forces, corresponding to the newly deformed contour, will not influence the snake. These field forces may be situated far enough from the snake, as an effect of the deformation, so that the snake is out of range from them. In the case of cell tracking "large deformations" will, in general, not occur as a cell deforms slowly and continuously. Other advantages will also become apparent when the snake model will be effectively applied to problems such as contour extraction and the tracking (Chapter 3) and shape description (Chapter 5) of deformable objects such as cells.

In this chapter, we first introduce the basic concepts of the snake model in section 2.1. The next two sections, "Dynamics of the Snake in the Continuous Domain" and "Discretization of the Model," constitute a complete summary of the roots of the snake model inspired by the very first papers on the subject by Terzopoulos and Kass *et al.* [145, 144, 78]. Section 2.4 concerning "Solving the Discrete Equations of Motion" is a brief summary of an efficient algorithm for matrix factorization proposed by Benson and Evans and referred to by Terzopoulos [18, 144]. The following section, "Snake Parameters: the Original Model," is a discussion that arose from our experience gained with the original snake model and which has its basis in papers by Terzopoulos and Kass *et al*. The real contribution to the subject though, comes in the last two sections, "The Original Snake Model: Advantages and Limitations" and "Improving the Original Snake Model," and in Appendix C dealing with the "Normalization of the Snake Forces." Finally, Appendix B, entitled "Inserting Discontinuities Along the Snake," is inspired by a paper by Terzopoulos [141].

2.1 Basic Concepts

A snake is a model of a deformable curve or contour (if closed) composed of abstract elastic materials. This elastic contour, in the model, is made of two types of materials: strings and rods. The former make the snake resistant to stretching while the latter make it resistant to bending. Such a deformable curve is activated by making it sensitive to the graph of a function of two variables (*e.g.*, the graph of an image $I(x, y)$) embedded in $\mathbb{R}^3$. Such a graph can be seen as a three-dimensional (3-D) surface, H. The snake is constrained to lie on H under the action of some gravitational force **g**. In other words, a weight is assigned to the snake to make it fall down the slopes of the surface H.

Depending on the nature of the surface considered in this thesis, the snake will be used for different purposes. A typical case is to treat the image data as the surface H. The



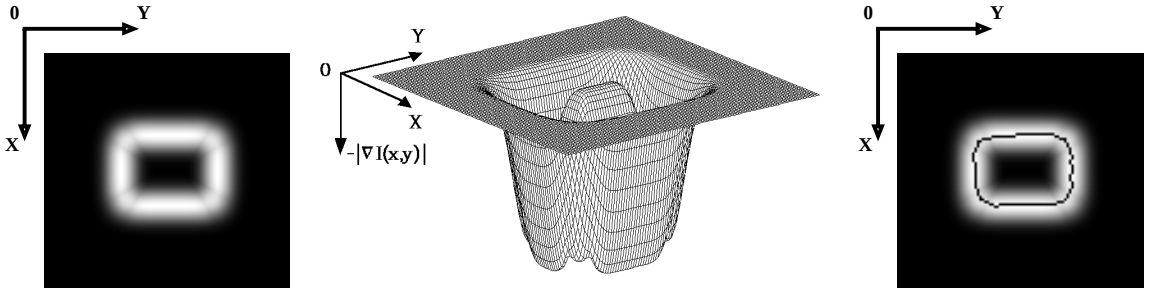
(a) (b)

Figure 2.1: Image seen as a 3-D surface. In (a) is shown the image of intensities of a blurred rectangle. In (b) is shown its representation as a 3-D surface. This representation corresponds to the graph of the function $-I(x, y)$.

image $I(x, y)$ can be represented as a 3-D surface with coordinates (x, y) used as the planar Cartesian reference system. Then, the surface $H(x, y)$ may corresponds to image intensities (*i.e.*, $H = \pm I(x, y)$) or to contrast values (*e.g.*, $H = \nabla I(x, y)$). Once H is determined, the idea is to have the snake lie on the surface allowing it to deform according to the surface topography, while under the influence of **g**. An illustration of how the image may be seen as a 3-D surface is given in Figure 2.1. Throughout this thesis, we will fix the magnitude of the gravitational force to be constant, that is, $|\mathbf{g}| = g = m\mathcal{G}$, where m is the constant mass of the snake and $\mathcal{G}$ is the magnitude of the constant gravitational acceleration.

Other useful types of surfaces can be obtained from range data [59], distance transforms applied to shapes or regions (see Chapter 5), and tangent fields [164, 47, 48]. The snake is positioned on such a surface, and, under the influence of gravity, will seek out valleys or ridges of the surface until it achieves some sort of equilibrium state. Such behavior is obtained by associating with the snake an energy function that depends on the height of the surface at the snake's location (similar to a potential energy). The snake will then attempt to minimize its energy by seeking out local minima in height in a neighborhood around the contour.

A critical step in the process is obtaining a surface where deep valleys correspond to significant events or features of the function represented by that surface. For example, let us again consider the function $H = \pm I(x, y)$, the intensity values of the image. Significant or interesting features in an image are edges and contours, that is, regions of high contrast, or homogeneous regions possessing some uniform characteristic. A proper surface should have valleys positioned at places corresponding to such features as edges, contours or regions boundaries. Therefore, in general, the original data, such as image intensities, will first have to be filtered in an appropriate way to enhance such features. An example of the kind of filtering used on an image to obtain an appropriate surface for an active contour



(a) (b) (c)

Figure 2.2: Example of a surface, obtained by filtering an image, to be used with a snake. In (a) is shown the image of intensities of the gradient of a blurred rectangle. In (b) is shown its representation as a 3-D surface. This representation corresponds to the graph of the function $-|\nabla I(x, y)|$. In (c) is shown the position of a snake superimposed over the image of intensities (a). This snake has crawled down the 3-D surface until it reached the bottom of a valley of the surface. Note that the snake is not perfectly symmetrically positioned because of discretization effects.

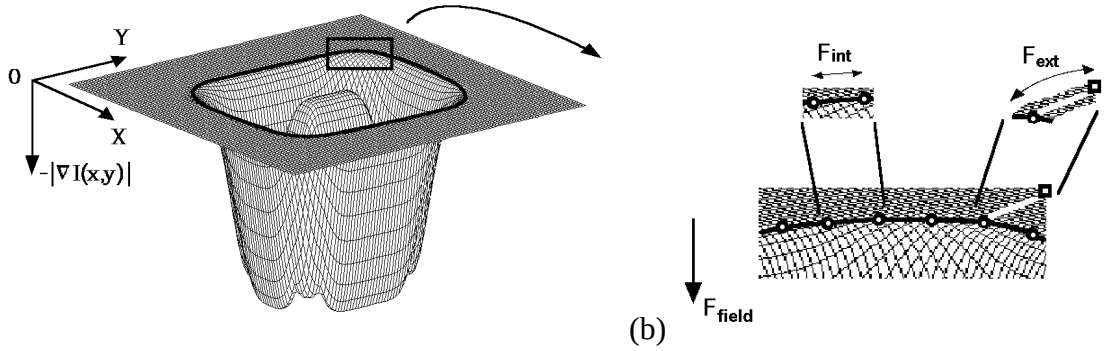
model is given in Figure 2.2. More details and examples will be given in Chapter 3 when applying the snake model to the problems of segmentation and tracking.

An obvious question arises: what if the snake gets "stuck" at some local minimum or valley corresponding to noise or some other undesirable feature in the image? The original snake model [78] provides an "indirect" solution to such problems by permitting the addition of external constraints or forces that will push the snake away from undesirable features. A complementary kind of external force is also provided, which permits the snakes to be pulled towards significant features.

Such a method of imposing external constraints or forces is indirect because it relies on other external mechanisms, such as a pre-analysis of the image data or user interaction to place these constraints near the identified features. This process of imposing external constraints will be clarified with examples of image segmentation (Chapter 3) and shape skeletonization (Chapter 5).

In summary, we have an elastic contour positioned on a 3-D surface, and on which external forces can be imposed. In addition, a mass is assigned to the snake which is then embedded in an uniform *gravitational field*.[1] Furthermore, the surface, H, on which it lies

[1]Gravitational field: A region of space in which a body (*e.g.*, the Earth) exerts a force, g, on another body (*e.g.*, the snake) through space, $\mathbb{R}^3$. We consider in the case of the snake model that the former body is much more massive than the second one, the snake, and that it exerts a constant force of magnitude $g = m\mathcal{G}$ on this snake.



(a) (b)

Figure 2.3: The snake and the forces acting on it. In (a) is shown the potential surface H obtained from the gradient of the blurred rectangle shown in Figure 2.2. Also in (a) is shown a closed snake positioned on the verge of the valley of H (contour in black). A small square window (in (a)) has been selected to illustrate (in (b)) the different forces acting on the snake. In (b) a portion of the snake is shown. Points of the snake are shown as circular dots. The black links between the snake points represent the internal constraints of elasticity that exist between them (*i.e.*, $\mathrm{F}_{\mathrm{int}}$). A square dot represents the position of an external constraint (acting on one snake point here). A white link represents the external force (*i.e.*, $\mathrm{F}_{\mathrm{ext}}$). Finally, $\mathrm{F}_{\mathrm{field}}$ represents the gravitational force which will have a tendency to make the snake crawl down the slopes of H.

can be seen to specify its *gravitational potential energy function*}.[2] Hereafter, in this thesis, we will refer to such a surface by using the term *potential surface*, to make clear how it influences the snake motion. The gravitational field is defined spatially by the surface coordinates (x, y), while the gravitational potential energy values are given by the surface height, $z = H(x, y)$. Depending upon where a snake is positioned on the potential surface, that is, at a certain height, it will have a given gravitational potential energy term. Other influences will also come from the previously mentioned internal and external constraints (Figure 2.3).

We observe that a natural way of causing the snake to move in order to reach a lower gravitational potential energy, by seeking valleys in the potential surface, is to convert potential energy to kinetic energy. The potential energy of the snake is given by the sum of its gravitational potential energy and of the potential energy terms obtained from the internal and external constraints acting on it. Then, the kinetic energy is dissipated by damping, resulting in the snake reaching a new lower equilibrium, that is, lower in terms of potential energy and, thereby, lower in terms of height. By following such a natural description of the

[2]Gravitational potential energy: The work an object at height can do by falling under the influence of a gravity force. The "gravitational potential energy function" is then obtained as the possible states (or heights) this object can assume in a gravitational field. In the case of the snake model these states are fixed by the surface H on which the snake is constrained to lie on.

dynamics of an active contour, the snake model was first described in terms of a Lagrangian formulation of the motion [145, 144]. The dynamics of the snake defined within such a formalism in the continuous domain are described in the next section. A discretization of the model for practical implementation will then follow.

2.2 Dynamics of the Snake in the Continuous Domain

Consider a deformable curve $v(s,t)$, with parameters s (spatial index) and t (time), defined on given open intervals Ω and T respectively. Let us consider this deformable curve to be a function of two variables x and y (*e.g.*, spatial coordinates) having the same parameterization as v:

$$v(s,t) = (x(s,t), y(s,t)) \ : \ s \in \Omega, \ t \in T. \tag{2.1}$$

The potential energy function of the snake, $E_{snake}(v)$, is defined as [78]:

$$E_{snake}(v) = \frac{1}{2} \int_{\Omega} \left[E_{int}(v) + E_{ext}(v) + E_{field}(v) \right] \, ds, \tag{2.2}$$

E_{int} represents the internal potential energy of the snake. It is a function of both bending and stretching forces applied to the snake. E_{ext} gives rise to external constraint forces. E_{field} gives rise to gravitational field forces.

The internal potential energy of the snake, E_{int}, is defined as follows:

$$E_{int}(v(s)) = \omega_1(s) \left| v_s \right|^2 + \omega_2(s) \left| v_{ss} \right|^2, \tag{2.3}$$

where $v_s \equiv \frac{\partial v}{\partial s}$ and $v_{ss} \equiv \frac{\partial^2 v}{\partial s^2}$. The first order term, $\omega_1(s)|v_s|^2$, makes the snake behave like a *string* (*i.e.*, resists stretching), while the second order term, $\omega_2(s)|v_{ss}|^2$, makes the snake behave like a *rod* (*i.e.*, resists bending). The weight $\omega_1(s)$ regulates the tension of the snake, while $\omega_2(s)$ regulates its rigidity. Two types of discontinuities, position or tangent discontinuities, may be introduced along a snake by setting these weights to zero [145, 144]. If we set $\omega_2(s_0) = 0$, thereby removing any rigidity constraint at s_0, a tangent discontinuity can occur at this point, or element, on the snake. Furthermore, if we set both $\omega_1(s_0) = \omega_2(s_0) = 0$, no internal constraint exists at s_0, forcing a position discontinuity at this snake element. Hereafter, in this thesis, we will refer to such points or elements of the snake by using the term *snaxel*, a contraction of the two words "snake element."

The external potential energy, E_{ext}, arises from two complementary types of forces: springs and volcanos. A *spring* force, $f_{spring} = -k_{spring}(\mathbf{P}_1 - \mathbf{P}_2)$, may be created between a fixed point $\mathbf{P}_1(x_1, y_1)$ on the potential surface and a snaxel with coordinates $\mathbf{P}_2(x_2, y_2)$ by adding the energy term, $E_{spring} = -\frac{1}{2} k_{spring}(\mathbf{P}_1 - \mathbf{P}_2)^2$, to E_{ext}. A *volcano* or cone of energy may be created by defining a repulsive force, $f_{volc} = \frac{1}{2} \frac{k_{volc}}{r^2}$, where r represents the radius of the volcano base. The effect of this volcano force is implemented by adding the

cone of energy associated with the force, $E_{volc} = -\frac{k_{volc}}{r}$, to E_{ext}. This cone of energy can be seen as a small conic surface or patch which can be used to fill-in valleys and to create mounts or peaks on the potential surface. Springs are used to force the snake to attach to desirable features, such as curvature extrema of a contour, while volcanos are used to push the snake away from undesirable local minima of the potential field, like those due to noise. Hence we see the complementary nature of the spring and volcano forces, the former being an attractive force, while the latter is a repulsive force.

The potential field energy, E_{field}, can be derived from the classical gravitational potential energy equation (see for example [68]) in a point-by-point fashion as follows:

$$E_{field}(v(s,t)) = \mu \mathcal{G} z(v(s,t)), \tag{2.4}$$

where μ is the constant *mass density* of the snake, $\mathcal{G}$ is the magnitude of the gravitational acceleration discussed previously, and $z(v(s,t))$ is the height or potential value at snaxel s and at time t on the potential surface H.

Given the potential energy function, E_{snake} (equation (2.2)), for a specific initial position, a minimization procedure can be applied to reach a more stable energy state by converting potential energy to kinetic energy and then dissipating this kinetic energy through an energy dissipation function [145, 144].

The minimization process can easily be seen as a problem of classical mechanics. According to the *principle of least action* [68], the motion of a deformable curve under the influence of conservative forces, that is, derivatives of a potential function, during a time interval T, is described by those functions $v(s,t)$ for which the line integral I_L:

$$I_L = \int_T L(v) dt, \tag{2.5}$$

is a minimum. The Lagrangian $L(v)$ is, by definition, given as $L(v) = T(v) - U(v)$, where $T(v)$ expresses the kinetic energy of the curve, v, and $U(v)$ is its potential energy. The kinetic energy function $T(v)$ is defined as:

$$T(v) = \frac{1}{2} \int_\Omega \mu \left| v_t \right|^2 ds, \tag{2.6}$$

where $v_t \equiv \frac{\partial v}{\partial t}$. The potential energy function $U(v)$ is the previously defined $E_{snake}(v)$, that is, the total instantaneous potential energy of the snake. Therefore, combining equations (2.2), (2.5) and (2.6), I_L can be rewritten as the double integral:

$$I_L = \frac{1}{2} \int_T \int_\Omega \left[\mu \left| v_t \right|^2 - E_{int}(v(s)) - E_{ext}(v(s)) - E_{field}(v(s)) \right] ds\, dt\,. \tag{2.7}$$

From the calculus of variations, it is known that an extremum of I_L must satisfy the Euler-Lagrange equations. From equations (2.3), (2.4) and (2.6) we have that the Lagrangian is a function such that:

$$L = L(s,t,x(s,t),y(s,t),x_t,y_t,x_s,y_s,x_{ss},y_{ss})\,. \tag{2.8}$$

In such a case the Euler-Lagrange equations of motion are the following [62]:

$$\begin{aligned}
\frac{\partial}{\partial x}(L) - \frac{\partial}{\partial t}\Big(\frac{\partial}{\partial x_t}(L)\Big) - \frac{\partial}{\partial s}\Big(\frac{\partial}{\partial x_s}(L)\Big) + \frac{\partial^2}{\partial s^2}\Big(\frac{\partial}{\partial x_{ss}}(L)\Big) &= 0, \\
\frac{\partial}{\partial y}(L) - \frac{\partial}{\partial t}\Big(\frac{\partial}{\partial y_t}(L)\Big) - \frac{\partial}{\partial s}\Big(\frac{\partial}{\partial y_s}(L)\Big) + \frac{\partial^2}{\partial s^2}\Big(\frac{\partial}{\partial y_{ss}}(L)\Big) &= 0.
\end{aligned} \tag{2.9}$$

These equations describe the motion of a *conservative* system, that is, a system where all the forces acting on it are derived from a potential function. With such forces, potential energy is transformed solely into kinetic energy, leading to a constant increase of speed. In order to reach a new lower equilibrium, in terms of potential energy, at a rest state ($v_t = 0$), kinetic energy must be dissipated by damping. A natural and simple way of describing such behavior is by including within the equations of motion, obtained for a conservative system (equations (2.9)), the effect of additive frictional forces proportional to snaxel velocities, that is, forces not derived from a potential function or, equivalently, non-conservative. Frictional forces of this type may be derived from *Rayleigh's dissipation function* [68]:

$$D(v_t) = \frac{1}{2}\int_\Omega \gamma\,|v_t|^2\,ds, \tag{2.10}$$

where γ is the constant damping density or viscosity factor. Then, the frictional forces acting on the snake are derived as:

$$\begin{aligned}
D_f(x_t) &= -\frac{\partial}{\partial x_t}\Big(D(v_t)\Big) = -\gamma x_t, \\
D_f(y_t) &= -\frac{\partial}{\partial y_t}\Big(D(v_t)\Big) = -\gamma y_t.
\end{aligned} \tag{2.11}$$

Incorporating these forces into the Euler-Lagrange equations of motion, gives:

$$\begin{aligned}
\frac{\partial}{\partial x}(L) - \frac{\partial}{\partial t}\Big(\frac{\partial}{\partial x_t}(L)\Big) - \frac{\partial}{\partial s}\Big(\frac{\partial}{\partial x_s}(L)\Big) + \frac{\partial^2}{\partial s^2}\Big(\frac{\partial}{\partial x_{ss}}(L)\Big) &= -\gamma x_t, \\
\frac{\partial}{\partial y}(L) - \frac{\partial}{\partial t}\Big(\frac{\partial}{\partial y_t}(L)\Big) - \frac{\partial}{\partial s}\Big(\frac{\partial}{\partial y_s}(L)\Big) + \frac{\partial^2}{\partial s^2}\Big(\frac{\partial}{\partial y_{ss}}(L)\Big) &= -\gamma y_t.
\end{aligned} \tag{2.12}$$

Using equations (2.2), (2.3) and (2.6), we can replace the Lagrangian L by $T - U$ as follows:

$$L = T - U = \frac{1}{2}\int_{\Omega}\left[\mu\left|v_t\right|^2 - \omega_1(s)\left|v_s\right|^2 - \omega_2(s)\left|v_{ss}\right|^2 - E_{ext} - E_{field}\right] ds. \qquad (2.13)$$

Then, replacing L in equations (2.12) by this last equation (2.13), and after a few manipulations, these equations become:

$$\begin{aligned} \mu x_{tt} + \gamma x_t - \frac{\partial}{\partial s}\Big(\omega_1(s)x_s\Big) - \frac{\partial^2}{\partial s^2}\Big(\omega_2(s)x_{ss}\Big) &= -\frac{1}{2}\left(E_{ext_x}(v) + E_{field_x}(v)\right), \\ \mu y_{tt} + \gamma y_t - \frac{\partial}{\partial s}\Big(\omega_1(s)y_s\Big) - \frac{\partial^2}{\partial s^2}\Big(\omega_2(s)y_{ss}\Big) &= -\frac{1}{2}\Big(E_{ext_y}(v) + E_{field_y}(v)\Big) \end{aligned} \qquad (2.14)$$

where $x_{tt} \equiv \frac{\partial x^2}{\partial t^2}$, $y_{tt} \equiv \frac{\partial y^2}{\partial t^2}$, $E_{ext_x} \equiv \frac{\partial}{\partial x}(E_{ext})$, $E_{ext_y} \equiv \frac{\partial}{\partial y}(E_{ext})$, $E_{field_x} \equiv \frac{\partial}{\partial x}(E_{field})$ and $E_{field_y} \equiv \frac{\partial}{\partial y}(E_{field})$. Note that associated with such differential equations are appropriate initial and boundary conditions, at $t = 0$ and at the extremities of the interval Ω, respectively. We will address this issue in the next section when discretizing the equations of motion.

Let us summarize the way these equations model the dynamics of the snake. On the left-hand-sides (LHS's), the first two terms represent inertial (μv_{tt}) and damping (γv_t) influences; the former is proportional to acceleration and the second is proportional to speed, as one would expect. The next two terms represent the elasticity forces between snaxels in terms of tension ($\omega_1 v_s$) and rigidity ($\omega_2 v_{ss}$). We will see that the former is proportional to the spacing between snaxels, while the later is proportional to curvature. On the right-hand-sides (RHS's), the first term represents the influence of the external constraints as a function of the first spatial derivatives of the energy E_{ext} (*i.e.*, giving rise to external forces). The second term represents the influence of the potential surface as a function of its negative slope in the x and y directions. The latter will force the snake to follow the potential surface topography in the direction of highest negative slope in a fashion similar to a *steepest descent* technique. Therefore, the LHS's represent the intrinsic forces acting on or within the snake, while the RHS's represent the extrinsic forces which are independent of the snake's nature.

The description of the dynamics of the snake model in the continuous domain is now complete. Only the solution of the differential equations of motion remains. However, for the purposes of practical implementation, the model must first be discretized. This is the subject of the next section.

2.3 Discretization of the Model

Because today's computers are mostly based on digital devices, any problem which is to be solved using the computational power of these computers must be specified in a discrete

fashion. For the snake model, the space and time domains, the external forces, the snake itself and the equations of motion must be discretized so that the model is compatible with the digital nature of the computer.

Let us first consider the discretization of a region E of the space, $\mathbb{R}^3$, in which the potential surface, H, is defined. This region, E, can be discretized by defining three discrete[3] sets, $\overline{X}$ and $\overline{Y}$, for the spatial coordinates, and $\overline{Z}$ for the field values or valid potential surface heights. This permits the confinement of the triplet values, $(x, y, z) \in E$, to a restricted set of values, $(\overline{x}, \overline{y}, \overline{z}) \in \overline{E}$, where $\overline{E}$ represents the restricted discrete domain. The sets, $\overline{X}$ and $\overline{Y}$, consist of integer values varying from 0 to some given maximum indexing values, $M_{\overline{X}}$ and $M_{\overline{Y}}$, respectively: $\{\overline{x} \in \overline{X} \ : \overline{X} = \{0, \ldots, M_{\overline{X}}\}\}$ and $\{\overline{y} \in \overline{Y} \ : \overline{Y} = \{0, \ldots, M_{\overline{Y}}\}\}$. These two sets define indices on the potential surface. A third set is then defined for the surface values. It consists of real numbers varying from a given minimum, $\overline{Z}_{min}$, to a given maximum, $\overline{Z}_{max}$, such that $\{\overline{z} \in \overline{Z} \ : \overline{Z} = \{\overline{Z}_{min}, \ldots, \overline{Z}_{max}\}\}$. Examples of such discrete domains $\overline{E}$ that will be used in this thesis are the *image domain*, $\mathcal{I}$, containing digital intensity images ($I(\overline{x}, \overline{y})$) and their filtered versions, and the *distance transform domain*, $\mathcal{DT}$, containing digital distance transforms of 2-D regions or objects ($DT(\overline{x}, \overline{y})$).

Like the region E, the external forces must also be spatially discretized. This is done by using the same discrete spatial coordinates ($(\overline{x}, \overline{y}) \ : \ \overline{x} \in \overline{X}$ and $\overline{y} \in \overline{Y}$) when evaluating the spring and volcano forces.

Unlike the previous cases, the snake must be discretized in not one, but two domains: space and time. The spatial discretization is done by regularly tessellating the interval Ω into $M_{\overline{s}}$ nodes (*i.e.*, $M_{\overline{s}}$ snaxels), leading to a discrete set $\overline{\Omega}$ ($\{\overline{s} \in \overline{\Omega} : \overline{\Omega} = \{0, \ldots, M_{\overline{s}} - 1\}\}$). We use the symbol $\Delta\overline{s}$ to represent the distance or spatial step between successive snaxels. The time discretization is achieved by considering a discrete time interval, $\overline{T}$, defined to start at time 0. It consists of values $\overline{t}$ regularly separated by a constant time-step, $\Delta\overline{t}$. For example, $\Delta\overline{t}$ may represent the inter-frame time interval when analyzing sequences of images. Theoretically we can activate a snake for an infinite period of time; therefore $\overline{T}$ is the open set: $\{\overline{t} \in \overline{T} : \overline{T} = \{0, \Delta\overline{t}, 2\Delta\overline{t}, \ldots\}\}$. Having discretized the two parameters s and t which describe the snake (equation (2.1)), its discrete version can now be stated as follows:

$$\overline{v}(\overline{s}, \overline{t}) = (\overline{x}(\overline{s}, \overline{t}), \overline{y}(\overline{s}, \overline{t})). \tag{2.15}$$

This spatially relates the discrete snake to the same grid $(\overline{x}, \overline{y})$ used to discretize the domain E and the external forces.

The next step involves the discretization of the equations of motion of the snake (equations (2.14)). As previously, a discretization in both space and time is required. For such differential equations we use the *finite difference method* as the discretization technique to approximate the first and second order derivatives in equations (2.14), in both space and

[3]Throughout this thesis, barred symbols will refer to discretized variable (*e.g.*, $(x, y) \rightarrow (\overline{x}, \overline{y})$).

time.

First, let us consider the time derivatives (the first two terms on the LHS's of equations (2.14)). Since we do not have *a priori* knowledge of the future positions of the snake, we will position ourselves in time at $t = \overline{t} - \Delta\overline{t}$, that is, at a known position, looking for the next new position, at time $t = \overline{t}$, that will present itself as a solution to equations (2.14). This has the advantage of allowing us to use *central differences* in time, centered at $t - \Delta\overline{t}$, when evaluating time derivatives, as follows:

$$\begin{aligned} \mu v_{tt}(s, \overline{t} - \Delta\overline{t}) &\approx \frac{\mu}{(\Delta\overline{t})^2}\left[\overline{v}(\overline{s}, \overline{t}) - 2\overline{v}(\overline{s}, \overline{t} - \Delta\overline{t}) + \overline{v}(\overline{s}, \overline{t} - 2\Delta\overline{t})\right], \\ \gamma v_t(s, \overline{t} - \Delta\overline{t}) &\approx \frac{\gamma}{2\Delta\overline{t}}\left[\overline{v}(\overline{s}, \overline{t}) - \overline{v}(\overline{s}, \overline{t} - 2\Delta\overline{t})\right], \end{aligned} \tag{2.16}$$

where the two initial positions at $\overline{t} = -\Delta\overline{t}$ and $\overline{t} = -2\Delta\overline{t}$ must be given. Following [144, 78] we consider constant time-steps and replace $\Delta\overline{t}$ by 1 to simplify equations (2.16). We must keep in mind that to achieve good numerical stability, relatively small deformations of the snake are needed. This is the same as having a relatively small time-step $\Delta\overline{t}$, keeping the numerical errors (proportional to $(\Delta\overline{t})^2$ in equations (2.16)) relatively small. Therefore, replacing $\Delta\overline{t}$ by 1 and combining the two equations (2.16) leads to the following:

$$\mu v_{tt} + \gamma v_t \approx \overline{v}(\overline{s}, \overline{t})\left[\mu + \frac{\gamma}{2}\right] + \overline{v}(\overline{s}, \overline{t} - 1)\left[-2\mu\right] + \overline{v}(\overline{s}, \overline{t} - 2)\left[\mu - \frac{\gamma}{2}\right], \tag{2.17}$$

where the two initial positions $\overline{v}(\overline{s}, -1)$ and $\overline{v}(\overline{s}, -2)$ must be given. From this equation, we can conclude that the discretization of time derivatives has the effect of averaging the combined influences of inertia and damping over the present and two previous snake positions. This takes effect over all snaxels.

We now consider the derivatives in the spatial coordinates, s, of the snake, for the last two terms on LHS's of equations (2.14), at time $t = \overline{t}$, that is, at the new snake position. Assuming that we have access to the local neighborhood around each snaxel, we can use forward, backward or central differences, or even combinations of these. For reasons of symmetry and direct local influence, we combine forward and backward differences when approximating $\frac{\partial}{\partial s}\big(\omega_1(s)v_s\big)$, and use central differences for $\frac{\partial^2}{\partial s^2}\big(\omega_2(s)v_{ss}\big)$. If we make another choice, such as using only central differences, we may not necessarily have direct relations or links between snaxels as a function of ω_1, or we may not have symmetric relations. We also assume that the snake forms a loop or closed curve in order to have simple cyclic boundary conditions. The insertion of position discontinuities, considered in Appendix B, will permit us to model open active curves. We therefore obtain the following discrete equation:

$$-\frac{\partial}{\partial s}\big(\omega_1(s)v_s(s,t)\big) + \frac{\partial^2}{\partial s^2}\big(\omega_2(s)v_{ss}(s,t)\big) \approx$$

$$
\begin{aligned}
&\frac{1}{(\Delta\overline{s})^2}\left\{\omega_1(\overline{s})\left[\overline{v}(\overline{s},\overline{t})-\overline{v}(\overline{s}-\Delta\overline{s},\overline{t})\right]\right\} \quad + \\
&\frac{1}{(\Delta\overline{s})^2}\left\{\omega_1(\overline{s}+\Delta\overline{s})\left[\overline{v}(\overline{s},\overline{t})-\overline{v}(\overline{s}+\Delta\overline{s},\overline{t})\right]\right\} \quad + \\
\frac{1}{(\Delta\overline{s})^4}&\left\{\omega_2(\overline{s}-\Delta\overline{s})\left[\overline{v}(\overline{s}-2\Delta\overline{s},\overline{t})-2\overline{v}(\overline{s}-\Delta\overline{s},\overline{t})+\overline{v}(\overline{s},\overline{t})\right]\right\} \quad - \\
&\frac{2}{(\Delta\overline{s})^4}\left\{\omega_2(\overline{s})\left[\overline{v}(\overline{s}-\Delta\overline{s},\overline{t})-2\overline{v}(\overline{s},\overline{t})+\overline{v}(\overline{s}+\Delta\overline{s},\overline{t})\right]\right\} \quad + \\
\frac{1}{(\Delta\overline{s})^4}&\left\{\omega_2(\overline{s}+\Delta\overline{s})\left[\overline{v}(\overline{s},\overline{t})-2\overline{v}(\overline{s}+\Delta\overline{s},\overline{t})+\overline{v}(\overline{s}+2\Delta\overline{s},\overline{t})\right]\right\} \quad , \qquad (2.18)
\end{aligned}
$$

where the cyclic boundary conditions require that $\overline{v}(-2\Delta\overline{s},\overline{t}) = \overline{v}(M_{\overline{s}}-2\Delta\overline{s},\overline{t})$, $\overline{v}(-\Delta\overline{s},\overline{t}) = \overline{v}(M_{\overline{s}}-\Delta\overline{s},\overline{t})$, $\overline{v}(M_{\overline{s}},\overline{t}) = \overline{v}(0,\overline{t})$, $\overline{v}(M_{\overline{s}}+\Delta\overline{s},\overline{t}) = \overline{v}(\Delta\overline{s},\overline{t})$, $\omega_2(-\Delta\overline{s}) = \omega_2(M_{\overline{s}}-\Delta\overline{s})$, $\omega_2(M_{\overline{s}}) = \omega_2(0)$ and $\omega_1(M_{\overline{s}}) = \omega_1(0)$. As in the case of the time derivatives, we consider the spatial step-size, $\Delta\overline{s}$, as constant and replace it by 1 to simplify equation (2.18) [144, 78]. However, to achieve a good approximation of the derivatives, we must keep in mind that the distance between snaxels, $\Delta\overline{s}$, should remain small and relatively constant. This will keep the numerical errors, proportional to $(\Delta\overline{s})^2$ for the approximation of $\frac{\partial}{\partial s}\Big(\omega_1(s)v_s\Big)$ and $(\Delta\overline{s})^4$ for the approximation of $\frac{\partial^2}{\partial s^2}\Big(\omega_2(s)v_{ss}\Big)$, relatively small. Therefore, replacing $\Delta\overline{s}$ by 1, and rewriting equations (2.18) in terms of $\overline{v}$ leads to the following:

$$
\begin{aligned}
-\frac{\partial}{\partial s}\Big(\omega_1(s)v_s(s,t)+\frac{\partial^2}{\partial s^2}\Big(\omega_2(s)v_{ss}(s,t)\Big)\Big) \quad &\approx \\
\overline{v}(\overline{s}-2,\overline{t})\,\omega_2(\overline{s}-1) \quad &+ \\
\overline{v}(\overline{s}-1,\overline{t})\left[-2\omega_2(\overline{s}-1)-2\omega_2(\overline{s})-\omega_1(\overline{s})\right] \quad &+ \\
\overline{v}(\overline{s},\overline{t})\left[\omega_2(\overline{s}-1)+4\omega_2(\overline{s})+\omega_2(\overline{s}+1)+\omega_1(\overline{s})+\omega_1(\overline{s}+1)\right] \quad &+ \\
\overline{v}(\overline{s}+1,\overline{t})\left[-2\omega_2(\overline{s})-2\omega_2(\overline{s}+1)-\omega_1(\overline{s}+1)\right] \quad &+ \\
\overline{v}(\overline{s}+2),\overline{t})\,\omega_2(\overline{s}+1) \quad &, \qquad (2.19)
\end{aligned}
$$

where $\overline{v}(-2,\overline{t}) = \overline{v}(M_{\overline{s}}-2,\overline{t})$, $\overline{v}(-1,\overline{t}) = \overline{v}(M_{\overline{s}}-1,\overline{t})$, $\overline{v}(M_{\overline{s}},\overline{t}) = \overline{v}(0,\overline{t})$, $\overline{v}(M_{\overline{s}}+1,\overline{t}) = \overline{v}(1,\overline{t})$, $\omega_2(-1) = \omega_2(M_{\overline{s}}-1)$, $\omega_2(M_{\overline{s}}) = \omega_2(0)$ and $\omega_1(M_{\overline{s}}) = \omega_1(0)$. With such boundary conditions, equation (2.19) holds for all $\overline{s}$ and can be rewritten as a system of linear equations for all $\overline{s}$ in a more convenient matrix form:

$$
-\frac{\partial}{\partial s}\Big(\omega_1(s)v_s(s,t)\Big)+\frac{\partial^2}{\partial s^2}\Big(\omega_2(s)v_{ss}(s,t)\Big) \approx K\,\overline{\mathbf{V}}(\overline{s},\overline{t})\,, \qquad (2.20)
$$

where K, the *stiffness matrix*, representing all internal elasticity relations of the snake, is defined for $\overline{s} = 0,\ldots,\ M_{\overline{s}}-1$, as follows:

$$K = \begin{pmatrix} c_0 & b_0 & a_0 & & & & a_{M_{\overline{s}}-2} & b_{M_{\overline{s}}-1} \\ b_0 & c_1 & b_1 & a_1 & & & & a_{M_{\overline{s}}-1} \\ a_0 & b_1 & c_2 & b_2 & a_2 & & & \\ & a_1 & b_2 & c_3 & b_3 & a_3 & & \\ & & & & \cdots & & & \\ & & & a_{M_{\overline{s}}-5} & b_{M_{\overline{s}}-4} & c_{M_{\overline{s}}-3} & b_{M_{\overline{s}}-3} & a_{M_{\overline{s}}-3} \\ a_{M_{\overline{s}}-2} & & & & a_{M_{\overline{s}}-4} & b_{M_{\overline{s}}-3} & c_{M_{\overline{s}}-2} & b_{M_{\overline{s}}-1} \\ b_{M_{\overline{s}}-1} & a_{M_{\overline{s}}-1} & & & & a_{M_{\overline{s}}-3} & b_{M_{\overline{s}}-2} & c_{M_{\overline{s}}-1} \end{pmatrix}. \quad (2.21)$$

This matrix is *symmetric* and *pentadiagonal* which will prove to be useful when solving the discrete equations of motion of the snake. The following identities are used to relate K to equation (2.19):

$$\begin{aligned} a_{\overline{s}} &= \omega_2(\overline{s}+1), \\ b_{\overline{s}} &= -2\omega_2(\overline{s}) - 2\omega_2(\overline{s}+1) - \omega_1(\overline{s}+1), \\ c_{\overline{s}} &= \omega_2(\overline{s}-1) + 4\omega_2(\overline{s}) + \omega_2(\overline{s}+1) + \omega_1(\overline{s}) + \omega_1(\overline{s}+1). \end{aligned} \quad (2.22)$$

Finally, the vector $\overline{\mathbf{V}}$ contains all snaxel positions at time $\overline{t}$:

$$\overline{\mathbf{V}}(\overline{s},\overline{t}) = \left(\overline{v}(0,\overline{t})\, \overline{v}(1,\overline{t})\, \ldots\, \overline{v}(M_{\overline{s}}-1,\overline{t})\right)^T. \quad (2.23)$$

where T stands for the *transpose*. From such a system of equations we can observe the effect of discretization in terms of first and second order differences, on the types of relations linking the snaxels. Each snaxel ($\overline{s}$) is directly linked to at most its four direct neighbors ($\overline{s}-2,\ \overline{s}-1,\ \overline{s}+1$ and $\overline{s}+2$; see Appendix B). The weights given to each link, in terms of ω_1 and ω_2, are symmetrically distributed around each snaxel. Such properties of direct local influence and symmetry are desirable to obtain an isotropic behavior of the snake and to be able to apply fast methods for solving such a system of equations. Also, due to the local nature of the links between snaxels, discontinuities may easily be introduced along the snake by breaking these links. With the kinds of links we use, that is, ω_1 and ω_2, two useful types of discontinuities can be introduced: tangent and position breaks. A *tangent discontinuity* is enforced by breaking the rigidity links ($\omega_2 = 0$) at a given snaxel. A *position discontinuity* is enforced by breaking both the rigidity and the tension links ($\omega_2 = \omega_1 = 0$) between two given snaxels. A tangent discontinuity permits the snake to take the form of a corner, while a position discontinuity permits the creation of a snake which forms an open curve. The introduction of such discontinuities can be achieved by simply modifying certain entries in the matrix K corresponding to the broken links. A detailed analysis of how to retrieve the matrix entries that need to be modified is presented in Appendix B.

We are still left with the discretization of the RHS's of equations (2.14). Since we have already considered the discretization of the external forces and the potential surface, it remains to approximate the directional derivatives in x and y. Again, we do not have *a priori* knowledge of the future positions of the snake, so we can at best approximate these derivatives by the last known snake position, that is, at time $\overline{t}-1$. We will consider how to efficiently implement these approximations in subsequent sections. Therefore, we use the following notation for the discrete version of the RHS's of equations (2.14):

$$\begin{aligned} E_{ext_x}(v(s,t)) + E_{field_x}(v(s,t)) &\approx E_{ext_{\overline{x}}}(\overline{v}(\overline{s},\overline{t}-1)) + E_{field_{\overline{x}}}(\overline{v}(\overline{s},\overline{t}-1)), \\ E_{ext_y}(v(s,t)) + E_{field_y}(v(s,t)) &\approx E_{ext_{\overline{y}}}(\overline{v}(\overline{s},\overline{t}-1)) + E_{field_{\overline{y}}}(\overline{v}(\overline{s},\overline{t}-1)). \end{aligned} \quad (2.24)$$

We can now write the discrete version of the equations of motion for all snaxels. This results in a set of two systems of $M_{\overline{s}}$ linear equations. Using equations (2.17), (2.20) and (2.24) we can write these two systems of equations as follows:

$$\begin{aligned} A\,\overline{\mathbf{X}}(\overline{s},\overline{t}) &= B_{\overline{x}}(\overline{s},\overline{t}-1,\overline{t}-2)\,, \\ A\,\overline{\mathbf{Y}}(\overline{s},\overline{t}) &= B_{\overline{y}}(\overline{s},\overline{t}-1,\overline{t}-2)\,, \end{aligned} \quad (2.25)$$

where on the LHS's we use the following identities:

$$\begin{aligned} A &= \left[\left(\frac{\gamma}{2}+\mu\right)\mathbf{I}\right] + K\,, \\ \overline{\mathbf{X}}(\overline{s},\overline{t}) &= (\overline{x}(0,\overline{t})\,\overline{x}(1,\overline{t})\,\ldots\,\overline{x}(M_{\overline{s}}-1,\overline{t}))^T\,, \\ \overline{\mathbf{Y}}(\overline{s},\overline{t}) &= (\overline{y}(0,\overline{t})\,\overline{y}(1,\overline{t})\,\ldots\,\overline{y}(M_{\overline{s}}-1,\overline{t}))^T\,, \end{aligned}$$

with $\mathbf{I}$ denoting the identity matrix. Note that the matrix A has the same off-diagonal elements as K; the only differences occur on the principal diagonal where the term $[\frac{\gamma}{2}+\mu]$ is added to the diagonal elements of K. Therefore, A like K possesses the properties of being symmetric and pentadiagonal. Furthermore, A can be shown to be *positive definite*.[4] On the RHS's of equations (2.25) the following identities are used:

$$\begin{aligned} B_{\overline{x}}(\overline{s},\overline{t}-1,\overline{t}-2) &= \\ -\frac{1}{2}\left(E_{ext_{\overline{x}}}(\overline{v}(\overline{s},\overline{t}-1)) + E_{field_{\overline{x}}}(\overline{v}(\overline{s},\overline{t}-1))\right) &+ \\ [2\mu]\;\overline{x}(\overline{s},\overline{t}-1) + \left[\frac{\gamma}{2}-\mu\right]\;\overline{x}(\overline{s},\overline{t}-2) &\quad , \end{aligned}$$

[4]The easiest way to prove that A is positive definite is by showing that $\overline{\mathbf{X}}^{\mathrm{T}} A\,\overline{\mathbf{X}} > 0$ and $\overline{\mathbf{Y}}^{\mathrm{T}} A\,\overline{\mathbf{Y}} > 0$ hold for any $\overline{\mathbf{X}} \neq 0$ and $\overline{\mathbf{Y}} \neq 0$, respectively.

$$
\begin{aligned}
B_{\overline{y}}(\overline{s},\overline{t}-1,\overline{t}-2) \quad &= \\
-\frac{1}{2}\left(E_{ext_{\overline{y}}}(\overline{v}(\overline{s},\overline{t}-1)) + E_{field_{\overline{y}}}(\overline{v}(\overline{s},\overline{t}-1))\right) \quad &+ \\
[2\mu]\ \overline{y}(\overline{s},\overline{t}-1) + \left[\frac{\gamma}{2}-\mu\right]\ \overline{y}(\overline{s},\overline{t}-2) \quad &,
\end{aligned}
$$

with $\overline{s} = 0, \ \ldots\ , M_{\overline{s}} - 1$. Note that the RHS's depend on two prior snake positions, at $\overline{t}-1$ and $\overline{t}-2$. Then the solution of the system of equations (2.25) can be obtained by retrieving the new snake position at time $\overline{t}$:

$$
\overline{\mathbf{X}} = A^{-1} B_{\overline{x}}\ ,\ \overline{\mathbf{Y}} = A^{-1} B_{\overline{y}}\,.
$$

How to solve equations (2.25) is considered in the next section.

2.4 Solving the Discrete Equations of Motion

Equations (2.25) can be solved in linear time ($\mathcal{O}(M_{\overline{s}})$) by using efficient factorization techniques to obtain $A^{-1}\ B_{\overline{x}}$ and $A^{-1}\ B_{\overline{y}}$ indirectly. These techniques take full advantage of the properties of matrix A being pentadiagonal, symmetric and positive definite. For example, the algorithm of Benson and Evans [18, 144] proves to be useful for our problem. The basic idea of their algorithm consists of using certain matrix A properties to factor it into $A = DLUD$, where D is a diagonal matrix, L is a lower triangular matrix and $U = L^T$ is an upper triangular matrix. The system of linear equations $A\ \overline{\mathbf{X}} = B_{\overline{x}}$ (similarly for $\overline{\mathbf{Y}}$)[5] becomes $DLUD\overline{\mathbf{X}} = B_{\overline{x}}$ which can be directly reduced to $LUD\overline{\mathbf{X}} = D^{-1}B_{\overline{x}} = Q$. The next step consists of solving for $Z = D\overline{\mathbf{X}}$ in the system $LUZ = Q$. This is done in two stages by first doing a forward substitution for $P = UZ = L^{-1}Q$. Then a backward substitution for $Z = U^{-1}P$ can be performed. The final step consists of solving for the system $\overline{\mathbf{X}} = D^{-1}Z$. Such a factorization scheme proves to be most efficient for iterative processes where the matrix A does not vary at each iteration [18]. This will be the case for the snake model in most situations, where the mass density μ and the damping density γ are considered to be constant, and where the elasticity constraints or links $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$ will be set to be constant most of the time (section 2.5). In the case where the matrix A changes with each iteration, for example, by varying the elasticity properties of the snake for every $\Delta\overline{t}$, the Choleski factorization[6] ($A = LU$) will be more efficient [18].

[5] In the following sections, we refer to the equations of motion of the snake often by solely using variables in $\overline{x}$. The equations in $\overline{y}$ are simply obtained by replacing $\overline{x}$ by $\overline{y}$ and $\overline{X}$ by $\overline{Y}$.

[6] The complexity of the Choleski factorization is also linear in time ($\mathcal{O}(M_{\overline{s}})$). See, for example, Jacobs [77] for a complete discussion of this technique.

We have now covered most aspects concerning the implementation of the snake model. But there remains one important aspect to be dealt with in order to obtain a desirable behavior for the snake (*i.e.*, stable, accurate, fast): how to fix or set values for the different intrinsic and extrinsic parameters γ, μ, ω_1, ω_2, $E_{ext_{\overline{x}}}$ and $E_{field_{\overline{x}}}$? We discuss this important subject in the following section.

2.5 Snake Parameters: The Original Model

In this section we consider how to fix the different snake parameters in the discrete equations of motion (2.25) following the original ideas of Kass *et al.* and Terzopoulos [78, 145]. In subsequent sections, problems emerging from these original descriptions of the snake model will be considered and solutions to them will be proposed. For now, let us examine the original framework in detail.

By observing the discrete equations of motion (equations (2.25)), one notes that μ and γ are primarily important to implement a memory effect by averaging the actual snake position $\overline{\mathbf{V}}(\overline{t})$ with the two prior positions $\overline{\mathbf{V}}(\overline{t}-1)$ and $\overline{\mathbf{V}}(\overline{t}-2)$. Both μ and γ are positive or null real numbers: $\mu, \gamma \geq 0 \ : \ \mu, \gamma \in \mathbb{R}$. By varying their combined effects, more or less weight will be put on the existing or prior positions, or on the differences between these positions. By setting both of them to zero, the memory effect is lost: the snake moves solely on the basis of its elasticity constraints and the extrinsic forces. It is worth noting that both μ and γ are set to be constant in the original snake model in order to simplify the equations of motion of the snake and the factorization stage. This constraint could be removed to define a more general active contour model. For example, γ could be a function of s to slow down snaxels one by one depending on their individual speed. The main drawback of such a technique would be that a less efficient factorization method would have to be used. Also, since this would imply the introduction of other derivatives in the equations of motion (equations (2.14)), it is not clear that it would help the stability of the model since we approximate derivatives by differences, a further source of instability. Therefore, it is in general better, from a numerical point of view, to keep μ and γ constant.

Let us consider the other two parameters which provide some control of the intrinsic behavior of the snake: the tension and rigidity weights, ω_1 and ω_2. To understand the effect of varying these weights, consider the internal potential energy $E_{int}(v(s)) = \omega_1(s)|v_s|^2 + \omega_2(s)|v_{ss}|^2$ (equation (2.3)). By minimizing E_{int} over all snaxels, that is, minimizing $I_{int} = \int_\Omega E_{int} ds$, we can deduce how ω_1 and ω_2 affect the behavior of the snake. Let us consider each effect separately:

$$I_{\omega_1} = \int_\Omega \omega_1(s) \, |v_s|^2 \, ds \ , \ I_{\omega_2} = \int_\Omega \omega_2(s) \, |v_{ss}|^2 \, ds \, .$$

where $I_{\omega_1} + I_{\omega_2} = I_{int}$. If $\omega_1(s) > 0$, then $I_{\omega_1} > 0$ and to reduce it, $|v_s|^2$ $(= x_s^2 + y_s^2)$ must be reduced. In the discrete domain, this means that $\Delta\overline{v}^2$ must be reduced or that

the average distance between snaxels must be decreased; thus the snake shrinks. On the contrary if $\omega_1(s) < 0$, then $I_{\omega_1} < 0$ and to reduce it, $|v_s|^2$ has to increase. Therefore, in the discrete domain $\Delta\overline{v}^2$ must increase or, equivalently, the average distance between snaxels has to increase; here the snake dilates. Let us now consider I_{ω_2}. If $\omega_2(s) > 0$ then $I_{\omega_2} > 0$ and to reduce it, $|v_{ss}|^2$ ($= x_{ss}^2 + y_{ss}^2 \geq 0$) must be reduced. This implies that both $|x_{ss}|$ and $|y_{ss}|$ have to be reduced or equivalently that the *curvature* of the curve v is reduced.[7]} If $\omega_2(s) < 0$, then $I_{\omega_2} < 0$ and to reduce it, $|v_{ss}|^2$ must increase. In this case we can hardly predict the behavior of the snake since even if the sum $x_{ss}^2 + y_{ss}^2$ increases, it does not describe the behavior of the curvature of v in a trivial way.

The previous analysis provides us with some understanding of how $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$ may affect the snake behavior. Still we need some sort of rules to fix the values of these weights. Since $\omega_1(\overline{s})$ is linked to the notion of distance between snaxels, Terzopoulos [145] has proposed to associate with the snake a point-wise metric function $\mathcal{L}(\overline{s})$ used to prescribe the "natural arc length," that is, the "desirable" step-size $\Delta\overline{s}$ between each snaxel. Then, the tension weights $\omega_1(\overline{s})$ can be fixed with respect to $\mathcal{L}(\overline{s})$ as follows:

$$\omega_1(\overline{s}) = \Delta\overline{s}(\overline{s}) - \mathcal{L}(\overline{s})\,, \tag{2.26}$$

where $\Delta\overline{s}(\overline{s}) = \sqrt{\Delta\overline{x}^2 + \Delta\overline{y}^2}$ is the actual distance between snaxels. Therefore, when snaxels are too far apart (*i.e.*, $\Delta\overline{s}(\overline{s}) > \mathcal{L}(\overline{s})$) $\omega_1(\overline{s})$ is positive and the snake should shrink. For snaxels too close to each other, $\omega_1(\overline{s})$ is negative and the snake expands. Also proposed by Terzopoulos [145] is a point-wise metric function $\mathcal{C}(\overline{s})$ used to prescribe the "natural curvature" at each snaxel. Then, similarly to the tension, the rigidity weight $\omega_2(\overline{s})$ can be fixed with respect to $\mathcal{C}(\overline{s})$ as follows:

$$\omega_2(\overline{s}) = \overline{k}(\overline{s}) - \mathcal{C}(\overline{s})\,, \tag{2.27}$$

where $\overline{k}(\overline{s})$ is the actual discrete curvature at snaxel $\overline{s}$. There are advantages as well as disadvantages that arise from such techniques used to set $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$. These will be considered in sections 2.6.1 and 2.6.3.

A final subject that must be examined with respect to the parameters $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$ is the introduction of discontinuities. In previous paragraphs, we mentioned that two types of discontinuities were defined: tangent and position discontinuities (section 2.3). The former corresponds to fixing $\omega_2(\overline{s})$ to 0 at snaxel $\overline{s}$, while the second corresponds to fixing both $\omega_2(\overline{s})$ and $\omega_1(\overline{s})$ to 0. Two important questions remain: where to position discontinuities and how to choose these locations? Since a position discontinuity is primarily used for obtaining an open active curve, the particular snaxel at which one introduces such a break can be chosen arbitrarily. On the other hand, tangent discontinuities were incorporated within the snake model to be able to track or fit corners or cusps when these were recognized in

[7]The curvature of v (in the parametric form) is given by: $k = \frac{x_s y_{ss} - y_s x_{ss}}{v_s^3}$. As x_{ss} and y_{ss} tend toward zero, the curvature k will tend to zero.

an image at a higher level of processing [78]. Therefore, tangent discontinuities, although useful for obtaining a closer fit to the data, that is, matching valleys or folds of the potential surface forming corners or cusps, require a prior interpretation stage which is able to recognize the need for introducing such breaks. Such an interpretation stage can be implemented by looking for a minimization of the energy function E_{snake} when introducing a tangent break. For example, after a snake reaches a stable state (*e.g.*, $\overline{v}_{\overline{t}} = 0$), one may examine its curvature by looking for a region where the snake bends considerably. Tangent breaks can be introduced at these locations. When the snake reaches its new stable state, if any changes have occurred in position and energy, we can assume the validity of creating a corner or cusp at the chosen position. Such a scheme of reducing the energy of an active contour by introducing tangent or even position breaks is similar to the idea of Blake and Zisserman [19] who define a penalty function which permits breaks in the active curve if it leads to a reduction of the energy of the curve.

Similar to the introduction of discontinuities, external forces have to be situated and activated based on some other processing level to identify those points or regions where external forces are needed. Attractive forces, such as springs, may be used to attach the snake to edges found in a filtered image or to corners of identified objects. Similarly, repulsive forces, such as volcanos, may be used to push away a snake from some identified noisy region or isolated edges.

After knowing *where* to use external forces comes the question of *how* to efficiently measure or create them. The spring effect is best implemented by directly considering the spring force rather than the energy term it gives rise to. This is so because only two known points, the concerned snaxel and the fixation point, are involved in the computation of the force linking them which can easily be evaluated based on the distance separating them. The case of a volcano constraint may be considered differently. Because a volcano constraint involves a region or neighborhood of points, it is not practical to directly evaluate the possible repulsive forces between the volcano and each snaxel in its neighborhood. Instead, the cone of energy $E_{volc} = -\frac{k_{volc}}{r}$ may be added to the potential field energy E_{field}. The evaluation of the slopes on the modified potential surface will then activate the repulsive effect of a volcano. Such a method for mixing volcano energy with potential field energy avoids keeping track of a volcano and reduces the two evaluation steps, $E_{ext_{\overline{s}}}$ and $E_{field_{\overline{s}}}$, to a single one. We can mix these two kinds of energy terms because they are both represented as surfaces, a cone for E_{volc}, a potential surface for E_{field}; this is not the case for the spring energy which is better defined in a point-wise fashion.

Finally, let us consider the essential subject of evaluating field forces constrained by a potential surface. These forces emerge in the discrete equations of motion (2.25) from the directional derivatives of the potential energy terms, $-E_{field_{\overline{x}}}$ and $-E_{field_{\overline{y}}}$. They are best understood as the slopes, in the $\overline{X}$ and $\overline{Y}$ directions, evaluated at each snaxel $\overline{s}$ for the last known snake position, that is at time $\overline{t} - 1$. Therefore, the effect of evaluating these field forces is to make the snaxels move by following the steepest slope of the surface evaluated

in the neighborhood of each snaxel. Depending on the direction of the gravity force $\mathbf{g}$, that is its sign, the snaxel will fall towards valleys, the physical case of a gravity force directed downward, or will climb mountains. Since the goal of the snake approach in solving minimization problems is to find these extrema of the potential surface, a critical step in the method will be to obtain "adequate" surfaces $H(\overline{x}, \overline{y})$. Adequate here means that these surfaces must possess suitable properties in order for the snake approach to be applicable. An essential property is that valleys, or peaks, and folds of the surface correspond to significant features of the function represented by that surface. In general, the surface $H(\overline{x}, \overline{y})$ will first have to be filtered in order to emphasize those valleys or folds which may map to the desired significant features.

In the previous applications that can be found in the literature, only the image domain $\mathcal{I}$ was considered. Kass *et al.* [78] have proposed to use a combination of different filtered versions of an image I to construct the potential surface. For example, smoothed versions as well as gradients of the image can be used to attract the snake to lines and edges. A scale-space scheme might be used in a first step at a coarse scale to get closer to the global energy minimum represented by the significant searched contour. This removes most of the noise effects. In further steps, the optimal valley or contour would be sought at finer and finer scales. On the other hand, Zucker *et al.* [164] propose to build the potential surface indirectly from the image in a second stage involving reconstruction of the image contours. An initial image analysis stage is used to recover the discrete trace of a curve or contour on the basis of a relaxation labeling scheme. Points which have a "good" support with respect to their eventual curve point neighbors in terms of orientation, curvature, contrast and proximity are retained as potential curve points. The potential surface is then built by digging little oriented pockets at each of these points in an initially flat surface. The snake model is then used to interpolate the curve points by following the valleys obtained from the successive pockets.

Once a relatively adequate potential surface is obtained, the last step consists of evaluating the directional slopes iteratively at each snaxel position. In general, only coarse approximations of the first derivatives in $\overline{x}$ and $\overline{y}$ are used.

We have covered all of the aspects of the original snake model from an informal description of the snake, to the description of the dynamics of the snake in the continuous and the discrete domains, to finally some observations on how to fix the snake parameters. In the following section, we summarize the advantages and problems associated with this original snake model.

2.6 The Original Snake Model: Advantages, Limitations and Shortcomings

The snake is best understood as an active curve possessing an energy function "whose minima [hopefully] comprise the set of alternative solutions available to higher level processes" [78]. Such a model possesses advantages as well as limitations when applied to problems pertinent to spaces of functions such as the image domain $\mathcal{I}$. These are summarized in this section. Furthermore, shortcomings related to the original description of the model are also emphasized. Solutions to them will be proposed in section 2.7.

2.6.1 Advantages

When applied to the image domain $\mathcal{I}$, the snake model possesses numerous advantages over more traditional methods of recovering image contours. Within the snake model framework, images are considered as potential surfaces that may require some filtering technique to emphasize some of their features. Thus, the extraction of these features is based on a minimization procedure which retains all the relevant information that may be present in the image. This is radically different from most techniques used to extract contours or edges in intensity images which usually assume "ideal" models for these features, such as step edges or roof edges [87], and which rely on global measures of significance to fix threshold values to identify valid edges or contour points. Such methods are generally too specific in their way of imposing ideal feature models, and too global in their way of choosing threshold values. The snake model offers the advantage of extracting the same features without committing itself with respect to the exact nature of these features and to their required significance.

Another advantage of the snake model over the traditional edge and contour extraction methods is its intrinsic connectivity. Being a connected contour, information along the length of the snake is integrated in an implicit manner. This is useful when analyzing noisy images or natural scenes. In such cases one often encounters contours with missing parts, such as edge gaps, generated by regions along the boundary of an object having low contrast or significantly corrupted by noise. Applied to such problems, an edge-linking or graph search technique [87] will generally fail, being unable to bridge the gaps, while a snake will often encompass such problems thanks to its intrinsic connectivity.

A snake may also be used to extract subjective contours [78]. This capability is obtained, again, as a result of the implicit snake connectivity and also because of the energy minimization scheme used to recover the best possible solution given the available data and snake constraints.[8] The snake model is one of the few technique that addresses the problem

[8]Note that for such features as subjective contours, other energy constraints are imposed on the snake. For example, constraints relating the snake to contour terminations may be used [78].

of the extraction of such features.

Due to its explicit description of external forces and the possibility of modifying its intrinsic constraints by varying its elasticity properties or by introducing discontinuities, the snake model provides the ability to easily interface it to higher levels of processing. Of course this also imposes limitations on the application of snakes (§ 2.6.2). But, the intervention of other higher computational levels becomes necessary for many difficult applications, especially in computer vision. Such higher computational levels can be used to set intrinsic elasticity constraints and to position external forces. These higher level interactions can also be seen as ways of introducing some interactive capabilities into an automatic image understanding system.

Due to its dynamic behavior, the snake is a contour model that is well suited to tracking continuous deformations or movements of nonrigid natural shapes. The snake will follow the same local minimum or image contour from frame to frame in a video sequence, assuming that the deformations are not "large." The same idea has been applied in an attempt to solve the correspondence problem in stereo vision, where the video sequence uses the two frames given by each eye and where an additional energy term is employed to account for the disparities that occur between views [78]. This ability to track deformations also finds application to the scale-space analysis problem [156] where the different versions of an image are obtained at different scales. A sequence of frames may then be ordered from a coarse to fine scale.

The snake model is one of the few that has emerged in the computer vision literature that provides a general framework for solving different, but related, visual problems. Essentially, any problem for which the set of possible solutions can be represented as a potential function can have a snake approach applied to it, the main constraint being that this potential function must be smooth "enough" so that a snake is able to crawl and seek significant valleys. The snake approach can be applied to various problems such as extracting contours, tracking contours, stereo matching, motion correspondence, shape skeletonization, scale-space tracking, range image segmentation, path planning for robots. Furthermore, it can be generalized to higher dimensional problems. In this thesis we consider only 1-D snakes, that is active contours, but the model has been generalized by Terzopoulos to higher dimensions [142, 146]. For example *active surface* models can be defined to reconstruct or extract surface regions rather than contours [148].

2.6.2 Limitations

There are of course limitations to the usefulness of such a model based on a regularization principle [142, 146]. The latter assumes smoothness constraints when fitting the data that might be unrelated to their nature. The snake model requires potential surfaces that are *sufficiently smooth*, mainly for two reasons: so that the snake does not remain blind to the desired solution, and so that the computations involved at each iteration remain numerically

stable. We will see later, in Chapter 3, that even when using smoothness constraints, the snake may remain "blind" to some extent (*e.g.*, the snake is trapped in a local minimum). Clearly, this limits the application of such a model to non-textured images. In general, even raw data such as the intensity image $I(\overline{x}, \overline{y})$ of a scene containing only smooth surfaces or objects cannot be directly used because of its "roughness" due to quantization and sensor noise.

Another strong limitation of the model is the fact that it requires powerful initialization processes which are able to position a snake close enough to the desired solution. This especially pertains to the static image segmentation case where the problem of extracting an approximation of the trace of a contour is difficult since it is underconstrained. In the dynamic case though, especially for tracking deformable shapes in long sequences, this limitation is not as important if some interaction with higher level processes is permitted for the initialization (Chapter 3).

The fact that the snake model relies on higher level processes or *a priori* knowledge to set most of the snake parameters, to introduce discontinuities, and to position external forces also imposes limits on the usefulness of the snake model when applied to complex problems where such knowledge is not available. This is usually the situation in the image domain, particularly for the static case, where no *a priori* knowledge is assumed available except, possibly, through an extensive analysis. In other words, the segmentation problem often requires a prior solution in order to fix appropriate values for the snake parameters or to introduce discontinuities! The snake model seems better suited for more constrained problems where knowledge is available about the features being sought or for dynamic situations where previously analyzed images may be used to provide the required knowledge, and this is the case for the cell tracking problem.

2.6.3 Shortcomings

Besides the limitations of the snake model that restrict its applicability, the original description of the model possesses a number of difficulties at the implementation level which, if carefully considered, can be solved, leading to a more efficient algorithmic description.

The most important shortcoming that arises from the original description of the snake model is its relatively unstable numerical behavior. Major instabilities emerge from the fact that the terms of the discrete equations of motion (equations (2.25)) are unrelated to each other with regard to the amplitude they might reach. This is best understood by considering equations (2.25) in detail. Consider the first system of equations in $\overline{x}$: $A\overline{\mathbf{X}} = B_{\overline{x}}$ (similar results are derived by symmetry for the system in $\overline{y}$). Take a particular equation for snaxel $\overline{s} = i$ of this system:

$$a_{i-2}\overline{x}(i-2,\overline{t}) + b_{i-1}\overline{x}(i-1,\overline{t}) + c_i^*\overline{x}(i,\overline{t}) + b_i\overline{x}(i+1,\overline{t}) + a_i\overline{x}(i+2,\overline{t}) \quad =$$

$$-\frac{1}{2}\left(E_{ext_{\overline{x}}}(\overline{v}(i,\overline{t}-1))+E_{field_{\overline{x}}}(\overline{v}(i,\overline{t}-1))\right)+[2\mu]\,\overline{x}(i,\overline{t}-1)+\left[\frac{\gamma}{2}-\mu\right]\overline{x}(i,\overline{t}-2) \quad ,$$

where ${c_i}^* = c_i + \frac{\gamma}{2} + \mu$. Such an equation may be rewritten by isolating $\overline{x}(i,\overline{t})$, the snaxel coordinate that comes as one solution of the system of equations (2.25), in terms of three interaction groups that apply different types of forces to the given snaxel. This is done as follows:

$$\begin{aligned}\overline{x}(i,\overline{t}) &= \frac{1}{c_i^*}\left\{\mathcal{F}_{pot_{\overline{x}}}\left(\overline{v}(i,\overline{t}-1)\right)+\mathcal{F}_{memo}\left(\overline{x}(i,\overline{t}-1,\overline{t}-2)\right)\right.\\ &\quad \left.+\mathcal{F}_{stiff}\left(\overline{x}(i-2,i-1,i+1,i+2,\overline{t})\right)\right\}\ , \end{aligned} \tag{2.28}$$

where the following identities are used:

$$\begin{aligned}\mathcal{F}_{pot_{\overline{x}}}\left(\overline{v}(i,\overline{t}-1)\right) &= -\frac{1}{2}\left(E_{ext_{\overline{x}}}(\overline{v}(i,\overline{t}-1))+E_{field_{\overline{x}}}(\overline{v}(i,\overline{t}-1))\right)\ ,\\ \mathcal{F}_{memo}\left(\overline{x}(i,\overline{t}-1,\overline{t}-2)\right) &= [2\mu]\,\overline{x}(i,\overline{t}-1)+\left[\frac{\gamma}{2}-\mu\right]\overline{x}(i,\overline{t}-2)\,,\\ \mathcal{F}_{stiff}\left(\overline{x}(i-2,i-1,i+1,i+2,\overline{t})\right) &= -\left(\ a_{i-2}\overline{x}(i-2,\overline{t})+b_{i-1}\overline{x}(i-1,\overline{t})\right.\\ &\quad \left.+b_i\overline{x}(i+1,\overline{t})+a_i\overline{x}(i+2,\overline{t})\ \right)\ ,\end{aligned}$$

where $\mathcal{F}_{pot_{\overline{x}}}$ indicates the effects of the extrinsic forces that derive from a potential, $\mathcal{F}_{memo}$ indicates the memory effects due to inertia and damping and $\mathcal{F}_{stiff}$ indicates the elasticity constraints that directly link snaxel $\overline{s} = i$ to its four direct neighbors. A number of observations can be made from the analysis of equation (2.28). First, we note that these forces are of two distinct types, depending on whether or not they are dependent on the coordinates $(\overline{x},\overline{y})$. $\mathcal{F}_{memo}$ and $\mathcal{F}_{stiff}$ are explicitly related to $\overline{x}$ (or $\overline{y}$), being functions of it, while $\mathcal{F}_{pot_{\overline{x}}}$ is not, with the exception of one of its components, the spring force (§ 2.2). Therefore, $\mathcal{F}_{memo}$ and $\mathcal{F}_{stiff}$ can easily be limited in their extent by imposing on them saturation values ensuring that a new coordinate position $(\overline{x}(i,\overline{t}),\overline{y}(i,\overline{t}))$ will in general remain within the permitted discrete intervals $\overline{X}$ and $\overline{Y}$ (Appendix C). Under the influence of $\mathcal{F}_{memo}$ and $\mathcal{F}_{stiff}$ problems may occur only near the border of the discrete grid defined by $\overline{X}$ and $\overline{Y}$ where snaxels might be pushed out of the grid by these forces. On the other hand, extrinsic forces obtained from the directional slopes of a potential surface, that is, field forces and repulsive volcano forces, can be the cause of major instabilities because their amplitudes may assume unlimited high values without any respect for the permitted values of $\overline{x}$ and $\overline{y}$ (*i.e.*, values within the discrete intervals $\overline{X}$ and $\overline{Y}$). The maximum possible amplitudes of $\mathcal{F}_{pot_{\overline{x}}}$ (or $\mathcal{F}_{pot_{\overline{y}}}$) are obtained for vertical orientations of the potential surface where the slope takes an infinite value; for example, vertical cliffs on the potential surface will occur in correspondence to step edges in an image $I(\overline{x},\overline{y})$. Such high force amplitudes may drive

a snaxel, a group of snaxels, or the entire snake, out of bounds, that is, out of the spatial domain defined by the discrete grid.

Another different source of instability, which results in an oscillatory behavior of the snake emerges from the iterative way in which some snake parameters may be fixed. Consider the tension and rigidity weights $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$. If one uses equations (2.26) and (2.27) as proposed by Kass *et al.* [78], $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$ will in general be modified at each iteration and for all snaxels. This has the effect of making the snaxels oscillate around the values prescribed by the natural arc length function $\mathcal{L}(\overline{s})$ and the natural curvature function $\mathcal{C}(\overline{s})$.[9] Furthermore, equations (2.26) and (2.27) impose no limitations on the amplitudes $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$ may reach, which can be the cause of unrealistic displacements of the snaxels.

The approximations of the derivatives that have to be used to discretize the optimization process and to solve it on a digitized grid are in general of an unstable nature and inaccurate. The same problems apply to the approximations of the derivatives used for fixing parameters; for example, when approximating the local curvature $\overline{k}$ in equation (2.27). Furthermore, the assumption we made, following Terzopoulos [144], that the step-size $\Delta\overline{s}$ between snaxels is constant and small and can be fixed to 1 to simplify the discrete equations of motion, may be an incorrect approximation if no mechanism is implemented to enforce snaxels to stay close to each other with an almost constant inter-distance or step-size $\Delta\overline{s}$.

Besides numerical instabilities and oscillatory behavior, another important problem exists with regard to the optimization process used to activate the snake. Following the original description of the snake, one usually uses as many iterations as necessary until the snake stops moving, that is until no snaxels move. This is equivalent to having the snake reach its lowest possible total potential energy, E_{snake}, given an initial configuration on the potential surface. This criterion of having to reach a *steady-state* (*i.e.*, $\overline{v_t} = 0$) provides an excessively strong stability condition for the optimization process that often results in the snake "missing" the desired solution. By this is meant that the snake might traverse the desired valley of the potential surface at some point during the iterative process, but might, in further iterative steps, recede from that desired position. This may happen whenever the bottom of the valley is not at the same height all along its length. In such cases, snaxels will have a strong tendency to pile up [2] in the deepest pockets of the valley which may lead to an inaccurate final solution. There are two major reasons that explain this behavior of the snake. First, the snake is always seeking to reach the lowest possible point of the potential surface because of the effect of minimizing E_{field} in equation (2.2). The other constraints E_{int} and E_{ext} may be used to counteract partially this effect, but this only makes the snake more unstable. Second, because we try to minimize E_{field} all over the length of the snake, there is an arbitrary bias to favor shrinking snakes, that is, shorter snakes. In summary,

[9] This oscillatory effect due to variable stiffness parameters is different from the oscillation of the snaxels going up and down in height on the potential surface when crossing the bottom of a valley. This other type of oscillation is progressively reduced as a result of the damping introduced in the equations of motion.

using the steady-state criterion, we cannot ensure the snake will converge to the desired solution.

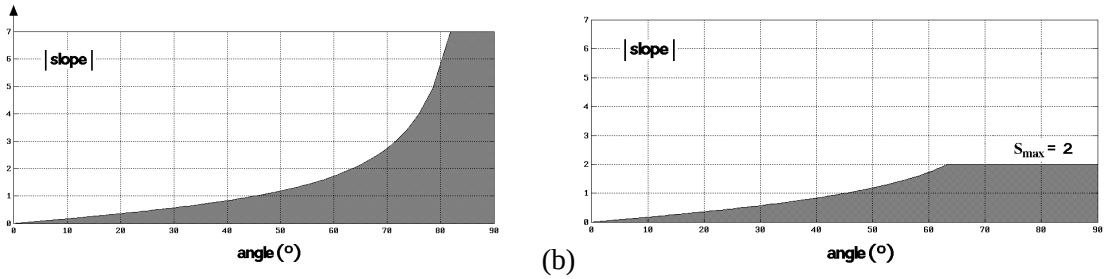
Even if a valley possesses the extraordinary quality of being at the same height all along its length, the snake will in general attain steady-state long after having reached the optimal spatial solution. This is due to the previously mentioned problems of numerical instability and oscillations. Furthermore, shrinking snakes are still favored. This again emphasizes the fact that the steady-state criterion is inadequate as a terminating condition for the optimization process used to activate the snake.

In the following section, solutions are proposed for the problems arising from the original description of the snake that were discussed in the above paragraphs.

2.7 Improving the Original Snake Model

Let us first consider the major source of instability created by possibly high amplitudes of $\mathcal{F}_{pot_{\overline{x}}}$ (similarly for $\mathcal{F}_{pot_{\overline{y}}}$) acting on each snaxel. The most obvious solution to this problem is to normalize or restrict the permitted values that any components of $\mathcal{F}_{pot_{\overline{x}}}$ might take. The directional slopes in $\overline{x}$ and $\overline{y}$ of the potential surface must be normalized in some way, or have to be redistributed on a range of values compatible with the other force values and with the intervals $\overline{X}$ and $\overline{Y}$.

The problem with redistributing the directional slopes of the potential surface $H(\overline{x}, \overline{y})$ is a non-trivial one because a positive slope function varies nonlinearly from 0 to infinity (Figure 2.4.(a)). Similar results are obtained for negative slopes by symmetry. This possibly infinite value, corresponding to a vertical orientation of the potential surface, will of course make the snake completely intractable. In fact, a quite broad range of almost vertical slopes will have the same effect. There are many ways of solving this problem. The solution we propose for the snake model consists of clipping the absolute value of the slope function for some saturation value $\mathcal{S}_{max}$ (Figure 2.4.(b)). This is an adequate transformation for the snake model because in order to make the snake seek significant valleys, what is principally needed is information about *where* it should crawl to. Therefore, the clipped version of the slope function fits our need and it is a simple solution. This clipped slope function starts by increasing slowly for small slope values, making the snake rather somnolent. Then, it increases faster and faster for higher slope values, until it saturates for some maximum value $\mathcal{S}_{max}$. This saturation value, $\mathcal{S}_{max}$, being the maximum amplitude for the group of forces $\mathcal{F}_{pot_{\overline{x}}}$, disregarding the spring forces which are set with respect to the spring constant k_{spring}, its maximum value should be chosen to relate it to the maximum desirable or permitted movement for a snaxel. A constraint on the minimum value for $\mathcal{S}_{max}$ can also be formulated by considering the minimum slope that should be accepted as significant before clipping the slope function. This minimum should not be too small because one does not want to put gentle slopes on the same level as steep slopes. We propose to use a slope



(a) (b)

Figure 2.4: The slope-amplitude problem. In (a) is shown the absolute values of the slope function. This function is asymptotic to infinity at an angle of 90^{o}. In (b) is shown the clipped version of this function for $\mathcal{S}_{max} = 2$.

of ± 1 (*i.e.*, angles of $\pm 45^{\text{o}}$) as the minimum for $\mathcal{S}_{max}$. This seems to be an appropriate choice for most applications. For example, in the image domain $\mathcal{I}$ this means that roof edges with angles of $\pm 45^{\text{o}}$ are seen as the minimum acceptable type of significant edges. We have used a value ± 2 (*i.e.*, angles of $\sim \pm 63.4^{\text{o}}$) as the maximum for $\mathcal{S}_{max}$. This means that angles from $\sim \pm 63.4^{\text{o}}$ to $\pm 90^{\text{o}}$ are considered as having similar significance (Figure 2.4.(b)). The maximum possible $\mathcal{S}_{max}$ value should not be too large because of the exponential growth of the slope function where similar angles may have very different slope values. By restricting $\mathcal{S}_{max}$ and other snake parameters (Appendix C), the maximum possible horizontal position displacement due to the influence of $\mathcal{F}_{pot_{\overline{x}}}$ can be fixed to a desirable number of horizontal discrete steps (similarly for vertical position displacements). Therefore this method of clipping the slope function of the potential surface offers a simple solution to the redistribution of this function to prevent arbitrarily large forces to be applied on the snake.

Let us now consider the problem of fixing the stiffness parameters $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$. Applying equation (2.26) to set values of $\omega_1(\overline{s})$ permits us to have snaxels at relatively constant and small distances apart from each other, thereby ensuring small numerical errors for the approximation of derivatives in the equations of motion. But, as mentioned previously in section 2.6.3, equation (2.26) imposes no limits on the values $\omega_1(\overline{s})$ may reach. This might be the source of unrealistic or arbitrarily large snaxel displacements. A solution to this problem is to apply the same "clipping" idea that was used for the slope function of the potential surface, but this time to a tension function τ used to determine $\omega_1(\overline{s})$. This tension function τ is added to equation (2.26) as follows:

$$\omega_1(\overline{s}) = \tau\left(\Delta\overline{s}(\overline{s}) - \mathcal{L}(\overline{s})\right) . \tag{2.29}$$

The function τ will be clipped for discrepancies between the snaxel inter-distance $\Delta\overline{s}$ and the metric function $\mathcal{L}(\overline{s})$ larger than a given distance $S_{\mathcal{L}}$, this at a maximum tension value τ_{max}. τ may take different forms. For example, David [47, 48] proposes to use for τ

a clipped ramp function. In such a case τ is defined as follows:

$$\tau = \begin{cases} \frac{\tau_{\max}}{S_{\mathcal{L}}} \left(\Delta\overline{s}(\overline{s}) - \mathcal{L}(\overline{s})\right) & if \quad |\Delta\overline{s}(\overline{s}) - \mathcal{L}(\overline{s})| < S_{\mathcal{L}} \,, \\ \tau_{\max} & if \quad \Delta\overline{s}(\overline{s}) - \mathcal{L}(\overline{s}) \geq S_{\mathcal{L}} \,, \\ -\tau_{\max} & if \quad \Delta\overline{s}(\overline{s}) - \mathcal{L}(\overline{s}) \leq -S_{\mathcal{L}} \,. \end{cases}$$

Applying equation (2.27) to specify $\omega_2(\overline{s})$ in the discrete domain is not a trivial task; the curvature $\overline{k}(\overline{s})$ must be approximated by second order differences, a further source of numerical instability. Also, since the weights $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$ are now varying in space and in time, the factorization step has to be recomputed at each time step. To keep the computations as simple as possible, we recommend fixing $\omega_2(\overline{s})$ to some small positive value to enforce some small degree of smoothness, that is, a weak tendency to straighten.[10] This decision can be interpreted as a least commitment rule that implies some degree of smoothing of the data combined with the advantage of avoiding the computation of $\overline{k}$. A positive value of $\omega_2(\overline{s})$ also offers the advantage that the snake will have less tendency to fold on itself.

The updating of $\omega_1(\overline{s})$ using equation (2.29) can be performed only occasionally rather than at every time step $\Delta\overline{t}$ to reduce as much as possible the number of times the factorization of matrix K has to be performed and also to reduce the oscillatory behavior of the snaxels. This updating rule finds its justification in the fact that for a stable snake every snaxel will move only in its immediate neighborhood. In general, the distance between snaxels will remain stable and the stability of the snake will be ensured by the normalization of the snake forces (Appendix B). In fact, we can push forward this idea of updating $\omega_1(\overline{s})$ occasionally by, instead, keeping it always constant and by, rather, updating occasionally the *snaxels sampling*, that is, the number of snaxels per arc length unit. For example, every N iterations, for some integer N, we can update the sampling of the snaxels by spatially *reinitializing* the snake. Reinitialization is performed by keeping the snake at its actual position and by enforcing snaxels to be equally spaced.[11] By keeping ω_1 constant, this last updating rule permits us to simplify the computations while obtaining the desired effect of having a snake with snaxels approximately equally spaced in time. We have therefore three possible updating rules for $\omega_1(\overline{s})$. We can update its values at every snaxel and for every time step $\Delta\overline{t}$ using equation (2.29). Or we can perform the updating only occasionally, every N iterations for example, still for every snaxel. Or we can keep $\omega_1(\overline{s})$ constant and rather spatially re-sample the snake occasionally. The first updating rule is the most accurate but also the most expensive computationally, while the last one is the coarsest one but the simplest one numerically.

The approximations to the derivatives that are used in the snake model are a significant source of numerical errors. In general, the presence of such errors is unavoidable. The best we can do here consists of reducing by as much as possible the coarseness of the

[10] In our experiments we have use a value of 0.01 for $\omega_2(\overline{s})$.

[11] See Appendix D for a variant of such a "snaxels resampling" updating rule.

approximations. Firstly, the numerical errors associated with the approximations of the spatial derivatives involving the stiffness constraints $\left(-\frac{\partial}{\partial s}(\omega_1(s)v_s) + \frac{\partial^2}{\partial s^2}(\omega_2(s)v_{ss})\right)$ can be kept low by using one of the previously mentioned updating rules for fixing the tension between snaxels. This ensures a relatively constant and small spacing between snaxels. Also, negative values of ω_2 should be avoided, since the bending of the snake using negative discrete rigidity constraint is of an unpredictable nature. On the contrary, positive values of ω_2 are a factor apropos stability since they enforce smoothness constraints. This is in agreement with our prior rule of fixing ω_2 to some constant small positive value. Secondly, the numerical errors associated with the approximations of the time derivatives, involving the inertial and damping constraints, can be kept small by enforcing only relatively small movements for each snaxel at each time-step. This is best performed by normalizing the forces acting on the snake (Appendix B). Finally, the numerical errors associated with the approximations of the slope of the potential surface can often be lowered by performing some smoothing filtering of the surface while extracting the directional slopes. However, there are some exceptions where the approximations of derivatives can be bypassed. This is the case whenever a symbolic derivation exists. For example, spring forces can be exactly derived from the energy of spring links between fixed points and snaxels. Another notable exception is the case of potential surface obtained in the distance transform domain, $\mathcal{DT}$. Their, the slope can be exactly determined for most points (Appendix D).

As we have seen previously in section 2.6.3, the steady-state criterion proposed in the original snake model as a terminating condition of the optimization process is an inadequate one. We propose a different criterion which is called the *steady-support* criterion. The basis of this criterion consists of examining the data for which we are seeking the features. For the snake model, we have the potential surface $H(\overline{x}, \overline{y})$, and the features we are seeking are the valleys and folds of $H(\overline{x}, \overline{y})$. Instead of seeking a minimum of the total energy of the snake E_{snake}, as a terminating condition of the optimization process, we propose to search only for a minimum of the potential field energy E_{field}. In other words, the stability criterion for the snake becomes one in which we seek a minimum and stable height of the snake on $H(\overline{x}, \overline{y})$ based solely on $H(\overline{x}, \overline{y})$ topography and not on other snake constraints (*i.e.*, E_{int} and E_{ext}). A simple implementation of this stability criterion, that provides an adequate solution to finding this minimum of E_{field}, consists of summing the local heights of the snake all along its length and of dividing this sum by the snake length. This summation is computed for each snaxel and each snaxel interval interpolating pairs of snaxels. Experimentally, we have used simple straight lines between snaxels for the interpolation, which is sufficient for most applications. We therefore seek to really minimize only the averaged potential field energy term, E_{field}, in equation (2.1); that is, we wish to find a minimum of the following energy function:

$$E_{snake_{field}}(v) = \frac{\int_\Omega E_{field}(v)\, ds}{\int_\Omega ds}\,. \tag{2.30}$$

We average E_{field} over the length of the snake so that this length has no effect on the selection of a preferred stable state. This averaging is our solution to the bias toward shrinking snakes that was present in the original optimization scheme. Although we seek a minimum of $E_{snake_{field}}(v)$ the snake still remains constrained by the internal and external potential energy terms, E_{int} and E_{ext}. The minimum of $E_{snake_{field}}(v)$ is used to provide a terminating criterion for the snake activity which is still being governed by equation (2.1). In general, this simple criterion provides a way for the snake to avoid the problem of "missing" the desired solution. This is because when the snake traverses a valley, its global height remains constant for a few time-steps, a "steady support" we can use as a terminating condition for the optimization process. Using such a topographical criterion also permits us to bypass the instability problems caused by the snake oscillatory behavior.

A final problem that needs to be examined is the *border-effect*. The border of the discrete grid defining the domain of survival of the snake should be considered with particular attention in order to ensure that the snake will not traverse it. A simple solution consists of ensuring that snaxels will not cross the borders of the discrete grid by adding wells or cliffs to the border of the potential surface. This will have the effect of pushing the snake away from these borders whenever it comes too close to them. Such a solution possesses the advantage of reducing the time complexity of the iterative process which governs the snake motion by avoiding the testing of snaxels coordinates to detect their proximity from the border of the discrete grid.

Let us summarize the modifications to the original snake model that we have proposed in order to improve its efficiency and accuracy:

- Normalization of the slopes of the potential surface with respect to the other forces acting on the snake and to the limits imposed on snaxel movements.
- Fix $\omega_2(\overline{s})$ to some small positive value (least commitment rule). Update $\omega_1(\overline{s})$ following one of the three proposed rules:
    1. Update $\omega_1(\overline{s})$ for every time-step, $\Delta\overline{t}$, and every snaxel, on the basis of a tension function τ.
    2. Update $\omega_1(\overline{s})$ relatively infrequently using the snaxel inter-distance stability assumption and on the basis of a tension function τ.
    3. Set $\omega_1(\overline{s})$ to be constant ($= \omega_1$) and re-sample the snake occasionally.
- Keep numerical errors due to the approximation of derivatives as low as possible using simple rules.
- Use a steady-support criterion as a terminating condition of the optimization process rather than a steady-state criterion.

- Solve the border-effect problem by preprocessing the border of the potential surface $H(\overline{x}, \overline{y})$.

With such improvements, the snake model will prove to be a more powerful and adequate framework for solving both the problems of tracking and describing the shape of amorphous objects such as cells.[12] In the following chapter, we address the first problem of tracking the trace of cells contours in noisy intensity images using the snake model. In a following chapter the shape description problem will be addressed by also using the snake model (Chapter 5).

[12] Addendum (February 2003): For an analysis of the snake model via a characterization of its constraints and parameters in a probabilistic framework, see the work of Boult *et al.* [33].

Chapter 3

Image Segmentation and Cell Tracking

3.1 Introduction

In this chapter, we address two fundamental issues related to the image analysis of deformable objects such as cells, namely image segmentation and tracking of deforming and moving objects.[1] We make use of the snake model as the main tool to solve these problems. We also consider hierarchical filtering techniques useful within the context of the snake model.

3.1.1 The Segmentation Problem

First, we consider the segmentation problem which consists of extracting the trace of a closed contoured cell membrane in a noisy image (Figure 3.1). The image was obtained from a video camera mounted on a microscope. It is first digitized and then transmitted to the computer where a real-time analysis can be performed. Alternatively, the image may be recorded on video tape or stored on the computer's disk system for a more detailed analysis, non real-time (Figure 3.2).

Many difficulties can arise in acquiring and segmenting images. They can be noisy and suffer from uneven illumination (Figure 3.1). Due to variations in the shape and thickness of the cell, the grey level intensities along the cell membrane may not be homogeneous. The cell is really a 3-D entity, which partially explains why its boundary is not homogeneously defined. Variations in the cytoplasm constituents and noise in the imaging process are other reasons which explain the nature of this problem. The intensity values within the cell may be similar to the values of the background. The cytoplasm might also contain organelles with strong edges that make the segmentation problem even more difficult to solve. The scale, or range of scales, at which the trace of the cell membrane is best defined

[1] Addendum (February 2003): A journal article based on this chapter was published in IEEE-PAMI in 1993 [98].

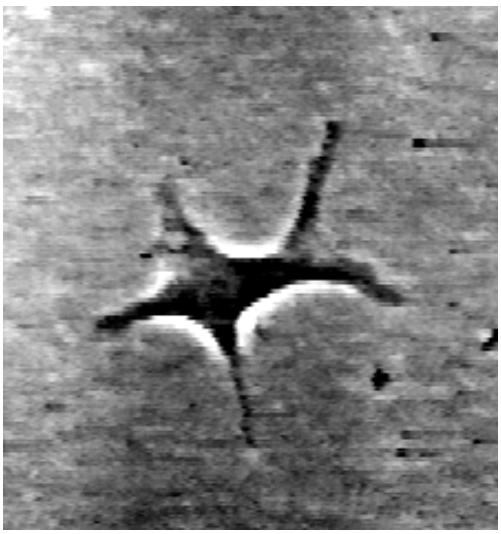


Figure 3.1: Example of a digitized cell image. The image is quantized into 256 grey levels using a frame-grabber and is normalized.

and perceived in the scene is unknown *a priori*. Finally, we must cope with a general problem found in "applied perception-cognition fields" (*e.g.*, A.I., computer vision, or pattern recognition) where no well established criteria exist to judge the quality of a machine segmentation result. There are essentially two reasons for such a lack of agreement on criteria to judge the value of a segmentation result. First, experiments in psychophysics have not yet yielded a general understanding of the perception processes. Second, the "applied-science" community has not yet agreed on general criteria for the evaluation of machine-based image segmentations. Neither has produced sets of images on which to test the extremely large variety of proposed segmentation algorithms.

The other important issue we are concerned with in this chapter is the tracking of non-rigid shapes such as cells in sequences of digitized images. We are interested in tracking a cell from frame to frame as it moves on a flat surface and as it deforms while moving. The same difficulties that exist in the static case for the segmentation problem exist here. The major difference, though, is that we can use segmentation results and motion knowledge derived from previous frames to facilitate the segmentation process in each new frame and thereby also facilitate the tracking. An example of an image sequence of a deforming and moving cell is shown in Figure 3.3.

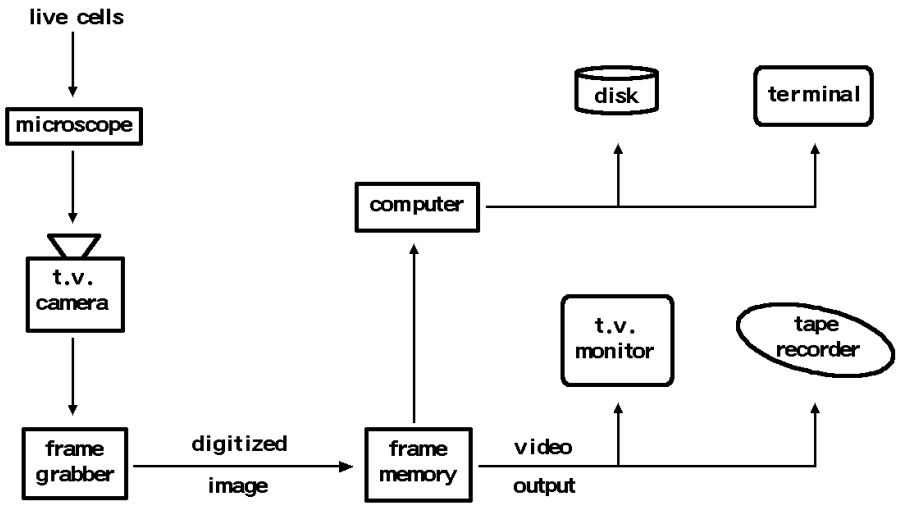


Figure 3.2: Laboratory set-up consisting of a computerized video recording system. First, an analog image is obtained from a phase-contrast microscope using a `t.v.` camera. Then, the image is digitized using the frame grabber. Motion sequences are recorded, on disk or on tape, using the time-lapsed video technique. Adapted from [115, Figure 7, Chapter 4, p. 57].

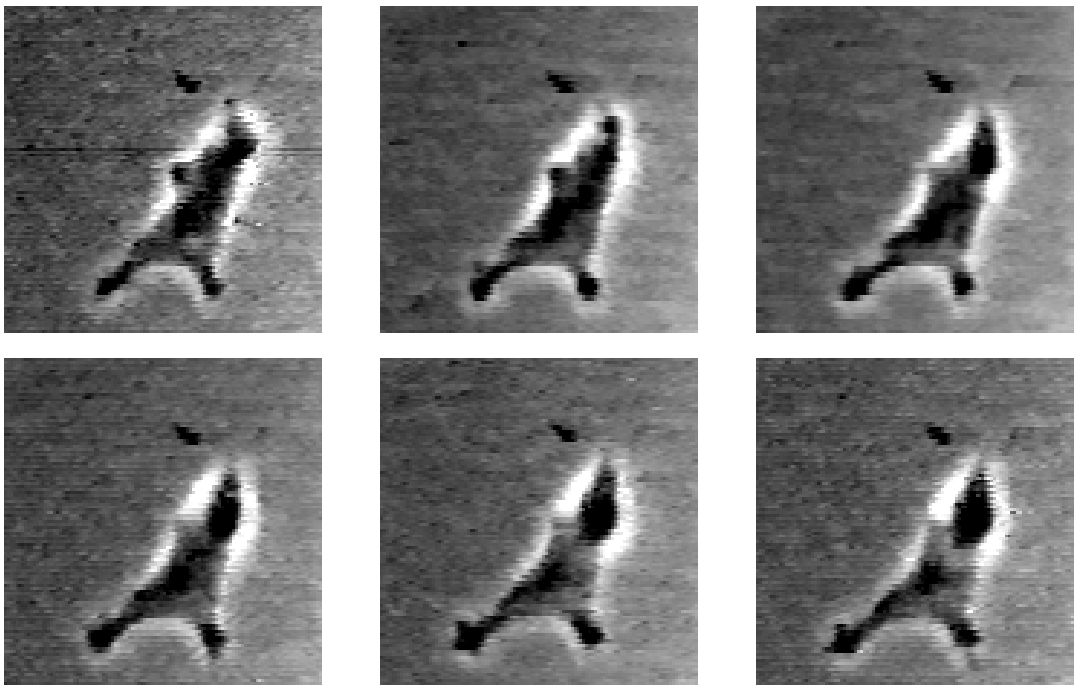


Figure 3.3: Example of an image sequence of a deforming and moving cell.

3.1.2 Assumptions

With such difficult problems as segmentation and tracking, it is necessary to make a number of assumptions if real-time image analysis is to be achieved. Therefore, we require the following two assumptions:

1. Only "small" deformations and movements of the cell occur from frame to frame.

2. User interaction, external constraints and *a priori* knowledge are permitted to initialize the first frame of the tracking sequence.

The first assumption is necessary because only small deformations, that is, of a few pixels, can be tracked using the snake. This is required as the snake must be on the slope of the valley corresponding to the newly deformed contour to effectively track this deformation. Assumption 1 is achievable for off-line analysis since we have control over the sampling rate at which cell images are acquired. For real-time analysis, this "small deformation" limitation will not be critical for many of the cell types we are studying since they move relatively slowly. For example, lymphocytes and neutrophils move at most a few microns per minute; they also often stop locomoting for various periods of time [115]. Furthermore, assumption 1 relies on a more general assumption about living organisms such as cells by which the motion and deformation of the cell's body are assumed to occur continuously in time. Therefore, we should always obtain small deformations by sampling a motion sequence at a high enough rate. In our case, the hope is that the cell will always remain flat on its surface, thereby justifying the simpler 2-D image analysis. If the cell moves in height as well, exhibiting 3-D motion, assumption 1 will remain valid, but in a 3-D space. This may create problems for our 2-D analysis scheme. In section 3.4.3 we will give an illustration of this problem.

The second assumption is in general not a critical one for the tracking of nonrigid shapes. It is the automation of the image analysis process over tens or hundreds of image frames which must be achieved. The effort spent on the first frame to initialize the process is of little concern if the correct tracking of the cells can be secured. However, assumption 2 is also seen as necessary in the context of real-time image segmentation at a video rate. Therefore, we have to ensure that the initialization is done fast enough so that criterion 1 is still met.

3.1.3 Quality Criteria

Besides computation efficiency requirements we need to address the issue of judging the quality of the results. Since no well-defined criteria exist for such a purpose we will rely on experimenter judgment. To do so we propose the following two criteria:

1. A machine-vision segmentation result will be judged "good" if the line-drawing obtained from the boundary of the segmented object permits the experimenter to "qualitatively" describe the cell shape in the same way he would by looking directly at the intensity image.

2. A machine-vision tracking result will be judged "good" if the deformations on the boundaries of the segmented object, obtained from the analysis of a sequence of images, correspond "qualitatively" to shape deformations, such as the formation of pseudopods, when observed by the experimenter.

Such "qualitative" criteria are considered to be sufficient for solving the segmentation and tracking problems by giving results which are useful, but not necessarily spatially accurate. The "qualitative" basis for these criteria also finds its motivation in psychophysics experiments. These suggest that vision is primarily concerned with "determining the general spatial layout of objects in the world" rather than a "detailed quantitative description of the visual scene" [128].

3.1.4 Solutions

To solve both the segmentation and the tracking problems, we propose to use the snake model, presented in detail in Chapter 2. The main motivation for using this model emerges from its active or dynamic nature. This permits us, first, to link both problems by solving them simultaneously and, second, to simplify them by reducing the space of possible solutions. The snake model permits us to reduce the search within the space of possible solutions by the use of heuristic constraints; that is, constraints imposed by elastic and external forces and limitations imposed by the small deformations assumption.

The potential surfaces on which the snake crawls are obtained from the image domain, $\mathcal{I}$, for the two problems of segmentation and tracking. Digital intensity images or filtered versions of them are used for this purpose. Since filtering is an essential step in obtaining a useful potential surface, that is, one where valleys correspond to interesting features of an image, we consider the subject of intensity image filtering in the next section (3.2). Once we have shown how potential surfaces are obtained, experiments on the image segmentation and tracking of cells using snakes will follow in subsequent sections (3.3 and 3.4, respectively).

3.1.5 Contributions

In this chapter, one essential contribution is made:

- We show the usefulness and limitations of the snake model for solving both the image segmentation and tracking problems in the case of nonrigid bodies such as cells. We give typical cases in which the method fails and we explain why it does so.

A number of less important contributions are also made. They are summarized below.

- We give clear explanations of what image filtering is, of its links to *scale-space* representations (section 3.2.2), and of how to efficiently implement it by the use of a particular class of pyramidal image representation called the *Hierarchical Discrete Correlation* or HDC (section 3.2.3).
- We propose a refinement procedure for the implementation of the HDC which improves its descriptive power by combining it with the *cascaded correlation* technique (section 3.2.3.4).
- Finally, we define the notion of a family of potential surfaces which are well suited within the context of the snake model (section 3.2.3.5).

3.2 Image Filtering

3.2.1 Introduction

Digital image filtering is a fundamental task which is used extensively in computer vision applications [38, 87]. *Filtering* can be understood as the processing of an image in order to discriminate, modify or compare its attributes. In contrast to a point operator which affects brightness or contrast, filtering serves to affect the spatial information in the digital image. For example, digital filtering is used to enhance boundaries and for the removal of noise. Filtering can be implemented by having a filter, or discriminating device, applied to every sampled signal point or element [126]. In the image domain, $I(\overline{x}, \overline{y})$ is a 2-D discrete signal and the filter is known as a mask or kernel. An example of digital filtering is *discrete correlation*. The correlation operation involves comparing the filter to pixels within the digital image to measure their degree of similarity [126].

Within such a filtering scheme, essentially two forms of computation may be performed: global and local. *Global filtering* is principally used to correlate an image with models of structures or objects that it may be desirable to recover in that image; this is the case in *image matching* [130]. In *local filtering*, small masks are correlated with the larger image in order to extract or modify some of the local properties of that image [38]. For our needs, that of producing useful potential surfaces, we will consider the latter form.

Essentially, we shall use two types of local image filters: lowpass and bandpass. The former will be necessary to remove noise and to smooth the original image data, while the latter will be necessary to emphasize image features such as edges, contours or region limits. As we shall see in subsequent sections, these two types of local filtering can be efficiently combined into a hierarchical method. However, before proceeding to a description of the filtering algorithm, the notion of scale will need to be examined carefully. As we will see in the following subsection, scale and filter-size are directly related notions.

3.2.2 Image filtering and the Notion of Scale

In computer vision applications, *scale* can be defined as the perceived size at which an object or structure is best isolated and identified in a scene. For example, let us consider one frame of a cell motion sequence (Figure 3.3.(a)). The maximum scale (δ_{max}) is bounded by the size of the image, whereas the minimum scale (δ_{min}) is limited by the pixel size. Between these extrema there exists a range of scales in which different objects or structures are best described at their specific *natural* scale. Natural scale refers to the scale or perceived size at which a structure in a scene emerges. A cell is described at an intermediate scale ($\delta_{cell} < \delta_{max}$) which is related to its size. An intra-cellular organelle would be described at a smaller scale ($\delta_{organelle} < \delta_{cell}$). The cellular membrane segments are described at an even smaller scale ($\delta_{membrane} < \delta_{organelle}$). Artifacts in the scene such as other organisms, gel constituents,[2] glass impurities, and shadows are described at yet smaller scales ($\delta_{texture} < \delta_{membrane}$). Also noise events arising from the imaging process (sensor) and the quantization process (digitization) can be described at small scales ($\delta_{noise} \approx \delta_{texture}$). The visual perception of these different structures or events can thus be understood as one of discrimination where we wish to "separate events at different scales" [155].

As mentioned previously, local image filtering can be used to discriminate between various image structures. It is by varying the width or size of the filter mask that the discrimination can be achieved for different image structure sizes. Therefore, we can establish a direct relation between the mask size of a filtering process and the scale at which we may desire to seek image structures or features. Such a mechanism of seeking image structures at various scales by performing image filtering for varying mask sizes has been described in the computer vision literature [102, 40, 87]. Events at different scales are discriminated depending on the frequency response of the filter. Basically, two types of filtering methods are used for scale discrimination. Lowpass filters of varying mask size are used to blur or average details at varying scales, while bandpass filters of given sizes can be used to extract features at corresponding scales. A popular example of the former filter type is the Gaussian blurring scheme in which Gaussian filters of varying size are used to blur the image by "averaging" image details at scales smaller than or similar to the filter size [87]. Popular examples of the latter filter type are obtained by the use of the Difference of Gaussians filters and the Laplacian filters which permit us to extract or emphasize structures at varying scales [35, 87].

Although filters of varying size can be used to perform image analysis or description at varying scales, the problem of setting the filter mask size remains. How can one know at which scale to process an image in order to identify events or image structures whose sizes are unknown *a priori*? The most valuable solution to this problem proposed in the literature consists of filtering the image at many scales simultaneously to retain "all available

[2]The gel is a lattice of collagen fibers immersed in a growth media which keeps the cells viable and allows them to move and grow. Some constituents of the gel may be opaque creating wire-like shadows or a texture.

structure" [81]. Furthermore, by continuously varying scales, thereby creating a continuous "space" containing all these filtered versions of the original image, a *scale-space* image representation is defined. This image representation is simplified by implicitly "relating one scale to another" [155], due to the intrinsic continuity along the scale axis of the scale-space.

It is by introducing such a scale-space representation of an image that we will generate a sequence or hierarchy of potential surfaces ordered from a coarse scale for large filter mask size to a fine scale for small filter mask size. Optimal image feature extraction, such as the extraction of the trace of a cell contour, will then be obtained by a *continuation method* [156, 147] of tracking the given feature from a coarse scale to its fine natural scale. In the following section, a computationally efficient scale-space representation will be described in order to generate such a hierarchy of potential surfaces. With this hierarchy or family of potential surfaces, a continuation method using the snake will be used to recover the trace of the cell contours and to track cells from frame to frame.

3.2.3 Efficient Local Image Filtering

For the purpose of producing potential surfaces which have valleys corresponding to significant events in the image domain, we must process or filter the original image to emphasize these events or features. Furthermore, this filtering should be performed in a multiscale fashion since the natural scales of the events we seek are unknown *a priori*. Finally we need to perform two types of filtering: lowpass and bandpass or differentiation filtering.

There exists a filtering framework, known as the *pyramid*, which exhibits all the characteristics we are seeking.[3] Furthermore, it provides computationally efficient methods of performing multiscale image correlation or filtering. For example, correlation using pyramid-like filtering methods can be performed in one to two orders of magnitude faster than with the Fast Fourier Transform method [38]. In the following paragraphs, we briefly describe how to produce families of potential surfaces ordered by scale using a generalized pyramid filtering method called the *Hierarchical Discrete Correlation* method originally proposed by Burt [38]. We will also propose to make this method more flexible by combining it with the *cascaded correlation* technique [45, 152].

3.2.3.1 Hierarchical Discrete Correlation

Hierarchical Discrete Correlation (HDC) is a filtering method performed on a digitized image, $I(\overline{x}, \overline{y})$, to produce a family of different filtered versions of that image. Following

[3] The pyramid also provides an image representation scheme which shares some links with the human visual system and the so-called "multichannel model" [154] in which a fixed number of frequency-tuned "channels" are used to carry representations of an original image at varying scales before additional high level analysis is performed [40, 87].

Burt's notation [39], a sequence of HDC images $\{g_0,\ g_1,\ g_2,\ \ldots\}$, can be produced as follows:

$$g_0(\overline{x},\overline{y}) = I(\overline{x},\overline{y})\ ,\ \ g_l(\overline{x},\overline{y}) = W_l(m_l,n_l) \otimes g_0(\overline{x},\overline{y})\,, \tag{3.1}$$

where l indicates the level in the hierarchy or sequence, W_l is the *weighting function* or mask of dimensions (m_l, n_l) proportional to l, and $\otimes$ is the correlation operation. A typical example of weighting functions are the discrete Gaussian kernels of varying size (Figure 3.4). As the mask W_l gets larger the corresponding HDC image g_l becomes more and more blurred; this HDC family produces a sequence of lowpass filtered images. Since direct correlations with increasing mask sizes are computationally expensive [38], another way of generating a HDC sequence is required. The HDC method itself offers a solution to this problem since "the correlation of a function [or signal] with certain large kernels can be computed as a weighted sum of correlations with smaller kernels" [38]. Therefore, a hierarchy can be produced by recursively filtering HDC images with a small mask, $w(m,n)$, of fixed dimensions, (m,n). We call this recursive property of the filtering process the *chain rule.*[4] Again, following Burt [39], equation (3.1) can be rewritten as follows:

$$g_0(\overline{x},\overline{y}) = I(\overline{x},\overline{y})\ ,\ \ g_l(\overline{x},\overline{y}) = \sum_{m=-2}^{2}\sum_{n=-2}^{2}\left(w(m,n) * g_{l-1}(\overline{x}+m2^{l-1},\overline{y}+n2^{l-1}\right)\ , \tag{3.2}$$

where a 5 x 5 mask is used as the recursive filter. Other mask sizes can be used, but a 5 x 5 mask has proven to be adequate with respect to efficiency and quality of the results for many image filtering applications [39]. In similarity to any discrete filtering process, special care must be taken near image boundaries where the mask may require sample points outside the image domain. One satisfying solution to this problem is to assume reflections along the image boundaries by applying the so-called *mirror effect* [39].

Equation (3.2) implies that from level to level in the hierarchy, the sampling distance d_l between elements (at level $l-1$) that contribute to the new HDC image (at level l) is doubled with each iteration. Sampling rates other than 2^l can be used to produce different types of HDC structure [38]. In the HDC method, the weighting mask $w(m,n)$ is constrained to possess certain characteristics that ensure normalization, unimodality, symmetry, a centered position of this mask and convergence of the recursive process [38]. To make the computations simpler, an additional constraint may be enforced by having the weighting mask $w(m,n)$ separable [38]:

$$w(w,n) = w_{\overline{x}}(m) \otimes w_{\overline{y}}(n)\,, \tag{3.3}$$

[4]This recursive property is also shared by certain morphological filters that will be used to process the cell contour in Chapter 4. It is within the mathematical morphology context that this property is called the "chain rule" property (Appendix A), which explains its use here.

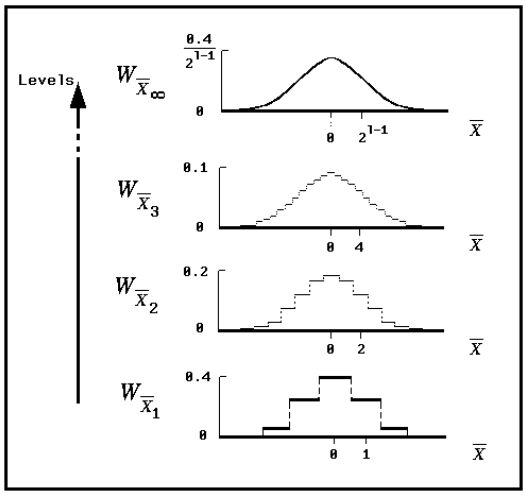


Figure 3.4: Equivalent 1-D weighting functions (here $W_{\overline{x}_l}$, along the $\overline{X}$ axis) converging to a 1-D Gaussian kernel (adapted from [38, Figure 2]).

where $w_{\overline{x}}$ and $w_{\overline{y}}$ are 1-D masks of length five used to filter the HDC images in the $\overline{X}$ and $\overline{Y}$ directions, respectively. This reduces the computational complexity of each correlation evaluated at any sample point from $\mathcal{O}(m*n)$ to $\mathcal{O}(m+n)$. In terms a 256 x 256 size image this represents a decrease of one order of magnitude in complexity. Figure 3.5 illustrates how the HDC is built in the 1-D case (in the $\overline{X}$ or $\overline{Y}$ direction) with a sampling rate of 2^l.

With these characteristics and constraints, the weights of the $w_{\overline{x}}$ mask (similarly for $w_{\overline{y}}$ by symmetry) are as follows [38]:

$$\begin{aligned} w_{\overline{x}}(0) &= a\,,\ where\, 0.25 \leq a \leq 0.5\,, \\ w_{\overline{x}}(-1) &= w_{\overline{x}}(1) = b = 0.25\,, \\ w_{\overline{x}}(-2) &= w_{\overline{x}}(2) = c = 0.25 - 0.5a\,, \end{aligned} \tag{3.4}$$

where a is a real value parameter to be chosen. Setting this value generates different convergence effects for the HDC method. For example, Burt [38] has shown that a value of $a \approx 0.4$ makes the HDC weighting functions converge to a Gaussian-like kernel. This convergence is best visualized by looking at the effects of recursively convolving the mask $w_{\overline{x}}$ (similarly for $w_{\overline{y}}$) with itself to produce the sequence of weighting functions[5] $\{W_{\overline{x}_1},\ W_{\overline{x}_2},\ W_{\overline{x}_3},\ \ldots\}$ (Figure 3.4).

A final subject of interest for the construction of potential surfaces using a method such as the HDC is its computational complexity. It has been shown by Burt that the HDC

[5]Since $w(m,n)$ is separable (equation (3.3)), $W_l(m_l,n_l)$ is also separable. This can be expressed as follows: $W_l(m_l,n_l) = W_{\overline{x}_l}(m_l) \otimes W_{\overline{y}_l}(n_l)$.

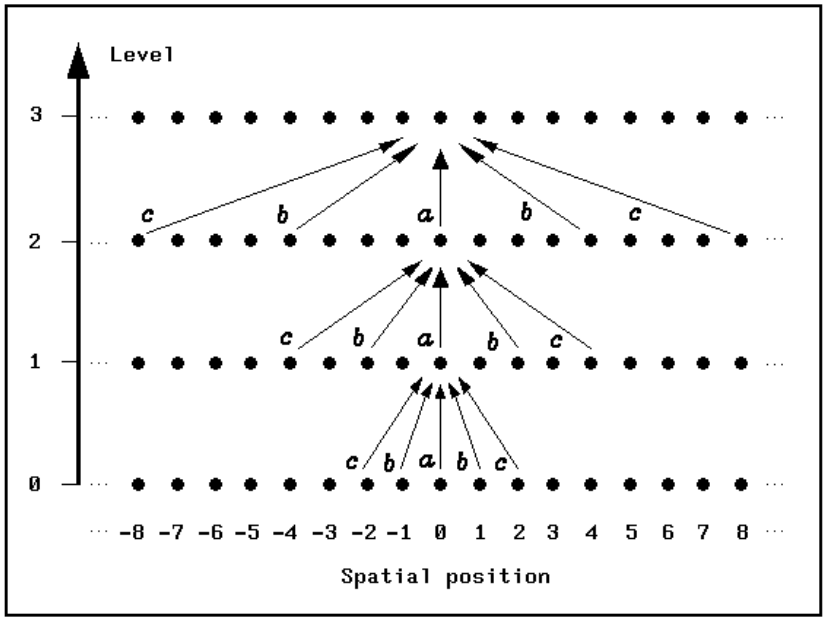


Figure 3.5: A sketched representation of the HDC method (1-D case). The letters a, b and c represent the weights of a mask ($w_{\overline{x}}$ or $w_{\overline{y}}$) of length five. The arrows represent the neighborhood relations from level to level of this recursive filtering method (*i.e.*, how one level is built from the preceding one). Here nodes represent sample points of the HDC image. Adapted from [38, Figure 1].

method is at least one order of magnitude faster than conventional correlation methods or even the Fast Fourier Transform method [38] when correlation results are needed at a single specific scale. If measurements are needed at different scales, then the HDC becomes even more advantageous over other traditional techniques [39]. In the following subsection we describe a particular class of HDC structures based on the use of Gaussian-like kernels, thereby producing lowpass filtered versions of an image. In section 3.2.3.3, another HDC representation will be described to produce bandpass filtered versions of an image.

3.2.3.2 Gaussian HDC

Using the original HDC method as defined in the preceding section, we can construct a structure of HDC images which constitutes a sequence of lowpass filtered versions of the original image. The particular structure of HDC images built by choosing the weights of the mask $w(m,n)$ to approximate Gaussian kernels (equation (3.4)) is named the *Gaussian* HDC.[6]

Each level l of the Gaussian HDC can be seen to correspond to a filtered version of the original image $I(\overline{x},\overline{y})$ at a different scale δ_l. A particular level l can be seen to be equivalent to a correlation of $I(\overline{x},\overline{y})$ with the weighting function W_l. Moreover, W_l approximates a discrete 2-D Gaussian kernel G_l as follows:

$$W_l(m_l,n_l) \approx G_l(\overline{x},\overline{y}) = \frac{1}{\sigma_l\sqrt{2\pi}} \exp\left[\frac{-(\overline{x}^2+\overline{y}^2)}{2\sigma_l^2}\right], \tag{3.5}$$

where σ_l stands for the standard deviation of the approximated Gaussian kernel at level l. By climbing in the HDC structure, coarser and coarser versions of $I(\overline{x},\overline{y})$ ($\approx G_l \otimes I(\overline{x},\overline{y})$) are obtained, where image structures at scales smaller than a scale δ_{min_l}, where δ_{min_l} is a function of the sizes (m_l,n_l) of the weighting function, W_l, are completely blurred or averaged with respect to neighboring image structures. An example of a Gaussian HDC created from one frame of a cell motion sequence is shown in Figure 3.6. Because the fixed sampling rate of two was used to generate HDC images, each one of them has a band limit which is one octave lower than the one of its predecessor [39]. This can be shown by deriving the values taken by the standard deviation σ_l of the approximated Gaussian kernel G_l at each level of the Gaussian HDC. Burt has derived the following equation for the standard deviation [38]:

$$\sigma_l = 0.56 \cdot 2^l\,. \tag{3.6}$$

Since σ_l gives us the degree of blurring or averaging obtained by Gaussian filtering, it can be used to approximate the scale of the smallest meaningful structures in an HDC

[6]Note that the constraint of separability we have enforced on the HDC filters does not interfere with the approximation of a 2-D Gaussian filter since separability is a property shared by the latter [152].

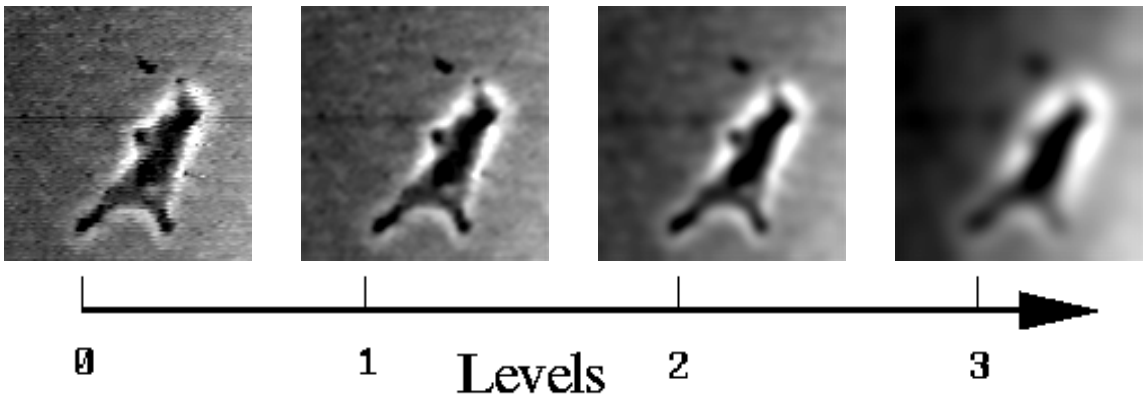


Figure 3.6: Example of a Gaussian HDC. The original image (level 0) corresponds to the first frame in the cell motion sequence given in Figure 3.3.(a).

image. This scale δ_{min_l} can be viewed as a linear function f of σ_l, that is, $\delta_{min_l} \approx f(\sigma_l)$. Therefore, using equation (3.6), we obtain the following relation between the minimum scales at each HDC level:

$$\delta_{\min_l} = 2 \cdot \delta_{\min_{l-1}} \; or \; r_{\delta_{\min_l}} = \frac{\delta_{\min_l}}{\delta_{\min_{l-1}}} = 2 \; for \; l \geq 1 \,, \tag{3.7}$$

where $r_{\delta_{min_l}}$ represents the *inter-level minimum scale ratio* of the HDC and where, at level 0, $\delta_{min_0} = \delta_{min}$, that is, the pixel size.

The Gaussian HDC can be seen as a technique for producing a "discrete" version of the so-called "scale-space," where each level has its scale related to the scales of its neighboring levels by equation (3.7), but with quantized scales.[7] For our need to produce families of potential surfaces, a discrete scale-space representation will prove to be sufficient.

3.2.3.3 Bandpass HDC's

We have obtained a multiscale image representation with the Gaussian HDC in terms of lowpass transforms of the original image $I(\overline{x}, \overline{y})$. What would be more useful for our needs is to have a multiscale representation in terms of bandpass transforms of $I(\overline{x}, \overline{y})$ in order to emphasize certain image structures such as edges and contours at different scales. One simple possibility is to take each level of the Gaussian HDC and apply a differentiation filter to it, for example the Sobel operator[8] [87], to produce approximations of the gradient of the

[7] Note that "continuously varying scale" versions of HDC-like methods have been studied in the literature; see for example [81, 76].

[8] Note that the Sobel operator also possesses the separability property which can be used to improve the efficiency of its filtering implementation [153].

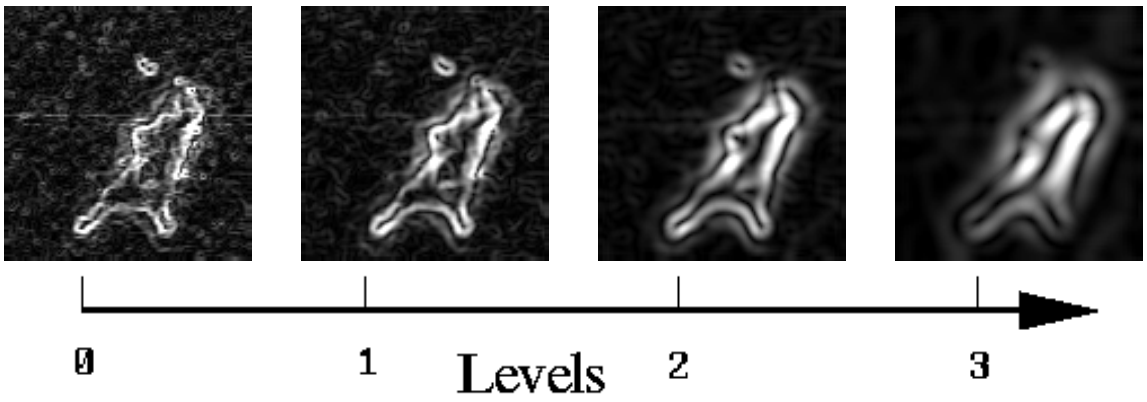


Figure 3.7: Example of a Gradient of Gaussian HDC (GGHDC) for a constant inter-level minimum scale ratio ($r_{\delta_{min_l}} = 2$). The original image was obtained from the first frame in the cell motion sequence shown in Figure 3.3.(a). The corresponding Gaussian HDC (Figure 3.6) is filtered at each level with simple 3 x 3 differentiation masks (here the Sobel masks) at each of four possible discrete orientations ($0^o, 45^o, 90^o$ and 135^o) [87]. The maximum response, or the sum of the four orientation responses, may then be taken as the new sample value (here we take the maximum).

Gaussian filtered images ($\approx \nabla |I(\overline{x}, \overline{y}) \otimes G_l|$). We call this multiscale image representation the "Gradient of a Gaussian HDC" or GGHDC (Figure 3.7). Masks other than the Sobel may be used. For example, Burt has proposed using simple first difference masks on the Gaussian HDC to approximate band limited directional derivatives [38].

Another bandpass transform derived from the HDC model has been proposed by Burt and Adelson [41]. They use a well known property of Gaussian kernels that permits the approximation of the second derivative of a Gaussian by the Difference of Gaussians (DoG's) [102]. By comparing two successive levels in the HDC, a *Laplacian Pyramid* or Laplacian HDC can easily be built [41, 39]. Such a structure also generates a multiscale image representation, but this time it produces approximations to the Laplacian filtered images ($\approx \nabla^2(I(\overline{x}, \overline{y}) \otimes G_l)$). The Laplacian HDC could be used in conjunction with the snake model to recover valleys which map to zero-crossings of $I(\overline{x}, \overline{y})$ [78], rather than extrema of $I(\overline{x}, \overline{y})$ as is the case with the GGHDC.

Both the GGHDC and the Laplacian HDC, if taken as the generators of families of potential surfaces, will require a subsequent slope evaluation in the $\overline{X}$ and $\overline{Y}$ directions in order to make the snake active.

3.2.3.4 Cascaded Correlations and the HDC

One limitation of the HDC method described in the previous paragraphs is the fact that the inter-level minimum scale ratio $r_{\delta_{min_l}}$ of the HDC is fixed at two as shown in equation (3.6). This limits the "resolution" or fineness of the HDC structure, by having scales δ_{min_l} that increase rapidly with l, thereby giving a discrete scale-space which may be "too coarse." If we require a finer scale-space representation, one obvious option is to reduce and fix the sampling rate between successive HDC levels. By fixing the sampling rate to be constant the HDC method becomes equivalent to the *cascaded correlation* method,[9] where a new filtered image or signal is produced by recursively filtering it with the same Gaussian kernel. In the discrete domain the easiest choice is to reduce and fix the sampling rate from 2^l to 1. However, Burt has proposed variable sampling rates other than integers for the HDC model. He describes how fractional sampling rates (between 1 and 2^l) can be defined to construct a *Fractional HDC* [38]. We do not consider such types of HDC structures because they are more difficult to implement and process, and because the image representation hierarchies we have discussed so far are sufficient for our needs.

We propose to combine both methods, the original HDC and the cascaded correlation, to produce a "hybrid" HDC, where the inter-level minimum scale ratio, $r_{\delta_{min_l}}$, can vary if required. This can be useful if potential surfaces or filtered images are required at or around a certain smoothing scale, σ^*. Then, an expanding sampling rate of 2^l could be used to rapidly approach σ^* (*e.g.*, up to level l^*), and a fixed sampling rate of one could be used to produce HDC levels with smoothing scales around the desired σ^* (Figure 3.8). In such a case, equations (3.6) and (3.7) are valid only up to level l^*.

3.2.3.5 Families of Potential Surfaces

In this final section related to image filtering we give a definition of the notion of a "family of potential surfaces" that we will use within the context of the snake model in subsequent sections.

We have seen in this section how the HDC model could be used to produce computationally efficient discrete scale-space image representations in terms of lowpass and bandpass image transforms. We will use such hierarchies to produce potential surfaces at varying quantized scales on which a continuation method will then become applicable to recover the trace of image contours.

For each image or 2-D signal to be processed using the snake model and where a multiscale analysis is required, the Gaussian HDC or the GGHDC will produce a *potential surface family* $\mathcal{H}$ ($\mathcal{H}_{[GHDC]}$ or $\mathcal{H}_{[GGHDC]}$), where each HDC level corresponds to a *potential surface member* $\mathcal{H}_l$ of the family. Each member $\mathcal{H}_l$ possesses a *child* (at level $l+1$) and

[9] Note that, in general, "cascaded correlation" is referred to as *cascaded convolution* in the literature. For symmetric kernels, such as the Gaussian, both filtering operations, correlation and convolution, are equivalent. Image hierarchies based on the idea of cascaded correlation have been studied in the literature [45, 152].

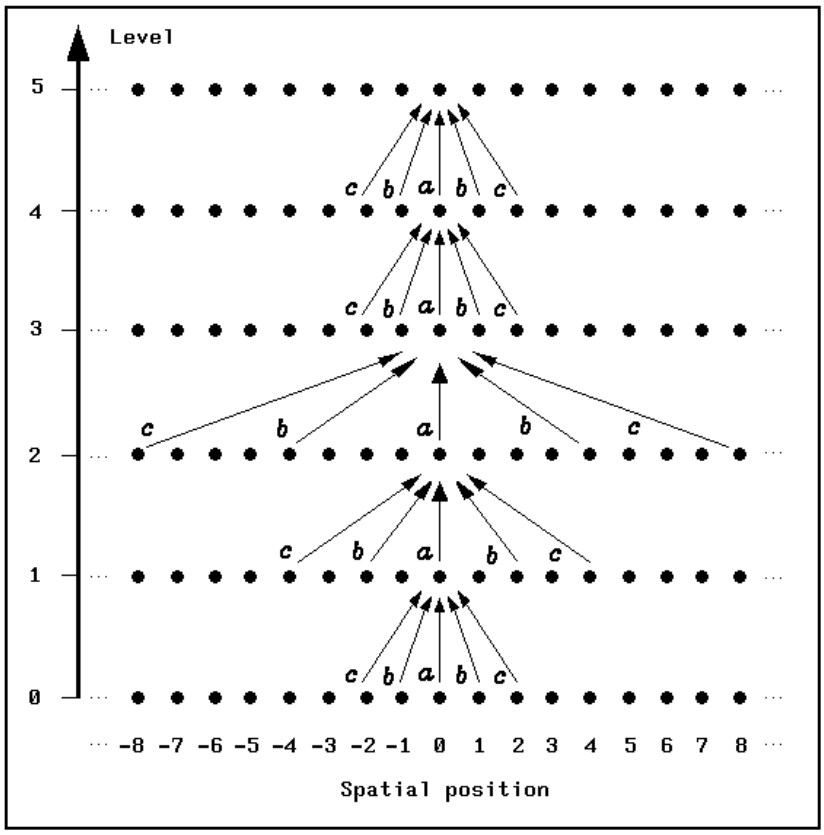


Figure 3.8: Construction of a "hybrid" HDC. An expanding sampling rate of $2^{\,l}$ is used up to a level l^* ($= 3$, for example) to obtain HDC levels to rapidly reach a given smoothing scale σ^* ($> \sigma_3$, in this example). Then a fixed sampling rate of 1 is used to produce HDC levels possessing smoothing scales around this σ^*.

a *parent* (at level $l-1$), with the exception of the *patriarch* ($l = 0$) which has no parent, and the *youngest child* ($l = l_{max}$) which has no child. Each child of the family will be filtered at a larger scale than its parent. A continuation method will then first be used to extract the trace of an image contour for a young member of $\mathcal{H}$ ($H(\overline{x}, \overline{y}) = \mathcal{H}_{l^*}$; *i.e.*, at a coarse scale $\delta_{min_l^*}$). Once an optimal solution has been reached for $\mathcal{H}_{l^*}$ using the snake model, this solution will be used as an initialization step for the direct parent $\mathcal{H}_{l^*-1}$. Again an optimal solution will be found and used as an initialization step for the next parent in the family ($\mathcal{H}_{l^*-2}$). By "continuously" tracking the best solution from generation to generation, a final solution will be generally reached with the eldest child (level 1) of the family. Terminating the continuation method before attaining the patriarch (level 0) can be understood as a need for discriminating edge or contour structures (at $\delta_{min_l} \approx \delta_{membrane}$) from noise and artifacts ($\delta_{min_l} \approx \delta_{texture} \approx \delta_{noise}$). Noise and artifacts usually have their natural scale at the pixel level ($\delta_{noise} \approx \delta_{texture} \approx \delta_0$), while segments of the trace of the cellular membrane are generally defined at a larger scale (*i.e.*, $\delta_{membrane} > \delta_{noise}$ or $\delta_{texture}$). Also, an image will in general require some degree of smoothing or blurring so that the snake remains stable on valleys corresponding to image contours. Therefore, we will generally end the continuation method at level 1 of the HDC structure, or at higher levels, unless we have *a priori* knowledge about the degree of smoothness of the original image.

3.3 Image Segmentation

In the following paragraphs we will show how the snake model combined with the HDC method can be used to recover the trace of a cell membrane in a noisy digitized image, thereby solving the segmentation problem. We note that some *a priori* knowledge will be assumed or that some user interaction will be permitted to spatially initialize the snake. The hybrid HDC method of filtering, combined with a continuation method from coarse to fine scales, will be used to discriminate noise and artifact from the trace of the cell membrane. The snake model will be helpful in bypassing to some extent the difficulties caused by inhomogeneous contrast along the trace of the cell membrane. It will also permit to extract a connected curve which approximates the trace of a cell membrane.

3.3.1 Initialization of Image Segmentation

Typically, the scene we are looking at contains more than one cell. As a first initialization step, we assume that a window is centered on each of the existing cells in the scene, the windows being positioned with the help of some high level interaction. Secondly, a closed snake is coarsely positioned by high level user interaction around the perceived cell contour in each of these windows. For example, this initial snake position can be drawn on the scene as a polygon by interactively specifying a set of points around the cell (Figure3.9). Note that

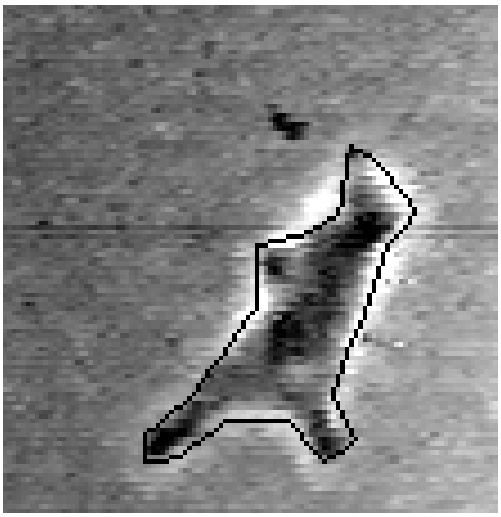


Figure 3.9: Initialization of the segmentation process where an initial polygonal snake positioned around the cell is shown superimposed on the image. This image corresponds to the first frame in the cell motion sequence shown in Figure 3.3.(a). It was selected interactively in a larger scene containing many cells by defining a square window approximatively centered on the cell. The polygonal snake was initialized by the user who selected a number of knot points around the cell.

the snake is initially positioned close to the solution we seek to recover. This is necessary because the snake is relatively blind in its search for a best solution. Otherwise it could easily be trapped in local minima of the potential surface which do not correspond to the trace of the cell membrane.

3.3.2 Building the Potential Surface Family

Once the initialization procedure has been completed, we apply the HDC method to build a family of potential surfaces $\mathcal{H}$. For example, Figure 3.6 and Figure 3.7 show the Gaussian HDC family ($\mathcal{H}_{[GHDC]}$) and the corresponding GGHDC family ($\mathcal{H}_{[GGHDC]}$), respectively, for the initial frame of the cell motion sequence illustrated by Figure 3.3. Figure 3.10 illustrates how one member of each family, $\mathcal{H}_{[GHDC]}$ and $\mathcal{H}_{[GGHDC]}$, can be visualized as a 3-D surface on which the snake will crawl seeking valleys.

3.3.3 Finding an "Optimal" Solution Using a Continuation Method

In order to recover the best possible approximation to the trace of the cell membrane, we start at a coarse HDC level l^*. This permits us to have the snake converge to the desired

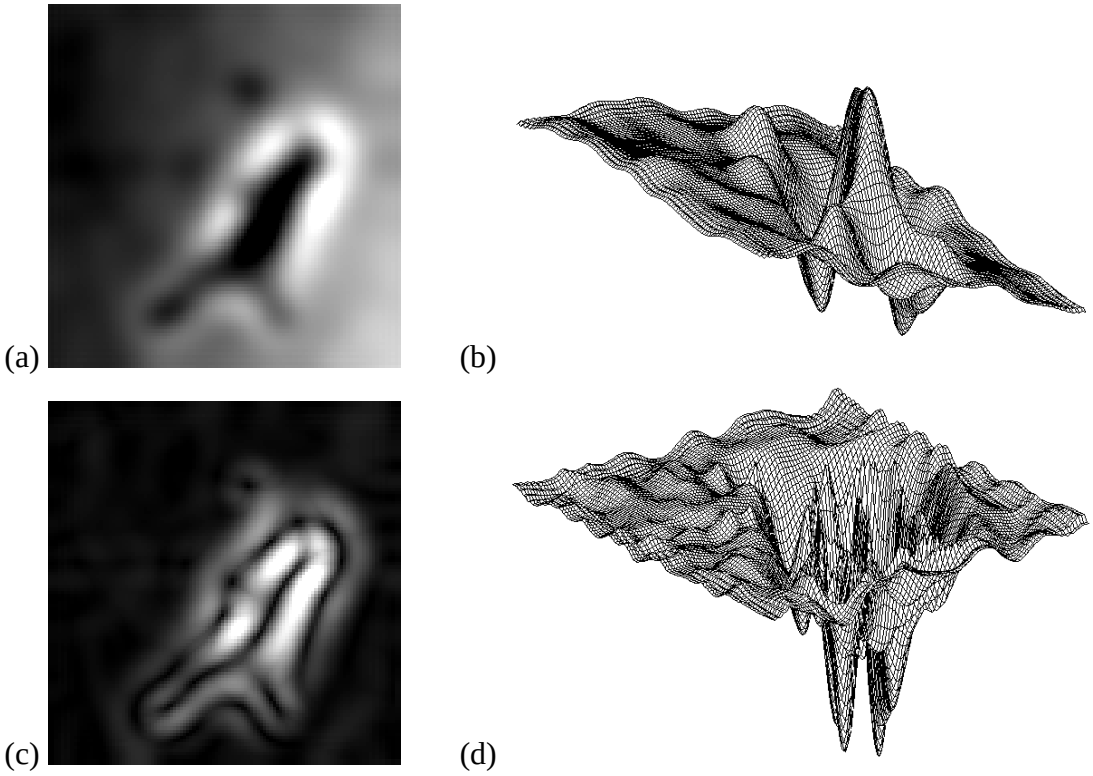


Figure 3.10: HDC images visualized as 3-D surfaces. In (a) is shown the HDC family member $\mathcal{H}_{[GHDC]_3}$ as an image of intensity. This HDC level was obtained from the original frame in the cell motion sequence shown in Figure 3.3. In (b) is shown the same family member $\mathcal{H}_{[GHDC]_3}$ as in (a), but this time represented as a 3-D surface where height corresponds to the grey level intensity. White is considered the zero level height, while black is the highest level. In (c) is shown the HDC family member $\mathcal{H}_{[GGHDC]_3}$ as an image of intensity. This HDC level was obtained from $\mathcal{H}_{[GHDC]_3}$ using the Sobel edge operator. In (d) is shown its representation as a 3-D surface (same convention as (b)).

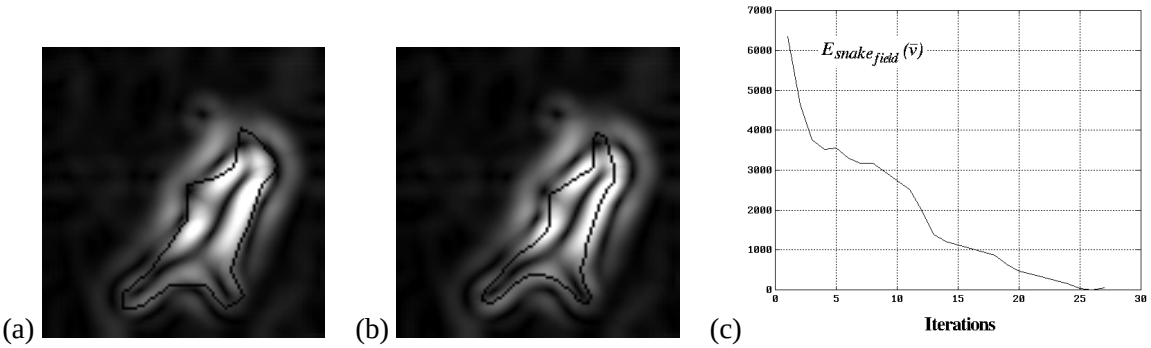


Figure 3.11: Finding the best solution at a coarse HDC level ($l = 3$). In (a) is shown the initial position of the snake superimposed over $\mathcal{H}_{[GGHDC]_3}$. This initial position was fixed interactively. In (b) is shown the optimal position for this snake when starting from the position in (a). This position is optimal with respect to the snake energy term $E_{snake_{field}}(\overline{v})$ which is then minimized. The evolution of $E_{snake_{field}}(\overline{v})$ as a function of iteration is shown in (c). Note that in (c) $E_{snake_{field}}(\overline{v})$ was normalized so that its minimum is the zero energy level.

feature, the valley of $\mathcal{H}$. At such a coarse level, the valley is wide enough because of the blurring so that the snake will be able to follow the slope of the valley and crawl toward its bottom. We assume that the initialization step, in which the snake is positioned around the cell, is done in a coarse fashion; this is a kind of least commitment assumption. In all of our experiments with cell scenes, only three levels of a Gaussian HDC were necessary (*i.e.*, $l_{max} = l^* = 3$), with sampling rates of 2^l for the first two levels (*i.e.*, $r_{\delta_1} = r_{\delta_2} = 2$) and a sampling rate of one for the last level. For other types of images, such as these with different noise levels, artifacts or contrast properties and different initialization mechanisms, other amounts of smoothing may be required.

Figure 3.11 shows the result of applying the improved snake model presented in Chapter 2 to the potential surface $\mathcal{H}_{[GGHDC]_3}$. With the GGHDC family, the directional slopes in the $\overline{X}$ and $\overline{Y}$ directions are evaluated by simple first differences of the potential surface values at each snaxel position. Also shown in this figure is how the potential field energy of the snake, $E_{snake_{field}}(\overline{v})$ is minimized by the iterations. As we can observe, as the snake crawls down the potential surface starting from its initial position shown in Figure 3.11.(a), $E_{snake_{field}}(\overline{v})$ is gradually reduced. The final position shown in Figure 3.11.(b) corresponds to the optimal one when starting from the position shown in (a). Here optimality refers to $E_{snake_{field}}(\overline{v})$ which is then at a minimum.

The same process can be repeated on the potential surface $\mathcal{H}_{[GGHDC]_2}$, but this time using the optimal solution obtained at the previous HDC level, that is, for $\mathcal{H}_{[GGHDC]_3}$, as the initial snake position. Results at this intermediate level ($l = 2$) for the recovery of the cell contour are shown in Figure 3.12. It is worth noticing that this time only five iterations

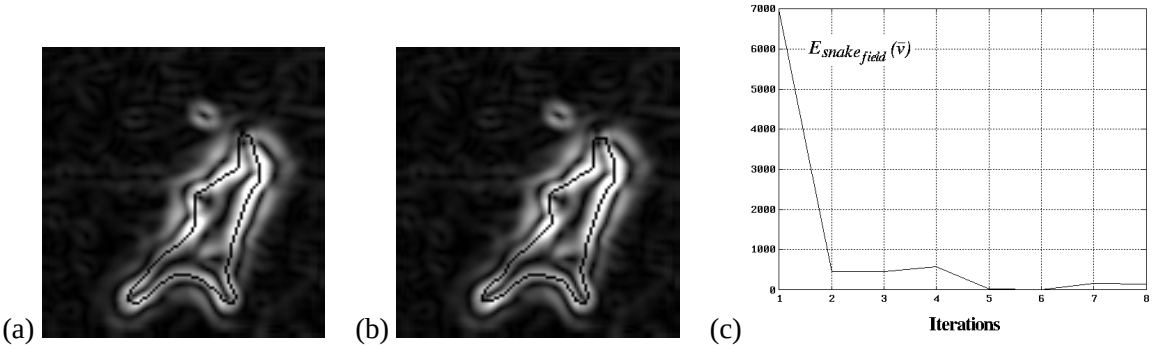


Figure 3.12: Finding the best solution at an intermediate HDC level ($l = 2$). In (a) is shown the initial position of the snake superimposed over $\mathcal{H}_{[GGHDC]_2}$. This initial position was obtained as the optimal snake position at the preceding level in the HDC family (*i.e.*, for the child $\mathcal{H}_{[GGHDC]_3}$; see Figure 3.11.(b)). In (b) is shown the optimal position for this snake when starting from the position in (a). The evolution of $E_{snake_{field}}(\overline{v})$ as a function of iteration is shown in (c) ($E_{snake_{field}}(\overline{v})$ normalized).

were sufficient to reach the minimum of $E_{snake_{field}}(\overline{v})$ (Figure 3.12.(c)) while twenty-six iterations were required for the preceding child level $\mathcal{H}_{[GGHDC]_3}$ (Figure 3.11.(c)). This is due to the fact that in the case of $\mathcal{H}_{[GGHDC]_3}$ the initial snake position was relatively far from the bottom of the valley (Figure 3.11.(a)), in comparison to the case of $\mathcal{H}_{[GGHDC]_2}$ (Figure 3.12.(a)). Then, when starting from the position shown in Figure 3.12.(a), the snake crawls down on a relatively short distance to reach its final optimal position shown in Figure 3.12.(b). Because we were initially close to the optimal solution, the deformations of the snake shown in Figure 3.12, from (a) to (b), are relatively small (compare to Figure 3.11, (a) and (b)).

Finally, for $\mathcal{H}_{[GGHDC]_1}$, the same process is repeated using the optimal result obtained on $\mathcal{H}_{[GGHDC]_2}$ as the initial snake position (Figure 3.13). Again, relatively few iterations are required (nine here; see Figure 3.13.(c)) compared to the initial case of $\mathcal{H}_{[GGHDC]_3}$. Similar to the case of $\mathcal{H}_{[GGHDC]_2}$, the initial snake position for $\mathcal{H}_{[GGHDC]_1}$ is relatively close to the bottom of the valley (Figure 3.13.(a)). Then, starting from this position, the snake crawls down on a relatively short distance to reach its final optimal position shown in Figure 3.13.(b). Being initially close to the optimal solution, the deformations of the snake shown in Figure 3.12, from (a) to (b), are relatively small as expected. Note that because we are at a low level in the family hierarchy, the bottom of the valley on which the snake crawls is relatively narrow. This explains why $E_{snake_{field}}(\overline{v})$ in Figure 3.13.(c) is not as well behaved as it was the case for the two preceding children in the hierarchy (compare with Figure 3.12.(c) and Figure 3.11.(c)).

Such a process of tracking the best solution from coarse to fine scales in a scale-space representation of the signal, here an image, defines the so-called *continuation method*

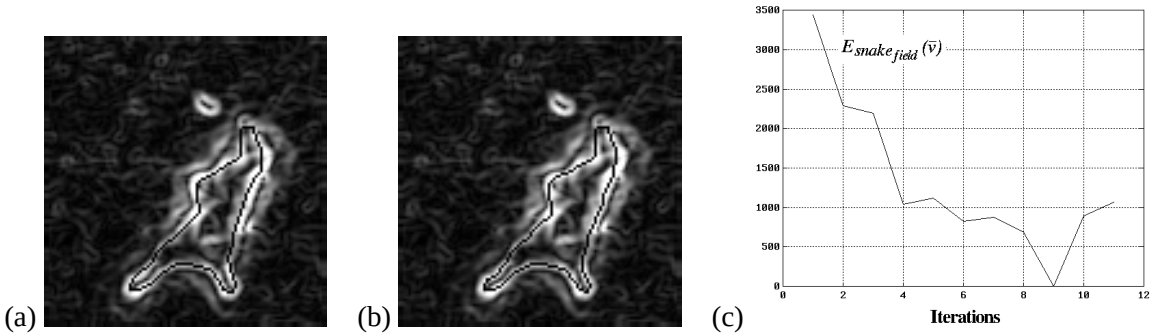


Figure 3.13: Finding the best solution at a fine HDC level ($l = 1$). In (a) is shown the initial position of the snake superimposed over $\mathcal{H}_{[GGHDC]_1}$. This initial position was obtained as the optimal snake position at the preceding level in the HDC family (*i.e.*, for the child $\mathcal{H}_{[GGHDC]_2}$; see Figure 3.12.(b)). In (b) is shown the optimal position for this snake when starting from the position in (a). The evolution of $E_{snake_{field}}(\overline{v})$ as a function of iteration is shown in (c) ($E_{snake_{field}}(\overline{v})$ normalized).

[156, 147]. We have shown here, and in other experiments we have conducted, that a discrete version of this continuation method is sufficient in most cases to recover the trace of an object in a noisy scene under appropriate assumptions.[10] Furthermore, we claim that, under the "small deformation" and the "initialization" assumptions, such a technique provides us with "good" segmentation results that satisfy our quality criterion, that is, criterion (a) in section 3.1.3. The snake model provides a line drawing representation of the cell shape which permits the experimenter or observer to produce a qualitative shape description similar to the one he obtains by looking directly at the intensity image (Figure 3.14). Although, in general, the snake model combined with a discrete continuation method gives good results, counterexamples can be given where the method we have described so far may fail. We discuss the limitations of the snake model for image segmentation in the following subsection.

3.3.4 Image Segmentation Using Snakes: A Critique

We have shown in the previous sections how the snake model can be combined with a quantized or discrete multiscale image representation to recover an "optimal" approximation of the trace of the contour of a blob-like shape; where optimality is defined in terms of the "steady-support" criterion.

An advantage as well as a limitation of the snake model emerges from the global way in

[10] It is interesting to note that such results could be in agreement with the multichannel model of human vision which may also be seen as a discrete or quantized version of a scale-space image representation, where only 4 or 5 channels are used to send scaled versions of an image to other brain processes [87].

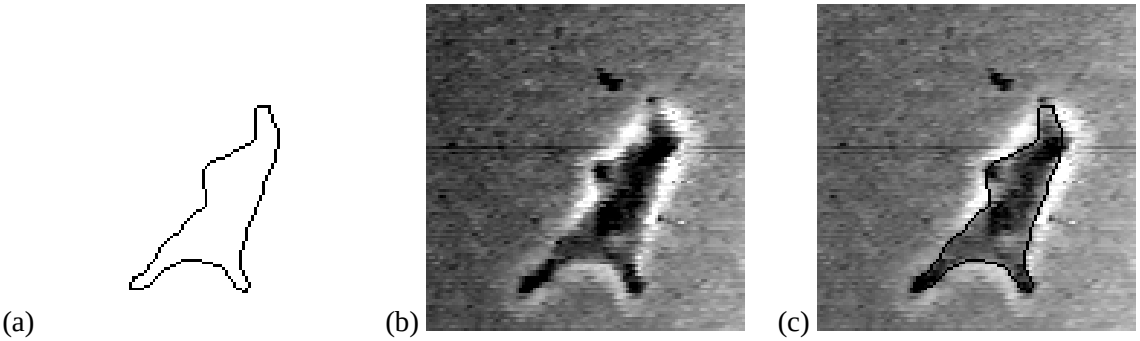


Figure 3.14: Judging the quality of the segmentation. In (a) is shown the line drawing obtained from the optimal snake position (*i.e.*, for $\mathcal{H}_{[GGHDC]_1}$; see Figure 3.13.(b)). In (b) is shown the original intensity image (*i.e.*, unblurred). In (c), the snake in (a) is superimposed on the image in (b). From (c), we claim that the segmentation obtained using the snake model is "close" enough to the perceived shape of the cell, when a human observer looks at the image in (b), to judge the segmentation good.

which an optimal solution is evaluated. This approach to the segmentation problem makes the active contour or snake seek a *global* minimum of its energy functional (equation (2.2), section 2.2). It offers the advantage of integrating information about the derived potential surface feature along the entire length of the closed snake. This is implemented by seeking a global minimum of this snake energy functional. As pointed out in Chapter 2, this is an attempt at having the snake "bridge" the gaps in regions of low contrast, along the cell contour. In other words, it should permit the snake to link two ends of deep valleys using the best "pass" between them. Note that this bridging capability is better implemented if one adopts the "steady-support" criterion rather than the original "steady-state" criterion.

The "globality" of the energy minimization process also points out a weakness related to the efficiency of the snake model for segmentation. In the original snake model we saw how the "steady-state" criterion could result in the snake missing a desired solution. Alternatively, it could force the snake to require an excessive number of iterations before reaching a stable state. It could make the snake shrink artificially by causing snaxels to pile up in deep pockets of a valley of the potential surfaced. To remedy these difficulties we proposed the "steady-support" criterion which proved to be better suited to that purpose. Still, even this more "intelligent" energy minimization criterion possesses certain limitations.

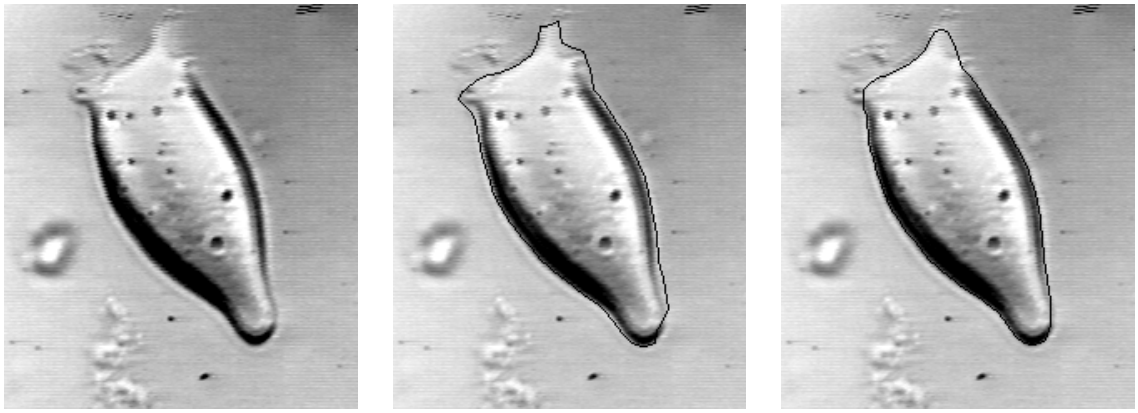
In our attempts to improve the original snake model, the "steady-support" criterion was suggested as the global energy function to be minimized. This was proposed in order to stay as close in spirit as possible to the original definition of the snake model. It was computed by summing the support of the snake (*i.e.*, its height) over its entire length (equation (2.30)). Because this minimization is performed in a global fashion, due to the summation of the support of all points of the snake contour, we cannot ensure in a strict sense that the "best"

possible solution will be retained during such an iterative minimization process. This is because we have no efficient way of discriminating a "good" individual snaxel displacement from a "bad" one. By "good" we are referring to a snaxel movement that reduces the energy $E_{snake_{field}}(v)$. Similarly, "bad" refers to a snaxel movement that increases the energy $E_{snake_{field}}(v)$. Therefore, the effect of snaxels which make $E_{snake_{field}}(v)$ increase *could* be counteracted by the motion of other snaxels along the snake contour which would make $E_{snake_{field}}(v)$ decrease by a larger amount. Thus, the summation of both effects would result in a better energy value although, locally, only the latter of the two effects is desirable.

Figure 3.15 illustrates how such a malfunction of the snake optimization process might occur. Here contour regions of low contrast exist at the tip of cell pseudopods. The corresponding segment of valley in the potential surface is at a relatively high height with respect to its neighborhood, since we are at a pass between two relatively deep valleys. Elsewhere along the snake contour, snaxels may be positioned on the slope of relatively deep valleys, such as when a snake is initialized using the solution obtained from another potential surface. Thus, a first group of snaxels positioned at the tip of the pseudopod may have a tendency to move down one or both deeper sides of the pass. In such a case the corresponding continuous snake, obtained by linking snaxels by straight lines, would be forced to cross a plateau rather than the pass itself, thereby increasing $E_{snake_{field}}(v)$ (Figure 3.15). The second group of snaxels moving down the slope of a deep valley might mask this local increase of $E_{snake_{field}}(v)$ at the tip of the pseudopod. Therefore, the locally better approximation of the trace of the cell contour at the pseudopod tip could be lost due to an overall decrease of the global snake potential field energy.

The limitation of the applicability of the "steady-support" criterion as first defined in Chapter 2 represents our best attempt within the scope of this thesis, that is, under the constraint of assumptions 1 and 2 of section 3.1.2, to solve the image segmentation problem. With this stability measure we were able to achieve relatively good segmentation results with an implementation which lent itself to real-time applications although, as we have seen, we cannot certify that the snake will find in all cases the solution we seek. Thus a complete understanding of this problem should provide a general method for refining the "steady-support" criterion. What is missing is some "local" constraint on the global energy function $E_{snake_{field}}(v)$. The notion of support should be refined to become more descriptive of what is meant by "good" local support for each snaxel and its intermediate joining segments. We have defined a simple but excessively "coarse" definition of support based solely on the notion of height or potential energy. A more complex model should incorporate a measure of the shape of the bottom of a valley or a pass. That is, we should evaluate the support of a candidate contour point not only with respect to a global potential energy but also with respect to the actual local topography or shape of the surface on which the point lies. However, maintaining optimality in the strict sense remains a difficult problem.

A few different implementations of the snake model and other similar active contour models have been proposed in the recent literature in an attempt to solve some of the short-



(a) (b) (c)

Figure 3.15: Example illustrating where the minimization process could fail due to its global definition. In (a) is shown the original image (note the two pseudopod tips at the top of the cell which have relatively low contrast boundaries). In (b) is shown the solution obtained from a previous frame. In (c) is shown the final solution. We note that the pseudopod at the top-left of the cell is almost completely lost by the snake, while the pseudopod on the top-right of the cell is partially recovered.

comings of the snake model or to provide possibly better models. In his "alternative snake", Scott [134] defines a snake energy function E_{snake} in the Fourier domain. Although he claims his snake gives him a more powerful contour representation, where the shape is summarized by the Fourier coefficients, his model also lacks the ability to recognize locally significant features: "the snake cannot [always] smell" [134] all of the significant contour points. Staib and Duncan [137] also propose a parameterization of the snake in the Fourier domain. Leclerc and Fua [84, 66] use an active contour model similar to the original snake model. They rely on the direction of the gradient to make the snake active, as is the case with the Kass *et al.* model [78]. Amini *et al.* [3, 2] use the model of Kass *et al.*, but with a different implementation of the minimization procedure based on a dynamic programming framework. Their model relies on the direction of the gradient of the potential surface to activate the snake.

All of these other similar techniques share the same lack of ability to accurately sense the local shape of the potential surface. Therefore, aside from the fact that they also are constrained under assumptions 1 and 2 (§ 3.1.2), they are also limited in their applicability.

3.4 Tracking Deformable Shapes

After having demonstrated how the snake model can be used for image segmentation to recover the trace of cell contours in noisy images, we now show in the following paragraphs how the same model leads to an automatic tracking method for deformable objects such as cells. We also present the limitation of this method for cell tracking.

The image motion sequence we will analyze consists of an ordered list of frames: $f_1, f_2, f_3, ..., f_N$. For the purpose of the present demonstration, a sequence containing $N = 120$ frames was used as a test (Figure 3.16). This sequence was obtained from a time-lapse recorded video tape. A relatively fast moving cell was selected for this illustration of the snake model applied to tracking. It was only necessary to use a subset of the full sequence. One frame out of ten was used to perform the tracking with the snake. Therefore, only frames $f_1, f_{10}, f_{20}, ..., f_{120}$ were used, in total 13 frames. Figure 3.16 shows only five of these selected frames to illustrate the deformation and motion of the cell during this sequence of $N = 120$ frames. This same sequence was used in Figure 3.3 as an example of a deforming cell.

3.4.1 The Initial Frame

Processing the initial frame (f_1) to recover the trace of the cell contour is performed using assumptions 1 and 2 and the discrete continuation method on a HDC family as described in section 3.3. Once an optimal solution is found at a fine scale, that is, for a HDC level $l = 1$, it is used as the input or initialization data for the tracking procedure; in other words, it is used for processing the following frame in the selected sequence (here frame f_{10}).

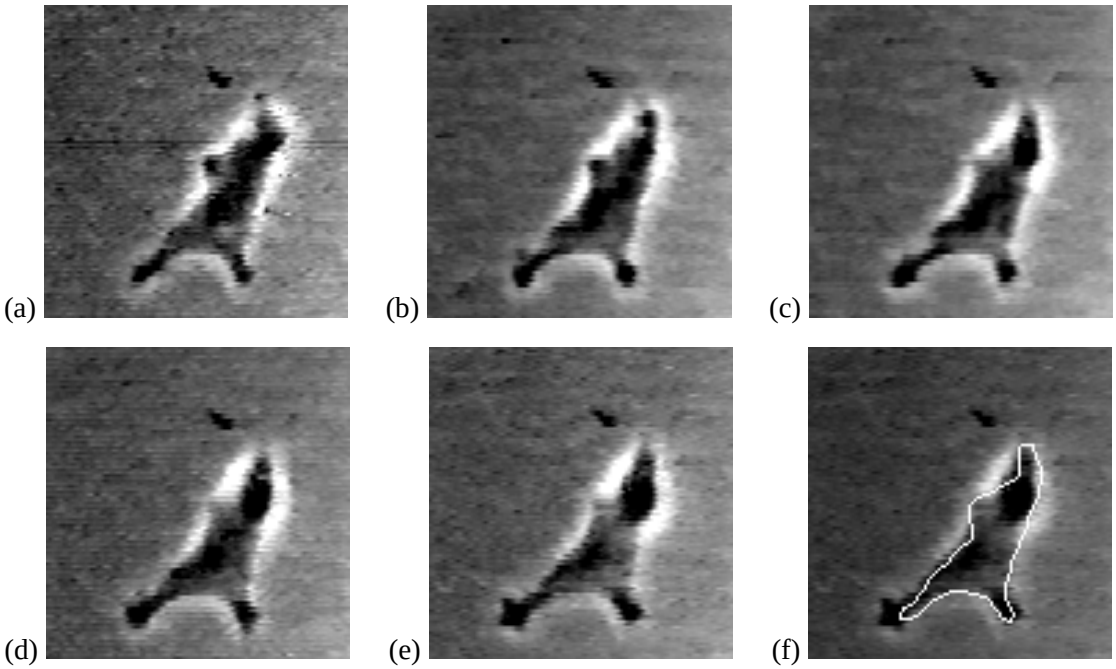


Figure 3.16: The motion sequence used to demonstrate the ability of the snake model to perform the tracking. From (a) to (e) is shown the cell in frames f_1, f_{30}, f_{60}, f_{90} and f_{120}, respectively. In (f) the cell in frame f_{120} is again shown, but with the "optimal" snake position found for frame f_1 superimposed on it. This illustrates the actual deformation that occurred during the $N = 120$ frames sequence.

3.4.2 The Following Frames

Let us consider a given frame f_n ($1 < n \leq N$). For simplicity we assume that there is only one window to be processed. If multiple cells require that more than one window must exist, the same procedure will simply be applied to each window. The initial snake position is obtained from the optimal solution found in frame f_{n-1}. Thus the first step consists of obtaining a new family, $\mathcal{H}$, of potential surfaces for the frame f_n. As we explain in the following paragraphs, the filtering in a dynamic sequence can be performed in a much more efficient way than in the static case.

Because we are tracking relatively slow objects, and because of the assumption of "small deformations" less smoothing will be required to permit the snake to seek the new trace of the cell contour. For example, in the test sequence, we generated families of potential surfaces up to level $l_{max} = 2$ only, for frames f_n following the initial one (*i.e.*, $n > 1$).

Furthermore, the filtering used to generate the potential family $\mathcal{H}$ in each frame, f_n, can be restricted to a much smaller area than the entire cell window. We can use the initial snake position obtained from frame f_{n-1} to delimit an area in the vicinity of this closed curve in order to spatially restrict the filtering. Here again we can use the HDC method to our advantage since the filtering computations within the HDC context are always local. Thus it is not necessary to process the full image in order to complete an HDC computation [38].

Figure 3.17 shows the results of the tracking using the snake model for five of the thirteen frames of the analyzed sequence. All deformations which occur are seen to be tracked with good accuracy; this result satisfies our quality criterion 2 (§ 3.1.3). This illustrates the power of the snake method for tracking objects.

3.4.3 Limitations of the Snake Model for Cell Tracking

In this section we analyze a typical problem that illustrates the limitations of the snake model for cell tracking in 2-D.

Figure 3.18 shows a series of four frames selected at every ten frames following the sequence previously presented in Figure 3.16 (*i.e.*, frames f_{130}, f_{140}, f_{150} and f_{160}). The growing pseudopod at the bottom of the window (South side) appears to be forming in a noncontinuous fashion since a slightly dark oval region first appears at the tip of the pseudopod in frame f_{130}. In the following frames, this region becomes even darker and darker until it merges with the rest of the pseudopod in frame f_{160}.

Such discontinuous pseudopod formation does not correspond to the accepted biological growth process which of course can only be continuous. What we are emphasizing here is an imaging artifact which is a result of the way in which we look at the cell. We are actually observing the cells in only one focal plane, assuming that they are flat (2-D) objects. However, this does not exactly correspond to reality since the cell is really a 3-D entity. What is happening in the present case is that the pseudopod *continuously* grows above the

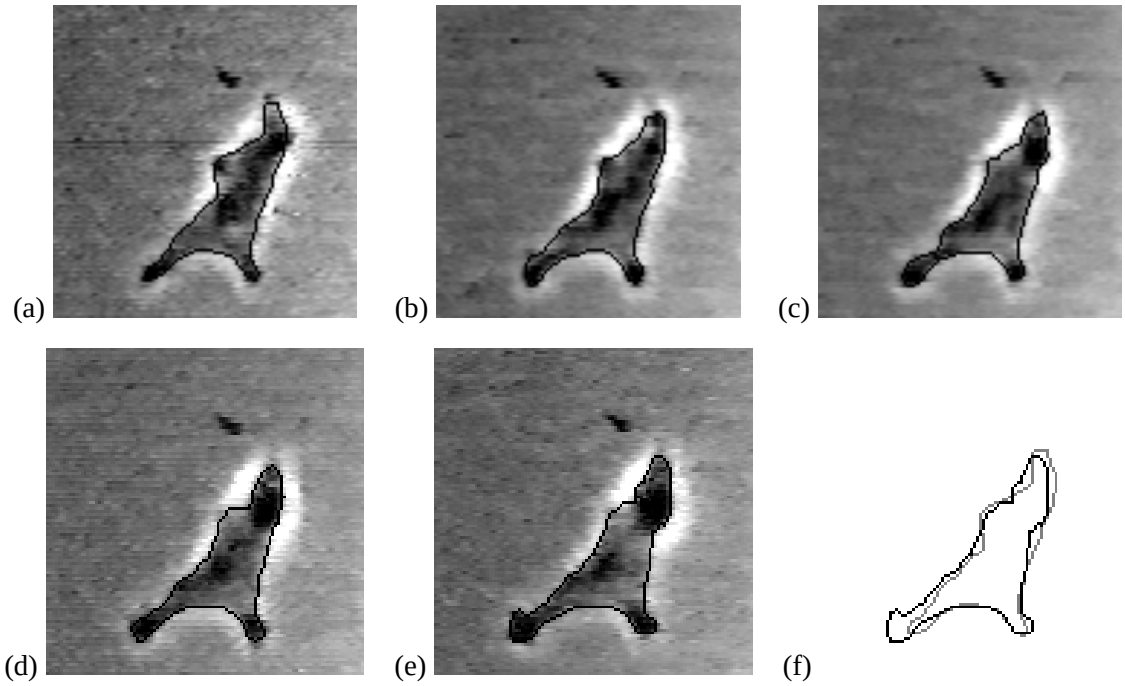


Figure 3.17: Cell tracking. All frames have their optimal snake superimposed on them to show the extracted contour. The results for frames f_1, f_{30}, f_{60}, f_{90} and f_{120} are shown in (a), (b), (c), (d) and (e), respectively. In (f) are shown the extracted contours in frames f_1 (grey dotted contour) and f_{120} (dark connected contour) to illustrate the total deformation that was recovered from the observed sequence.

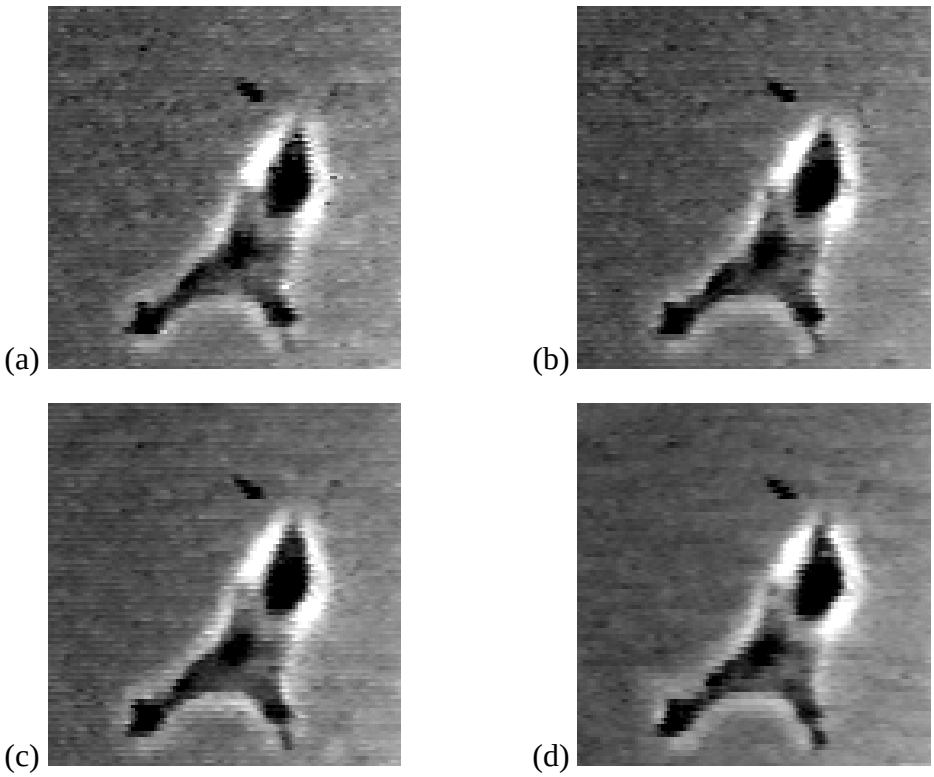


Figure 3.18: Illustration of the limitations of the snake model for tracking. Frames f_{130}, f_{140}, f_{150} and f_{160} are shown in (a), (b), (c) and (d), respectively.

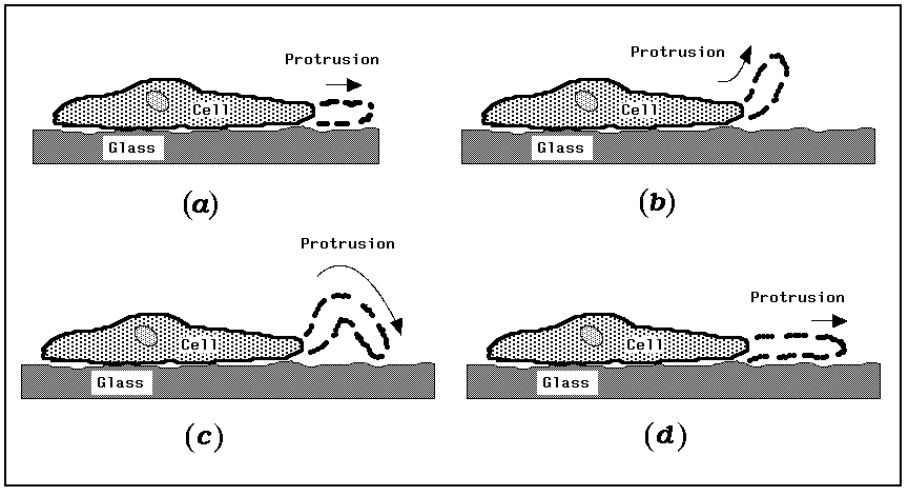


Figure 3.19: How a pseudopod, or protrusion of the cell's body, might form above the glass surface and consequently out of the focal plane.

focal plane until its tip comes back in contact with the glass upon which it is constrained to move. This is illustrated in Figure 3.19.

Although assumption (a) of "small" deformations is not violated in reality, that is, the cell really deforms continuously, but in 3-D, it is violated in the image sequence which only represents one 2-D slice of the 3-D world. Therefore we expect the snake to fail to track such deformations. This is indeed what we observe (Figure 3.20). In frame f_{130} the snake remains on a contour near the end of the pseudopod, that is, where the pseudopod quits the glass surface or the focal plane. The snake behaves in the same way in subsequent frames (f_{140}, f_{150}). Thus it remains blind to the actual 3-D deformation process of the pseudopod until the complete pseudopod tip returns to the focal plane in frame f_{160}. In this last frame the snake segment in the vicinity of the pseudopod tip is now situated on a plateau of $\mathcal{H}$. Thus no contour segment exists in the image in correspondence to these snaxel forming this snake segment. Because these snaxels are now too far from the tip of the valley to sense it, the snake has no way to detect that it is wrongly positioned. Furthermore, these snaxels are immobilized by tension forces since this snake segment is now like an elastic stretched over the plateau.

In a sense the snake is rather blind in its search for the best valley. Rather than sensing the potential surface, $\mathcal{H}$, by literally looking at its shape or topography to recognize the bottom of its valleys when crawling on it, the snake relies on sensing only the slope of $\mathcal{H}$ in a simple manner in the vicinity of each snaxel. Again, as pointed in section 3.3.4, we see that we would need some kind of more powerful way to characterize the bottom of a valley in order to recover this kind of feature in a potential surface.

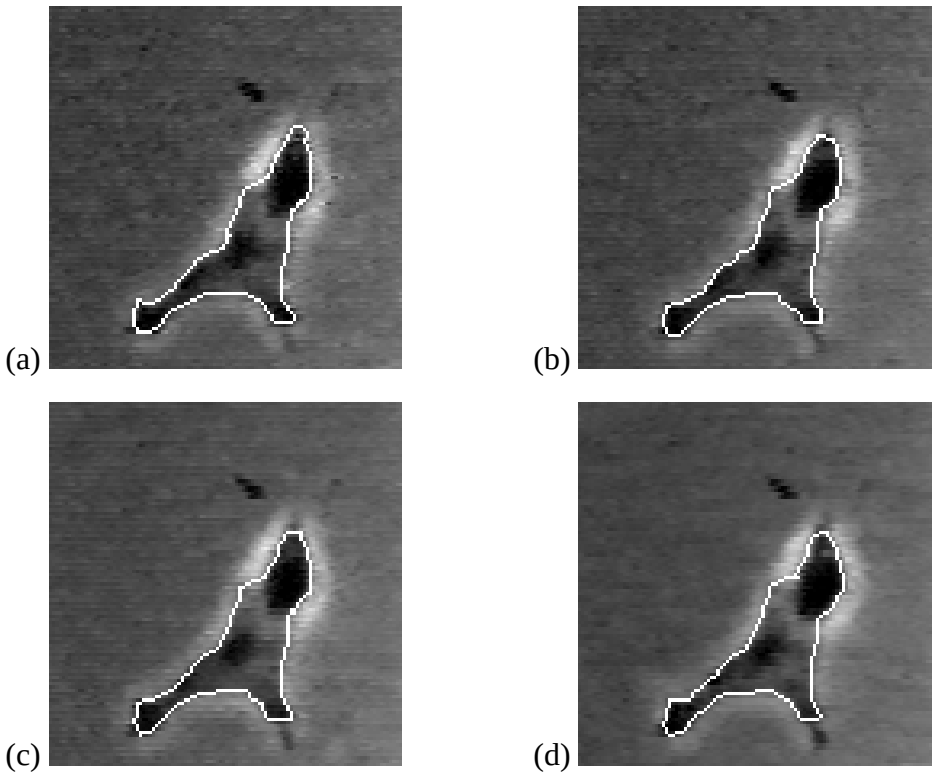


Figure 3.20: Illustration of the snake being unable to track the growing pseudopod in the lower right part of the cell. Frames f_{130}, f_{140}, f_{150} and f_{160} are shown in (a), (b), (c) and (d), respectively.

3.5 Conclusions

In this chapter we have shown the abilities and limitations of the snake model for the image segmentation and tracking of deformable objects such as cells. We have shown that under appropriate assumptions, which limit the extent of the deformations and permit user initialization, the snake model is powerful enough to provide solutions to both the segmentation and tracking problems.

In order to obtain potential surfaces to be used by the snake in an efficient manner, we have also described how the HDC method was well suited for image filtering and image representation at multiple scales. Furthermore, this leads to a discrete scale-space representation to which a continuation method can be applied in a natural way to recover image features for which their natural scale was unknown *a priori*.

We have also demonstrated the limitations of such an approach based on the idea that computer vision problems may be embedded in a variational scheme which in turn can be solved by a discrete optimization process seeking to minimize some global energy function. We discussed a counterexample to the snake model which showed its limitations for actually recovering or keeping track of "good" segmentation solutions. We have also demonstrated that the snake is relatively "blind" in its search for image features in the dynamic case, when the feature formation can occur in a noncontinuous fashion. These failures of the snake model have lead us to the conclusion that the recovery of the trace of a contour in an image by seeking the bottom of valleys in a potential surface, needs a more powerful sensing mechanism than the simple gradient descent search favored by the snake approach and by the other similar active contour models.

There is yet another path that could be followed in the dynamic case to analyze a sequence of images. Since we have the results of the previously analyzed images available, we could make better use of them. Instead of simply considering the trace of a contour in the immediately preceding frame to help us in our search for future deformations of that contour, we could make use of more complex information. For example, we could use the curvature extrema of the contour and other shape features of the deforming object. Furthermore, this kind of information could be integrated over many frames to detect growing and shrinking parts of the cell in order to predict possible future deformations. This information could then be used to perturb or deform the snake when searching for a new solution. In addition, the integration of information over many frames could lead to the possibility of eliminating extracted contours in frames where they do not correspond to or match preceding and subsequent extracted contours.

In light of the above comments regarding descriptors of deformable objects other than simply the trace of the contour, we next address two complementary issues in object description: contour analysis and region analysis. In the following chapter we consider the extraction of contour features such as curvature extrema. In a subsequent chapter we will examine the subject of region representation and analysis.

Chapter 4

Shape Features using Curvature Morphology

4.1 Introduction

Shape representation and analysis is fundamental to computer vision. Within the scope of this thesis, our interest in it is an outgrowth of a study of the dynamic changes in cell shape [115], coupled with the need for more informative shape descriptors than simply the outline of a shape.[1] Of the many approaches to shape that have been proposed, the notion of curvature of planar curves has emerged as one of the most powerful for the representation and interpretation of objects in an image [87, 14, 100]. Curvature is a measure of the rate of change in orientation at each point along a curve. There is psychophysical, physiological, as well as computational and mathematical support in favor of using curvature as a representation for contours. Curvature extrema seem to be used by the human visual system to segment contours into meaningful parts [15, 21, 75]. Endstopped neurons in the visual cortex can be interpreted as performing local curvature measurements [54]. Local estimations of curvature and tangent information are sufficient for the recovery of the trace of a curve in an image [120]. From differential geometry, the fundamental theorem of the local theory of curves states that any regular planar curve is uniquely defined by its curvature [53] — regularity implies continuity of a curve and its derivatives. Non-regular points of a curve are *singular* ones where, for example, the curvature goes to infinity, that is, where the change in orientation is undefined, or where there exist a break or step in curvature. They often correspond to visually salient contour features such as corners or protrusions. Therefore, given the finite set of singular points of a contour, some of which can be represented as extrema and steps in curvature, as well as the curvature values for all intervening points, a contour is uniquely defined. This representation by curvature is invariant to rigid motion,

[1] An early version of this chapter was first published as a technical report [90]. Two shorter versions were also published recently as conference papers [91, 93].

that is, with respect to translation or rotation.

4.1.1 Contour Characterization by Curvature Features

Curvature extrema can be used to characterize the outline of a shape, but not all of them will be perceptually salient. Depending on their importance or significance along the contour, curvature extrema will be perceptually selected or noticed. We embody the term "curvature extremum significance" by appealing to two complementary notions: the *relative amplitude* or *curvature measure* and the *region of support* [90]. Both measures are defined in the neighborhood of each curvature extremum, along the contour. The "region of support" defines how well an extremum is isolated from other curvature extrema [90, 140]. As we will see in section 4.3, this notion of isolation or localization is closely linked to the scale or size of this curvature feature.

Although significant curvature extrema can be used to summarize a shape, they are not the only useful curvature features for this purpose. Other effective curvature features include: the first order discontinuities or "steps" in curvature, the arcs of constant curvature,[2] and the zero-crossings of curvature.

These four types of curvature features can in turn be shown to correspond to specific contour features which characterize the outline of a shape. Let us first consider curvature extrema. In the discrete domain we define curvature extrema to consist of any significant peaks of the discrete smoothed curvature function, $\overline{k}_\sigma$ (see section 4.2 for more details). For each of these peaks the rate of change of the discrete orientation function to the contour is maximized at a single contour point, $\overline{S}_{max}$. Three cases are possible. The first case is characterized by any contour point where the orientation changes smoothly but with the highest speed relative to a finite neighborhood on the contour. This corresponds to the definition of a curvature extremum in the continuous domain (Figure 4.1.(a), on the left). The second case is denoted by a contour point where there exists a break or step in orientation and which is flanked by contour segments having curvature functions of the same sign. Three subcases exist: *(i)* both segments are of positive sign (Figure 4.1.(a), central picture), *(ii)* both segments are of negative sign (Figure 4.1.(c), central picture), *(iii)* both segments are of "no-sign" (*i.e.*, zero curvature or straight line segments). The third type of curvature extremum is defined by a contour point where there again exists a break or step in orientation, but which is flanked by contour segments having curvature functions of the opposite sign. Then, six subcases exist which correspond to the permutations of the pairs formed by positive $(+)$, negative $(-)$ and zero (0) curvature segments (*i.e.*, the pairs $(+,-)$, $(-,+)$, $(0,+)$, $(+,0)$, $(0,-)$ and $(-,0)$; Figure 4.1.(a), on the right, shows the case of $(-,+)$ for a positive curvature extremum, while Figure 4.1.(c), also on the right, shows the same case

[2] Note that with respect to the other curvature features which are localized at single points (*i.e.*, 0-D objects), arcs of constant curvature are further differentiated by the fact that they are defined over a connected set of points (*i.e.*, 1-D objects).

for a negative curvature extremum). Note that in the continuous domain the last two cases would actually be described as curvature discontinuities. But, since in the discrete domain we lack the same notion of continuity, we do not need to distinguish them. Extrema can further be classified into two subclasses depending on the way in which the change in orientation occurs. If in the neighborhood of an extremum location on the contour, the tangent to the contour, or orientation function (section 4.2), rotates by enveloping the contour, that is, from the exterior of the shape, a *corner* or *protrusion* is obtained (Figure 4.1.(a)). On the contrary, if at the extremum location the tangent to the contour rotates inwardly towards the shape, a *concavity* or *depression* is obtained (Figure 4.1.(c)).

The second type of curvature feature, the "step" in curvature, corresponds to a *smooth join* [14], $\overline{S}_{smooth}$, where the orientation to the contour varies smoothly, but not the curvature. Smooth joins can be classified into two types. Firstly, if the jump in curvature occurs between two flanking curvatures of the same sign we call this curvature feature a *smooth join of the first type*. Two subcases exist; Figure 4.1.(b) shows the case of two positive flanking curvature segments. Secondly, if the flanking curvatures are of different sign, we call it a *smooth join of the second type*. Again, two subcases exist; Figure 4.1.(d) shows the case of a negative flanking curvature segment followed by a positive one. The third curvature feature, an arc of constant curvature, can again be classified into two types. A non-zero curvature arc corresponds to an *arc of a circle*, while a zero curvature arc corresponds to a *straight line segment*. Finally, the fourth curvature feature is the zero-crossing of curvature which corresponds to an *inflection point* of the contour.[3]

4.1.2 Chapter Overview

In a typical computer vision system, discrete traces of contours of objects are first extracted from an image. The curvature of these discretized contours is then approximated and used to detect important features of the boundary of an object. Although curvature extraction from an object contour would seem to be a rather simple task, few methods exist that are simultaneously easy to implement, fast and reliable in the presence of noise. In this chapter we will first propose a scheme for obtaining the discrete curvature function of opened or closed planar contours based on the *chain code* representation of a boundary [63]. This approach has been previously reported and we emphasize only our own attempts to optimize its implementation.

We will also present a new method of extracting important features from the curvature function. We are interested in localizing features such as peaks of curvature and arcs of

[3]Note that here we simplify the definition of an inflection point to occur at any point were the curvature changes its sign. This is valid in the discrete domain. In the continuous domain, there is a second condition: at an inflection the curvature has to be analytic (*i.e.*, regular point) [37]. For example, a smooth join of the second type is necessarily also an inflection point (even in the continuous domain), while, in the discrete domain, a corner and an indentation can possibly be situated at an inflection point (Figure 4.1, (a) and (b)).

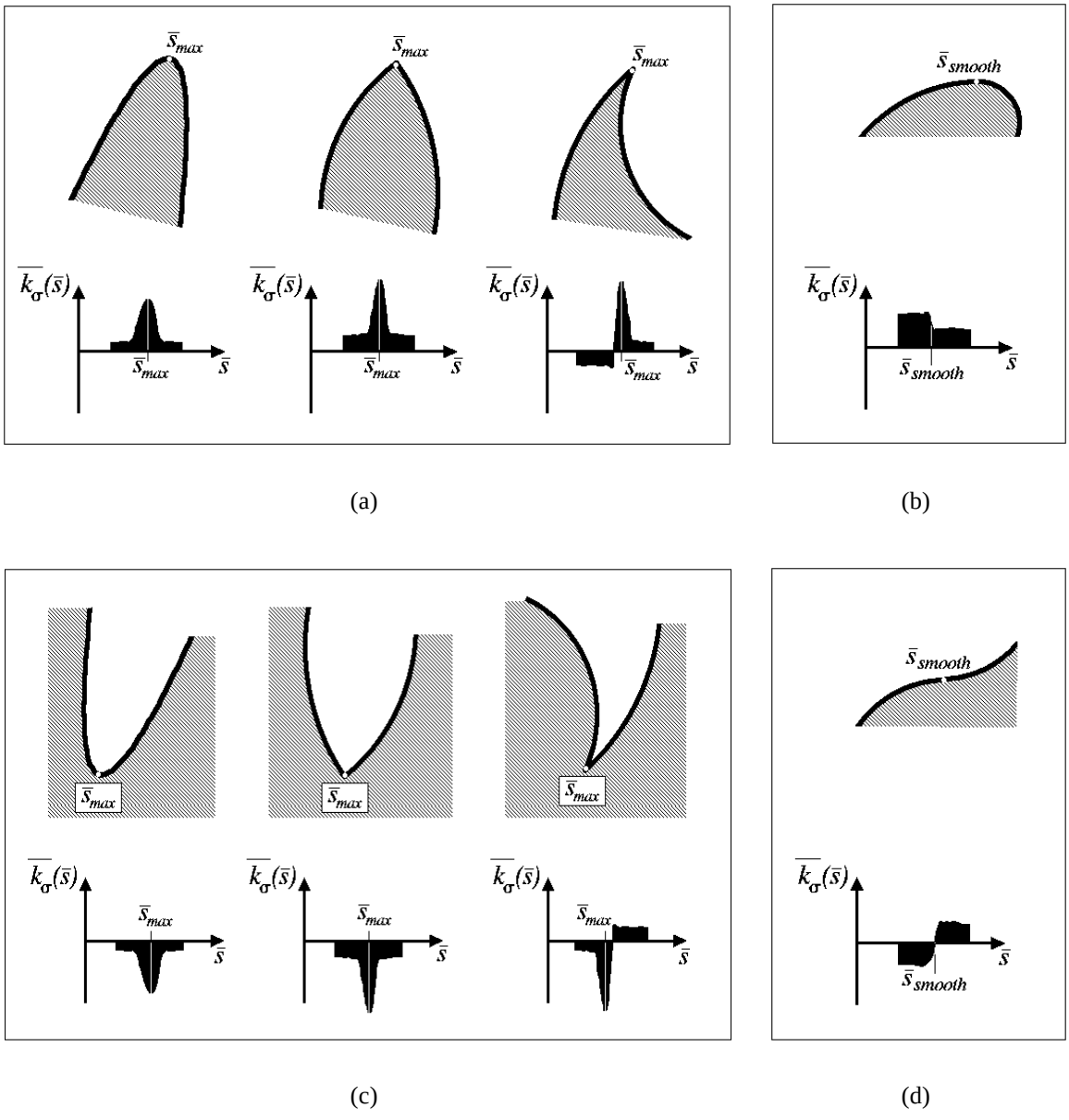


Figure 4.1: Some curvature features and their contour counterparts in the discrete domain: (a) protrusion, (b) smooth-join of the first type, (c) depression, (d) smooth-join of the second type. Only segments of a contour are shown. Contour pixels are traversed in a fixed counterclockwise direction. Therefore, the interior of the object delimited by the contour is defined as being to the left of the traversal direction (shaded area).

constant curvature. Furthermore, we would like to differentiate these features based on their relative significance, that is, by their degree of isolation from nearby features and their relative amplitude. Consequently, we seek methods for segmenting the curvature function into its basic and significant events. To achieve this goal we propose using morphological operations on functions [135]. These operations permit us to create a representation of curvature, not only in terms of feature localization and identification, but also in terms of significance and scale. We will demonstrate that morphological operators can be used to locally remove insignificant details of a signal, in our case the curvature function. This can be applied to an increasing sequence of sizes without blurring the shape of these details and while preserving the global shape features. This property of morphological operations permits us to suggest a new scale-space representation for curvature referred to as the *Morphological Curvature Scale-Space*. Advantages over the usual scale-space approaches [155, 14, 108, 109] will be presented.

In the following sections, we present a procedure for retrieving the discrete orientation and curvature functions of the discrete trace of a contour from a chain code representation. In this context, quantization errors and "protrusion-depression definition" problems inherent to the chain code representation are discussed. We also indicate how smoothing and differentiation of the discrete orientation data should be performed to extract the curvature function, while at the same time minimizing noise effects. We then discuss the morphological operations that are applicable to curvature analysis. Certain morphological measures are also introduced for the purpose of describing curvature peaks extracted using these morphological operations. The Morphological Curvature Scale-Space is then defined and its main advantages are demonstrated.

4.1.3 Contributions

The following contributions on contour segmentation are made:

- We demonstrate a problem inherent in the chain code representation which was not previously reported. We call it the "protrusion-depression definition" problem. We also propose a procedure to detect its occurrence and a means of eliminating the possibility of errors (section 4.2.2).
- We propose to apply morphological operators to the curvature function to extract its main features. These particular operators are called "hat transforms" (section 4.3.1).
- On the basis of the morphological operations performed on the curvature function of a contour, a new scale-space representation called "Morphological Curvature Scale-Space" is introduced. This multiscale representation of the curvature function possesses a number of advantages over the more traditional scale-space representations; it satisfies the three criteria of "causality," "immediate localization" and "piecewise smoothing" (section 4.3.3).

A number of less important contributions are also made:

- We emphasize two quantization problems inherent in the chain code definition first mentioned by McKee and Aggarwal [105], but not often subsequently considered in the literature (section 4.2.2).
- We describe how to efficiently filter, by smoothing, the orientation signal obtained from the chain code to compute the curvature function. We also emphasize that the smoothing must be performed in a conservative way to retain as much as possible of the relevant details of the curvature signal (section 4.2.3).
- We define four morphological measures to describe extracted peaks of the curvature function (section 4.3.2).

4.2 From Discrete Contour to "Not Too Smooth" Curvature

Since any visual input to a computer is discrete as a result of quantization, contour extraction must address the issue of synthesizing a continuous representation from a discrete one [101]. This involves going from the discrete trace of an image contour (*e.g.*, edge pixels) to a continuous or linked set of points through which the curve passes. Essentially, two kinds of encoding and representation schemes for smooth curves have been proposed in the computational vision literature: curve fitting algorithms[4] and orientation chaining algorithms.[5]

In the first approach, a set of curves is fit to the contour by minimizing a certain error measure. The points where curves meet are retained as knot points and are essentially arbitrary. Most such techniques employ polygonal approximations, circular arcs, or splines. Their main drawback is that the knot points are not always related to our perception of the salient points of a contour, such as for example the curvature extrema. Furthermore, the *a priori* assumption made about the nature of contour segments between knot points, that is, that they are composed of straight lines, circular arcs, or splines, can often be seen to be rather rigid.

In the second scheme, an opposite approach is adopted where orientation or curvature along the contour is first approximated and then processed further to extract the knot points. These points are usually obtained by examining the rate of change of orientation (curvature function) along the contour. Curve segments can then be fitted between the knot points. Since this representation seems to be supported at the physiological level and is also ad-

[4]The following constitutes a non-exhaustive reference list of "curve fitting algorithms" covering the last fifteen years of research work on such methods: [124, 123, 151, 43, 70].

[5]The following constitutes a non-exhaustive reference list of "orientation chaining algorithms" covering the last twenty-five years of research work on such methods: [63, 105, 64, 36, 14, 116, 107].

vantageous at the computational level, we have chosen it to encode and represent discrete contours.[6]

4.2.1 Curvature Function Extraction

Two different approaches have been reported for computing curvature of a discrete curve from orientation: filtering and differentiation (FD) methods; and difference of slopes (DOS) methods. FD methods evaluate smoothed local curvature by convolving a discrete orientation representation of the contour with a template. The discrete orientation representation is usually given by the well-known chain code representation of Freeman [63]. The derivative of a Gaussian of variable size is often employed as the template in order to extract an approximation of the curvature [14]. Alternatively, the orientation data can be differentiated by computing the first difference of the angle formed by nearby points and then smoothed with a Gaussian filter to reduce noise effects. This can yield a multiscale representation if a set of different sized filters is used [52]. However, this alternate method usually has a poorer signal-to-noise ratio. This can be attributed to a specific reason: filtering discrete orientation data with a template that directly extracts the derivative greatly amplifies noise.

In DOS methods the curvature is estimated at each point by taking the angular difference between slopes of two line segments fitted to the data before and after each point [133, 65, 116, 117]. This can be repeated for line segments of varying length.

For both FD and DOS methods, the extent of data smoothing is governed by the size variable; the template size for FD's and the line size for DOS's. DOS methods yield good results in the presence of boundary noise, but are computationally more expensive than FD methods. In this chapter, we will propose an alternative FD method which requires less complex computations than most current methods. But first, in the following subsection, we describe how the discrete orientation representation is obtained.

4.2.2 From the Trace of the Discrete Contour to a Discrete Orientation Representation

Let us consider the discrete trace of a curve on a square sampling lattice with integer coordinates $(\overline{x}, \overline{y})$ and denote it by $\overline{v}(\overline{s}) = (\overline{x}(\overline{s}), \overline{y}(\overline{s}))$, with parameter $\overline{s}$ (spatial index). Because $\overline{v}$ consists of a connected set of points, the spatial step-size $\Delta\overline{s}$ between successive contour points should correspond to the distance between centroids of two neighboring pixels. On a square sampling lattice, $\Delta\overline{s}$ should take the value 1 for vertical or horizontal neighbors or the value $\sqrt{2}$ for diagonal neighbors.

[6] Some recent methods proposed in the literature can be seen as a compromise between the two above-mentioned schemes. For example, Medioni and Yasumoto process cubic B-splines fitted to a contour by examining their curvature. Extrema of curvature are then extracted and the fitting process is refined [106].

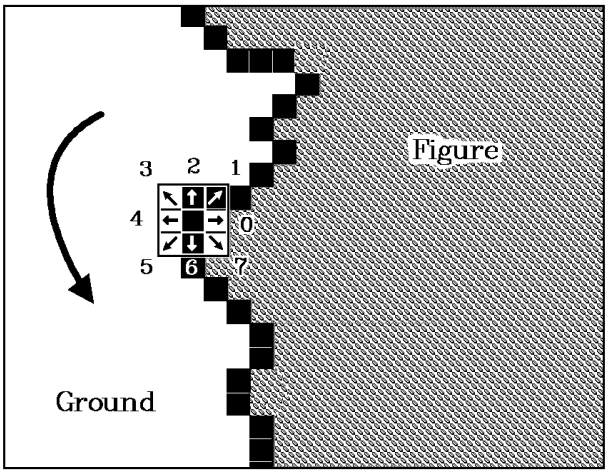


Figure 4.2: Example of chain code generation.

Given a discrete trace of a curve such as $\overline{v}$, a common first step is to consider the chain code [63] of the bounding contour. We observe that the chain code is in fact a discrete representation of the orientation along the contour. It can be stated as a vector of integers $C(\overline{s})$, where each entry represents the discrete angle formed by adjacent pixels. Due to the nature of the square sampling lattice, angles in the chain code are constrained to a discrete range of eight values (0,...,7) representing multiples of 45^o. Contour pixels are traversed in a fixed counterclockwise direction.[7] Therefore, the figure or interior of the object is defined as being to the left of the traversal direction (Figure 4.2).

Two quantization problems inherent in the chain code definition must be solved before embarking on further processing [105]. First, the angle quantization in such a narrow range (0,...,7) may introduce angle discontinuities of more than 180^o when, for example, a 0 follows a 7. The solution here is to avoid such discontinuities by using a modulo 8 operation that puts a bound of 180^o on angle discontinuities. For example, replace 0 by 8, if 0 follows 7. This *modified chain code*, C_m, is generated as follows:

- $C_m(0) = C(0)$;
- if $|C_m(\overline{s}-1) - C(\overline{s})| > 4$ then $C_m(\overline{s}) = C(\overline{s}) + 8c$, s.t. $|C_m(\overline{s}-1) - [C(\overline{s}) + 8c]| \leq 4$;
- otherwise, $C_m(\overline{s}) = C(\overline{s})$.

The second quantization error is introduced by the discretization of the index $\overline{s}$, which is usually simply incremented by one for each new pixel found while traversing the contour.

[7] The direction of traversal is arbitrarily fixed as clockwise or counterclockwise. Either choice gives similar results. It is essential to remain consistent throughout the traversal of the contour points and the assignment of the chain codes. Otherwise, the shape of the object could be erroneously interpreted (cf. the "protrusion-depression" problem).

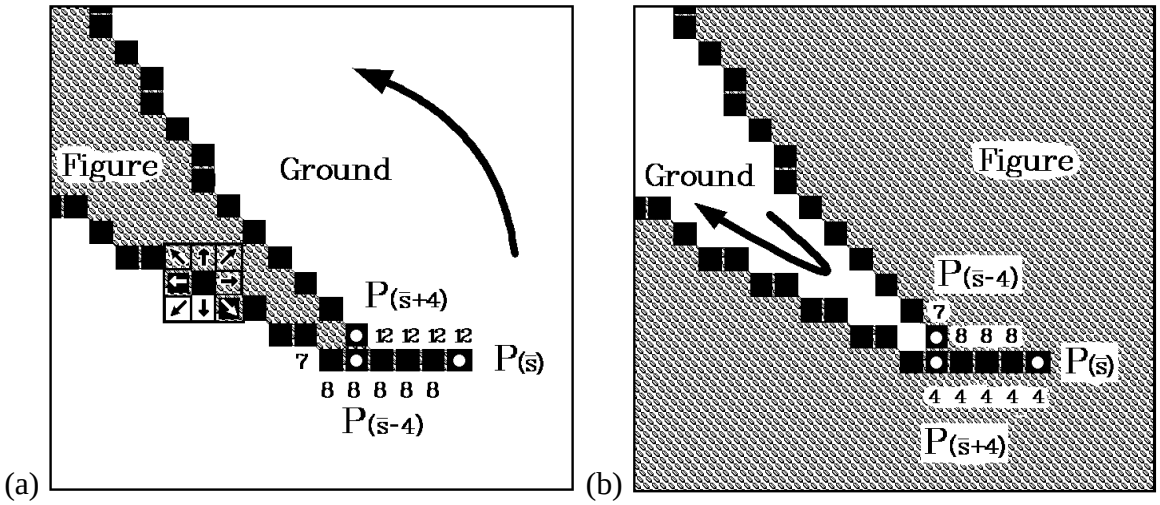


Figure 4.3: Protrusion-depression definition problems. A protrusion is shown in (a) where an angle of $\pm 180^{o}$ occurs at the contour pixel $P(\overline{s})$. The right choice for the chain code value is 12. If the usual chain code definition were used, a value of 4 would be obtained, leading to an inconsistent definition of the tip as being part of a depression, as shown in (b). Using our proposed procedure, since pixel $P(\overline{s}+i)$ lies to the left of pixel $P(\overline{s}-i)$, a protrusion is recognized. In (b), since pixel $P(\overline{s}+i)$ lies to the right of pixel $P(\overline{s}-i)$, a depression is recognized.

However, a diagonal step is $\sqrt{2}$ times longer than a step along the grid axes. This can be corrected by counting two units ($2\overline{s}$) for entries of $C_m(\overline{s})$ with directions along grid axes and three units ($3\overline{s}$) for entries along diagonal directions; that is, we approximate a step-size of $\sqrt{2}$ by 1.5 while keeping integer values. We denote this normalization of the distance along the trace of the curve by $\overline{u}$.

In addition to these quantization errors, there is a third source of error which is related to the definition of the chain code. Very sharp protrusions ending in a line of pixels of width one (Figure 4.3) could be erroneously defined as a depression, depending on the preceding values of C_m, as one gets closer to the tip of the protrusion. Therefore, *protrusion-depression definition problems* occur at angles of $\pm 180^{o}$ or equivalently, where $|C_m(\overline{s}-1) - C(\overline{s})| = 4$. The same problem can also occur for depressions, depending on how the trace of the contour is obtained.[8] In the general case, a protrusion or depression with an angle of $\pm 180^{o}$ that occurs at a contour pixel $P(\overline{s})$ can be checked by examining

[8] A simple object contour following algorithm used to extract the chain code from a segmented edge map would not usually tolerate, as part of the contour, depressions made of a string of pixels of width one. A more "intelligent" contour following algorithm would extract contour pixels in two passes. It would turn around the object (interior on the left) as well as "around" the background (exterior on the left) thereby generating sharp protrusions as well as sharp depressions.

a neighborhood of $P(\overline{s})$ to ensure that the interior of the object is still defined to be on the left of the direction of contour traversal. A simple procedure consists of looking at contour pixels preceding and following $P(\overline{s})$. The procedure is as follows:

If $|C_m(\overline{s}-1) - C(\overline{s})| = 4$, then:

- Compare the coordinates of the neighboring contour pixels $P(\overline{s}-i)$ and $P(\overline{s}+i)$ adjacent to $P(\overline{s})$ ($i = 1, 2, 3, ...$) until two are found which are located at different spatial positions.
- Check whether or not $P(\overline{s}+i)$ is to the *left* of $P(\overline{s}-i)$. If it is, then there is a *protrusion*. That is, $C_m(\overline{s}) = C_m(\overline{s}-1) + 4$ (*i.e.*, $+180^o$). Otherwise, $C_m(\overline{s}) = C_m(\overline{s}-1) - 4$ (*i.e.*, -180^o), indicating a *depression*.

In summary, the original chain code $C(\overline{s})$ must be corrected for three types of errors: discontinuities of more than $|180^o|$, arc length normalization, and protrusion-depression definition problems. These three types of errors can be detected in parallel by applying the procedures described above, thereby generating a modified chain code $C_m(\overline{u})$. The following section describes how to obtain smoother orientation data and then derive an estimate of the curvature.

4.2.3 From Discrete Orientation to a Smoothed Curvature Representation

Let us use the symbol $\overline{\theta}(\overline{u})$ to refer to the modified chain code after both quantization and protrusion-depression errors have been corrected, that is, $\overline{\theta}(\overline{u}) = C_m(\overline{u})$, to emphasize its equivalence with a discrete orientation representation of the trace of a boundary. We now must process $\overline{\theta}(\overline{u})$ to obtain the curvature measurements. By definition, for a parameterization of the curve by arc length s, the curvature $k(s)$ is given as the first derivative of the orientation function, $\theta(s)$, that is, $k(s) = \theta'(s)$. In the discrete case, we wish to retrieve $\overline{k}(\overline{u})$, which can be approximated by $\overline{k}(\overline{u}) \approx \overline{\theta}'(u)$.

Noise amplification problems occur when differentiating a discrete signal such as $\overline{\theta}(\overline{u})$ because it is coarsely quantized in 45^o steps. For example, a straight line at an angle between 0^o and 45^o is represented by a succession of 0^o and 45^o orientations, a very noisy signal indeed. This aliasing phenomenon cannot be alleviated unless the original sampling grid is refined or the orientation sampling is done over a larger neighborhood. Still, the effect of aliasing can be reduced by the application of lowpass or bandpass filtering to the signal. It is common to filter $\overline{\theta}(\overline{u})$ with a Gaussian template G_σ [36, 52], with standard deviation σ, defined as:

$$G_{\sigma(s)_{s=\overline{u}}} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-\overline{u}^2}{2\sigma^2}\right). \tag{4.1}$$

Furthermore, the convolution theorem [126] can be used to differentiate the filtered signal:

$$\left(\overline{\theta}(\overline{u}) \star G_\sigma\right)' = \overline{\theta}(\overline{u}) \star G'_\sigma \,. \tag{4.2}$$

This approach is often used to obtain $\overline{k}(\overline{u})$. The smoothing and differentiation of $\overline{\theta}(\overline{u})$ with a bandpass filter G'_σ is performed concurrently [36, 14]. However, in terms of computational complexity, there is an advantage to using Gaussian templates in an initial smoothing step instead of computing the derivatives of the Gaussian directly, as is done in other FD methods. Gaussian filtering can be efficiently implemented by combining *Hierarchical Discrete Correlation* (HDC) techniques with *cascaded correlation* (Chapter 3, § 3.2). Reusing the convolution theorem and the cascaded correlation property of Gaussian templates, we can obtain a smoothed curvature signal $\overline{k}_\sigma(\overline{u})$ as follows:

$$\begin{aligned} \overline{k}_\sigma(\overline{u}) &= \left(\overline{\theta}(\overline{u}) \star G_\sigma\right)' = \left(\overline{\theta}(\overline{u}) \star G_{\sigma_1} \star G_{\sigma_2}\right)' \\ &= \overline{\theta}(\overline{u}) \star G_{\sigma_1} \star G'_{\sigma_2} \,, \end{aligned} \tag{4.3}$$

where σ_1 and σ_2 are the standard deviations for the two Gaussian functions G_{σ_1} and G_{σ_2}, respectively, and $\sigma^2 = {\sigma_1}^2 + {\sigma_2}^2$. Therefore, in the first step we filter $\overline{\theta}(\overline{u})$ with a lowpass filter using an HDC-like algorithm and denote this operation by:

$$\overline{\theta}_{\sigma_1}(\overline{u}) = \overline{\theta}(\overline{u}) \star G_{\sigma_1} \,. \tag{4.4}$$

Then in a second step, we filter $\overline{\theta}_{\sigma_1}(\overline{u})$ with a bandpass filter to obtain its smoothed derivative:

$$\overline{k}_\sigma(\overline{u}) = \overline{\theta}_{\sigma_1}(\overline{u}) \star G'_{\sigma_2} \,. \tag{4.5}$$

The first step involving G_{σ_1} is efficiently computed using the HDC procedure. The second step is performed with a derivative of a Gaussian template, G'_{σ_2}, over a smaller neighborhood than for the original G'_σ in equation (4.2). Our complete procedure for extracting curvature from discrete orientation data is illustrated in Figure 4.4.

Although Gaussian filtering is desirable to reduce noise effects and to evaluate the derivative of $\overline{\theta}(\overline{u})$ with a numerically stable procedure, this smoothing process must be performed in a conservative way. By "conservative smoothing" we imply that we wish to retain as far as possible the relevant details of $\overline{\theta}_{\sigma_1}(\overline{u})$ and $\overline{k}_\sigma(\overline{u})$. However, Gaussian filtering considers noise and peaks of $\overline{k}_\sigma(\overline{u})$ without distinguishing between them on the basis of size [116]. We would like to describe the features of $\overline{k}_\sigma(\overline{u})$ by detecting them without having to significantly modify $\overline{k}_\sigma(\overline{u})$. Gaussian filtering cannot be used for this later purpose since peaks corresponding to features would be attenuated and blurred. In summary, linear filtering using Gaussian templates and their derivatives is ineffective because the scope of

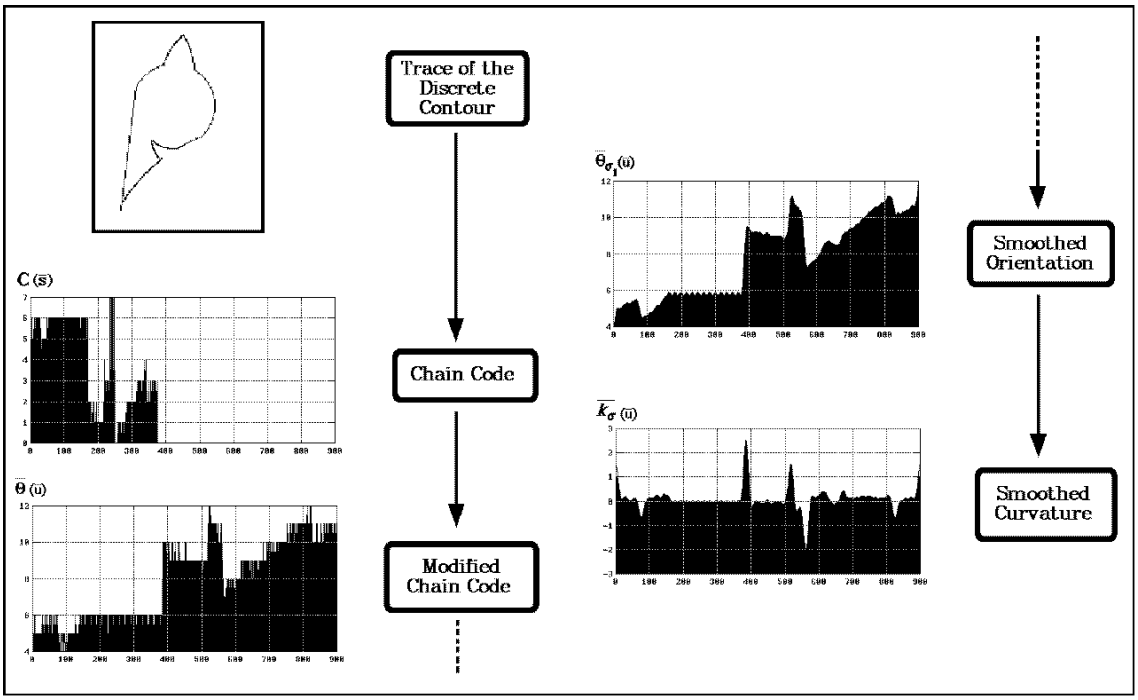


Figure 4.4: From the trace of the discrete contour to smoothed orientation and curvature representations. The object shown in here and its discrete smoothed curvature $\overline{k}_{\sigma}(\overline{u})$ will be used throughout this chapter to illustrate the complete curvature morphology approach.

the filtering is too global, that is, it is indiscriminate with respect to significant curvature features [20, 125, 116, 91].

We seek a description of curvature features in terms of their annihilation, as proposed in the so-called *scale-space* approaches [155, 36, 81]. However, instead of using Gaussian smoothing to generate a scale-space, which has an undesirable global effect, we propose the use of *morphological operators* [135]. These permit the removal of details from a signal such as $\overline{k}_\sigma(\overline{u})$ without modifying its global morphology. Since they exert only local influence, morphological operators have the desirable property of generating uniform scale-spaces that are unambiguous and therefore easily interpreted. Morphology for curvature is the subject of the next section.

4.3 Curvature Analysis

Although the curvature at each point along a curve uniquely defines its behavior, it is the morphology of curvature that permits the retrieval of useful visual information. Thus, we would like a description of $\overline{k}_\sigma(\overline{u})$ in terms of any well identified features.

As mentioned earlier, peaks in curvature correspond to the critical points of a contour and are useful for visual perception, as well as mathematical and computational representation. Constant regions are defined by straight lines ($\overline{k}_\sigma(\overline{u}) = 0$) and arcs of constant curvature ($\overline{k}_\sigma(\overline{u}) = constant$). They can be used to compress information and reduce the complexity of the interpretation process. Such constant regions, along with their associated peaks of curvature, can also be used to compute region-based representations of an object. Examples are the so-called skeletons or symmetric axes (Chapter 5) and local rotational symmetries [61]. Zero crossings of $\overline{k}_\sigma(\overline{u})$ map to inflection points of a contour. Since they are a natural way of segmenting the contour into convex and concave regions, we will separate $\overline{k}_\sigma(\overline{u})$ into two functions, $\overline{k}_{\sigma+}(\overline{u})$ for convex regions and $\overline{k}_{\sigma-}(\overline{u})$ for concave regions (Figure 4.5).

In addition to localizing the curvature features, we wish to be able to quantify their significance. This includes their relative dominance or amplitude as well as their degree of isolation along the contour. This concept of feature significance leads naturally to a multiscale representation in which irrelevant details of $\overline{k}_\sigma(\overline{u})$ can be removed before its interpretation.

All of these notions relating to feature shape, localization, size, isolation and scale are associated and can be treated using the mathematical technique known as *mathematical morphology* [135].

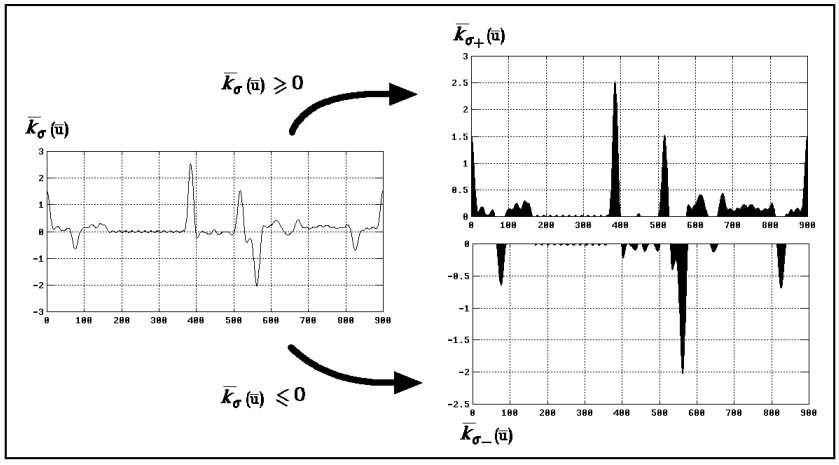


Figure 4.5: Curvature function $\overline{k}_{\sigma}(\overline{u})$ as the sum of two functions, $\overline{k}_{\sigma+}(\overline{u})$ (positive curvature values) and $\overline{k}_{\sigma-}(\overline{u})$ (negative curvature values).

4.3.1 Curvature Morphology

Mathematical morphology applied to discretized functions such as $\overline{k}_{\sigma}(\overline{u})$ provides us with useful tools for the extraction of primitives or dominant shapes found in such functions.[9] We have coined the name *curvature morphology* not only to emphasize the fact that we apply mathematical morphology operators to the curvature function, but also because this method permits us to study or analyze the "form" or morphology of the curvature function by segmenting it into its essential primitives or features. It is also interesting to note that with curvature morphology we consider the curvature function as an image to which we apply image processing techniques. This represents a rather different approach with regard to other more traditional techniques of contour analysis.

Two dual operations, *erosion* and *dilation*, are the keystones of mathematical morphology. As their names indicate, erosion is a shrinking operation while dilation is an expanding one. These operations are performed locally by observing the structure of the neighborhood at each point of the function. The neighborhood over which local computations are performed is delimited by a *structural element* defined by a set of resolution cells constituting a specific shape such as a line or square. This is analogous to the concept of a 2-D window that we can slide over an image to perform convolution-like computations. In our application to curvature, only flat structural elements, that is, lines of increasing width, are considered. Furthermore, we employ *symmetric* flat structural elements centered on curvature function points. Therefore, we use only widths of odd values and the minimal width is

[9] See Appendix A for mathematical definitions and properties of morphological operations on functions. For further mathematical details about this subject, the reader is referred to chapter twelve of Serra's book [135]. For applications to grey level images, see the paper by Sternberg on grayscale morphology [138].

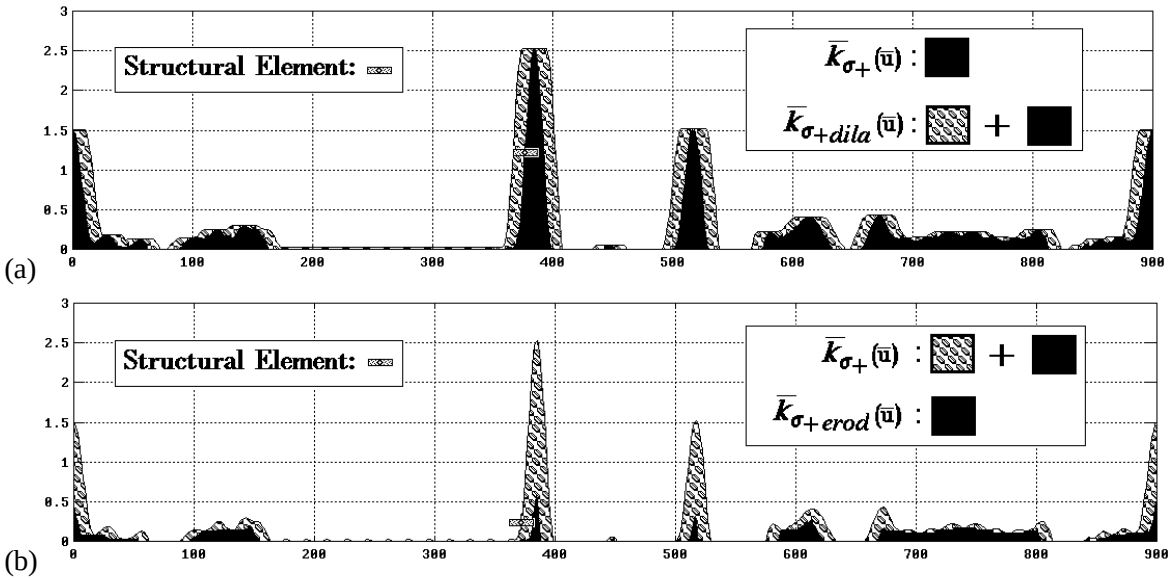


Figure 4.6: Examples of dilation and erosion operations applied to $\overline{k}_{\sigma+}(\overline{u})$ (positive curvature function of the object shown in Figure 4.4). The structural element shown in (a) and (b) defines the neighborhood over which local "min-max" computations are performed (here 21 arc length units wide). In (a), a dilation is performed by using a maximum operation. The new dilated curvature function is represented with the symbol $\overline{k}_{\sigma+dila}(\overline{u})$. In (b), an erosion is performed by using a minimum operation. The new eroded curvature function is represented with the symbol $\overline{k}_{\sigma+erod}(\overline{u})$.

fixed to three arc length units $\overline{u}$ (see Appendix A, \S A.1.7 for more details). Symmetric flat structural elements are sufficient for the extraction of peaks and flat regions of $\overline{k}_{\sigma}(\overline{u})$.

Eroding a function by a segment of length R is equivalent to replacing the function values at every point by the minimum of all the points in a neighborhood of size R. Likewise, dilating a function by a segment of length R is equivalent to a maximum transformation over a neighborhood of size R. Examples of dilation and erosion applied to $\overline{k}_{\sigma+}(\overline{u})$ are given in Figure 4.6 (similar results are obtained for $\overline{k}_{\sigma-}(\overline{u})$).

By combining dilation and erosion, two new operations can be defined: *opening* and *closing*. Opening is the dilation of an eroded function, while closing is the erosion of a dilated function. In both cases, by combining the two dual operations of dilation and erosion, the original function is only partially recovered since some details are eliminated from it. In the case of opening, convexities or "bumps" of increasing size are removed with the use of different sized structural elements. Closing, on the other hand, is used to fill-in concavities or holes. These concepts are illustrated in Figure 4.7. An opening (closing) will remove from the function all details, such as peaks, that are smaller than the structural element size and which are oriented toward the top (bottom) of this function. The result is

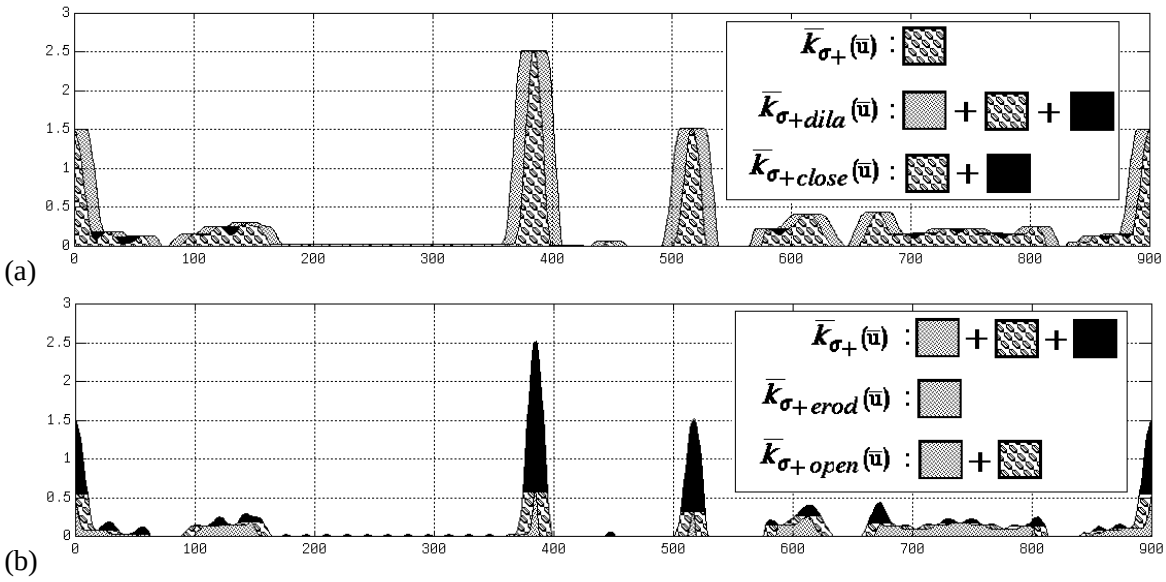


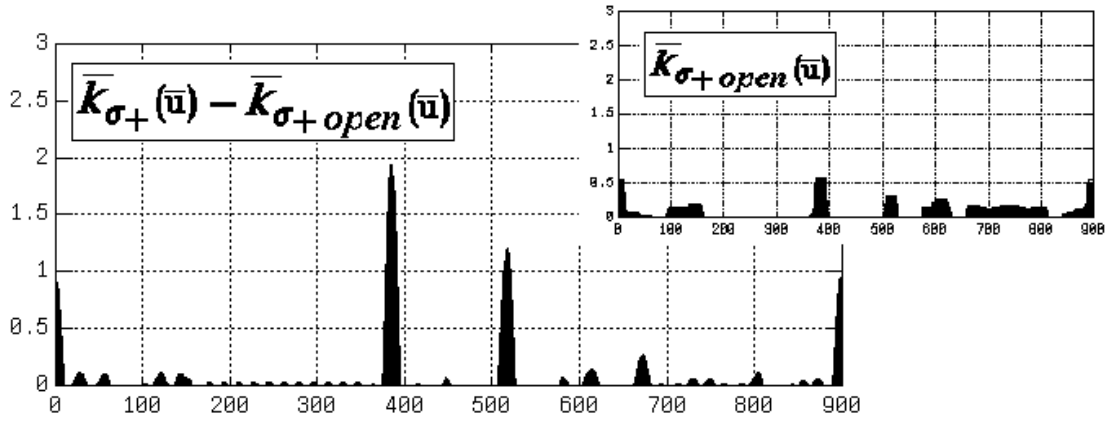
Figure 4.7: Examples of closing and opening operations applied on a positive curvature function for a structural element of fixed width (here 21 arc length units wide).

a new function which is smoother than the original one.

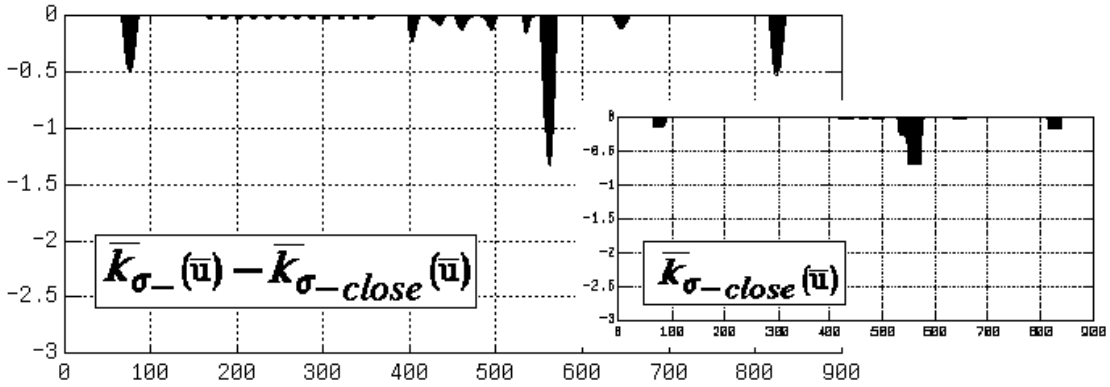
We observe that mathematical morphology provides us with a method for removing from the signal $\overline{k}_{\sigma}(\overline{u})$ details of increasing size. This is indicated in Figure 4.8.(a) where we compare the function $\overline{k}_{\sigma+}(\overline{u})$ to its "opened" version $\overline{k}_{\sigma+open}(\overline{u})$ and thereby isolate the peaks and bumps. Thus the residual $\overline{k}_{\sigma+}(\overline{u}) - \overline{k}_{\sigma+open}(\overline{u})$ is defined as the *top-hat transform* of $\overline{k}_{\sigma+}(\overline{u})$. We can also define its dual, the *bottom-hat transform* in which $\overline{k}_{\sigma+}(\overline{u})$ is compared to its "closed" version $\overline{k}_{\sigma+close}(\overline{u})$ according to the residual $\overline{k}_{\sigma+}(\overline{u}) - \overline{k}_{\sigma+close}(\overline{u})$. These two residuals are similarly defined for the negative curvature function as $\overline{k}_{\sigma-}(\overline{u}) - \overline{k}_{\sigma-open}(\overline{u})$ and $\overline{k}_{\sigma-}(\overline{u}) - \overline{k}_{\sigma-close}(\overline{u})$ respectively. The bottom-hat transform applied to $\overline{k}_{\sigma-}(\overline{u})$ is illustrated in Figure 4.8.(b). Both these operators will be used as the basic morphological operations for curvature. Hat-transforms are so called because they can be visualized as a covering of peaks with a hat of fixed size.

4.3.2 Peak Description

Peaks with different sized bases can be extracted by varying the size of the flat structural elements. By increasing the width of a structural element, more detail can be extracted from the curvature signal, $\overline{k}_{\sigma+}(\overline{u})$ or $\overline{k}_{\sigma-}(\overline{u})$. Once a peak is isolated from the curvature signal, its morphology or shape can be analyzed. We define four morphological measures to describe an isolated peak (Figure 4.9):



(a) Top-hat transform on $\overline{k}_{\sigma+}(\overline{u})$.



(b) Bottom-hat transform on $\overline{k}_{\sigma-}(\overline{u})$.

Figure 4.8: Examples of hat-transforms for a structural element of fixed width (21 arc length units wide). In (a) the top-hat transform is applied to $\overline{k}_{\sigma+}(\overline{u})$ to extract its peaks. In the top-right hand corner is shown the effect of removing the extracted peaks from $\overline{k}_{\sigma+}(\overline{u})$. This is equivalent to $\overline{k}_{\sigma+open}(\overline{u})$. In (b) the bottom-hat transform is applied to $\overline{k}_{\sigma-}(\overline{u})$ to extract its valleys. In the bottom-right hand corner is shown the effect of removing the extracted valleys from $\overline{k}_{\sigma-}(\overline{u})$. This is equivalent to $\overline{k}_{\sigma-close}(\overline{u})$.

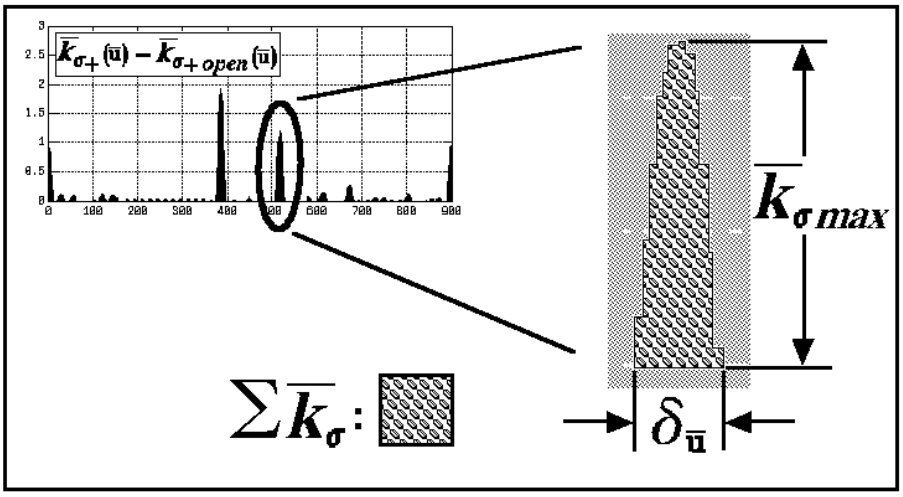
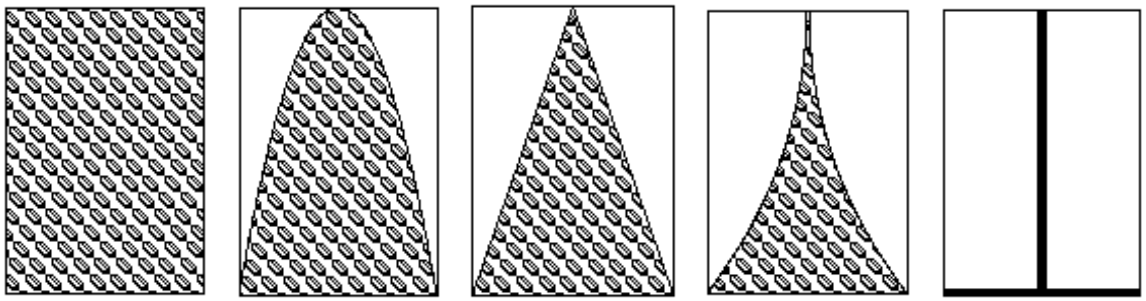


Figure 4.9: The features used to describe an isolated peak of curvature.

- The *extent of the peak*, $\delta_{\overline{u}}$, which is equal to the width of the flat structural element.
- The *maximal peak amplitude* or maximal relative curvature, $\overline{k}_{\sigma max}$.
- The *average relative curvature* of the peak, $\overline{k}_{\sigma avg}$, given by the area under the curve defined by $\Sigma\overline{k}_{\sigma}/\delta_{\overline{u}}$.
- The *shape factor*, r, given by $\overline{k}_{\sigma max}/\overline{k}_{\sigma avg}$ ($r \geq 1$).

Each of the four morphological measures derived from the peak features $\overline{k}_{\sigma max}$, $\delta_{\overline{u}}$ and $\Sigma\overline{k}_{\sigma}$ can be used to describe the nature of the peak. For example, peak significance can be determined not only by large values of $\overline{k}_{\sigma max}$ with respect to $\delta_{\overline{u}}$ but also by the shape factor r. Values of r greater than one may indicate a very narrow and well-defined peak. On the other hand, a value close to one indicates a region of constant curvature (Figure 4.10.(a)). By examining its extent $\delta_{\overline{u}}$, a constant curvature region could be retained as being significant or be disregarded. The different possible kinds of shapes of peaks described by the shape factor r are illustrated in Figure 4.10. Stable curvature peaks are those whose shape factor is within the range $1 < r \leq 2$ (Figure 4.10, (b) and (c)). By "stable" we mean that if $\delta_{\overline{u}}$ is increased, $\overline{k}_{\sigma max}$ of the given peak will significantly increase. On the other hand, when $2 < r < \delta_{\overline{u}}$, the peak tends to flatten for increasing $\delta_{\overline{u}}$ (Figure 4.10.(d)). The case of $r \approx \delta_{\overline{u}}$ corresponds to noise in the curvature data (Figure 4.10.(e)).

Although these shape measures seem to be useful, the scale or size at which they should be sought is variable and unknown *a priori*. Therefore, curvature morphology analysis is best performed at different scales, where scale is defined as the variable size of the structural element. This leads to a multiscale representation of curvature which is developed in the next section.



(a) $r = 1$ (b) $1 < r < 2$ (c) $r = 2$ (d) $2 < r < \delta_{\overline{u}}$ (e) $r = \delta_{\overline{u}}$

Figure 4.10: The five possible types of peak shapes described by the shape factor r for given $\overline{k}_{\sigma max}$ and $\delta_{\overline{u}}$. In (a) $\overline{k}_{\sigma avg} = \overline{k}_{\sigma max}$. In (b), (c) and (d) are shown the three most common types of curvature peaks. In (e) $\Sigma\overline{k}_{\sigma} \approx \overline{k}_{\sigma max}$.

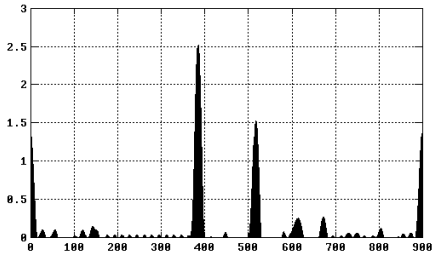
4.3.3 Morphological Curvature Scale-Space

We generate the multiscale representation of the curvature signal, $\overline{k}_{\sigma+}(\overline{u})$ or $\overline{k}_{\sigma-}(\overline{u})$, by uniformly increasing the size of the structural element. A sequence of peaks is generated using the hat-transforms. A scale history can then be associated with each peak, starting with the scale at which it appears and terminating with the scale at which it ceases to increase in height. As soon as a peak stops increasing, it no longer needs to be considered as part of $\overline{k}_{\sigma+}(\overline{u})$. We define the *region of support* of a peak to be the scale $\delta_{\overline{u}max}$ at which it stops increasing. $\delta_{\overline{u}max}$ can be interpreted as the region of influence a peak has along the contour.[10]

Once a peak has been removed from $\overline{k}_{\sigma+}(\overline{u})$, no other peak is permitted to grow over the extent of the former. This is simply because only one curvature feature should be associated with a given contour segment. An example of the scale-space generated in this way is given in Figure 4.11. Here, the position of an event, that is of a peak of $\overline{k}_{\sigma+}(\overline{u})$, is given by the position $\overline{u} = \overline{u}(\delta_{\overline{u}max})$ of the maximal relative curvature value $\overline{k}_{\sigma} = \overline{k}_{\sigma max}(\delta_{\overline{u}max})$ associated with the peak.

Our scale-space representation for curvature, referred to as the *Morphological Curvature Scale-Space* or MCS, possesses certain desirable advantages over previous approaches such as the "Curvature Primal Sketch" [14]. For example, each branch in the scale-space diagram remains isolated, whereas in the usual scale-space approaches reported in the literature instabilities occur because of branch merges [155, 14, 108]. The MCS bears some similarity to the kind of scale-space generated under the *weak-continuity* constraints scheme of Blake and Zisserman [20, 19], although in their representation, only discontinuities in

[10] See [140] for a variation of the concept of "region of support" for a curvature extremum.

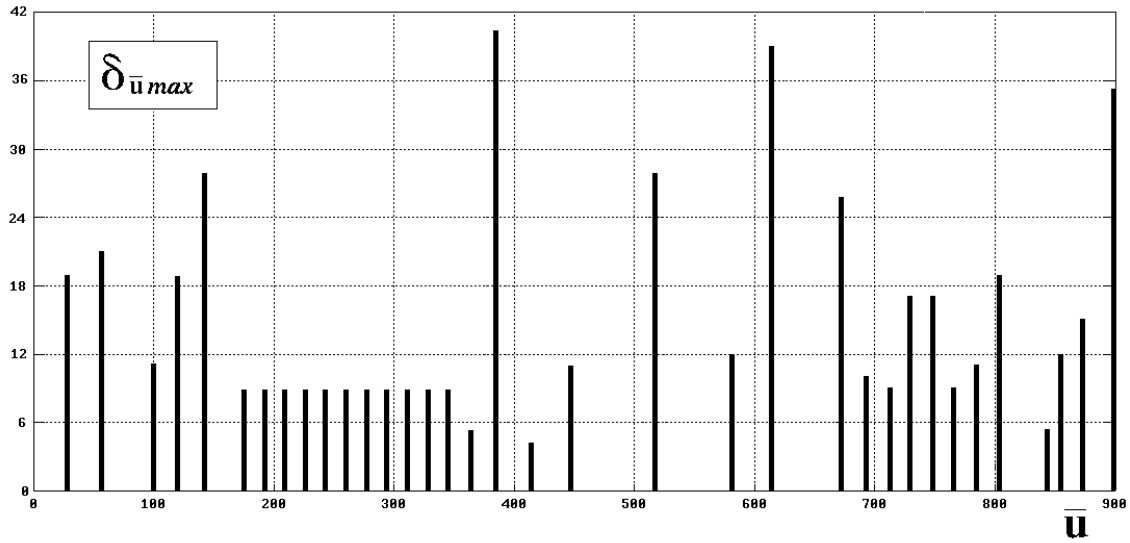


(a) Peaks of $\overline{k}_{\sigma+}(\overline{u})$

No. of peaks = 35

Peak#	$\delta_{\bar{u}\,max}$	$\bar{u}$	$\bar{k}_{\sigma\,max}$	Peak#	$\delta_{\bar{u}\,max}$	$\bar{u}$	$\bar{k}_{\sigma\,max}$
0	35	899	1.3958	18	40	384	2.5194
1	19	27	0.1081	19	4	414	0.0076
2	21	55	0.0997	20	11	446	0.0619
3	11	100	0.0164	21	28	516	1.5228
4	21	119	0.1072	22	12	580	0.0699
5	28	142	0.1516	23	39	613	0.2597
6	9	175	0.0291	24	26	672	0.2991
7	9	192	0.0291	25	10	693	0.0156
8	9	209	0.0291	26	9	712	0.0180
9	9	226	0.0291	27	17	729	0.0528
10	9	243	0.0291	28	17	748	0.0528
11	9	260	0.0291	29	9	765	0.0180
12	9	277	0.0291	30	11	784	0.0164
13	9	294	0.0291	31	19	803	0.1166
14	9	311	0.0291	32	5	843	0.0057
15	9	328	0.0291	33	12	855	0.0437
16	9	345	0.0291	34	15	872	0.0573
17	5	362	0.0047				

(b) Peak features



(c) Morphological Curvature Scale-space

Figure 4.11: Example of the morphological curvature scale-space.

curvature are tracked explicitly. Furthermore, the MCS satisfies the three criteria proposed by Perona and Malik [125], that any candidate paradigm for generating multiscale representations should satisfy.[11] These criteria are:

- *Causality*: No spurious details should be generated when the scale ($\delta_{\overline{u}}$) is increased.
- *Immediate Localization*: At each scale, feature boundaries should be sharp (unblurred) and correspond to meaningful boundaries at that scale.
- *Piecewise Smoothing*: At all scales it is preferable that intra-feature smoothing occur rather than inter-feature smoothing.

The MCS is an elegant way of removing noise from $\overline{k}_{\sigma+}(\overline{u})$ while *preserving* its significant features. Noisy elements of $\overline{k}_{\sigma+}(\overline{u})$ which exhibit a short history at small scales in the MCS can be eliminated. Note that both the top-hat and the down-hat transforms can be used here to remove small convex bumps and to fill-in small concave depressions of $\overline{k}_{\sigma+}(\overline{u})$, respectively. An example of filtering $\overline{k}_{\sigma+}(\overline{u})$ in this fashion is shown in Figure 4.12, where the threshold on $\delta_{\overline{u}}$ for removing bumps and valleys was set at 12 in Figure 4.11.

A second threshold, this time on the minimal acceptable $\overline{k}_{\sigma max}$ value, is also used for peak significance. This threshold was set at 0.20 in Figure 4.11. The same filtering process is also applied to $\overline{k}_{\sigma-}(\overline{u})$. The combined results are shown in Figure 4.13. This ability of the MCS to deal with noise justifies our previous conservative attitude toward Gaussian smoothing (section 4.2.3).

The MCS can also be used to build a hierarchy of significant peaks. Peaks with long histories starting at a small scale are classified as prominent. Peaks starting at higher scales, on the other hand, can be classified as corresponding to constant curvature regions because their initial larger extent $\delta_{\overline{u}}$ maps to a constant curvature arc. Once these peaks are extracted, they can be subtracted from the curvature signal. The resultant signal can then be further processed to isolate other constant curvature arcs. An example of this segmentation process applied to both positive and negative curvature signals is shown in Figure 4.14.

Once the main features of the curvature signal have been segmented, that is when both positive and negative peaks and constant curvature arcs have been extracted, the contour of the object itself can be segmented. Peaks of curvature map to knot points of the object contour, while constant curvature arcs map to their respective contour arcs. This is shown in Figure 4.15. All knot points are extracted and precisely localized. Note that no knot point is identified between arcs 2 and 3 because there is no significant discontinuity in orientation between these two arcs. Therefore, there is no peak in curvature where these two arcs join. In such a case a smooth join of the first type can be identified. Constant curvature arcs are also well identified (*e.g.*, arc 3: a straight line). Gaps between identified arcs and knot points

[11] Note that scale-space representations based on the usual Gaussian blurring approach only satisfy the causality criterion.

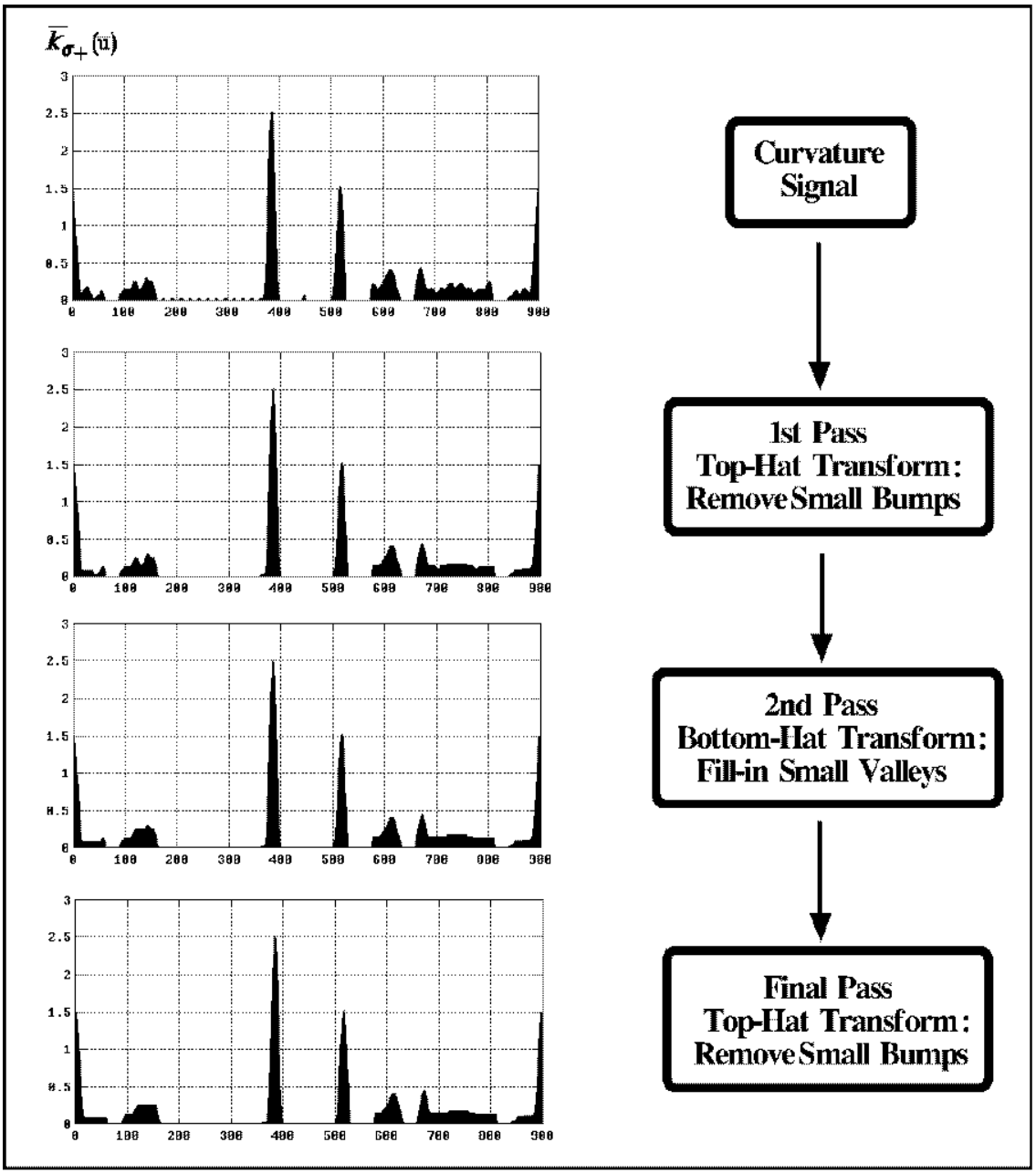
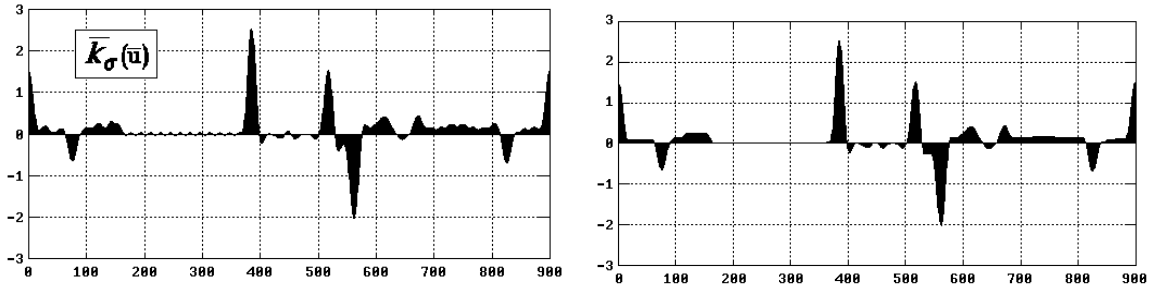
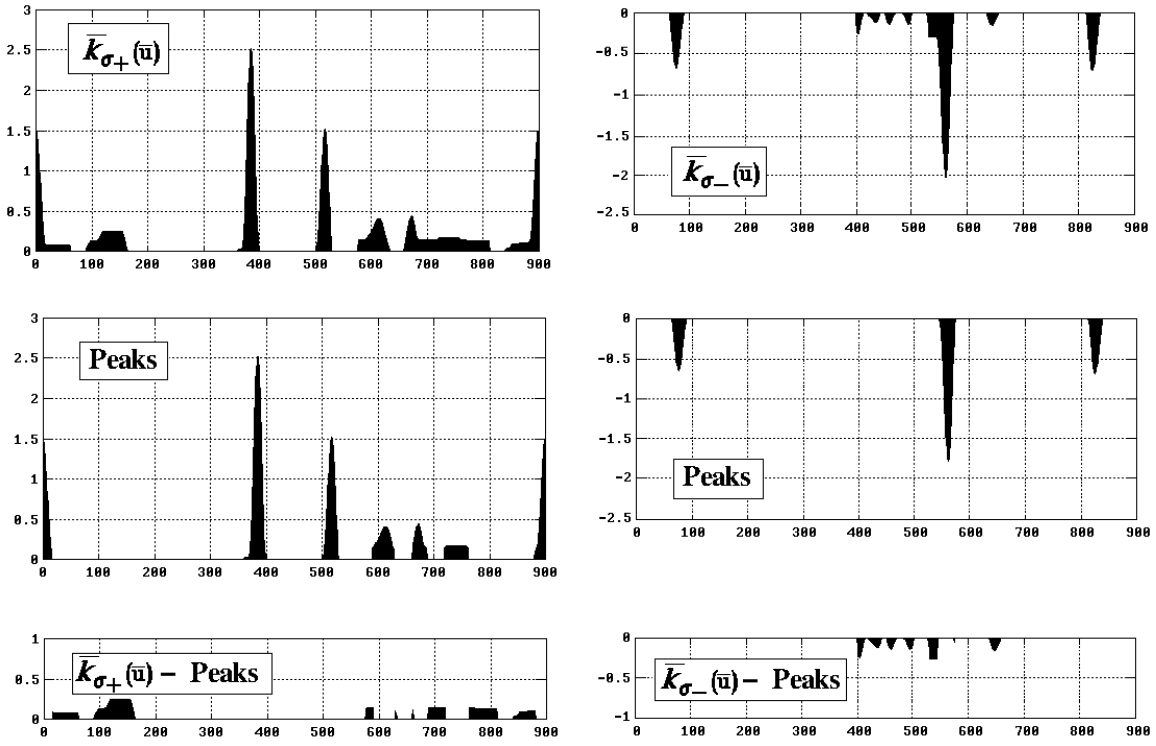


Figure 4.12: Removing noise from $\overline{k}_{\sigma+}(\overline{u})$. At each one of the three passes, the MCS is used to extract small bumps and depressions.



(a) Original signal (b) Filtered signal

Figure 4.13: Example of removing noise from $\overline{k}_{\sigma}$ using the hat transforms and the MCS.



(a) Positive curvature segmentation (b) Negative curvature segmentation

Figure 4.14: Segmentation of the curvature signal.

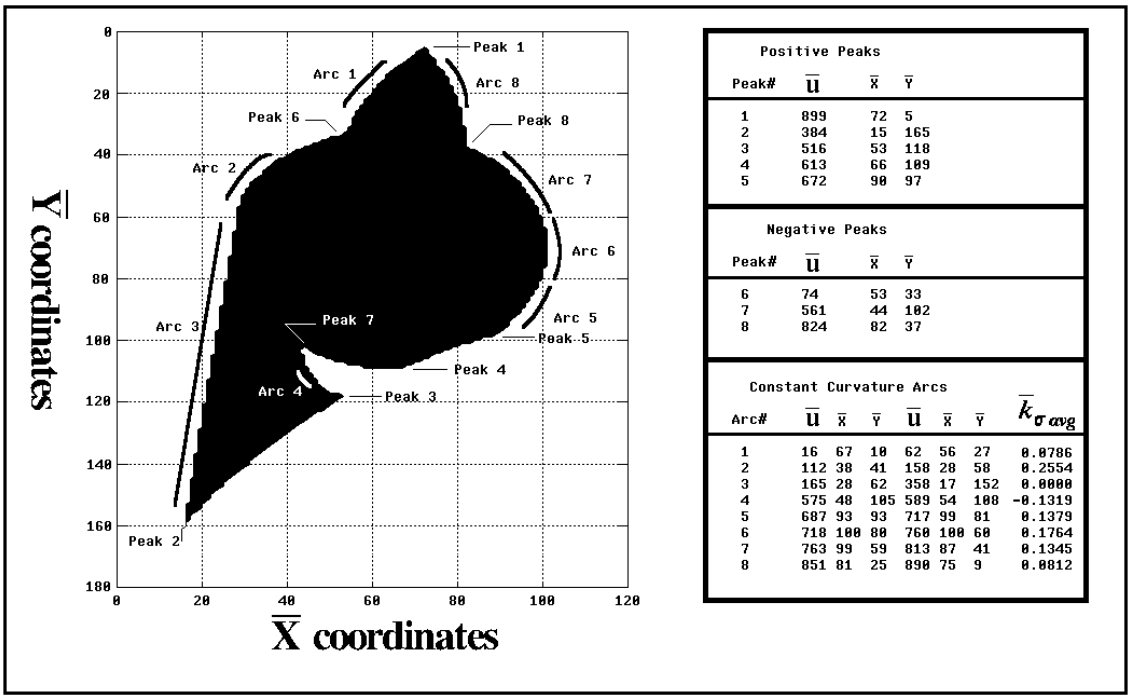


Figure 4.15: Contour segmentation based on the previously extracted curvature features.

which are caused by filtering can be filled-in by extrapolation, while unidentified arcs can be approximated by fitting splines or circular arcs between the bounding knots. The average curvature of these unidentified arcs, obtained by integrating the curvature signal between the knots, can be used to impose a constraint on the fitting curve segment. For example, between peaks 2 and 3 (Figure 4.15), if we integrate the curvature function we obtain a small negative value ($\overline{k}_{\sigma avg} \approx -0.0148$). Therefore, we can approximate this contour segment by a (slightly) concave arc.

4.4 Conclusions

In this chapter we have proposed efficient ways of representing curvature and its features for object contours found in noisy images. We have discussed how the curvature function should be retrieved from the discrete trace of a contour using a modified chain code representation. Special care has been taken with quantization errors, "protrusion-depression definition" problems, and implementation issues. We have also presented a new method for extracting features from the curvature function employing morphological operations.

Unlike linear transformations of functions such as Gaussian filtering, morphological operations are characterized by their local effect. They remove information of increasing

extent as the size of the structural element increases without blurring the remaining important features. Therefore, signal processing through iterative morphological transformations can be thought of as a process of selective information removal, where irrelevant details are subtracted from the signal, enhancing the contrast of essential features. We have proposed such an iterative scheme by defining a new scale-space representation referred to as the *Morphological Curvature Scale-Space* (MCS). We believe that the MCS is a more powerful tool for the interpretation of curvature than previous scale-space approaches. There are four reasons: first, the MCS is uniform and unambiguous; second, the MCS satisfies the three criteria of causality, immediate localization and piecewise smoothing; third, morphological measures can be combined in the MCS [90] to strengthen the analysis and interpretation of curvature primitives; and fourth, the MCS is a strictly local representation.

Another important aspect of curvature morphology that compares favorably with other analysis approaches is its low computational complexity. The computations are simple and hierarchical (Appendix A, § A.1.8), and lend themselves to applications where fast computations are imperative. This is the case when sequences of images need to be analyzed in real time [115].

However, a number of issues still need to be addressed so that the approach we have proposed for contour segmentation can be viewed as being complete. In particular, we need to study the problem of determining how much smoothing is necessary in the first step of recovering a useful curvature function (section 4.2). We have suggested that smoothing was necessary but it should be done in a conservative way. We have not quantified and mathematically defined these concepts. Also we have not given a detailed analysis on how to choose the two thresholds on curvature extremum significance when performing noise-filtering. These issues need to be addressed in detail. The purpose of our description and analysis of curvature morphology was to show its feasibility, its performance and some of its advantages over other contour segmentation methods.

Finally, it was mentioned in section 4.3 that curvature features of an object outline could be used to compute a region-based shape representation. We will consider this issue in the next chapter. We will show how the shape features obtained using curvature morphology can help us to apply the snake model to the shape description of deformable objects such as cells.

Chapter 5

Shape Description Using Snakes

5.1 Introduction

This chapter presents a new method for shape description of planar objects in which both region and boundary features are extracted. We propose a new algorithm for shape skeletonization to achieve this.[1] Skeletons are object representations which are particularly appropriate for the description of the amorphous or biological shapes commonly found in nature for which other representational schemes based on ordinary geometry are inappropriate [25]. They can be used to encode visual cues such as symmetry, shape primitives, width information and the process-history of deformable objects. We argue that boundary information can be combined with a region-based representation such as the skeleton to produce a richer, more powerful and efficient shape representation. For example, extrema of curvature along the contour can be used to generate significant skeleton branches [52, 95].

Our method for shape description is an implementation of a 2-D dynamic grassfire that relies on a distance surface on which elastic contours minimize an energy function. A Euclidean distance transform combined with an active contour model, such as the snake, is used for this minimization process. Boundary information is integrated into the model by the extraction of curvature extrema and arcs of constant curvature.

5.1.1 Biological Basis for Shape Analysis

Within the context of this thesis, we are principally concerned with the notion of *biological shape* or the shape of natural forms assumed by living organisms. How can we categorize this notion of shape? What do we mean by the shape of an object? Following the early ideas

[1] An early version of this chapter was first published as a technical report [94]. Two shorter versions were also published recently as conference papers [95, 92]. Addendum (February 2003): A journal article based on this chapter was published in IEEE-PAMI in 1992 [97].

of Blum [25] we observe that biological shape is concerned with three kinds of problems. First, is the "taxonomic" or descriptive problem concerned with a hierarchically structured representation of a form in terms of its primitives. For example, a cell can be hierarchically represented by a body and its subparts such as pseudopods. Second is the "psychological" or sociological problem; how do organisms perceive and characterize the shape of other organisms? Third, is the "developmental" problem which is concerned with how shape is coded internally by organisms. The aim of biological shape description and analysis is to provide models to solve these problems. There are at least two different classes of descriptors that have been proposed in the literature, one based on the analysis of the external structure of forms, the boundary, and one based on their internal structure, that is, the region delimited by the boundary [121, 87].

One may consider 2-D shape as "the outward form of an object defined by [its] outline" [71]. In essence, shape is interpreted as the form or features of the contour or boundary of an object that makes it distinct from other objects. Still, there is an opposite or rather a complementary view when looking at shape description, in which the region or the interior of an object is considered in preference to the contour [87]. On the basis of a region description, visual cues such as symmetry, region primitives or subparts, and width or compactness are emphasized. An important advantage of region descriptors over contour descriptors is that they provide a correct topological representation of the object. Indeed, contour descriptors lack the ability to describe how close in the plane contour points are to each other. In other words, contour "points which are geometrically close may be quite far apart along the boundary" [121]. As to whether or not there are actually explicit region descriptors used by the brain, the question remains open. The best conjecture at present is that shape description at the physiological level is essentially feature-based [51].

5.1.2 Region or Boundary?

Are the two classes of descriptor, that is, boundary- and region-based, distinct or are they related? Can we express shape description by a continuum of information ranging from purely boundary descriptors to purely internal descriptors? Can we exploit both their descriptive powers and relate them in natural ways or is this dichotomy inevitable? Answers have been proposed in the past. For example Leyton [99, 100] has shown that in the case of regular curves, there is a direct relation between boundary and region features. More precisely, to each extremum of curvature, there is a corresponding *symmetric axis* which emerges or terminates at it. Furthermore, Leyton has proposed that "shape [may be] understood as the outcome of processes that formed it." The "process-history is recovered" by these features which are emphasized by both the internal and external descriptors, namely the curvature extrema and symmetric axes endings.

Axes of symmetry of an object may be defined in a number of ways in both the continuous and discrete domains. Essentially, a point in the interior of a shape is a *symmetry point*

(part of a symmetric axis) if it is at an equal "distance" from two or more boundary points, where the distance is any valid metric in the continuous or the discrete domain. An equivalent definition is based on the law of propagating wavefronts where the boundary of the shape is taken as the source of a *grassfire* [24, 25]. The *skeletal points* are then defined as the locus of meeting wavefronts, here fire fronts, where the fire is quenched. The complete set of skeletal points forms the *skeleton* of the shape. With such definitions, the skeleton and the complete set of symmetric axes are equivalent [25].

5.1.3 A model for shape description

We are seeking a model of shape description that should answer three types of problems: developmental, perceptive and descriptive. We know from psychophysics and neurophysiological experiments that shape in the visual cortex seems to be mainly feature-based and that the curvature along a curve or contour is a key descriptor. From topological considerations, we know that contour-based descriptions alone are not sufficient to resolve biological shape. Furthermore, from psychophysical experiments, we know that any valid model for shape description should be invariant under certain transformations, such as those arising from variations in illumination, color, rigid motion and scale [16].

In order to extract "pertinent" shape features we first need to *transform* the figure-ground image, that is, transform the binary image (object – non-object) obtained from a preceding segmentation step. Extracted shape features should emphasize both the form of the object boundary and the structure of its interior. These features should also be invariant under the above-mentioned image transformations.

In the subsequent sections of this chapter we will propose a method for biological shape and its implementation that should address these needs. But first we will briefly survey the different techniques used to obtain the skeleton representation. Then, we will propose a novel implementation of the grassfire transform. We have adopted this formulation for the extraction of shape symmetries because it explicitly emphasizes the relation between boundary features and region features. We will see how the formation of the skeleton is initiated at curvature extrema or centers of curvature where, in some sense, the "shape is concentrated." Here the fire fronts start merging, or in other words, the transforming shape, under the grassfire process, starts interacting with itself. The issue of integration of boundary information in our shape model will be addressed. Following this we will describe how a graph representation of the skeleton is easily obtained using the grassfire transform. Such a hierarchical representation permit us to naturally keep only these *branches* of the skeleton which are significant. We will summarize the advantages of this model for biological shape; variations in the implementation of the algorithm are also proposed. Finally, we will conclude this chapter by showing how the proposed model for biological shape can be used for the tracking and description of deformable objects such as cells.

5.1.4 Contributions

In this chapter, a number of contributions are made. They are summarized below.

- We present a new method for representing and describing shape on the basis of an active contour model. This permits us to efficiently simulate the grassfire transform. For the first time the fire fronts are considered as distinct entities with regard to the field of grass or the interior of a shape.
- We emphasize the relationship that exists between the grassfire transform and the notion of a "distance surface", a relation first mentioned by Blum [25] and for the most part ignored since. We take advantage of this relationship to efficiently simulate the grassfire propagation by an active contour.
- The use of an active contour on a field of grass, represented as a distance surface, permits us to employ an Euclidean metric while bypassing discretization problems found in other skeletonization algorithms also based on the Euclidean metric.
- Our method for shape explicitly relates boundary information to region information, a departure from most shape description techniques found in the literature.
- The use of both types of shape descriptors gives us a powerful tool for obtaining robust multiscale descriptions. In particular, this permits us to specify significance criteria and measures for each skeleton branch, thereby producing skeletons without spurious branches, a major advantage over most other skeletonization techniques.
- We propose a new concept for branch significance based on the notion of local deformation introduced by symmetry points on the distance surface. We call this the "ridge support". Furthermore, we show how the ridge support can be evaluated in terms of the velocity of formation of a symmetric axis or in terms of the slope amplitude of the tangent to the symmetric axis.
- We propose the new concept of a deformable skeleton useful for tracking deformable shapes. We refer to this as the "dynamic skeleton".
- Finally, we consider a number of implementation details to optimize the numerical performance of our skeletonization algorithm. In particular, we give the details of an optimized implementation of Danielsson's algorithm [46] for computing discrete Euclidean distance maps.

5.2 Shape Description by Skeletonization

A skeleton is a representation of an object by idealized thin lines that retain the connectivity of the original shape [72, 52]. Skeletal points, when connected, form a skeleton. They can be defined in a number of ways in both the continuous and discrete domains. The first definition had its roots in the concept of *nearest-point mapping* (or distance mapping; see section 5.3.1) for a closed region or set, O. In this case, the set of points in the background with more than one nearest point in O, that is, the set of skeletal points of the exterior of a shape, is considered important for characterizing the geometry of O [32]. This concept was further explored by considering the locus of *centers of osculating circles* to O (from the exterior) at more than one point to characterize the convexity or non-convexity of the set O [114, 113]. Much later came Blum who turned his attention to the interior of a region O [23, 25]. Inside O, skeletal points can be defined as the locus of *centers of maximal inscribed circles*, that is, circles in O not included in any other circle. Such circles are touching the boundary of O at more than one point. An equivalent definition is based on the law of propagating wavefronts where the boundary of the shape is taken as the source of a *grassfire* [23, 22, 24]. The skeleton is then defined as the locus of meeting wavefronts. Brady and Asada [36] define their skeleton as the locus of the *chord midpoints* of maximal inscribed circles. They call this shape representation *smoothed local symmetries* (SLS).[2] Leyton [100] defines skeletal points as the locus of *arc midpoints* of maximal inscribed circles or minimal circumscribed circles. He calls this representation *process inferring symmetric axis* (PISA). The process or transform by which the complete set of skeletal points is recovered is referred to by different names such as: SAT (Symmetric Axis Transform), MAT (Medial Axis Transform), *grassfire* transform, shape *skeletonization* and shape *thinning*.

5.2.1 Algorithms for Skeleton Computation

Over the years, since the first ideas of Blum emerged, various algorithms and implementations have been published. These can be classified into three different groups [94]: 1) direct thinning of an object, 2) analytic computation of a skeleton based on an approximation of the object contour, and 3) ridge following based on a distance mapping of the object. We briefly mention the characteristics of each class of algorithm and its main drawbacks.

Most algorithms in the literature are based on the thinning concept [135] in which one iteratively peels off the contour of an object; this is an approximation to the grassfire process [72]. Sequential as well as parallel algorithms exist.[3] These are iterative procedures whose computational time depends on the maximum width of the object. Their accuracy is limited

[2]Note that Blum also studied such a definition of skeletal points based on the locus of chord midpoints and called them *symmetric chord coordinates* [25].

[3]For example, for sequential algorithms see [72, 6, 5, 52, 158, 157]; for parallel ones see [122, 49, 73, 57, 42].

since the flow of information when performing a peeling-off step is biased by the intrinsic connectivity of the digitized grid (4- or 8-connected grid in general).

Another class of algorithm uses a direct (analytic) computation of the symmetric axes based on the polygonal approximation of a shape.[4] Here the problem of accuracy is more critical for biological shape description since a polygonal approximation is often not sufficient. Objects with noisy boundaries would require very fragmented polygons with short sides which could have the effect of making the results less accurate. The computation of symmetry points is generally more complex than for the other two classes of algorithms. In addition, objects with holes are generally difficult to process by such methods.

A third class of algorithm uses ridge following techniques based on *Distance Transforms* (DT's) applied to the object shape. Different DT's can be used: the *city block* DT [131, 13], the *chessboard* DT [7], the *hexagonal* DT (on an hexagonal grid) [107, 31], *chamfering* and other *quasi-Euclidean* DT's [110, 55] and the *Euclidean* DT [46, 74, 79, 94]. Skeletons based on the city block or chessboard DT can be computed very fast and are assured to be connected within a fixed number of passes. This is due to the simplicity of the connectivity of skeletal pixels [9, 10, 11]. The drawback is that these two DT's are not accurate, yielding a 40% to 50% error in distance values. Furthermore they are not as consistent as their Euclidean or quasi-Euclidean counterparts, in that they generate artificial spurious skeleton branches.[5] Methods based on hexagonal and quasi-Euclidean or Euclidean Distance Transforms rely on ridge following on the surface of a distance map to obtain the skeleton. They produce accurate[6] and smoother results, but thickness and connectivity of the skeleton branches must be carefully checked. For example, gaps often occur due to the spatial quantization of the digital grid and must be filled in, perhaps by using gradient ascent methods [55, 74, 79].

5.2.2 Advantages of Algorithms based on Distance Mapping

The most attractive type of algorithm to date seems to be those based on distance transforms. The main advantages are:

- Only simple computations such as mask convolutions are required, this for a fixed number of passes which are independent of the complexity of the object boundary.
- The skeletal pixels are labeled directly by the DT. The distance value from the background corresponds to the radius of a maximal osculating circle to the boundary.

[4] For example, see [111, 27, 136, 85, 36, 103]. Other, more complex, approximations such as arcs of circles and splines may also be used.

[5] These problems are also found in the first-mentioned class of "peeling-off" methods due to the fact that most of them rely on 4- or 8-connectivity as do the city block and chessboard DT's.

[6] Worst-case error of less than 0.3 pixel unit in the case of Danielsson's EDT algorithm [46].

- On a computational cost performance basis, these methods yield more accurate and smoother results than those of any other type of algorithm.

Consequently, we have decided to study this class of algorithm for shape skeletonization. We have attempted to improve its performance in order to use it to track the shape of large numbers of cells in real time [115]. This has lead to a new algorithm which is presented in the following sections.

5.3 Shape Skeletonization by Wavefront Propagation

We now introduce a new algorithm for computing skeletons which combines the advantages of algorithms based on the Euclidean distance metric with certain important additional features: connectivity which is implicitly ensured, integration of both contour and region information, and a multiscale description which is immediately available.

5.3.1 The Grassfire Transform as a Potential Surface

Our method for computing a skeleton of a binary image is based on an efficient implementation in the discrete domain of the original grassfire algorithm. It incorporates a physical analogy of grassfire propagation. This is achieved by igniting a "fire" on the object's boundary points which are the limits of the field of grass. The fire then propagates at constant speed within the object. Points at which the fronts of the fire merge are retained as being on a symmetric axis. These correspond to points that are at a maximal Euclidean distance from two or more boundary points, or equivalently, at the center of the maximal inscribed circle.

If we consider the grassfire as a function of time, a space-time graph known as a *two-dimensional dynamic grassfire* [25] is obtained. This generates a 3-D surface, as shown in Figure 5.1, where the time variable is along the vertical axis, $\overline{T}$, and where the $\overline{X}$ and $\overline{Y}$ axes correspond to the fire front coordinates. Each $(\overline{X} - \overline{Y})$ plane at a particular time $\overline{t}$ contains the fire front at that time. We obtain a *distance surface* by replacing time by a distance coordinate.[7] Such a surface has a maximal directional slope of one everywhere except along ridges where it is undefined [69]. These ridges correspond to the locus of the skeleton (Figure 5.1).

The resulting distance surface is equivalent to the distance map obtained by computing a Euclidean Distance Transform (EDT) on the initial figure-ground image [94]. A distance transformation on a binary image (object O, non-object O') produces a mapping from a double-valued function f_1 in a space S_1 ($f_1 : f_1(p) = 1$ iff $p \in O$ and $f_1(p) = 0$ iff $p \in O'$)

[7] The reader is referred to [69] and to Appendix D (§ D.2) for a mathematical definition of a distance surface. The 3-D surface representation of the grassfire transform obtained by replacing time for distance or height was first proposed by Kotelly [82].

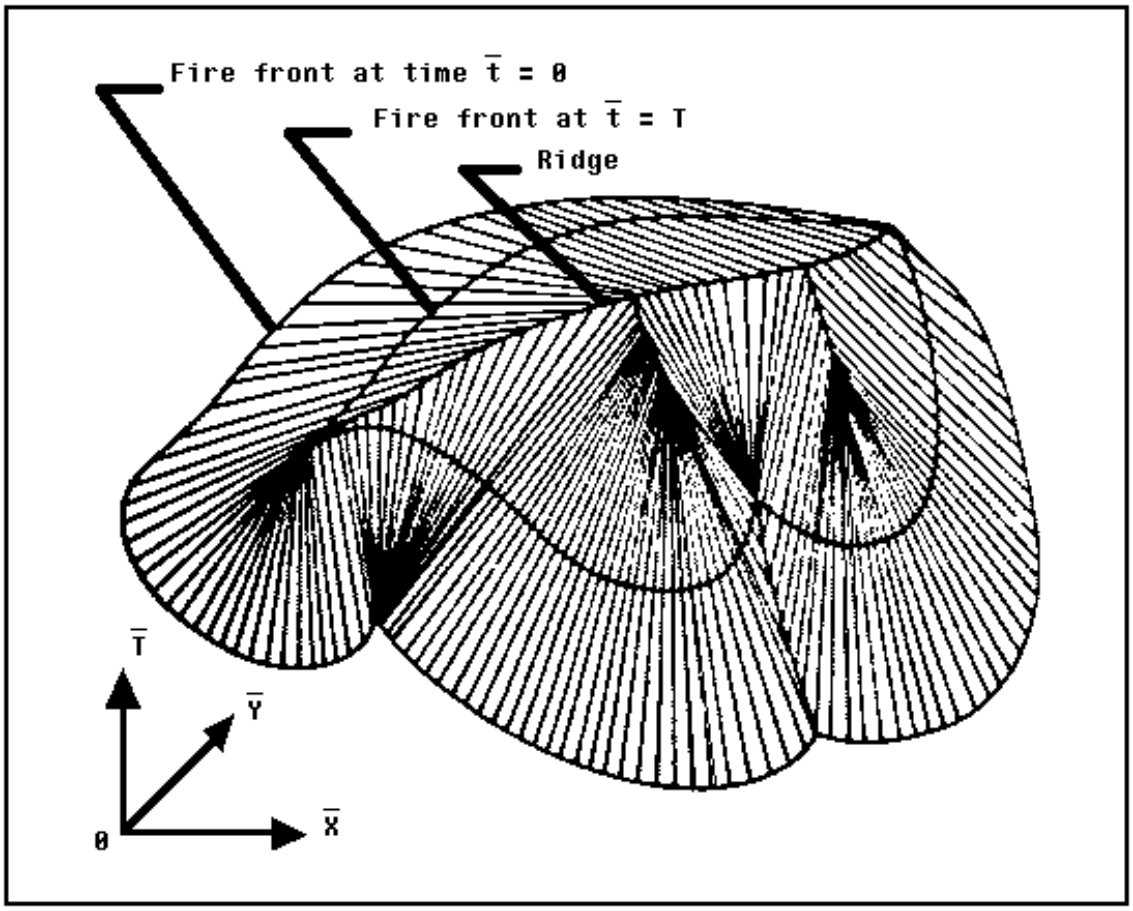


Figure 5.1: 2-D dynamic grassfire, where the vertical coordinate represents time or distance from the ground state. The maximal directional slope of the obtained surface, which is the reciprocal of the grassfire velocity, is everywhere one except at the symmetric axis locus (ridges), where it is undefined. Adapted from [25, Figure 11].

to a multi-valued function f_2 in a space S_2 of identical dimension. The mapping is applied to f_1 by minimizing a distance metric $dist(p, p')$, where p and p' are two points in the space S_1. The distance values of f_2 are given by:

$$f_2(p) = \min_{p' \in O'} [dist(p, p')] \ , \tag{5.1}$$

where $p = p(\overline{x}, \overline{y}) \in O$. Therefore, the first step in our method consists of computing the EDT of a discrete figure-ground image.[8] A simple sequential algorithm exists for computing EDT's [46]. In Appendix D (§ D.1) we describe a very fast implementation of this algorithm which has a computational complexity comparable to the simple chessboard DT.

The second step involves simulating the grassfire propagation. To achieve this, we initiate an *active contour*, in our case a snake, on the figure boundary. A snake consists of an elastic curve that tends to minimize its energy by sensing the *potential surface* on which it is located. In our case the potential surface, H, is taken to be the negative of the discrete distance surface:

$$E_{pot}(p) = \begin{cases} -f_2(p) < 0 \, , & p = p(\overline{x}, \overline{y}) \in O \, , \\ 0 \, , & elsewhere \, . \end{cases} \tag{5.2}$$

This means that potential values correspond to null or negative distance values. It is necessary to invert the distance map in this way so that the snake will reduce its energy by falling down the potential surface. Ground is taken to be at potential 0, the figure contour corresponds to potential values of -1, and inside the figure the potential values vary from -1 to the negative of the maximal possible distance (Figure 5.2). We call the space containing 3-D potential surfaces, H, obtained from equation (5.2), the *distance transform domain* $\mathcal{DT}$.

Having initialized the snake on the locus defined by potential values of -1, it is then activated by sensing the potential surface (ignition of the fire). This is accomplished by observing the first directional slopes of the potential surface, H, in the $\overline{X}$ and $\overline{Y}$ directions[9] and using a *gradient descent* approach (Chapter 2, § 2.5). At each time step $\overline{t}$ of the iterative process, each snaxel position is reevaluated based on the gradient, the elastic internal constraints and certain other external constraints (Chapter 2). Points on the potential surface where the snake folds into thin lines, that is, where the fire fronts merge, are retained as the points of the skeleton. This occurs when the snake reaches an equilibrium, under the

[8] In our case, the figure-ground image is easily obtained from the chain code representation of the boundary of an object (Chapter 4) after the image segmentation has been completed (Chapter 3). To do so we use a region-filling algorithm which takes full advantage of the chain code representation. For a detailed description of such an algorithm see [139] or [1].

[9] In the present case where H is a distance surface, the evaluation of the directional slopes can be directly obtained from the EDT computation (Appendix D, § D.3).

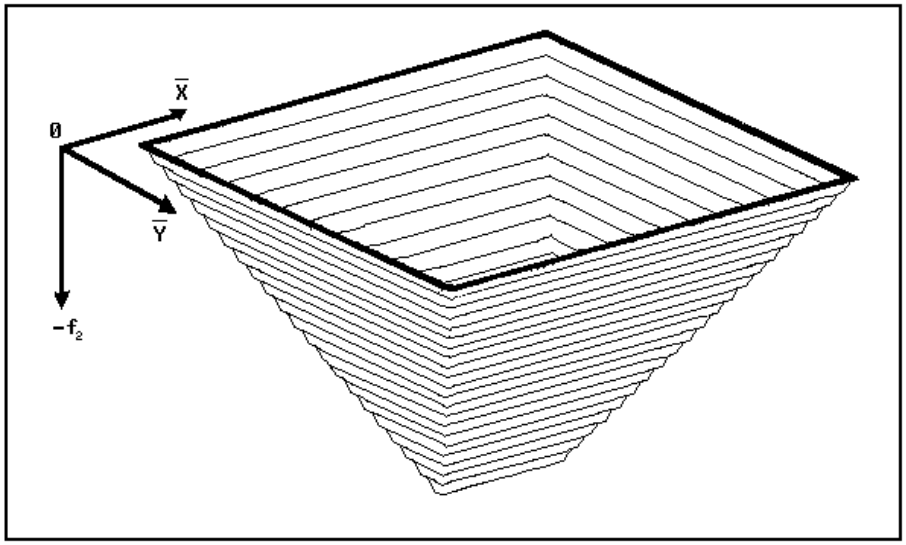


Figure 5.2: Example of a potential surface for a rectangular shape obtained by taking the negative of the Euclidean distance transform.

"steady-support" criterion, or stops moving. An example of the simulation of the grassfire transform is given in Figure 5.3.

5.3.2 Simulation of the Grassfire Transform for Regions Containing Holes

When applying an active contour model to planar regions that are not topologically equivalent to a disc, such as 2-D objects containing holes, a slightly more complex implementation of the grassfire transform is required.

Essentially we must define as many fire fronts as there are boundaries on the object. For each *internal contour* of a region, that is, each contour delimiting a hole, we need to generate a supplementary active contour.[10] Therefore, given a binary object limited by one external boundary and possibly N internal contours, where N is some fixed positive integer, we simply need to initialize N supplementary snakes on the potential surface, one at each internal contour position. Each of the $N+1$ snakes will simulate a particular fire front. We then activate each snake in parallel and use the same gradient descent approach to simulate the grassfire propagation as for the case of a region without holes. Two types of skeletal points are then defined. First, as before, skeletal points are extracted where a given snake

[10]Note that some preprocessing of the binary object may be required to eliminate holes of an insignificant small size. For example, mathematical morphology operations can be used to fill in holes of a predetermined size [135].

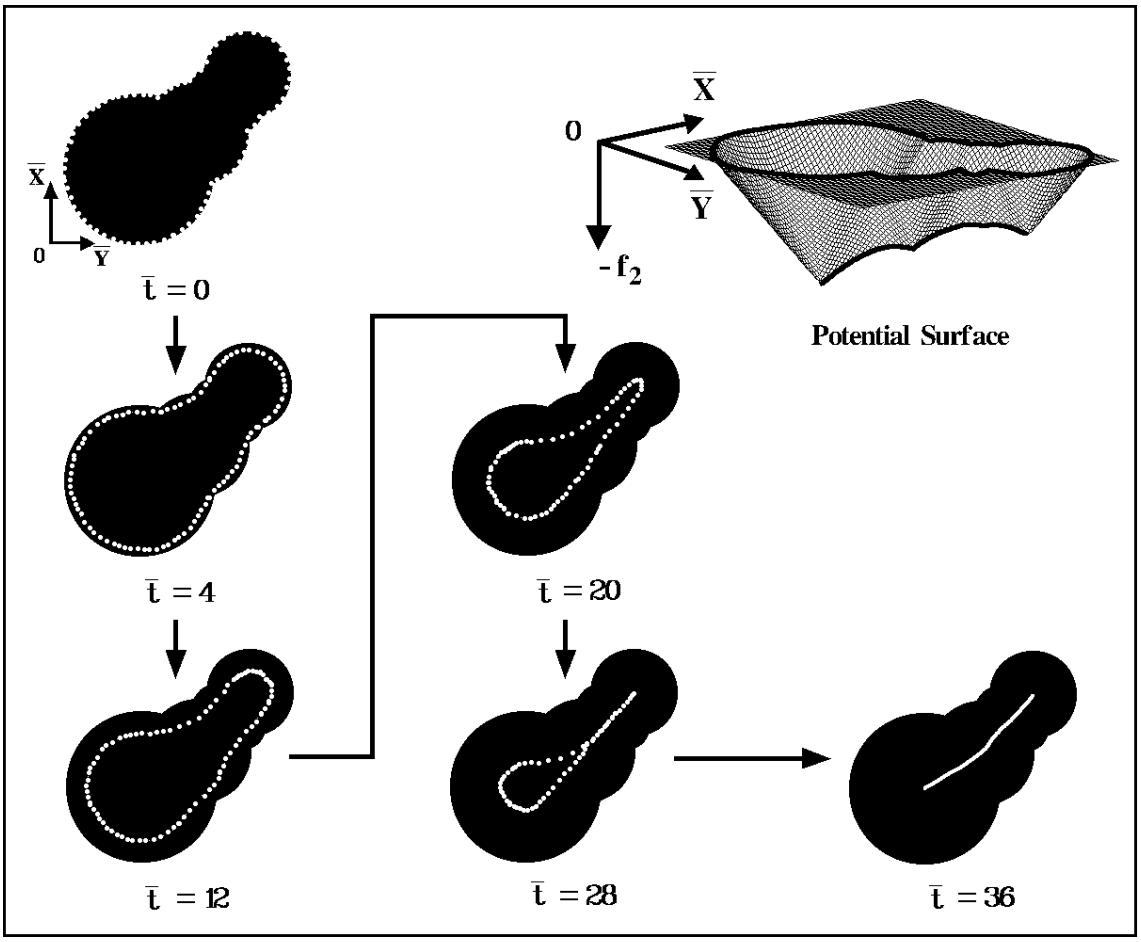


Figure 5.3: Example of grassfire propagation on a blob-like shape. The potential surface is shown at the top right corner. The dark lines superimposed on it represent the snake at its initial and final states. The fire is ignited at time $\overline{t} = 0$ where time corresponds to iteration. The stable state occurs at $\overline{t} = 36$ iterations. This last picture shows the final result when the number of snaxels is increased so that the snake is spatially connected, that is, without any gaps.

folds into a thin line. Second, skeletal points are also identified where two or more different snakes merge or meet each other. The final stage of the grassfire propagation also occurs when the snakes reach a stable state, that is, when the "steady-support" criterion is satisfied or when they stop moving.

Since the snake's activation and inhibition, by the use of the "steady-support" criterion, are dependent only on the forces applied directly to the snake itself (elasticity and external constraints) and on the topography of the potential surface, but not on the presence or absence of other snakes in their vicinity, the actual simulation of the grassfire transform can be computed in a sequential fashion. This approach permits simpler implementations on traditional sequential computers. Furthermore, we can activate the $N + 1$ snakes in any order since they are brought alive independently. An example of a sequential simulation of the grassfire transform for an annulus is given in Figure 5.4.

5.4 Integration of Boundary Information

The second building block of our method, the first being the grassfire process, is concerned with the integration of contour information within the skeleton computation. To do so, *critical points* of the boundary or equivalently, positive curvature extrema are extracted from the original object boundary. We refer to these positive curvature extrema by using the symbol C. Another type of critical point may exist within the interior of a shape, that is, the center of a maximal inscribed circle osculating O at a minimum of one connected set of points along the boundary (an arc of circle). We will consider this second type of critical point in detail in section 5.4.1.

The snake is then *attached*[11] to those critical points, the C's, that will ultimately correspond to the *branch end points* of the skeleton. It is necessary to attach the snake at C's to keep track of them as branch end points and also to keep track of the complete branches they are part of. Otherwise, if a snake was not fixed, it would fall down along the ridge. Critical points along the boundary define *snake segments* that correspond to individual fire fronts. Figure 5.5 illustrates grassfire propagation using an active contour descending the distance map of a rectangular shape and initially fixed at the positive curvature extrema.

The idea of fixing branch end points at positive curvature extrema is justified by the fact that at these points fire propagation collapses right at the start of the grassfire process.[12] The extraction and use of positive curvature extrema as part of the skeleton computation was previously reported and used in "peeling-off" methods [4]. The major difference here

[11] By "attached" we imply that the snaxel is pinned to the potential surface at the spatial position of the critical point using an external force. This is achieved using the so-called *spring* model (Chapter 2, § 2.5). Furthermore, a *tangent discontinuity* (Chapter 2, § 2.5) may be placed at that snaxel to enable the snake to easily fold along a ridge starting from the critical point.

[12] The same idea applies to objects with holes. Every boundary delimiting a hole is processed to extract its significant positive curvature extrema at which a snake is attached.

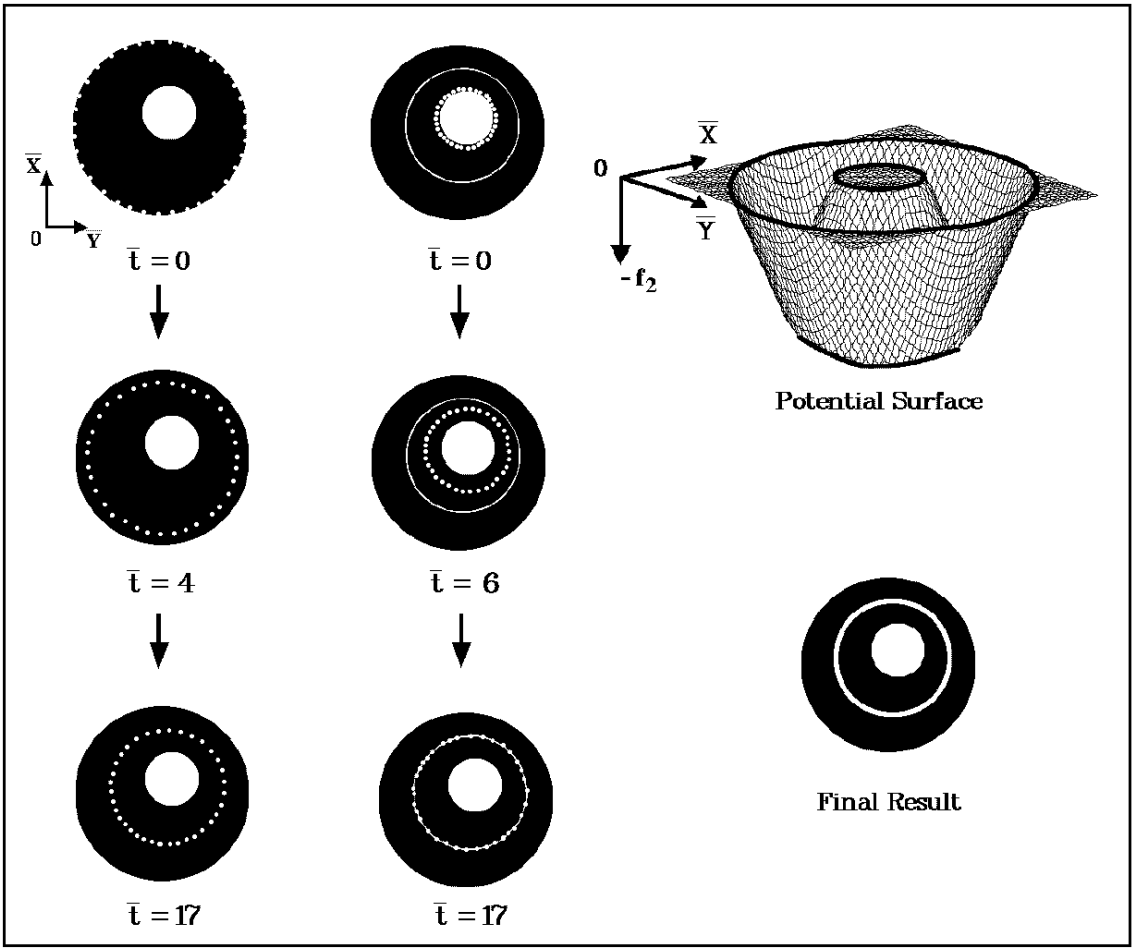


Figure 5.4: Example of grassfire propagation on an object containing an annular region. In the first column, on the left, is shown the fire propagation for a snake activated from the exterior boundary. In the second column is shown the fire propagation for a snake activated from the interior boundary delimiting the hole. In this second column the result of the first fire propagation is also shown as a connected snake (line of smaller width). In this particular case where the exterior and the interior contour do not possess any protrusions, both snakes converge to the same solution.

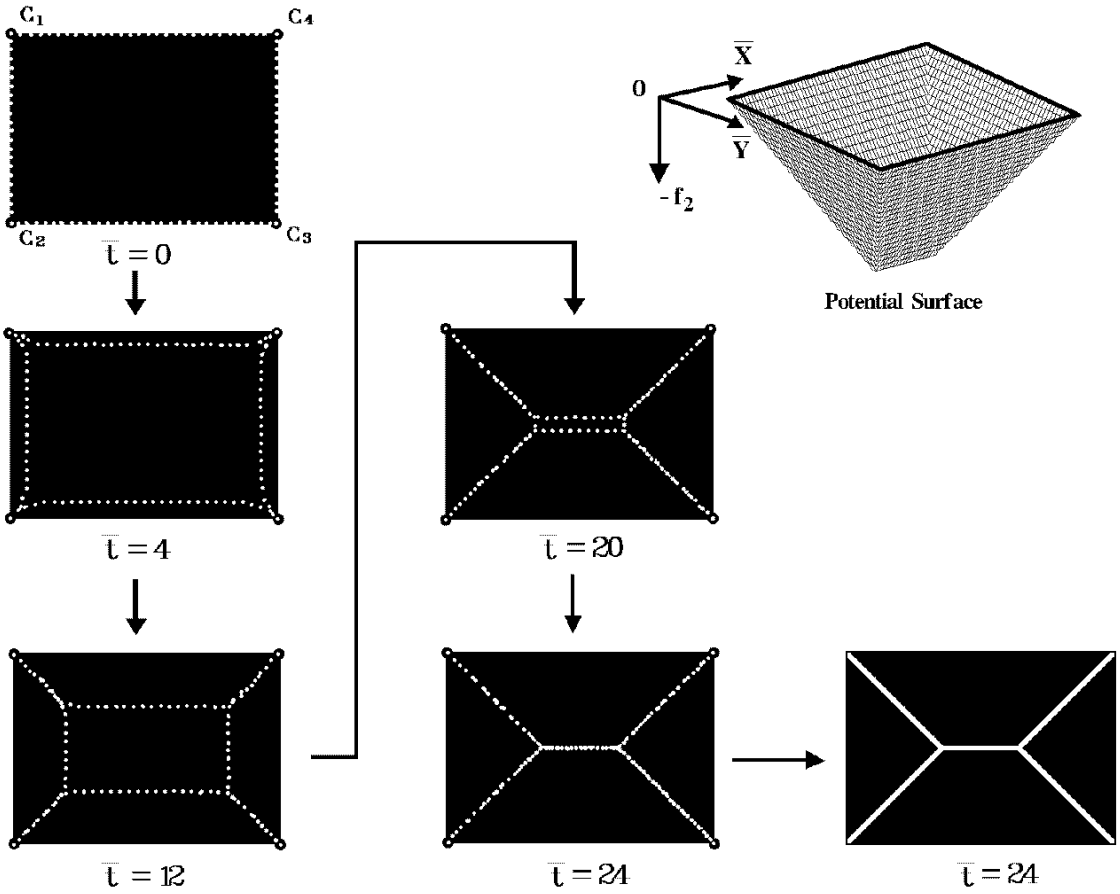
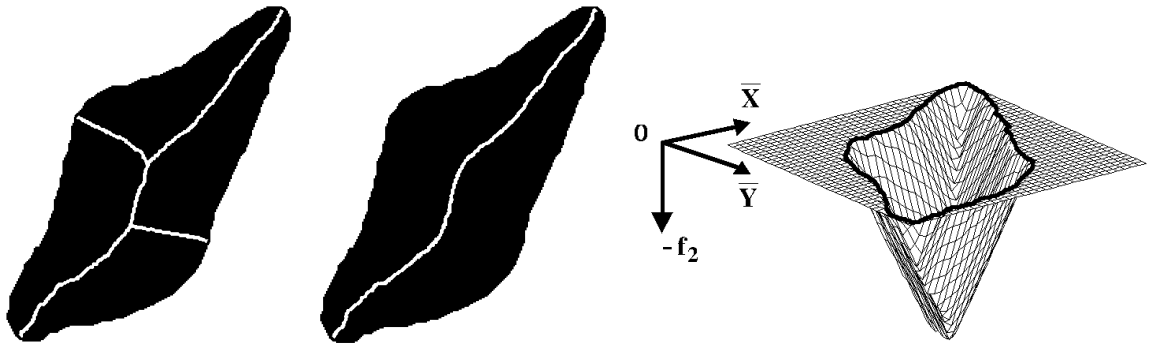


Figure 5.5: Example of a grassfire propagation on a rectangular shape using an active contour model. At ignition of the fire ($\overline{t} = 0$) the snake is attached at the four positive curvature extrema labeled C_1, C_2, C_3 and C_4 emphasized for display purposes. Time $\overline{t}$ corresponds to iteration. The stable state occurs at $\overline{t} = 24$ iterations. The last picture shows the final result when the number of snaxels is increased so that the snake is spatially connected.



(a) $\overline{t} = 42$ (b) $\overline{t} = 36$ (c) Potential Surface

Figure 5.6: Skeletons obtained from grassfire propagation combined with positive curvature extrema extraction at two different scales. In (a) the four most significant C's were retained. In (b) only the two most significant C's were retained. In the latter case, a stable state is reach earlier because the fire fronts are simpler. Time $\overline{t}$ corresponds to iteration.

is we only need to extract positive curvature extrema on the initial boundary of the object. Curvature extremum extraction permits us to adopt a multiscale approach for generating skeletons, as previously proposed in [52], which has the salutary effect of eliminating noise perturbations on the boundary. We have described in detail in Chapter 4 and in other publications [90, 91, 93] a new method, called *curvature morphology*, for extracting curvature features within the framework of a multiscale representation. In the remainder of this section, we emphasize its application to shape skeletonization.

Once the curvature features are extracted (Chapter 4, § 4.3), a skeleton representation at multiple scales can be obtained by employing different criteria for the *significance* of positive curvature extrema. Our own criteria for significance are based on both *maximum relative curvature amplitude* and *region of support*. The latter addresses the issue of whether a curvature extremum is sufficiently isolated from other nearby curvature features (Chapter 4, § 4.3.3). Figure 5.6 provides an example of the implementation of grassfire propagation coupled with positive curvature extrema extraction at two different scales.

5.4.1 Fire Front Propagation from Circular Arcs

There exists one case where the curvature extremum criterion is insufficient for determining the branch end node. This is the case of boundary segments which consist of arcs of circles[13] that have their center of curvature situated within the interior of the object, being

[13]We note that these arcs of circles are readily extracted using the morphological curvature scale-space representation (Chapter 4).

also part of the skeleton (*i.e.*, being a symmetry point), and furthermore being simultaneously an extremity of a ridge of the distance surface (i.e., being the end point of a skeleton branch). A simple procedure will be needed to stop fire propagation at the closest ridge point of the distance surface. Due to the uniformity of the maximal directional slope of the potential surface, and of the gradient descent technique used to update snaxel positions, we are assured that snaxels ignited on an arc of a circle will reach the closest ridge points. This follows from the fact that snaxels crawl down the distance surface following the direction of maximal slope. It is easily shown that this direction is along a normal or a radial line to a boundary point. For all boundary points such lines necessarily lead to the nearest ridge point due to the minimal distance constraint of equation (5.1). Blum called these lines of steepest descent, leading necessarily to symmetry points, *pannormals* [25].[14]

During fire propagation, it is necessary to check snaxels initially ignited on an arc of a circle to inhibit the fire propagation as soon as a ridge is reached. This is required to ensure that the end of the skeleton branch which corresponds to the center of curvature of an arc of circle on the boundary is retained in the case of a ridge starting with an increasing directional slope, that is, a skeleton branch consisting of increasing absolute distance values with regard to the branch end. In this case, if a snake were not fixed it would continue to fall down along the ridge as shown in Figure 5.7. With the snake model, one way to check if a snaxel has reached a ridge is to examine the directional slope which is used to activate the snake. If this slope is lower than one (*i.e.*, the snaxel is on a ridge) or if it changes abruptly in direction (*i.e.*, the snaxel went across a ridge), then we are in the vicinity of a ridge point of the potential surface (Appendix D, § D.3).

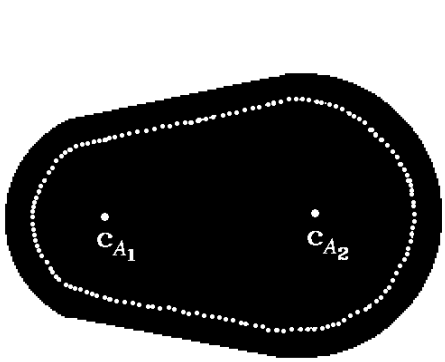
However, an even simpler procedure can be used to efficiently detect a center of curvature which is also a symmetry point and an end of a skeleton branch. Let us first examine points which are simultaneously the center of curvature of a circular boundary arc and a symmetry point. We will refer to these points by using the symbol C_A.[15] We note that the following two properties are shared by snaxels that should ultimately be merged at a C_A point:

1. From curvature morphology analysis, we can initially identify the snaxels on an arc of circle and thereby create a snake segment or fire front. We denote this snake segment S_A.

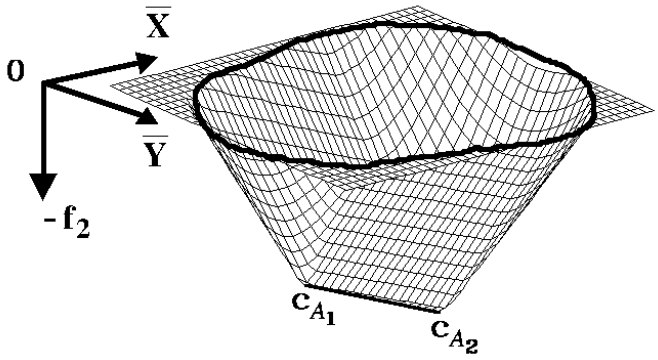
2. During the grassfire propagation, snaxels of S_A should get closer as the number of iterations progress.

[14]Matheron used a similar concept where the normals or radial lines to the boundary are limited in their extent by the skeleton, that is, they emerge from the boundary and terminate at a symmetry point. Matheron called such line segments *edges* [104].

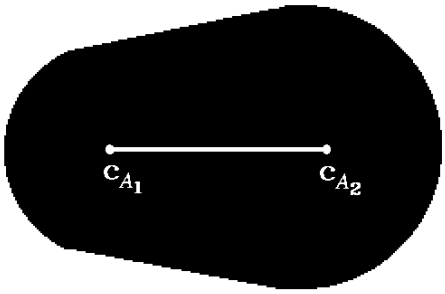
[15]Note that a C_A point, although being by definition a symmetry point, does not necessarily form the end of a skeleton branch (*e.g.*, think of an object with a "rounded-L" subpart or knee). Blum called C_A points f*inite contact symmetry points* [25, 26] to emphasize the fact that they correspond to maximum inscribed circles which osculate the boundary at connected sets of contour points.



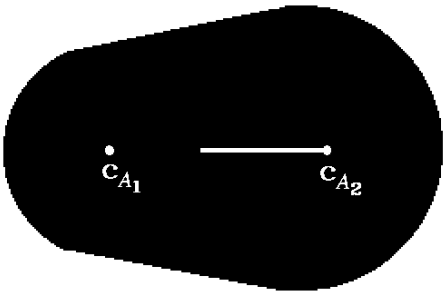
(a) $\overline{t} = 4$



(b) Potential Surface



(c) $\overline{t} = 24$



(d) $\overline{t} = 32$

Figure 5.7: Example of fire propagation on a shape made of two circular arcs joined by straight lines. C_{A_1} and C_{A_2} are the centers of the arcs. In the neighborhood of C_{A_1} the ridge is made of increasing absolute distance values as shown by the potential surface. If the snake is attached at C_{A_1} the true skeleton is obtained as shown in (b). Otherwise, the snake shrinks as shown in (d).

Point 2 arises from the fact that pannormals to the boundary arc S_A all merge at one point, the center of curvature of S_A. Since along any pannormal the slope of the potential surface is maximal, snaxels follow such directions and ultimately converge to one point if such a point exists in the interior of the object. This point of convergence is C_A. Therefore, we propose the following extraction procedure for C_A points which are also end points of skeleton branches:

- If the snaxels of segment S_A converge to one single point, C_A, of the object interior O, then:

  1. C_A is a ridge point (may be verified directly on H).
  2. C_A is the end of a skeleton branch if the snaxels that converged at its location are all constituents of a single fire front, namely of S_A. In other words, C_A becomes a point at which the fire front S_A collapses and gives birth to a symmetric axis.
  3. We consider C_A as a *critical point* like the C's (section 5.4). Therefore, the snake should be attached at C_A, and, optionally, the number of snaxels forming the fire front S_A can be reduced to one single snaxel.

We now have all of the elements required to obtain a skeleton made of merged fire fronts where the fire propagation is modeled and simulated by an active contour model.[16] The next stage in skeletonization consists of creating a graph representation of the skeleton, a step rarely considered in the literature (but see [26, 121, 112, 94]). This is the subject of the following section where we propose a method which takes full advantage of the ensured connectivity of our active contour model.

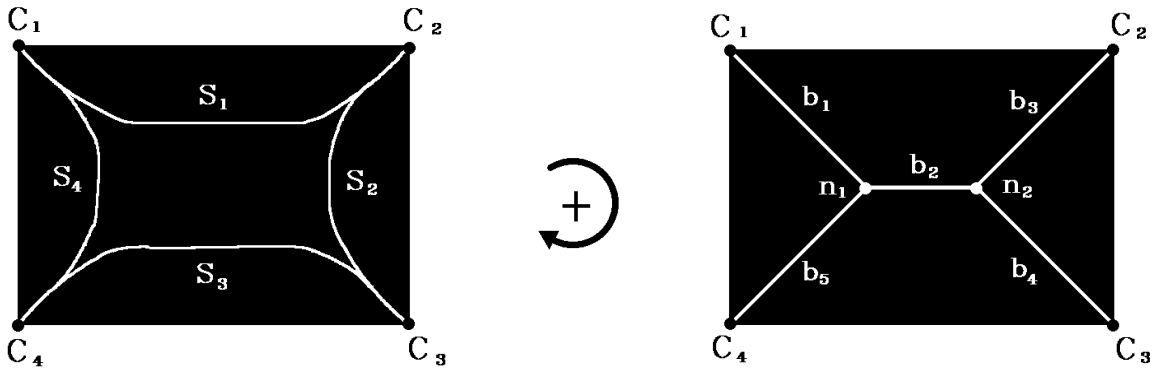
5.5 Graph Representation

The connectivity of our skeleton obtained by fire front propagation is ensured due to the internal constraints of the active contour model.[17] Thus, we can easily create a graph representation of the skeleton in terms of branches, end nodes, branching nodes as well as relations between these elements (*i.e.*, links).

The graph is generated at the termination of the propagation process which is modeled by the snake segments (Figure 5.8). These are completely connected and have no gaps. The actual curve approximation to the snake can be easily improved by increasing the number of snaxels once a "stable" skeleton has been obtained. Otherwise interpolation

[16] More detailed implementation issues are considered in Appendix D.

[17] As mentioned in section 5.2.1, methods for skeletonization based on ridge following on an Euclidean or quasi-Euclidean distance map do not, in general, directly produce a connected skeleton. This is because distances are evaluated only at integer coordinates.



(a) $\overline{t} = 10$ (b) $\overline{t} = 24$

Figure 5.8: Example of fire front labeling used to obtain a graph representation of the skeleton of a rectangular shape. The points C_1, C_2, C_3 and C_4 are "critical points" which are used to attach the snake. Four snake segments S_1, S_2, S_3 and S_4, are defined by the four critical points in (a). The direction of traversal is indicated as clockwise. In (b), five "branches" b_1, b_2, b_3, b_4 and b_5, are defined by five pairs of snake segments, that is $S_1 - S_4$, $S_1 - S_3$, $S_1 - S_2$, $S_2 - S_3$ and $S_3 - S_4$, respectively. Finally, "branch nodes" are defined where three or more snake segments meet. Here n_1 and n_2 are the junction of snake segments $S_1 - S_4 - S_3$ and $S_1 - S_2 - S_3$, respectively.

by straight lines can be used if one is not looking for high spatial accuracy. To obtain the graph representation, we first label each individual fire front, that is, each individual snake segment S defined between two consecutive critical points (Figure 5.8). Critical points may be of two kinds, positive curvature extrema on the boundary, C, or the center of curvature of circular boundary arcs, C_A, that map to ends of skeleton branches.

The traversal of the snake segments from an initial curvature extremum (labeled C_1 in Figure 5.8) is done by associating pairs of snaxels of different snake segments that correspond to the same skeletal locus. For example, in Figure 5.8 snaxels of both the first segment (label 1) and the last segment (label 4) should be found at each skeletal locus on the first *branch* (*e.g.*, b_1).[18] This will be so until a point is reached at the branch where more than two labels are found. This point corresponds to a *branch node* (*e.g.*, n_1). The traversal can then be pursued to identify other branches; for example, the branch consisting of pairs of snaxels labeled 1 and 3 in Figure 5.8. The traversal terminates when the initial snaxel is

[18]In general, snaxels of different snake segments will overlap. However, because we are using a digital grid to simulate the grassfire transform, ridges may be of thickness two (pixels) rather than one. Therefore, snaxels of different snake segments may not always overlap but rather be (connected) neighbors. A simple matching procedure then needs to be defined by taking into account the cases of overlapping and connected neighbors and using the spatial connectivity of the snake segments.

reached. Four simple rules can be used to generate the graph:[19]

- Each critical point (C or C_A) corresponds to an *end node*.
- Each *branch point* consists of two snaxels from two different fire fronts or snake segments (S or S_A).
- Each *branch node* consists of three or more snaxels from three or more different fire fronts.
- *Links*, that is the order relationship between branches are defined at branch nodes during the traversal of the snake.

Branches are processed twice by traversing all of the snake. However, this can be altered so that the pairs of snaxels forming branches are visited only once.

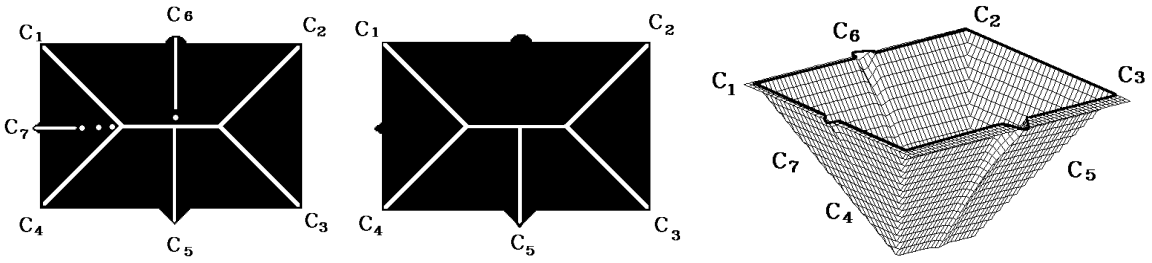
5.5.1 Graph Pruning and Branch Significance

Not all the branches of the skeleton are actually significant. So far the branches terminating at an end node have been generated based on the curvature criteria discussed in section 5.4 or based on the extraction procedure of center of curvature C_A discussed in section 5.4.1. In the case of the branches generated from positive curvature extrema, C, if the thresholds on relative curvature amplitude and region of support were relaxed, we might obtain spurious branches. However, it is possible to evaluate the skeletal branches on another basis, that is, by considering the *ridge support* along the branch.

We examine ridge support computed at each skeletal pixel by looking at the *local deformation* introduced by the ridge on the distance surface at that skeletal pixel position [94]. If the deformation is limited in its spatial extent after a certain point along the axis branch, then it is not "significant". The use of ridge support as a branch significance criterion is justified by the fact that noise or small deformations on the object contour generate small deformations of the distance surface along the corresponding axis branch. The relative number of skeletal pixels that lie on that part of the ridge that is changing significantly can be used in combination with the curvature significance measures to decide whether or not a branch should be pruned from the graph (Figure 5.9).

The notion of ridge support based on the local deformation introduced by a ridge on the distance surface is an intuitive or qualitative formulation; we need an easily computable measure to apply such a notion for graph pruning. For this let us first go back to the work of Blum who first proposed the use of the *velocity of skeleton branch formation*, that is, the speed at which fire fronts merge, as a measure of the "smoothness of [fire] fronts" as they collapse at symmetry points [24, 25]. We will refer to this velocity of formation of

[19] The same rules apply for objects with holes.



(a) (b) (c) Potential surface

Figure 5.9: Example to show how "ridge support" can be used as a "branch significance criterion". In (a), all possible branches have been generated. Branch segments made of skeletal pixels lying on a ridge that introduce very small deformations on the distance surface are shown as dotted lines. These dotted lines represent symmetry points for which the slope of the symmetric axis M_A is greater than a fixed threshold ($M_A > M_{A_{max}}$, $M_A < 1$; see text). In (b), only the branches with high ridge support are kept.

a skeleton branch or symmetric axis by the symbol V_A. Smoothness, as Blum refers to it, corresponds to our idea of local deformation of the distance surface. The distance surface is in fact a kind of photographic memory of the complete grassfire propagation. Where fire fronts progress "smoothly" the distance surface is locally smooth. This corresponds to points of maximal slope of one. Where fire fronts collapse or merge with each other the distance surface deforms and a ridge is created. The relative lack of smoothness of the deformation introduced by merging fronts depends on the angle at which they meet. Blum called such an angle the *generating angle* [24, 25]. The sharper the generating angle, the greater the velocity of formation of the symmetric axis, V_A, and the local deformation of the distance surface. The limiting case are for parallel fire fronts which merge at an infinite V_A and create a maximal deformation of the distance surface (Figure 5.10).

In the continuous domain, V_A can be defined as the time derivative of length along a symmetric axis, l_A:

$$V_A = \frac{\partial l_A}{\partial t}\,. \tag{5.3}$$

In the discrete domain we can approximate V_A by finite differences:

$$\overline{V_A} = \frac{\overline{l_A}}{\Delta \overline{t}}\,, \tag{5.4}$$

where $\overline{l_A}$ is the Euclidean distance of the discrete path along the symmetric axis from one symmetry point to another. But, since we are using a distance surface to "memorize" the

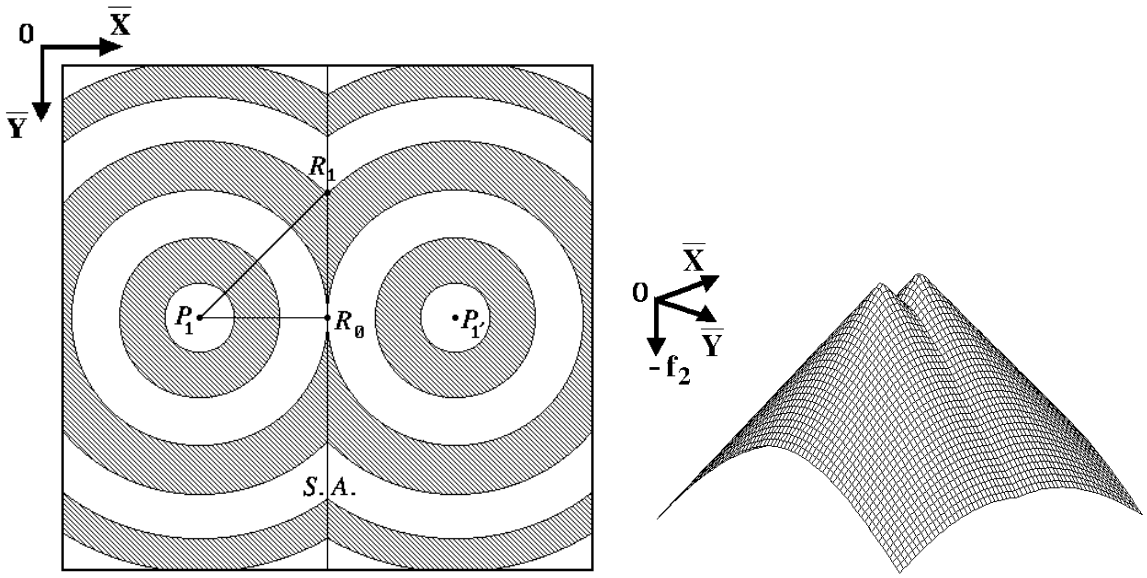


Figure 5.10: The velocity of formation of a symmetric axis and ridge support. In (a) are shown fire fronts generated by two isolated points, P_1 and $P_{1'}$, for an unlimited field of grass. The fire fronts are illustrated as an alternating wave pattern. The vertical centered line between P_1 and $P_{1'}$ represents the symmetric axis ($S.A.$) of infinite length, of the grassfire transform. The symmetric axis is first created at a midpoint, R_0, on a straight line segment joining the two excitatory dots, with an infinite speed ($V_A(R_0) = \infty$). As the symmetric axis grows, its velocity decreases asymptotically to the space velocity ($V_A \to 1$). For example, $V_A(R_1) = \sqrt{2}$. Note that as one travels along the symmetric axis, away from the initial symmetry point, the "generating angle" (see text) increases asymptotically from a completely closed angle of 0^o, at R_0, to a completely open angle of 180^o. For example, at R_1 the generating angle equals 90^o. Adapted from [24, Figure 1]. In (b) is shown the potential surface representation of this grassfire transform. The only ridge on this surface correspond to the vertical $S.A.$ between P_1 and $P_{1'}$. Note how the deformation produced by merging fire fronts (in (a)) varies along the $S.A.$. In correspondence to the symmetry point R_0 the deformation is maximal; at this point the surface folds with an angle of 90^o. The further away you are from R_0 along the $S.A.$, the smoother is the deformation.

complete grassfire we can replace time by the distance height of the distance surface obtained from the Euclidean Distance Transform, EDT:

$$\overline{V_A} = \frac{\overline{l_A}}{\Delta EDT(\overline{l_A})} \,. \tag{5.5}$$

As we have seen, in the case of parallel merging fire fronts, $\overline{V_A}$ may reach infinite values. Furthermore, $\overline{V_A}$ is always greater than one, that is, it is always greater than the space propagation velocity of non-merging fire fronts [24, 111]. Therefore, rather than using $\overline{V_A}$ directly as a measure of ridge support we propose[20] to use its inverse, that is, $\overline{V}_A{}^{-1}$ ($0 \leq \overline{V}_A{}^{-1} < 1$).

In fact, it is easily shown that the inverse, $\overline{V}_A{}^{-1}$, is nothing more than the amplitude of the directional slope at a symmetry point in the direction of the tangent to the symmetric axis.[21] We refer to this slope amplitude by using the symbol M_A:

$$M_A = \frac{1}{\overline{V_A}} = \frac{\Delta EDT(\overline{l_A})}{\overline{l_A}} \,. \tag{5.6}$$

M_A varies from null values (for parallel merging fronts) to unitary values at which no symmetry point can exist (*i.e.*, $M_A < 1$; see Appendix D, § D.2.1). Fixing a threshold, $M_{A_{max}}$, on the highest acceptable values for M_A ($M_A \leq M_{A_{max}} < 1$) provides us with a simple quantitative criterion for graph pruning on the basis of ridge support (Figure 5.9).

Finally, we note that the slope amplitude at a symmetry point, M_A, provides a good first approximation to the *boundary-axis weight* of Blum and Nagel [26]. The boundary-axis weight is another kind of branch significance measure assessing the "importance of the [symmetry points in] representing the boundary" [26]. Essentially, the boundary-axis weight is defined as a "limiting ratio of boundary length to symmetric axis length." Because this measure is defined in the continuous domain within the framework of differential geometry, it is in general difficult to evaluate it accurately and use it for practical applications. In contrast, M_A is easily evaluated in the discrete domain and provides for accurate results as a consequence of our use of an Euclidean metric. It can also be shown that M_A is closely related to the boundary-axis weight and is even equivalent to it for straight portions of a symmetric axis and for symmetry points with infinite $\overline{V_A}$.

[20] Montanari has previously used $\overline{V}_A{}^{-1}$ as a criterion for branch significance [110].

[21] Note that the directional slope of a symmetry point is not defined (geometrically) on the same basis as the directional slope of the distance surface (Appendix D). In fact, at symmetry point locations, the directional slope with respect to the distance surface is undefined (*i.e.*, the surface is locally non-regular). Rather the directional slope of a symmetry point is defined with respect to a 3-D curve, the symmetric axis, instead of a 3-D surface.

5.6 Advantages and Variations of the Proposed Skeletonization Algorithm

In this section we first summarize the main advantages of our algorithm for skeletonization of planar shapes and then present extensions which show the flexibility of our approach.

5.6.1 Advantages of Our Method

In addition to the benefits of using an Euclidean distance mapping, our algorithm for shape skeletonization possesses new features and has advantages over previously reported algorithms, as described below.

- The integration of boundary information permits us to reduce noise sensitivity and to obtain an explicit multiscale description in terms of boundary convexity significance.
- Our skeletons are always connected due to the nature of the active contour model we have used. We are actually fitting elastic strings to a potential surface, with these strings being implicitly connected due to their internal constraints.
- Combining convexity significance with ridge support along a skeleton branch permits us to further prune the skeleton graph, if necessary, thereby rendering skeletons with no spurious branches.
- Due to the connectivity of our skeleton model, a graph representation can be easily obtained. A hierarchical model of the object could be built by integrating the convexity significance and ridge support of a branch with the graph representation.
- The active contour model permits "friendly" user interaction when required. For example, in certain cases it might be useful to observe the effect of different curvature criteria on the multiscale representation. Also, we may wish to compare different kinds of skeletons that can be obtained: SAT-like, PISA-like [100], or augmented skeletons possessing branches terminating at concavities (see section 5.6.2). New branches can be added simply by using a spring model to attach the snake to particular points on the distance surface. Conversely, branches can be deleted by simply removing the corresponding spring forces.
- The active contour model also provides us with a natural way of defining *dynamic skeletons*. "Dynamic" refers to skeletons that evolve with time. This is especially important when processing nonrigid objects such as cells [115, 52, 94]. With an appropriate sampling rate, generally small deformations will occur from frame to frame and we can use the previously computed skeleton (at time $\bar{t} - \Delta\bar{t}$) to initialize the

snake on the new distance surface at time $\bar{t}$. A stable skeleton is then rapidly obtained since the snake is already close to the optimal solution. If larger deformations occur, that is, new curvature extrema appear or old ones disappear, we can use the previously computed skeleton as an initial guess, but must then add or delete spring forces to generate or drop branches, respectively (Figure 5.11).

- In comparison to the fastest known serial algorithms for skeletonization of 2-D shapes based on distance mapping, our algorithm has comparable numerical complexity. Our implementation of the Euclidean distance mapping is as fast as the chessboard distance mapping (Appendix D). The next step, grassfire propagation using the active contour model, necessitates less computation than most other methods. These require a "visit" to each pixel in the distance map to detect ridges, followed by an additional pass over the distance map to fill-in gaps between detected ridges. In general, a "visit" to a pixel of the distance map implies an access to the eight neighbors of this pixel. In our case, by using snake segments bounded by critical points, C's and C_A's, with a low spatial sampling rate, that is, using few snaxels, and due to the implicit connectivity of our active contour model, a solution is found in fewer steps (see also section 5.6.2 and Figure 5.12). On the other hand, our method requires a supplementary step when compared with most methods, namely the extraction of curvature extrema. This step can be accelerated by using techniques such as the HDC to filter orientation data along the boundary (Chapter A) and also by using the notion of ridge support to obtain a simple curvature significance criterion. Parallel implementation of our algorithm could also be achieved since both the EDT and active contour model permit a parallel computation. Furthermore, in addition to being fast, our method is more accurate and provides a simple way of building a graph from the skeleton.

One of the most interesting advantages of our method is the flexibility that the active contour model provides in order to compute skeletons in different or more efficient ways. This idea will become clearer in the next subsection.

5.6.2 Variations of the New Algorithm

Since all of the information about the skeletal locus is embedded or "memorized" in the distance surface and since we know the location of the branch ends by boundary curvature analysis, we can alter our active contour model to obtain even more efficient ways of computing a skeleton.

- The first obvious thing one can do is to compute an approximation of the grassfire propagation by having a low spatial sampling rate between knot points, C's and C_A's.

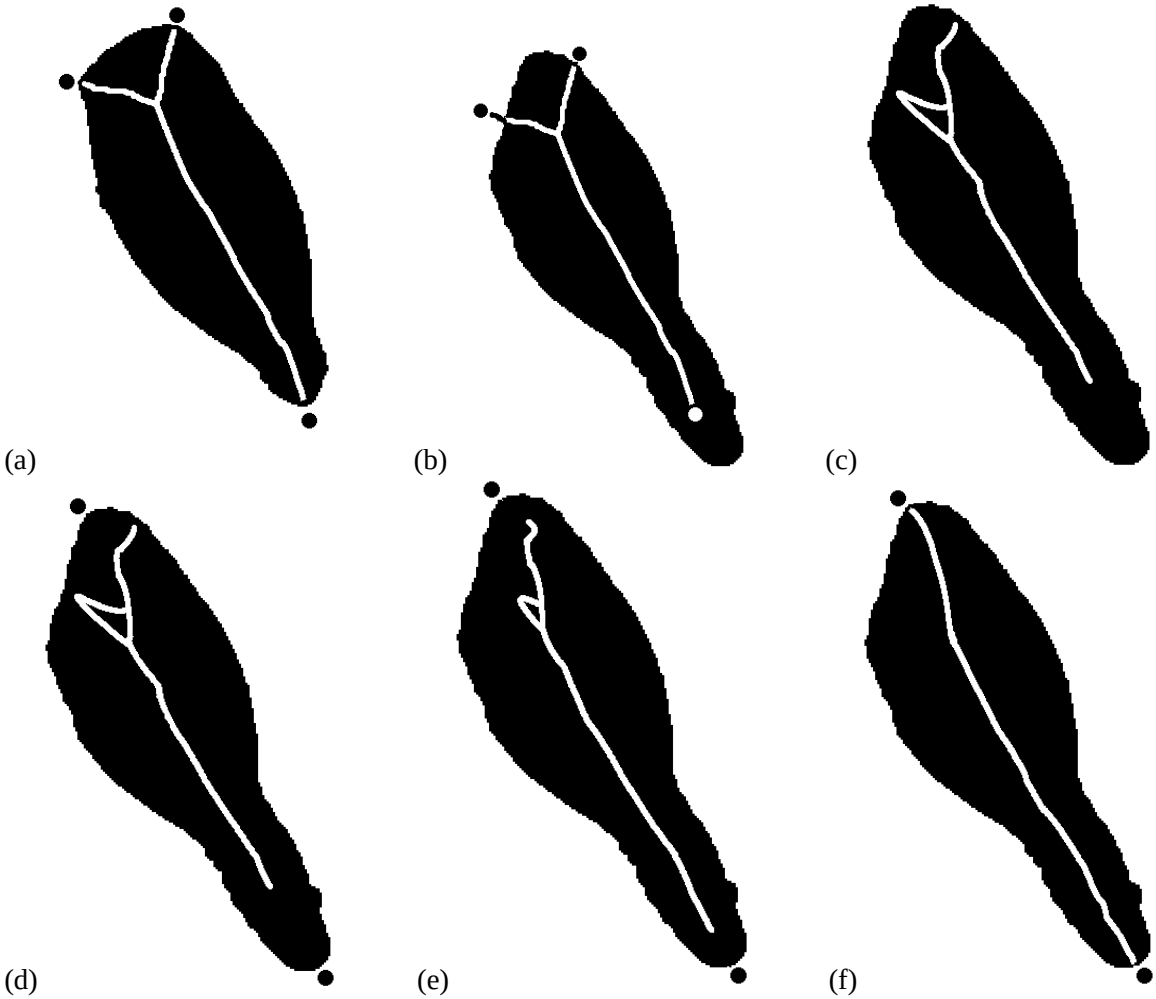
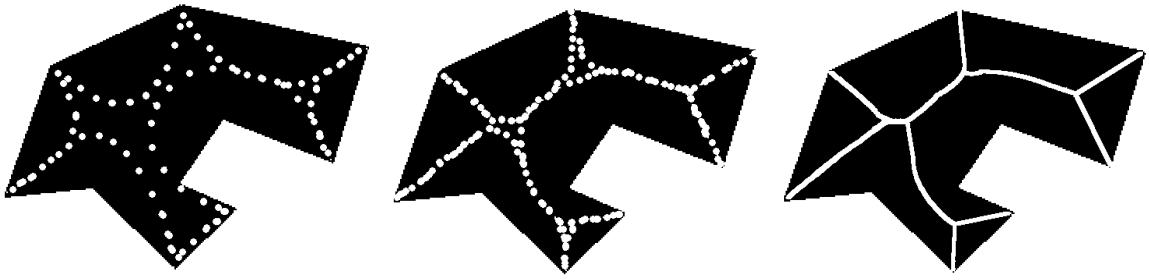


Figure 5.11: Example of the use of dynamic skeletons for the shape representation of a living cell in two successive frames, f_0 and f_1. Springs used to fix the snake at curvature extrema along the boundary are shown by dots. In (a) is shown the cell shape at frame f_0 with its skeleton (stable state reached after 35 iterations). The new cell shape is shown at frame f_1 in (b) to (f). In (b), the snake is initialized at the previously computed skeleton position. The effect of the removal of all springs corresponding to the curvature extrema found at frame f_0 is shown in (c). In (d), two new springs are used to attach the snake to the two most significant curvature extrema. An intermediate stage between (d) and the final result is shown in (e). Finally, in (f) we show the new stable state of the snake reached after only 15 iterations.



(a) $\overline{t} = 15$; 82 snaxels (b) $\overline{t} = 20$; 164 snaxels (c) $\overline{t} = 25$; 328 snaxels

Figure 5.12: Example of a grassfire propagation in three stages, with an increasing number of snaxels. In (a), a first coarse solution is obtained after 15 iterations. In (b), a more refined solution is then rapidly obtained by increasing the number of snaxels. In (c) is shown the final result for a continuous snake.

Once a stable solution is reached, one can increase the number of snaxels to obtain a better solution. This approach can dramatically reduce the computational time since the initial propagation step with few snaxels brings one close to a solution in a few short iterations; where the time to perform one iteration is proportional to the number of snaxels. Subsequently, only a few iterative steps using a large number of snaxels are necessary (Figure 5.12).

- Another improvement to reduce time complexity is to check the formation of the symmetric axes as fire fronts or snake segments merge. If snaxels of different snake segments overlap or are connected neighbors they create a new symmetry point thereby implying that no further processing is required for these snaxels. This further implies that as fire fronts merge, the grassfire propagation will move faster in terms of computer time since fewer snaxels will have to be updated. The best way to implement such a grassfire simulation is by "shortening" the snake as snaxels from different snake segments merge. The removal of snaxels can be performed in two ways: introducing position discontinuities by breaking the snake in parts (Appendix B) or reinitializing shorter snakes, one for each shortened snake segment or fire front.

- Different kinds of skeletons can be obtained by using boundary information. For example, if one locates the center of curvature of arcs of circles along the boundary, SAT-like skeletons are obtained. If instead, end nodes are placed at the middle of these arcs of circles (*i.e.*, on the boundary), PISA-like skeletons are generated.

- Our method seems to be naturally extendible to 3-D problems. In this case, the active contour model becomes an *active surface model* [143]. The potential surface

becomes a *potential volume* and is computed using the same EDT algorithm, but this time for 3-D objects (for examples of DT's in 3-D see [118, 28, 149]). Curvature information can also be used, this time by looking at the extrema of principle curvature of the bounding surface[22] defining the shape of the object [162]. The physical analogy becomes one of wave (surface) propagation in space instead of fire propagation on a planar surface. Loci where the waves merge define the skeleton in terms of three types of geometrical entities: points, curves and surfaces.

- Our method can be used as a refinement procedure for other fast but less accurate algorithms. For example, using Arcelli and di Baja's algorithms based on the city block DT or the chessboard DT [7, 13], one obtains a connected skeleton which usually has many spurious branches or misses some significant ones due to the errors introduced by the use of a non-Euclidean metric. This skeleton could be used as an initial solution for a snake. The distance map would be recomputed using the EDT algorithm and a stable skeleton could then be rapidly obtained. The last step could consist of pruning the skeleton by examining ridge support to compute branch significance measures as proposed in section 5.5.1. This method of using a skeleton defined on a coarse DT as an initial guess provides a possibility of bypassing the stage involving curvature analysis to extract critical points, C's and C_A's.

- Finally we note that an alternate sequential approach to skeleton generation based on the same main features we have used to implement the grassfire propagation algorithm is possible. This is based on a *ridge following* algorithm. The idea would be to allow snakes to "grow" from curvature extrema towards local maxima of the distance surface.[23] Thus snaxels would descend along ridges of the distance surface as they are generated. Branches would therefore be generated one by one and the graph representation would be directly obtained. However, problems might occur with the connectivity of the generated graph. Such "growing" snakes can also be used after the grassfire transform has been performed to segment a region into subparts. For example, branches can be generated in correspondence with concavities by initially fixing snakes at negative curvature extrema and then making them grow by descending the distance surface in the steepest gradient direction. This provides a new way of segmenting a planar shape [94] (Figure 5.13). Note that since concavities generate multiple pannormals which are orthogonal to the boundary [25], there exists more than one direction (of maximal slope amplitude equal to one) for the snake to crawl down the distance surface when starting at a concavity. In Figure 5.13 we use

[22] Extracting curvature information from a 3-D surface becomes the most difficult problem to solve in comparison to the 3-D EDT computation and the simulation of wave propagation. This might reduce the applicability of our method for 3-D objects.

[23] This idea of using growing snakes has been previously reported and applied to the contour extraction problem in noisy images [164, 47, 48].

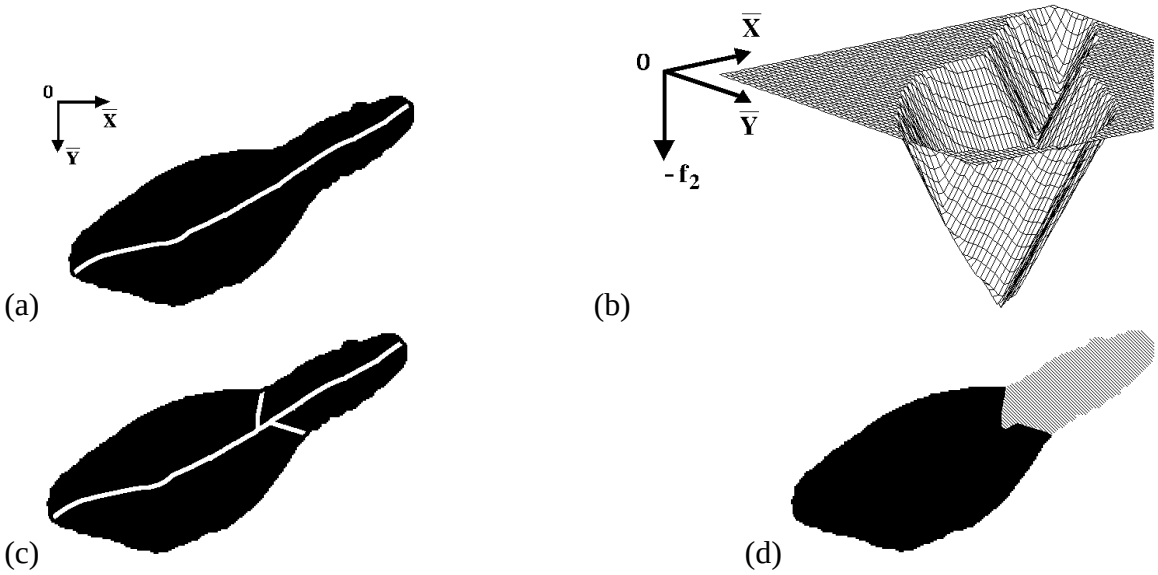


Figure 5.13: Example of the use of concavities for shape segmentation in the case of a shape of a cell. The usual skeleton generated for the two most significant convexities is shown in (a). (b) shows the potential surface used. Two branches are added in (c) by growing snake segments from significant concavities. In (d) is shown the segmentation of the cell shape into its two main parts, that is, the cell body (in black) and its pseudopod (in grey).

the direction of the radial orthogonal to the average orientation at the negative curvature extremum. The "average orientation" may be obtained from the smoothed or averaged chain code representation of the boundary (Chapter 4).

5.7 Conclusion

In this chapter we have presented a new method for shape description of amorphous planar objects on the basis of an active contour model. Such a model has permitted us to combine a class of contour-based shape descriptors (*i.e.*, curvature features) with their counterparts, a class of region-based shape descriptors (*i.e.*, symmetry points). We have been careful in optimizing numerous implementation aspects of our shape description method. We have used an Euclidean metric for optimal accuracy and the active contour model has permitted us to bypass some of the discretization limitations inherent in using a digital grid. We have performed noise filtering on the basis of both contour feature measures and region measures, that is, curvature extremum significance and ridge support, respectively, to obtain "robust" shape descriptors. We have also proposed other improvements and variations of the algorithmic implementation (section 5.6 and Appendix D).

Our motivations for the design of new shape descriptors were an outgrowth of our need to describe the form of a cell in both the static and the dynamic cases [115]. Furthermore, in light of the difficulties inherent in the image segmentation problem (Chapter 3), shape descriptors are thought to be helpful to ease the segmentation and tracking of cells in noisy intensity images.

In the following subsection we conclude by emphasizing the needs of cell tracking in terms of shape descriptors. We also outline how the new skeletonization algorithm we have presented in this chapter should be helpful to address these needs.

5.7.1 Skeletonization for Tracking Deformable Shapes

When tracking a deformable object such as a cell, two essential aspects of its morphological description are of concern to us: the shape segmentation of a cell into its subparts or primitives, and the recovery of its "process-history" or, in other words, the recovery of the evolution of its primitives and local deformations. The first aspect relates to the "descriptive" and "perceptive" problems of biological shape, and the third one, to some extent, to the "developmental problem" (§ 5.1.1). With the new tools described in this chapter we are capable of addressing these problems and providing solutions to them, thereby achieving our initial goal (§ 5.1.3). In the following paragraphs we summarize how our shape description method can be used in solving "biological shape problems."

Let us first consider the hierarchical shape description problem of a cell. Essentially, a cell is made of two types of primitives: its *body* or "central" shape subpart and its *pseudopod(s)* or "protruding" shape subpart(s) [115]. Pseudopods occur over a wide range of sizes (scales) and are "built" along significant or global convexities [52]. The SAT has been used in the past as the most efficient shape descriptor of protrusions [161, 115], and this for a continuum of scales [52]. The graph representation of the skeleton can then be used to segment the shape into its primitives [26]. Also, the branch data along a symmetric axis can be used to further segment a skeleton; for example, by locating necks or saddle points [112], or by using the ridge support criterion (Figure 5.9). Furthermore, since we have access to boundary features, the shape of an amorphous object can be segmented by relating concavities to the corresponding symmetric axis (Figure 5.13).

Segmentation of a shape using both boundary concavities and convexities also applies to the perception problem for which the outline of a shape is perceptually best summarized at these extrema of curvature (Chapter 4, § 4.1). As previously observed, it permits us to relate significant concavities, where we initiate segmentation, to the shape symmetries, along the symmetric axis, and, thereby, toward significant convexities. Also, our multiscale approach for defining curvature extremum significance addresses the issue of which extrema should be "perceptually selected" (Chapter 4, § 4.3.3).

Finally, shape deformation in time and the evolution of pseudopods can also be favorably described by a skeleton-based approach. The "history" of the skeleton is used for

this purpose. For example, branches corresponding to convex deformations which persist over a long period of time are kept or labeled as significant historical events which have the potential to lead to the formation of pseudopods. On the other hand, branches which do not survive can be removed from the process-history of a cell and are considered to be noisy events. This suggests another meaning to noise filtering which should be compared to the ridge support and curvature extremum significance criteria. Pseudopod activity can also be described using the skeleton representation. As the symmetric axis varies in its attributes, such as length and associated width function, or in comparison to other axes (*e.g.*, pseudopod dominance with respect to the cell motion [115]), the process-history of the cell subpart can be recovered *a posteriori* and predictions may be made, such as, future deformations, and motion direction. From the point of view of computation efficiency, our particular implementation on the basis of an active contour model permits us to ease the tracking of deformations and the evolution of protrusions (Figure 5.11).

Following the path pioneered by Harry Blum, we have attained our goal, within the scope of this thesis, of designing and implementing a powerful shape description system particularly well suited to tracking amorphous forms such as cells. We are now in a position to perform experiments with cell tracking in order to refine our understanding of the “social behavior” of cells and to achieve even more insight into biological shape problems.

Chapter 6

Conclusions

This thesis has attempted to address two types of problems associated with nonrigid objects deforming in the plane: (i) image segmentation and tracking, and (ii) biological shape. We decided to use a new concept in tackling these problems, namely the active contour model or "snake." The snake model has many interesting features; one of which — its dynamic behavior — motivated us to explore its applicability to our needs.

First, in Chapter 2, we analyzed the snake model in detail, pointing out its limitations and providing a number of improvements. The snake model has three major limitations: it requires smoothness constraints from the data, powerful initialization mechanisms, and high level processes to fix the numerous snake parameters. Of the few modifications we proposed to improve the original snake model, two of them are seen as essential. First, the different forces acting on the snake had to be normalized to ensure stability, and we proposed different mechanisms to do so; for example, by imposing a saturation value on the slope values of the potential surface. Second, since the original terminating criterion of the optimization process is inadequate, we proposed the "steady-support" criterion, based on a topography measure of the potential surface on which the snake crawls.

Then, in Chapter 3, we applied the snake model to the problems of image segmentation and tracking. Although we were successful to some extent in applying it to both problems, we gave counterexamples where failures of this method, in its present state, were unavoidable. Two types of failures were detected. First, in the static case, due to the global nature of the optimization process and to the relatively simple "steady-support" criterion, we demonstrated how the snake could miss or lose some of the details of the contour edges. Then, in the dynamic tracking case, we showed how the snake was limited to following only small deformations of the potential surface, or equivalently of the trace of the object contour, as a consequence of the use of a gradient descent search technique.

We were faced with two possibilities: try to improve the segmentation technique or explore other types of image analysis processes which may help the segmentation and the tracking. We opted for the second choice, looking at shape descriptors, motivated by the

results and conclusions of some of our predecessors [161, 115, 52]. Thus, in the second part of this thesis we described the result of our research on shape; however, we were not able to extend this work to actually use the shape data to improve segmentation and leave this for future work. We studied the two complementary classes of shape descriptors: boundary- and region-based. Starting with the outline of an object, we proposed in Chapter 4 a new method for contour segmentation. By combining linear filtering techniques, such as Gaussian smoothing, with nonlinear ones derived from mathematical morphology, we were able to produce a contour segmentation technique invariant under rigid transformation, robust toward noise, and providing an explicit multiscale representation. In particular, we showed how our scale-space representation of curvature features of the contour was well behaved in comparison to more traditional representations based solely on linear filtering.

Finally, in Chapter 5, we dealt with biological shape problems by proposing a method combining both contour- and region-based descriptors. This was done in an attempt to obtain a more powerful representation for shape on the premise that both types of descriptors provided useful, and to some extent complementary, features. Our original solution consisted of using the snake model to simulate the "grassfire transform" which implicitly relates contour features such as curvature extrema to region features such as shape symmetries. The principal characteristics of our shape analysis method are the following: use of the Euclidean metric, invariance under rigid transformations, a multiscale representation, an explicit relationship between contour and region features permitting us to segment a shape into its primitive subparts, a definition of significance measures for both types of shape descriptors — such as the "region support" of curvature extrema and the "ridge support" of skeleton branches.

6.1 Future Directions

6.1.1 Further Improvements to the Snake Model

The snake model requires further improvements. For one, convergence should be looked at carefully. Under which precise conditions will the snake converge to a desirable solution? Can we define more rigorously what a "desirable solution" is, other than just saying that it is the bottom of the "best" valley? What maximum deformations of the potential surface, H, should the snake be able to follow or track? Our insights into this method for the extraction of curves or contours — insights gained from our understanding of the nature of the failures we were faced with — suggest that one should seek appropriate 3-D local shape descriptors of H.

The snake model is used to recover the bottom of certain valleys of H, but this is done by simply relying on a gradient descent search technique. Furthermore, the bottom of a valley is characterized simply by minimal height or, equivalently, some gravitational potential energy measure. It seems that one should look at how to best characterize a valley or ridge

of H. For example, measures obtained using mathematical morphology for a 3-D surface such as H, in similarity to the measures obtained in Chapter 4 for the 2-D curve given by the graph of $\overline{k}_\sigma(\overline{u})$, may provide a useful solution.

Another avenue which may be worth exploring is in the area of *differential geometry for surfaces* [53]. For example, the topography or shape of the bottom of a valley may then be efficiently characterized by its local curvature measures (*e.g.*, Gaussian, mean and principals curvatures).

Besides modifying the way the snake senses the surface H on which it crawls, another issue is to consider more sophisticated filtering schemes for obtaining proper potential surfaces. We have expended some effort in obtaining families of potential surfaces by the use of the HDC technique. Yet, some of the filtering stages could be easily refined. For example, instead of using the Sobel kernels in order to derive the GGHDC family, or, equivalently, the gradient maps of a multiscaled image, other filtering kernels or techniques may be more appropriate. However, we suspect that such an approach would likely only diminish the number of instances of failures, such as the ones discussed previously, but would still not provide for a completely reliable segmentation technique.

6.1.2 Shape Analysis

On the subject of shape, many experiments now need to be conducted. Specifically, the dynamics of shape deformation should be studied and we believe that the shape descriptors we have proposed in this thesis would be useful in this regard. For example, in Chapter 5 we proposed to examine the evolution of skeletons to characterize the process-history of a nonrigid shape deforming in a plane. An interesting avenue to explore might be one of producing a *shape-history-space* indicative of, for example, the survival in time of shape subparts, skeleton branches, or contour curvature features. Consider skeleton branches surviving for a short period of time; these could be discarded from the process-history analysis using a shape-history-space. Such a dynamic representation for shape primitives shares strong similarities with a "scale-space." In the scale-space paradigm one tracks features along the scale axis to determine their significance. Instead, in the case of the shape-history-space, features are tracked along the time axis. Of course, the two should be combined into one composite representation.

Another issue that should be explored is the subject of how to efficiently recompute distance maps for objects or regions undergoing small localized deformations. At present, as soon as an object deforms, a complete distance map must be recomputed. It would be much more advantageous to relabel, if possible, only those object pixels in the vicinity of a shape deformation, rather than all object pixels.

Finally, as pointed out in Chapter 4, when processing an object's contour to retrieve some approximation of its discrete curvature function, the amount of smoothing required needs to be specified in some strict way. We have not addressed this important issue in this

thesis. What we think is needed though is some method that fixes limits on the required amount of smoothing based on the nature of the data (*e.g.*, image resolution and object perimeter). A minimum level of smoothing, for partial noise removal, might be fixed based on the nature of the discrete contour orientation generation method, such as the chain code. A maximum level might be fixed by deciding how much averaging should be permitted in order to avoid the merging of neighboring peaks of the curvature function.

6.1.3 Bottom-Up and Top-Down Approaches

As was our original intention, we have considered in this thesis only a *bottom-up* approach to the computer vision problems we were faced with. That is, we have focused our attention on only the essential first processing steps of image segmentation and shape description, in that order. Although we have suggested that, for example, information about shape could be used in a "feedback loop" to help the segmentation processes, we have not yet explored this other avenue.

The need for relevant *domain knowledge* about our specific task, cell tracking, may also become of interest, as the problems we are faced with, such as the specific failures of the snake model, become more precise and well understood.

Appendix A

Morphological Operations for Functions

Let f be a discrete function representing a signal, such as $\overline{k}_\sigma(\overline{u})$, the curvature along a contour. The graph of this function is defined as a set of points $(\overline{u}, f(\overline{u}))$. Let B be a (2-D) structural element indexed by a single parameter i. The superscript c in f^c stands for the complement of f such that $f^c + f = constant$; Ⓑ stands for the reflection of B, that is Ⓑ $= \{i : -i \in B\}$.

A.1 The Four Principal Operations

A.1.1 Erosion

An erosion is computed by taking the minimum of a set of differences. Its form is similar to convolution, with the summation of convolution replaced by the minimum operation and the product replaced by a subtraction operation. The erosion of f by B is defined as follows:

$$E_{fB} \equiv f \ominus B = \min_i \left[f(\overline{u} - i) - \text{Ⓑ}(i) \right] . \tag{A.1}$$

A.1.2 Dilation

Dilation can be performed by taking the maximum of a set of sums. Its complexity is the same as erosion and is related to correlation, where instead of doing summation of products, a maximum of sums is computed. Using the same notation as for erosion, the dilation of f by B is defined as:

$$D_{fB} \equiv f \oplus B = \max_i \left[f(\overline{u} - i) + B(i) \right] . \tag{A.2}$$

A.1.2.1 Significance of the Erosion and Dilation Operations

Erosion and dilation are dual operations but are not, in general, the inverse operation of each other. This is expressed below:

$$\begin{aligned} f \oplus B \ominus B &= f_1 \neq f\,, \\ f \ominus B \oplus B &= f_2 \neq f\,. \end{aligned}$$

Furthermore, in general

$$f_1 \neq f_2\,.$$

This is true whenever the function f is not too smooth or regular. A dilation (erosion) will remove from the top (underneath) of f all those details that are smaller than the structural element B which is of fixed size. The result is a new function f_{new} which is smoother than the original function f.

When the protuberances of f are not covered by the structural element B, then the combination of an erosion and a dilation becomes a reversible operation (*i.e.*, $f_1 = f_2 = f$). Therefore, a combination of an erosion and a dilation can give some information about the regularity of a function. Combinations of this type give rise to two new operations, which are defined next.

A.1.3 Opening

Opening is defined as the dilation of an eroded function. It is given by the following relation:

$$f_B \equiv (f \circ B)_i \equiv (f \ominus B) \oplus B = \sup\{\inf[f(\overline{u}) : \overline{u} \in B_j] : j \in \textcircled{B}_i\}\ . \tag{A.3}$$

At the point i, $(f \circ B)_{(i)}$ has the highest value of the infinima of f taken over all the B's containing i . This means that the opening of f is a new function defined by the highest points reached by any part of the 2-D (reflected) structural element as it slides *under* the whole extent of f [135].

A.1.4 Closing

Closing is defined by the following relation:

$$f^B \equiv (f \bullet B)_i \equiv (f \oplus B) \ominus B = \inf\{\sup[f(\overline{u}) : \overline{u} \in B_j] : j \in B_i\}\ . \tag{A.4}$$

The closing of a function can be interpreted as a new function defined by the lowest points reached by any part of the 2-D structural element as it slides *over* f [135].

A.1.5 Properties of the Four Operations

The principal properties of the operations discussed above are briefly discussed in this section. These properties are important in order to understand the effects of erosion, dilation, opening and closing on functions.

Increasing: Let f be a function smaller than another function F, that is $f(\overline{u}) \leq F(\overline{u})$, for all $\overline{u}$. Then:

$$f \ominus B \leq F \ominus B\,, \tag{A.5}$$
$$f \oplus B \leq F \oplus B\,, \tag{A.6}$$
$$f \circ B \leq F \circ B\,, \tag{A.7}$$
$$f \bullet B \leq F \bullet B\,. \tag{A.8}$$

The size of a function implies the size of the eroded or dilated function for a given structural element B. Erosion and dilation are said to be increasing operations. Since erosion and dilation are increasing operations, opening and closing are also increasing.

Expansivity:

$$f \ominus B \leq f\,, \tag{A.9}$$
$$f \oplus B \geq f\,, \tag{A.10}$$
$$f \circ B \leq f\,, \tag{A.11}$$
$$f \bullet B \geq f\,. \tag{A.12}$$

Erosion is antiexpansive, while dilation is expansive. The terms “erosion” and “dilation” have their origins in this property of expansivity. Opening is antiexpansive, as is its first constituent operation, erosion. Closing is expansive, as is dilation.

Duality:

$$f \oplus B = (f^c \ominus \check{B})^c\,, \tag{A.13}$$
$$f \ominus B = (f^c \oplus \check{B})^c\,, \tag{A.14}$$
$$f \circ B = (f^c \bullet \check{B})^c\,, \tag{A.15}$$
$$f \bullet B = (f^c \circ \check{B})^c\,. \tag{A.16}$$

Dilation is the dual of erosion. Thus dilation (erosion) is the erosion (dilation) of the complemented function. Opening and closing are also dual operations. The closing of f corresponds to the opening of the complemented function f^c.

Chain Rule:

$$(f \oplus B_1) \oplus B_2 = f \oplus (B_1 \oplus B_2) \,, \tag{A.17}$$
$$(f \ominus B_1) \ominus B_2 = f \ominus (B_1 \oplus B_2) \,. \tag{A.18}$$

This implies that the erosion or dilation of a function f by a wide or complex structural element B ($\equiv B_1 \oplus B_2$) can be performed using the basic components of B, namely B_1 and B_2. An image processor[1] can then be built using only basic components to perform any kind of dilation or erosion.

Idempotency:

$$(f \circ B) \circ B = f \circ B \,, \tag{A.19}$$
$$(f \bullet B) \bullet B = f \bullet B \,. \tag{A.20}$$

Opening and closing operations are performed only once for a specific structural element. Their is no equivalence to the chain rules of erosion and dilation. As discussed previously in section A.1.2.1, opening and closing are two ways to compute a "smooth" approximation of a function f by removing details of a given size. Applying these operations again with the same structural element B does not further modify the filtered function.

A.1.6 Significance of the Morphological Operations

Unlike linear transformations of functions, morphological operations are characterized by their non-invertibility. They remove information of greater and greater extent as the size of the structural element increases.

Signal processing through iterative morphological transformations can therefore be conceived as a process of selective information removal where irrelevant details are irrecoverably destroyed, thereby enhancing the contrast of essential function features [138].

A.1.7 Application to Curvature: Flat Structural Element

For the morphological analysis of a signal such as curvature where we are interested in the extraction of peaks and constant regions, flat structural elements (1-D) are used. Eroding

[1] Image processors based on mathematical morphology operations have been successfully applied to binary images since the early 1970's. For example, the "Texture Analyzer" [80] was the first system of its kind in industry. Its major advantage was its ability to extract quantitative measures of an image in real time. Its major flaw was its limitation to binary image analysis.

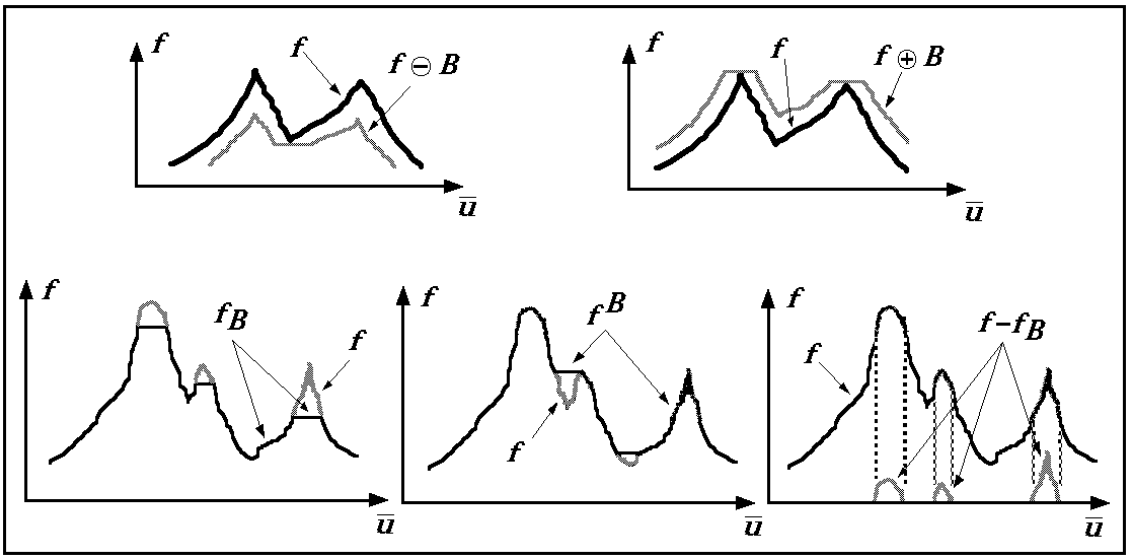


Figure A.1: Examples of the four morphological operations and of the top-hat transform (adapted from [135]).

a function by a segment of length R is equivalent to replacing the function values at every point by the minimum of all the points in a neighborhood of size R. Likewise, dilating a function by a segment of length R is equivalent to a maximum transformation over a neighborhood of size R. Openings and closings by flat structural elements maintain the vertical boundaries of the function they transform.

Hat-transforms, so named because they can be visualized as a covering of peaks with a hat of fixed size, can be used to extract sharp peaks and ridges. The residual $f - f_B$ (where f_B is the opened version of f as defined in equation (A.3)) is known as the *top-hat transform* and presents the possibility of extracting peaks. Its complement, the residual $f - f^B$ (where f^B is the closed version of f as defined in equation (A.4)), we call by analogy the *bottom-hat transform* and provides a way to extract valleys. Examples of the four morphological operations and of the residual $f - f_B$ with a flat structural element, are given in Figure A.1.

Morphological analysis of the residuals $f - f_B$ and $f - f^B$, and of the filtered signals f_B and f^B, can then be performed to extract and classify peaks, valleys and the resultant flattened regions.

A.1.8 Issues of Computational Complexity

Morphological operations for a function such as $\overline{k}_\sigma(\overline{u})$, using a flat structural element B are simple and easy to implement as *min–max* operations. Erosion is equivalent to taking the

minimum of $\overline{k}_\sigma(\overline{u})$ over the neighborhood defined by the width of B ($= R$), while dilation is equivalent to taking the maximum of $\overline{k}_\sigma(\overline{u})$ over the same neighborhood (section A.1).

Both erosion and dilation can be implemented as iterative processes by using the *chain rule* property (equations (A.17) and (A.18)). This property implies that an erosion or dilation with a relatively wide structural element (*e.g.*, 5 pixels wide) can be performed sequentially with smaller structural elements (*e.g.*, two structural elements 3 pixels wide), giving a more efficient way to perform *min–max* operations. However, the chain rule property does not apply directly to the opening and closing operations. These two are said to be *idempotent* (equations (A.19) and (A.20)). That is, their application to a given function with the same structural element does not further modify the filtered function. Thus we can only take advantage of the chain rule property for the first operation of an opening or a closing, that is, an erosion or a dilation, respectively. Using the latter for the first constituent operation of the closing and opening operation, a hierarchical-like scheme[2] can be implemented. For example, consider the hierarchical implementation of an opening operation. Let the basis of the hierarchy be the positive curvature function $\overline{k}_{\sigma+}(\overline{u})$. Subsequent levels are built up of eroded and opened versions of $\overline{k}_{\sigma+}(\overline{u})$: $\overline{k}_{\sigma+erod}(\overline{u}, l)$ and $\overline{k}_{\sigma+open}(\overline{u}, l)$, where l corresponds to the hierarchical level and $\overline{k}_{\sigma+open}(\overline{u}, l)$ is obtained by dilating $\overline{k}_{\sigma+erod}(\overline{u}, l)$. Therefore, an eroded version of $\overline{k}_{\sigma+}(\overline{u})$ at a particular level in the hierarchy, that is, $\overline{k}_{\sigma+erod}(\overline{u}, l)$, can be obtained efficiently (*i.e.*, with a small structural element) by eroding an eroded version of $\overline{k}_{\sigma+}(\overline{u})$ at a preceding level, that is, by the erosion of $\overline{k}_{\sigma+erod}(\overline{u}, l-1)$. Such a hierarchical scheme provides an efficient way of building the MCS, where each level in the hierarchy corresponds to an increasing $\delta_{\overline{u}}$.

Morphological measures are also simple to implement. Furthermore, due to the uniformity of MCS features, morphological measures are more rapidly computed for existing peaks, that is, for peaks that were detected in a preceding level in the hierarchy or at a smaller scale $\delta_{\overline{u}}$, since their localization is already known. Finally, this property of uniformity permits simple and easy interpretation, a major advantage in terms of computations over traditional scale-space approaches.

In summary, three aspects of curvature morphology lead to low computational complexity. Firstly, only simple computations for both morphological operations and measures are necessary. Secondly a hierarchical implementation using the chain rule property is easily obtained. Finally, the uniformity of the MCS and its simplicity of interpretation yields low computational complexity for the curvature morphology representation.

[2] Such a hierarchical representation of morphological operations shares many similarities with the "Hierarchical Discrete Correlation" representation (Chapter 3)).

Appendix B

Inserting Discontinuities Along the Snake

In this appendix, we make use of the notion of *computational molecules* discussed by Terzopoulos [141, 146] which give an explicit visualization of the links that exist between snaxels and which are useful to understand the effect of inserting discontinuities along the snake by removing some of these links.

B.1 Computational Molecules for the Snake

Let us reconsider the discretization of the stiffness constraints in the equations of motion of the snake. Assuming a constant step-size $\Delta\overline{s} = 1$, equation (2.18) is rewritten as follows:

$$\begin{aligned} -\frac{\partial}{\partial s}\Big(\omega_1(s)v_s(s,t)\Big) + \frac{\partial^2}{\partial s^2}\Big(\omega_2(s)v_{ss}(s,t)\Big) \;\approx\; & \\ \omega_1(\overline{s})\,[\overline{v}(\overline{s},\overline{t}) - \overline{v}(\overline{s}-1,\overline{t})] + & \\ \omega_1(\overline{s}+1)\,[\overline{v}(\overline{s},\overline{t}) - \overline{v}(\overline{s}+1,\overline{t})] + & \\ \omega_2(\overline{s}-1)\,[\overline{v}(\overline{s}-2,\overline{t}) - 2\overline{v}(\overline{s}-1,\overline{t}) + \overline{v}(\overline{s},\overline{t})] - & \\ 2\omega_2(\overline{s})\,[\overline{v}(\overline{s}-1,\overline{t}) - 2\overline{v}(\overline{s}\,\overline{t}) + \overline{v}(\overline{s}+1,\overline{t})] + & \\ \omega_2(\overline{s}+1)\,[\overline{v}(\overline{s},\overline{t}) - 2\overline{v}(\overline{s}+1,\overline{t}) + \overline{v}(\overline{s}+2,\overline{t})] \quad . & \end{aligned} \tag{B.1}$$

This equation can be understood as a *nodal equation* defining the elasticity links that exist at each snaxel or node, $\overline{s}$, of the snake. The terms on the RHS of equation (B.1) can then be represented graphically as a set of computational molecules (Figures B.1 and B.2). These molecules are made of atoms, indicated by ellipses, representing the coefficients at each nodal position, $\overline{v}(\overline{s},\overline{t})$, in equation (B.1). Between these atoms are links which represent the adjacency relations between them. Each molecule is positioned with respect to

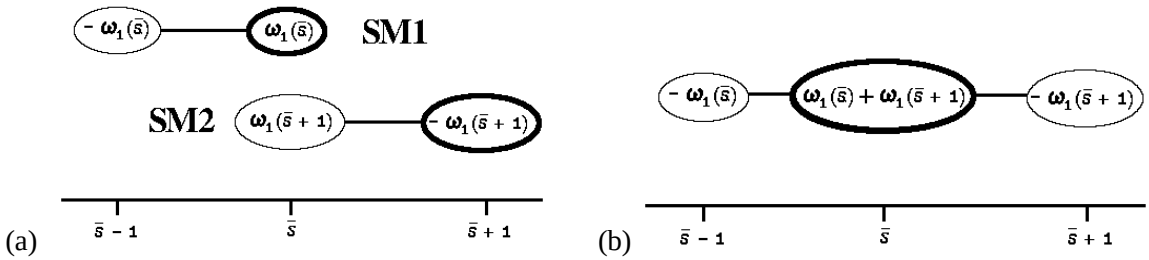


Figure B.1: String molecules for the snake model. In (a) are shown the two string molecules SM1 and SM2 defining the tension links. In (b) is shown the composite string molecule centered at a snaxel $\overline{s}$ obtained by the molecular summation of the molecules in (a).

a particular atom indicated by a dark contoured ellipse. This atom corresponds to the snaxel at which each term of equation (B.1) is defined. Each snaxel possesses such a "molecular" description of its local interaction with its neighbors. Figure B.1.(a) shows the two *string molecules*, SM1 and SM2, representing tension links between snaxels (*i.e.*, first two terms on the RHS of equation (B.1). Figure B.2.(a) shows the three *rod molecules*, RM1, RM2 and RM3, representing the rigidity links between snaxels (*i.e.*, last three terms on the RHS of equation (B.1).

The formation of nodal equations at any snaxel $\overline{s}$ can be obtained by *molecular summation*, where the primitive molecules combine centered at this snaxel $\overline{s}$. Coincident atoms sum together their coefficients. In Figure B.1.(b) is shown the *composite string molecule* obtained by molecular summation of the primitive string molecules SM1 and SM2. This illustrates the fact that tension constraints on each snaxel are symmetrically distributed around them and are only influenced by the two direct neighbors. In Figure B.2.(b) is shown the *composite rod molecule* obtained by molecular summation of the primitive rod molecules RM1, RM2 and RM3. This illustrates the fact that rigidity constraints at each snaxel are also symmetrically distributed around them, but this time on a larger neighborhood made up of the four direct neighbors. By molecular summation of both the composite string molecule and the composite rod molecule, the nodal equation (B.1) is recovered (Figure B.3). Such a composite molecule exists at every snaxel or node along the snake.

Computational molecules help us visualize the interactions between snaxels and how the stiffness weights, or atom coefficients, are distributed at each snaxel. The computational molecules are also useful for visualizing and implementing the introduction of discontinuities along the snake. Discontinuities which are breaks in the links or constraints between snaxels require *molecular inhibition* rather than molecular summation. Wherever a link is broken or removed, the molecular summation of the corresponding primitive molecule is inhibited. Such breaks correspond to setting the concerned tension, $\omega_1(\overline{s})$, and/or rigidity, $\omega_2(\overline{s})$, constraints to zero. In the two following sections, position and tangent discontinuities are defined using molecular inhibition in order to obtain the modified nodal equations.

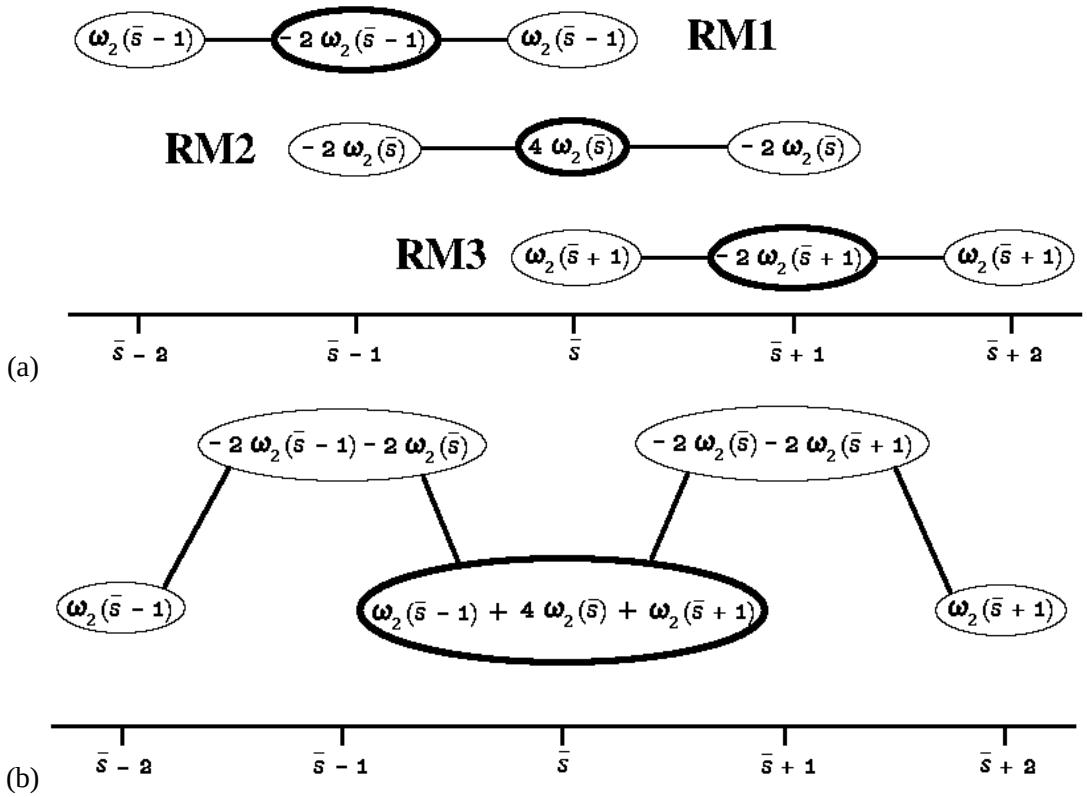


Figure B.2: Rod molecules for the snake model. In (a) are shown the three rod molecules RM1, RM2 and RM3 defining the rigidity links. In (b) is shown the composite rod molecule defined at a snaxel $\bar{s}$ obtained by the molecular summation of the molecules in (a).

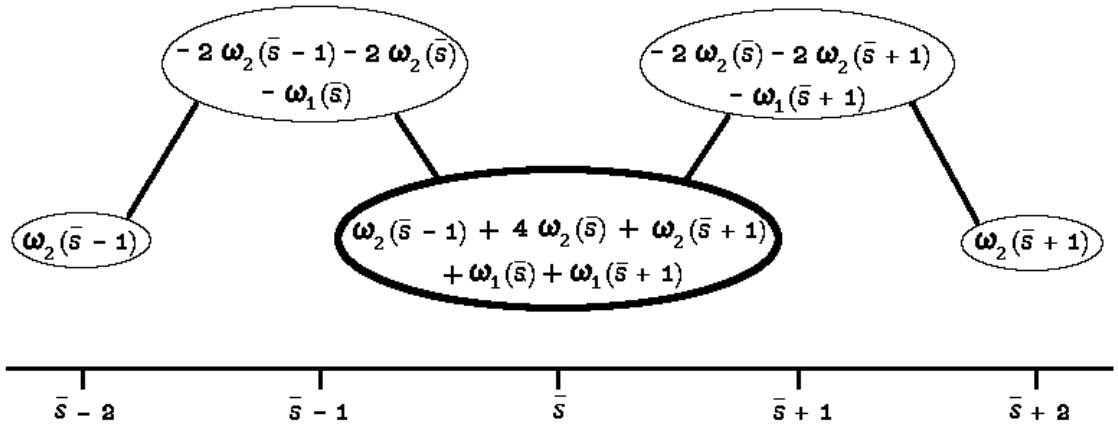


Figure B.3: The nodal equation obtained by molecular summation of both the composite string molecule and the composite rod molecule.

B.1.1 Introducing Position Discontinuities

A position discontinuity between two snaxels ($\overline{s} = i$ and $\overline{s} = i + 1$) consists of a complete break of all the links that relate these snaxels, that is, the snake is cut. Therefore, both the tension and rigidity constraints or molecules existing between these two snaxels must be inhibited when constructing the nodal equations. Since each snaxel has a nodal equation defined on its four direct neighbors, four snaxels, at nodes $\overline{s} = i - 1,\ i,\ i + 1$ and $i + 2$, share some links between nodes i and $i+1$. At node $\overline{s} = i-1$, one rod molecule, RM3, has to be dropped (Figure B.4.(a)). At node $\overline{s} = i$, two rod molecules, RM2 and RM3, and one string molecule, SM2, must be inhibited (Figure B.4.(b)). Similarly, due to the symmetry of the nodal equation, at node $\overline{s} = i+1$, two rod molecules, RM1 and RM2, and one string molecule, SM1, must be inhibited (Figure B.4.(d)). Finally, at node $\overline{s} = i + 2$ one rod molecule, RM1, has to be inhibited (Figure B.4.(c)).

The corresponding nodal equation is modified accordingly for each of the four molecular inhibitions. These modified nodal equations give us the entries of the stiffness matrix K that have to be modified. At node $i - 1$ the following three entries are modified:

$$\begin{aligned} c_{i-1} &= \omega_2(i-2) + 4\omega_2(i-1) + \omega_1(i-1) + \omega_1(i)\,, \\ b_{i-1} &= -2\omega_2(i-1) - \omega_1(i)\,, \\ a_{i-1} &= 0\,. \end{aligned} \tag{B.2}$$

At node i, four entries are modified, one of which was already mentioned previously (b_{i-1}), leading to the following three new entries to be modified:

$$\begin{aligned} c_i &= \omega_2(i-1)\,, \\ b_i &= 0\,, \\ a_i &= 0\,. \end{aligned} \tag{B.3}$$

Similarly, at node $i + 1$, four entries are modified. Two of these were already mentioned previously (b_i and a_{i-1}), leading to the following two new entries to be modified:

$$\begin{aligned} c_{i+1} &= \omega_2(i+2) + \omega_1(i+2)\,, \\ b_{i+1} &= -2\omega_2(i+2) - \omega_1(i+2)\,. \end{aligned} \tag{B.4}$$

Finally, at node $i + 2$, three entries are modified. Two of these were already mentioned previously (b_{i+1} and a_i), leading to the following new entry to be modified:

$$c_{i+2} = 4\omega_2(i+2) + \omega_2(i+3) + \omega_1(i+2) + \omega_1(i+3)\,. \tag{B.5}$$

In total, for any position discontinuity, nine entries of matrix K must be modified.

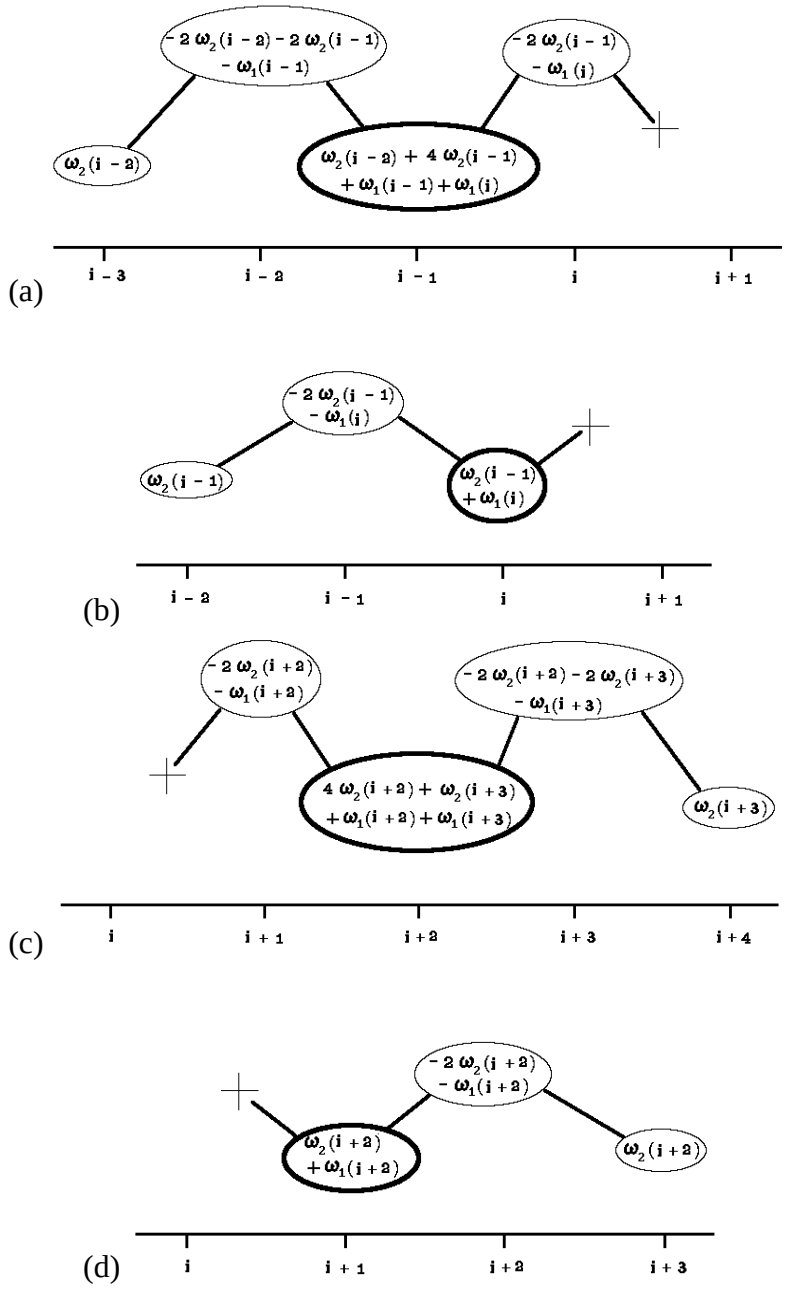


Figure B.4: Introducing position discontinuities by the molecular inhibition of rod and string molecules. A position discontinuity is enforced between two snaxels $\overline{s} = i$ and $\overline{s} = i + 1$. In correspondence, four snaxels ($\overline{s} = i - 1,\ i,\ i + 1$ and $i + 2$) have their nodal equation modified, that is, the link they possess between nodes $\overline{s} = i$ and $\overline{s} = i + 1$ is cut (indicated by a cross). In (a), (b), (d) and (c) are shown the modified nodal equations for snaxels $\overline{s} = i - 1,\ i,\ i + 1$ and $i + 2$, respectively.

B.1.2 Introducing Tangent Discontinuities

A tangent discontinuity at a snaxel $\overline{s} = i$ consists of a break of all rigidity constraints ($\omega_2 = 0$) at that snaxel or node. Only the tension links at that node are kept intact. Therefore, all the rod molecules that are centered at snaxel i must be inhibited in the molecular summation used to derive all nodal equations. Since a rod molecule is defined on a neighborhood made of three atoms (Figure B.2), three nodal equations, at nodes $\overline{s} = i-1,\ i$ and $i+1$, have to be modified. At node $\overline{s} = i-1$, one rod molecule, RM3, is dropped (same result as illustrated by Figure B.4.(a)). Similarly, at node $\overline{s} = i$, one rod molecule, RM2, is dropped (Figure B.5.(a)). Finally, at node $\overline{s} = i+1$, one rod molecule, RM1, is dropped (Figure B.5.(b)).

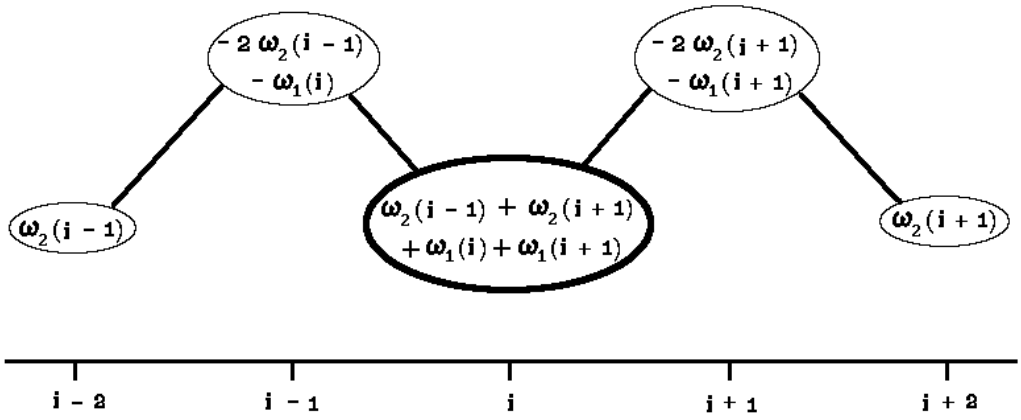
The corresponding nodal equation is modified accordingly for each of the three molecular inhibitions. As in the case of position discontinuities, these modified nodal equations give us the entries of the stiffness matrix K that must be changed. At node $i-1$ the same three entries ($c_{i-1},\ b_{i-1}$ and a_{i-1}) given in equation (B.2), for the case of a position discontinuity, are modified. This is because the same molecule, RM3, is inhibited from the molecular summation. At node i, three entries are modified, one of which was already mentioned previously (b_{i-1}, equation (B.2)), leading to the following two new entries to be modified:

$$\begin{aligned} c_i &= \omega_2(i-1) + \omega_2(i+1) + \omega_1(i) + \omega_1(i+1)\,, \\ b_i &= -2\omega_2(i+1) - \omega_1(i+1)\,. \end{aligned} \tag{B.6}$$

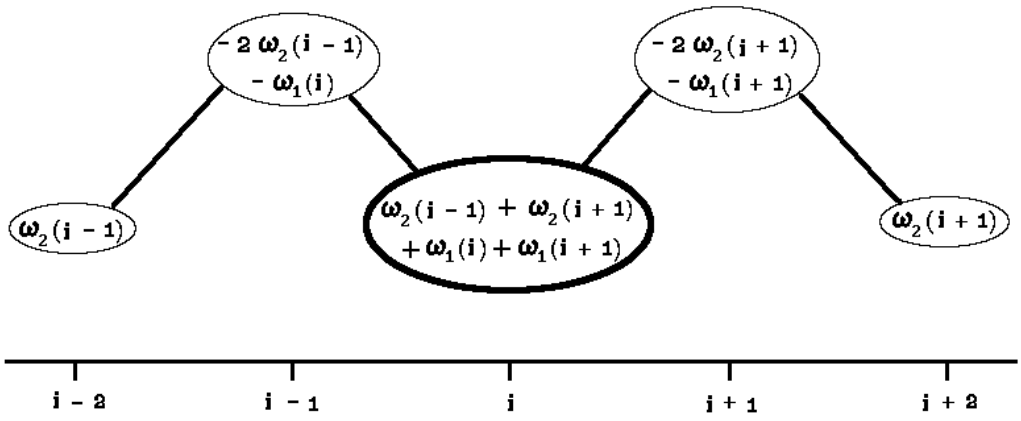
Finally, at node $i+1$, three entries are modified. Two of these entries were already mentioned previously (a_{i-1}, equation (B.2), and b_i, above), leading to the following new entry to be modified:

$$c_{i+1} = 4\omega_2(i+1) + \omega_2(i+2) + \omega_1(i+1) + \omega_1(i+2)\,. \tag{B.7}$$

In total, for any tangent discontinuity, six entries of matrix K have to be modified.



(a)



(b)

Figure B.5: Introducing tangent discontinuities by molecular inhibition of rod molecules. A tangent discontinuity is enforced at a snaxel $\overline{s} = i$. In correspondence, three snaxels ($\overline{s} = i - 1,\ i$ and $i + 1$) have their nodal equation modified, that is, the rigidity link they possess which is centered at $\overline{s} = i$ is inhibited from molecular summation. In (a) and (b) are shown the modified nodal equations for snaxels $\overline{s} = i$ and $i + 1$, respectively. The modified nodal equation for snaxel $\overline{s} = i - 1$ correspond to the case already illustrated by Figure B.4.(a).

Appendix C

Imposing Limits on the Snake Forces Amplitudes

In this appendix, we give a number of heuristics to impose limits on the amplitude of snake forces. Such limits can be fixed with respect to variations in position. The goal of such a "normalization" scheme is to keep these forces tractable so that the snake remains as stable as possible.

C.1 Maximum Elastic and Inertial Constraints

Consider equation (2.28) where both the external and the field forces are set to zero, that is, $E_{ext_{\overline{x}}} = E_{field_{\overline{x}}} = 0 = \mathcal{F}_{pot_{\overline{x}}}$. This is done to examine the influence of the stiffness and memory constraints, $\mathcal{F}_{stiff}$ and $\mathcal{F}_{memo}$, in obtaining a new snaxel coordinate (here $\overline{x}(i,\overline{t})$). Equation (2.28) becomes:

$$\overline{x}(i,\overline{t}) = \frac{1}{c_i^*}\left\{\mathcal{F}_{memo}\left(\overline{x}(i,\overline{t}-1,\overline{t}-2)\right) + \mathcal{F}_{stiff}\left(\overline{x}(i-2,i-1,i+1,i+2,\overline{t})\right)\right\} . \quad \text{(C.1)}$$

Let us first consider the effect of stiffness. The amplitude of $\mathcal{F}_{stiff}$ is a linear function of the weights $\omega_1(\overline{s})$ and $\omega_2(\overline{s})$ and of the spacing between the snaxels. Since we have fixed $\omega_2(\overline{s})$ to be a small positive and constant value, the maximum amplitude of $\mathcal{F}_{stiff}$ will be essentially dependent on the maximum possible amplitude of $\omega_1(\overline{s})$. $\omega_1(\overline{s})$ itself is a function of the spacing between snaxels (equation (2.29)) and its maximum value is given by τ_{max}. Therefore, what we need to do is to impose a maximal possible value on τ_{max}. Because $\omega_1(\overline{s})$ is used as a multiplying weight for the spacing between snaxels (equation (2.18)), the maximum value for τ_{max} can be set to an expected or desired maximal spacing. We have use experimentally a value of five for τ_{max}.

The effect of memory is imposed by $\mathcal{F}_{memo}$. Its amplitude is a linear function of the constants μ and γ, and of the motion, or spacing in time, of a given snaxel. Furthermore, μ and γ are used as multiplying weights for the spacing in time of a given snaxel (equation (2.16)). Therefore, the maximum values for μ and γ can be set in the order of magnitude of the expected maximal spacing in time of any snaxel. Also, we can fix them so that their combination gives a desired effect of damping. We have use experimentally a value of one for both μ and γ.

C.2 Normalization of the Field Constraints

In sections 2.6.3 and 2.7 it has been explained why and how the directional slopes in $\overline{x}$ and $\overline{y}$ of the potential surface, that is, $E_{field_{\overline{x}}}$ and $E_{field_{\overline{y}}}$, had to be redistributed on some acceptable interval. In this section, it is shown how one can normalize the field constraints $E_{field_{\overline{x}}}$ using the "clipped" slope function of section 2.7 so that the snake never undergoes unrealistic motion (similar results are derived, by symmetry, for $E_{field_{\overline{y}}}$).

In the following paragraphs we assume that the external constraints are inactive, that is, $E_{ext} = 0$. Therefore $\mathcal{F}_{pot_{\overline{x}}}$ may be rewritten as follows:

$$\begin{aligned} \mathcal{F}_{pot_{\overline{x}}} &= -\frac{1}{2}E_{field_{\overline{x}}}(i,\overline{t}-1) = -1\frac{1}{2}\mu\mathcal{G}h_{\overline{x}}\left[\overline{x}(i,\overline{t}-1),\overline{y}(i,\overline{t}-1)\right] \\ &= -\alpha h_{\overline{x}}\left[\overline{x}(i,\overline{t}-1),\overline{y}(i,\overline{t}-1)\right] , \end{aligned}$$

where we make use of equation (2.4) and where $\alpha = \frac{\mu\mathcal{G}}{2}$. Then, following the discussion of section C.1, equation (2.28) may be rewritten as follows:

$$\overline{x}(i,\overline{t}) \approx \overline{x}^*(i,\overline{t}) + \left(-\frac{\alpha}{c_i^*}\right) h_{\overline{x}} = \overline{x}^*(i,\overline{t}) + \Delta\overline{x}_{field} , \tag{C.2}$$

where $\overline{x}^*(i,\overline{t})$ is the new snaxel coordinate obtained from stiffness and memory constraints and where $\Delta x_{field} = \left(\frac{-\alpha}{c_i{}^*}\right) h_{\overline{x}}$. Therefore the influence of field constraints on a snaxel can be understood as a displacement contribution $\Delta\overline{x}_{field}$ (in pixel units) This displacement contribution may be normalized with respect to a maximal permitted displacement D_1 (also in pixel units). With respect to clipped versions of the slope function of the potential surface H and of the tension function τ, the maximal possible displacement contribution of field constraints ($\Delta\overline{x}_{field_{max}}$) is given by:

$$|\Delta\overline{x}_{field_{\max}}| = \left(\frac{\alpha}{c_{i\,\min}^*}\right) |\mathcal{S}_{max}| \leq D_1 , \tag{C.3}$$

where $\mathcal{S}_{max}$ is the maximal permitted slope value of the clipped slope function of section 2.7 and where ${c_i}^*{}_{min} = 6\omega_2 - 2\tau_{max} + \frac{\gamma}{2} + \mu$. With given constants ω_2, γ, μ, $\mathcal{S}_{max}$, τ_{max} and D_1,

the magnitude of the gravitational acceleration, $\mathcal{G}$, can be fixed to ensure that any snaxel displacement due to a field force will be limited in amplitude to a desirable value D_1.

C.3 Normalization of the External Constraints

External constraints are of two complementary types: volcanos (repulsive forces) and springs (attractive forces). Since, as proposed in section 2.5, volcano forces are best implemented by incorporating them within the potential surface, they do not have to be considered here. Spring forces, on the other hand, have to be normalized separately from the field forces. With no loss of generality, we may assume that the field constraints and the volcano forces are inactive, that is, $E_{field} = E_{volc} = 0$. Therefore $\mathcal{F}_{pot_{\overline{x}}}$ may be rewritten as follows (similar results are derived, by symmetry, for $\mathcal{F}_{pot_{\overline{y}}}$):

$$\mathcal{F}_{pot_{\overline{x}}} = -\frac{1}{2} E_{spring_{\overline{x}}}(i, \overline{t} - 1) = -\frac{1}{2} f_{spring} \mid_{\overline{x}} = \frac{1}{2} k_{spring}(\overline{x}_1 - \overline{x}_2)\,,$$

where we make use of the notation for spring constraints of section 2.2. Again, following the discussion of section C.1, equation (2.28) may be rewritten as follows:

$$\overline{x}(i, \overline{t}) \approx \overline{x}^*(i, \overline{t}) + \left(\frac{k_{spring}}{c_i^*}\right)(\overline{x}_1 - \overline{x}_2) = \overline{x}^*(i, \overline{t}) + \Delta\overline{x}_{spring}\,, \tag{C.4}$$

where $\overline{x}^*(i, \overline{t})$ is the new snaxel coordinate obtained from stiffness and memory constraints and where $\Delta\overline{x}_{spring} = \left(\frac{k_{spring}}{c_i^*}\right)(\overline{x}_1 - \overline{x}_2)$. Again, the influence of the spring constraint on a snaxel can be understood as a displacement contribution $\Delta\overline{x}_{spring}$ (in pixel units). This displacement contribution may be normalized with respect to a maximal permitted displacement D_2 (also in pixel units). In order to limit the possible amplitude of the distance "snaxel – spring fixation point", $\overline{x}_1 - \overline{x}_2$, a clipped version of f_{spring} should be used. Let us assume a saturation value $\Delta\overline{x}_{1,2_{max}}$ for this purpose. Then, the maximal possible displacement contribution of a spring constraint, $\Delta\overline{x}_{spring_{max}}$, is given by:

$$|\Delta\overline{x}_{spring_{\max}}| = \left(\frac{k_{spring}}{c_{i\ \min}^*}\right) |\Delta\overline{x}_{1,2_{\max}}| \leq D_2\,. \tag{C.5}$$

With given constants ${c_i}^*{}_{min}$, $\Delta\overline{x}_{1,2_{max}}$ and D_2, the spring parameter k_{spring} can be fixed to ensure that any snaxel displacement due to a spring force will be limited in amplitude to a desirable value D_2.

Appendix D

Euclidean Distance Mapping in the Discrete Domain

A distance mapping on a binary image ($I : I(\overline{x},\overline{y}) = 0$ iff $I(\overline{x},\overline{y}) \in O'$; $I(\overline{x},\overline{y}) = 1$ iff $I(\overline{x},\overline{y}) \in O$)[1] is produced by a distance transform, DT. A DT consists of a minimum operation on a distance metric $dist(p,p')$, where $p \in O$ and $p' \in O'$ (equation (5.1)), applied over a given space or domain (*e.g.*, an image I). Different metrics, $dist$, commonly used in discrete image processing are:[2]

$$\begin{aligned} dist_{CB}(p,p') &= |\overline{x}_p - \overline{x}_{p'}| + \left|\overline{y}_p - \overline{y}_{p'}\right| , \\ dist_{Chess}(p,p') &= \max\left(|\overline{x}_p - \overline{x}_{p'}| \ , \ \left|\overline{y}_p - \overline{y}_{p'}\right|\right) , \\ dist_{Euclid}(p,p') &= \sqrt{(\overline{x}_p - \overline{x}_{p'})^2 + \left(\overline{y}_p - \overline{y}_{p'}\right)^2} , \end{aligned}$$

where $p = p(\overline{x}_p, \overline{y}_p) \in O$ and $p' = p'(\overline{x}_{p'}, \overline{y}_{p'}) \in O'$. The DT based on the metric $dist_{CB}$ is called the *city block* DT. The DT based on the metric $dist_{Chess}$ is called the *chessboard* DT. And the DT based on the metric $dist_{Euclid}$ is called the *Euclidean* DT or EDT.

Applying a DT on I corresponds to "propagating" distances over I [131]. Points in the background ($p' \in O'$) are seen as sources from which distance values are propagated as waves over the complete image, I; for example, the propagation of circular waves in the case of an Euclidean metric. The first time an object point ($p \in O$) is reached by a given wave it is assigned a new label, that is, a minimum distance value from O'. Obviously such a wave propagation scheme is easily implemented in parallel (see for example [159]). But, it then becomes an iterative procedure, where iterations correspond to distances from the

[1] Notation. O : object or figure, O' : non-object or background. $\{\overline{x} \in \overline{X} \ : \ \overline{X} = \{0,\ldots,M_{\overline{X}}\}\}$ and $\{\overline{y} \in \overline{Y} \ : \ \overline{Y} = \{0,\ldots,M_{\overline{Y}}\}\}$.

[2] For extensive surveys of metrics used in the discrete domain see the work of Borgefors [28, 29].

punctual sources, dependent on the maximal width of the object O. However, for such a simple propagation transform, a sequential implementation is computationally preferable (*i.e.*, independent of the object's width) and provides equivalent results [131]. Several sequential algorithms exist for computing a distance map on a discrete grid. Essentially two categories of algorithms are available depending on the type of DT performed: EDT's or Weighted DT's such as the city block and chessboard DT's (hereafter referred to as WDT's, following Borgefors notation [30]. Common to both kinds of distance mapping is the fact that they are based on information flow or propagation using small local neighborhoods (*e.g.*, a 3 pixels x 3 pixels squared window). WDT algorithms require two passes over an image to obtain the distance map, while EDT algorithms require three or four passes [127].

The optimal distance metric is the Euclidean one, that is, $dist_{Euclid}$. The other metrics give coarse approximations of a circular propagation, that is, they approximate an Euclidean propagation. For example, the city block metric gives a diamond-shape propagation, while the chessboard metric gives a square-shape propagation [132]. DT's based on non-Euclidean metrics (*i.e.*, WDT's) have a long history in picture processing. In most cases they were used because they were faster to apply than EDT's. This was true until Danielsson's algorithm was published in 1980 [46]. In fact, only the city block DT can be evaluated much faster than Danielsson's EDT.[3] This is because of its simpler spatial complexity, though the price paid is poor accuracy (see section D.1.1).

In the following sections we will consider different aspects of the implementation and application of Euclidean distance mapping to binary images. In section D.1 we give a detailed analysis of an optimized implementation of Danielsson's Euclidean Distance Transform (EDT) algorithm.[4] In section D.2 we define the notion of a distance surface and briefly mention some of its properties. Finally, in section D.3 we show how the snake model can take advantage of the distance surface when seen as a potential surface.

D.1 Euclidean Distance Transform

In this section we present an optimized version, in terms of numerical complexity, of the sequential algorithm for computing the EDT first introduced by Danielsson. We also compare the computational complexity of this optimized algorithm to non-Euclidean DT's. Following Danielsson's notation, we consider a mapping from a binary image $I(\overline{x}, \overline{y})$ (a double-valued function)[5] to a multivalued image $L(\overline{x}, \overline{y})$. In order to evaluate L, Daniels-

[3] This must be contrasted with the generally incorrect belief appearing in the literature (even recently) that most WDT's are less computationally expensive than EDT's (see for example [28, 29, 55, 56, 8, 12].

[4] Addendum (February 2003): A version of this section was published in 1992 as a CVGIP-IU journal article [96].

[5] Initially, before applying the DT, objects points of I (*i.e.*, pixels $p \in O$) are assigned a large integer value M (*i.e.*, $I(p) = M$) greater than the square of the maximum possible width of any binary object that

son proposes to compute a multivalued vector image $\mathbf{L}$, at which each pixel of the image is assigned a vector (a doublet) rather than a singleton (*e.g.*, a distance value) as follows:

$$\mathbf{L}(\overline{x}, \overline{y}) = (L_{\overline{x}}, L_{\overline{y}}) \,, \tag{D.1}$$

where $L_{\overline{x}}$ and $L_{\overline{y}}$ will contain the minimum integer distance values in the $\overline{X}$ and $\overline{Y}$ directions, respectively, from the background O'. L is then simply obtained as the norm $\|\mathbf{L}\|$ by computing the square root as follows:

$$L(\overline{x}, \overline{y}) = \sqrt{L_{\overline{x}}^2 + L_{\overline{y}}^2} = |\mathbf{L}| \,. \tag{D.2}$$

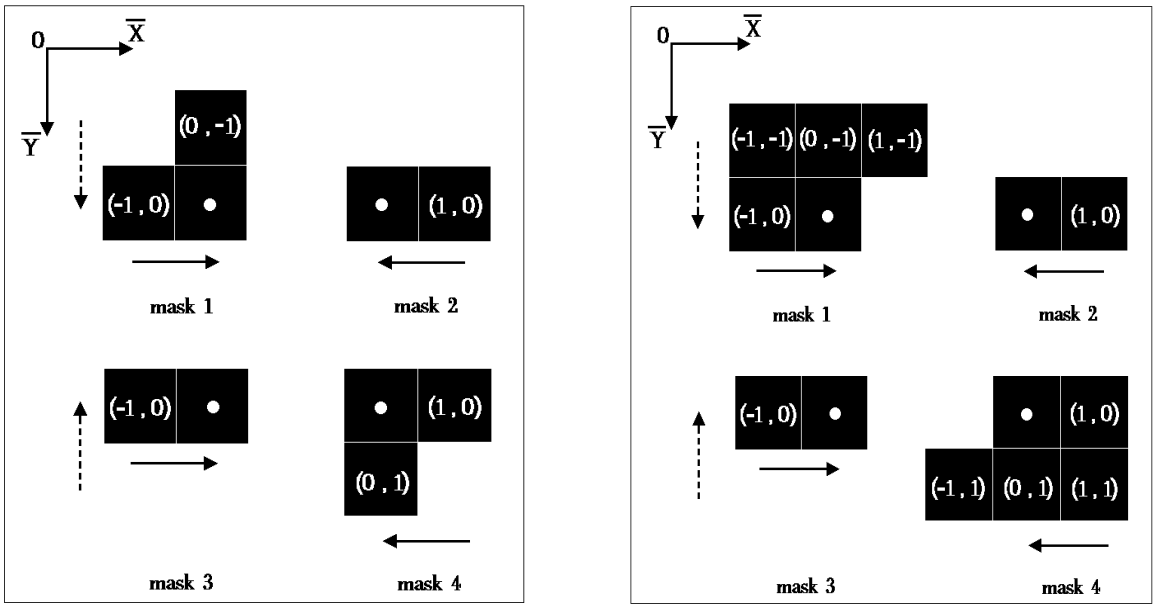
The propagation of distances over a complete image can be obtained with a four-pass algorithm (see [127] for a recent three-pass version). This is best visualized as a convolution of I with four masks (Figure D.1). Such a convolution is performed in two complete picture scans in opposite directions. Masks 1 and 2 are passed downwards over the image, while masks 3 and 4 are passed upwards (Figure D.1). Furthermore, each row is scanned in both horizontal directions to ensure an isotropic propagation of the distance values [127]. Depending on the required accuracy and numerical complexity, two versions of the EDT algorithm were described by Danielsson (see also [160]). The simpler one, named the $4SSEDT$, which stands for the "four-points Sequential Signed EDT," requires a visit to only the four direct neighbors (*i.e.*, horizontal and vertical neighbors) at each pixel $p \in O$ (Figure D.1.(a)), while the $8SSEDT$, which stands for the "eight-points Sequential Signed EDT," requires a visit to the 8 neighbors at each pixel p (Figure D.1.(b)). The $8SSEDT$ is numerically more complex than the $4SSEDT$, but it is also more accurate.

While scanning the image with four masks every object pixel, p, is updated by comparing it to some of its neighboring pixels (in a neighborhood defined by the masks of Figure D.1). Comparing a pixel p to its neighbors first implies seeking the neighbor, p_{min} (already visited at least once), having the minimum distance amplitude (incremented/decremented) and, second, assigning the pair $(L_x(p_{min}), L_y(p_{min}))$ to $\mathbf{L}(p)$ plus the increments or decrements in the $\overline{X}$ and $\overline{Y}$ directions given by the masks weights (indicating the relative position of p_{min} with respect to p). Therefore, the complete updating procedure generates the vector image $\mathbf{L}$. Seeking the minimum distance amplitude requires numerically expensive computations for every visited pixel $p \in O$.[6] Furthermore, floating point numbers, rather than integers, are then required. For example, if we wish to compare p with a neighbor p_{+x} in the positive $\overline{X}$ direction, then we must compare $L(p)$ with $L_{new}(p_{+x})$, where $L_{new}(p_{+x})$ is $L(p_{+x})$ incremented as follows:

$$L_{new}(p_{+x}) = \sqrt{\left(L_{\overline{x}}(p_{+x}) + 1\right)^2 + L_{\overline{y}}^2(p_{+x})} \,.$$

could exist in I (*e.g.*, $M = M_{\overline{x}}^2 + M_{\overline{y}}^2$).

[6] That is, two integer adds, two integer squares (multiplications), one floating square root and, possibly, two assignments and one or two increments/decrements.



(a) (b)

Figure D.1: The four masks used in the sequential signed EDT. Two sets of masks are shown. In (a) are shown the four masks for the $4SSEDT$. In (b) are shown the four masks for the $8SSEDT$. The masks are centered at pixel p (indicated by a white dot). The pairs of weights in each mask ($(\overline{x}, \overline{y})$ pairs) represent the distances from p in horizontal and vertical steps or increments. The reference frame ($\overline{X}$ and $\overline{Y}$ directions) is shown to be positioned in the top-left corner of an image, I. The arrows (next to the masks) indicate in which direction a given mask is passed over I. The convolution of these masks over I is performed in two picture scans (indicated by dashed arrows) in opposite directions (here downward for masks 1 and 2, and upward for masks 3 and 4). For each row of I the pairs of masks (1,2) and (3,4) are moved in opposite horizontal directions so that the propagation of distance values is isotropic (see text).

However, there is a much better way to perform these computations using integers only, getting rid of the multiplications and bypassing the square root operations completely.[7] Since we only wish to compare the amplitudes of $L(p)$ and its neighbors (*e.g.*, $L_{new}(p_{+x})$), it is equivalent and sufficient to compare the squares of their amplitudes (*e.g.*, $L(p)^2$ and L^2_{new}), thereby saving the floating point operation. Furthermore, we can augment the vector image $\mathbf{L}$ by also assigning the sum $L^2_{\overline{x}} + L^2_{\overline{y}}$ ($= L^2$) to each pixel p [44, 160], thereby obtaining a new vector image representation, $\mathbf{L}_+$, as follows:

$$\mathbf{L}_+(\overline{x},\overline{y}) = \left(L_{\overline{x}}, L^2_{\overline{x}} + L^2_{\overline{y}}\right) = \left(L_{\overline{x}}, L_{\overline{y}}, L^2\right) . \tag{D.3}$$

Then, $L_{new}(p_{+x})$ can be evaluated easily from the stored data as follows:

$$\begin{aligned} L^2_{new}(p_{+x}) &= \left(L_{\overline{x}}(p_{+x}) + 1\right)^2 + L^2_{\overline{y}}(p_{+x}) \\ &= \left(L^2_{\overline{x}}(p_{+x}) + L^2_{\overline{y}}(p_{+x})\right) + 2L_{\overline{x}}(p_{+x}) + 1 \\ &= L^2(p_{+x}) + 2L_{\overline{x}}(p_{+x}) + 1 . \end{aligned}$$

Indeed, since now L^2 is kept stored in $\mathbf{L}_+$, we can in this example (and similarly in all other cases; see below) evaluate $L_{new}(p_{+x})^2$ using only one add, one left-shift (*i.e.*, a multiplication by two) and one increment.[8] We summarize all the possibilities when evaluating L_{new} for both the $4SSEDT$ and $8SSEDT$ in the following four equations:

$$\left(L_{\overline{x}} \pm 1\right)^2 + L^2_{\overline{y}} = L^2 \pm 2L_{\overline{x}} + 1 , \tag{D.4}$$

$$L^2_{\overline{x}} + \left(L_{\overline{y}} \pm 1\right)^2 = L^2 \pm 2L_{\overline{y}} + 1 , \tag{D.5}$$

$$\left(L_{\overline{x}} \pm 1\right)^2 + \left(L_{\overline{y}} \pm 1\right)^2 = L^2 \pm 2\left(L_{\overline{x}} + L_{\overline{y}} \pm 1\right) , \tag{D.6}$$

$$\left(L_{\overline{x}} \pm 1\right)^2 + \left(L_{\overline{y}} \mp 1\right)^2 = L^2 + 2\left(\pm L_{\overline{x}} \mp L_{\overline{y}} + 1\right) = L^2 \pm 2\left(L_{\overline{x}} - L_{\overline{y}} \pm 1\right) \tag{D.7}$$

We can now evaluate the numerical complexity of both algorithms. Let us define the integer constant N^2 to represent the total number of object pixels found in an image I.[9] Let us first consider the case of the $4SSEDT$. Then, only equations (D.4) and (D.5) are to be considered. The complete updating procedure will require for each object pixel six comparison operations, which we refer to by using the symbol C_{4SSEDT}, with the six neighbors defined by the masks of the $4SSEDT$ (Figure D.1.(a)). A C_{4SSEDT} operation represents

[7] This optimized implementation was first brought to our attention in an unpublished work of one of our colleagues at McGill University [44]. Since then a similar idea has been presented by Ye [160], but without much detail.

[8] An increment (or decrement) is usually faster to compute (or at least as fast to compute) on most machines than an add (or subtract).

[9] $\mathrm{N}^2 = \sum_{(\overline{x},\overline{y})} i$, where $i = 1$ iff $I(\overline{x},\overline{y}) \in O$ and $i = 0$ iff $I(\overline{x},\overline{y}) \in O'$.

one left-shift, one add (or subtract), one increment and one compare. Therefore, finding the minimum distance values in the case of the $4SSEDT$ will require a total of $6\mathrm{N}^2$ C_{4SSEDT} operations. In the case of the $8SSEDT$ the four equations above are used. The complete updating procedure will require for each object pixel ten comparison operations. Six of them are just the C_{4SSEDT} operations as previously. The four other comparisons are derived from equations (D.6) and (D.7) where both $L_{\overline{x}}$ and $L_{\overline{y}}$ are incremented/decremented simultaneously. They are associated with the four diagonal neighbors (with respect to p) found in the masks of Figure D.1.(b). We refer to these four comparison operations using the symbol C_{8SSEDT}. A C_{8SSEDT} operation represents one left-shift, two add(s)/subtract(s), one increment/decrement and one compare. Therefore, finding the minimum distance values in the case of the $8SSEDT$ will require a total of $6\mathrm{N}^2$ C_{4SSEDT} plus $4\mathrm{N}^2$ C_{8SSEDT} operations; or in other words, $10\mathrm{N}^2$(left-shift, increment/decrement, compare) + $14\mathrm{N}^2$(add/subtract) operations.

Once we have completely updated the vector image $\mathbf{L}_+$, the real Euclidean distance map is recovered by computing the square roots of $\mathbf{L}_+$'s third elements (equation (D.3)). This accounts for N^2 floating point operations. However, in many practical cases the (integer) squared distance values may be sufficient. Furthermore, since we know the maximum possible size of an object ($\leq \sqrt{M} = \sqrt{M_{\overline{x}}^2 + M_{\overline{y}}^2}$), for fixed-sized images I (*i.e.*, an image of size $M_{\overline{x}} * M_{\overline{y}}$), we can store in a double-index lookup table all possible distance values L \begincitation [160]. This table can be indexed using the absolute values of the first two elements of the vector image $\mathbf{L}_+$, that is, $|L_{\overline{x}}|$ and $|L_{\overline{y}}|$. Such an approach is particularly useful for fixed-size images if distance maps are to be evaluated often.

This completes our detailed analysis of Danielsson's algorithms optimized for numerical performance. In the following subsection we show how these algorithms compare to the non-Euclidean ones (*i.e.*, the WDT's).

D.1.1 Comparison between Euclidean and Non-Euclidean Distance Transforms

Let us first consider the four simplest WDT's: the city block DT, the chessboard DT, and the *chamfer* DT's [28] for the two smallest neighborhood sizes (*i.e.*, 3x3 and 5x5). In Figure D.2 are shown the masks used for these WDT's. Note that only two masks are sufficient for each WDT while four were used for the EDT (compare with Figure D.1). This implies that the propagation of distances will not be isotropic and will generate errors in the distance map. However, this type of error is negligible "compared to the difference between the Euclidean distance and the non-Euclidean approximations" [127].

The chamfer DT's have different sizes and weights that can be chosen to satisfy various needs. A 3x3 chamfer DT is named *chamfer-*a*-*b where a and b represent the mask's

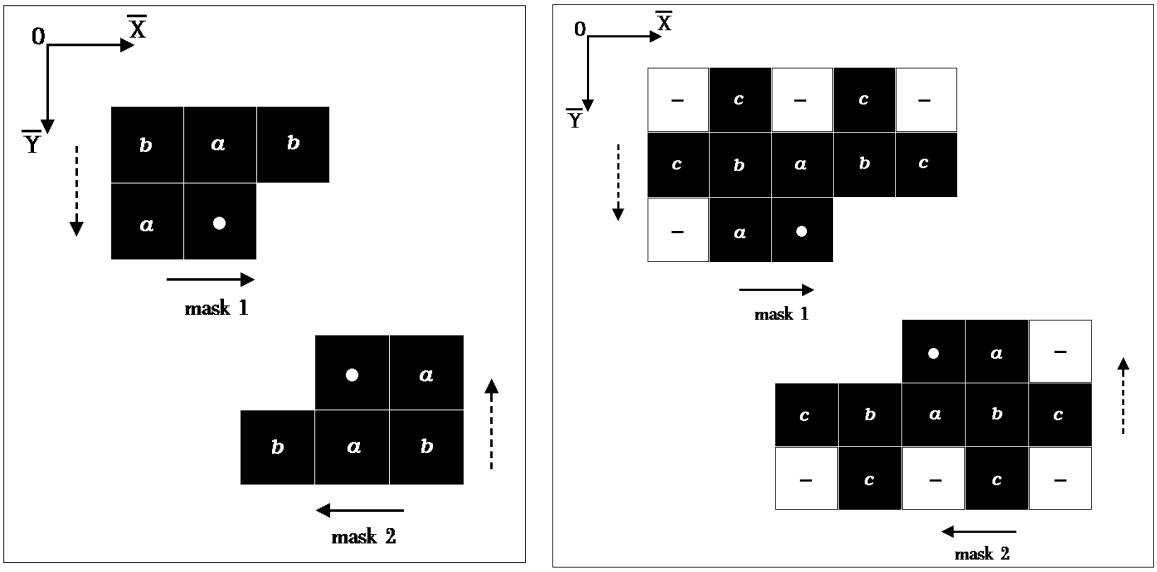
weights[10] (Figure D.2.(a)). The weights a and b are chosen to take into account accuracy and numerical efficiency. Chamfer DT's are scaled (*i.e.*, multiplied) by a factor r, usually by the weight a. This means that the computed distance values need to be divided by r, this to recover the best possible approximation to the Euclidean distance values. The "optimal" weights, for accuracy, are $a \approx 0.95509$ and $b \approx 1.36930$ [29]. However, for practical applications, the weights $a = 23$ and $b = 33$ are preferred,[11] this to maintain only integer computations [150]. A 5x5 chamfer DT is named *chamfer-*a*-*b*-*c where an additional weight c is needed. For such a neighborhood size, certain mask pixels do not contribute to the distance value propagation (*i.e.*, they are redundant). They are therefore suppressed from the local search for a minimum distance value (marked with a minus symbol ($-$) in Figure D.2.(b)). Again the weights a, b and c are chosen to satisfy the needs of accuracy and numerical efficiency. The optimal accuracy is obtained for $a \approx 0.98128$, $b \approx 1.40314$ and $c \approx 2.19529$ [150]. For practical applications, the weights $a = 5$, $b = 7$ and $c = 11$ are preferred to maintain only integer computations [29, 150].

For the city block DT (Figure D.2.(a)) a complete updating procedure will require for each object pixel four comparison operations, C_{CB}. A C_{CB} operation involves one increment and one compare. For the chessboard DT (Figure D.2.(b)), each object pixel will require eight comparison operations, C_{CB} (*i.e.*, same operations as for the city block DT). Chamfer-a-b DT's (Figure D.2.(a)) require eight comparison operations, $C_{chamfer}$, per object pixel. A $C_{chamfer}$ comparison consist of one add and one compare. Chamfer-a-b-c DT's (Figure D.2.(d)) require sixteen comparison operations, $C_{chamfer}$, per object pixel (*i.e.*, same operations as for chamfer-a-b DT's). In Table D.1 we compare both the EDT's and the WDT's we have studied so far from the point of view of both numerical complexity and accuracy.

Let us first note that both the $8SSEDT$ and the $4SSEDT$ have maximal possible errors (MaxDiff in Table D.1) expressed in absolute pixel unit values, pu, which are less than the sampling error of the digital grid ($\pm 0.5pu$). Therefore, these errors can be considered negligible in the discrete domain [127]. Moreover, as the distance values are augmented, these possible errors become more and more negligible (in relative percentage), that is, they tend rapidly toward zero. Also, these errors can only occur in a few rare and sparsely distributed locations [46] and therefore they do not propagate over the distance map. In the case of the WDT's, all errors are expressed as some percentage of the real Euclidean distance value. Therefore, the contrary to the EDT's, as the distance values, L, augment the absolute errors do also. Finally, these errors generally occur at groups of pixels (*i.e.*, non-isolated) and propagate rapidly to corrupt large portions of the distance map.

[10] Note that the city block and the chessboard DT's can be interpreted as the *chamfer*-1-2 and *chamfer*-1-1 DT's, respectively (Figure D.2.(a)). In the case of the city block DT the weight b becomes redundant (*i.e.*, $b = 2 * a$) and the masks can be further reduced in size.

[11] The weights $a = 2$ and $b = 3$ may be preferred over $a = 23$ and $b = 33$ when the scaling by $r = a$ is required. In this case $r = 2$ and the divide operation is simply replaced with a right-shift operation.



(a) (b)

Figure D.2: Masks used for WDT's for the two smallest neighborhood sizes (*i.e.*, 3x3 and 5x5). The masks are centered at pixel p (indicated by a white dot). The weights in each mask represent the (approximated, unsigned and scaled) distance from p. The reference frame ($\overline{X}$ and $\overline{Y}$ directions) is shown to be positioned in the top-left corner of an image, I. The arrows (next to the masks) indicate in which direction a given mask is passed over I. The convolution of these masks over I is performed in two picture scans (indicated by dashed arrows) in opposite directions (here downward for masks labelled 1 and upward for masks labelled 2). In (a) are shown the masks for the 3x3 chamfer DT's where a and b are weights to be optimized (see text). The city block and the chessboard DT's can be seen as particular cases of 3x3 chamfer DT's; that is, the *chamfer*-1-2 and *chamfer*-1-1 DT's, respectively. In (b) are shown the masks for the 5x5 chamfer DT's where a, b and c are three weights to be optimized, and where some mask pixels, shown by a minus symbol ($-$), are suppressed (see text).

Distance Transform	Numerical Complexity		Accuracy (MaxDiff)
	Comparison Op.	Floating Pt.	
$8SSEDT$	$10N^2(<<,++/--,<)$ $+14N^2(+/-)$	$N^2(\sqrt{})$	-0.09 pu (1)
$4SSEDT$	$6N^2(<<,+/-,++,<)$	$N^2(\sqrt{})$	-0.29 pu (1)
Chamfer-a-b-c	$16N^2(+,<)$	$N^2(\div)$	
$a \approx 0.981$, $b \approx 1.403$, $c \approx 2.195$	"	"	-1.872% (2)
$a = 5$, $b = 7$, $c = 11$	"	"	2.175% (2)
Chamfer-a-b	$8N^2(+,<)$	$N^2(\div)$	
$a \approx 0.955$, $b \approx 1.369$	"	"	4.491%(2)
$a = 23$, $b = 33$	"	"	6.209% (2)
$a = 2$, $b = 3$	"	$(>>)$	13.40% (3)
Chessboard	$8N^2(++,<)$	None	41.4% (3)
City block	$4N^2(++,<)$	None	-58.6% (3)

Table D.1: Comparison of Distance Transforms (EDT's and WDT's) for both numerical complexity and accuracy. DT's are ranked by accuracy performance. For chamfer DT's, the best real and integer weights are given. For numerical complexity performance, both comparison and floating point operations are shown. The significance of the symbols is: left-shift ($<<$), right-shift ($>>$), increment/decrement ($++/--$), add(s)/subtract(s) ($+/-$), compare ($<$), square root ($\sqrt{}$), divide ($\div$). For accuracy performance, the maximum possible difference (MaxDiff) or error with respect to the true Euclidean distance is given in percent ($\%$) or in absolute pixel units, pu (*i.e.*, the distance between two horizontal or vertical pixel centroids). References for accuracy performance: (1) $\equiv$ [46], (2) $\equiv$ [150], (3) $\equiv$ [28].

Let us now consider the numerical complexity of these different DT algorithms. We first ignore the floating point operations; this eliminates the chamfer DT's that use real-valued weights. Thus, the fastest DT is the city block DT (because it uses the smallest masks), but it is also the least accurate. In second place we find that the chessboard DT and the $4SSEDT$ have similar numerical complexity with a slight advantage for the chessboard DT. The $4SSEDT$ is slightly slower than the chessboard DT, even though the latter requires a visit to 25% more pixels, because the comparison operations for the $4SSEDT$ are more complex than in the chessboard case. In fourth place follows the chamfer-23-33 which has comparison operations as complex as the ones for the $4SSEDT$ (if we neglect the left-shift and the increment operations in front of the add/subtract operation), but which requires a visit to 25% more pixels. In fifth place comes the $8SSEDT$ and in last place is the chamfer-5-7-11 (followed by any other chamfer DT's defined on larger neighborhood; *e.g.*, 7x7 chamfer DT's).

If we do consider floating point operations, then the city block DT is still the fastest followed by the chessboard DT (they do not require any floating point operations). In third place comes the chamfer-2-3 DT which does not require floating point operations because the divide operation is a division by 2 (=a) which is simply implemented with a right-shift. In fourth place comes either the $4SSEDT$ or the chamfer-23-33 DT depending on whether or not a square root is faster then a divide operation on a given machine.[12] Then the $8SSEDT$, the 5x5 chamfer DT's, and chamfer DT's for larger neighborhoods, follow in that order.

In the worst possible case, with respect to the numerical complexity of the EDT's, where increment/decrement operations are not significantly faster than add/subtract per one and square root operations are slower than (floating point) divisions, we can still say that the $4SSEDT$ is faster than any WDT defined over neighborhoods larger than 3x3. Similarly, the $8SSEDT$ is faster than any WDT defined over neighborhoods larger than 5x5. Considering the inaccuracy of WDT's and their relative lack of speed improvement compared to the EDT's it seems that for practical applications they have little to offer.

The only possible advantage of WDT's over EDT's is that they have lower memory requirements. An efficient implementation of an EDT algorithm like the one we have described in section D.1, requires three times more space than the WDT's (to store the vector image $\mathbf{L}_+$). However, the extra information we obtain might be very useful (other than the fact that $\mathbf{L}_+$ gives us exact Euclidean distances at a low numerical complexity). For example, because we store both the $\overline{X}$ and $\overline{Y}$ distance values and their signed orientation (given by $L_{\overline{x}}$ and $L_{\overline{y}}$), we always know the exact position of the nearest background pixel for any object point. This information can become particularly useful when employed in conjunction with the skeleton of the object (Chapter 5). This information permits us

[12] For example, on our standard equipped SUN 3 workstation, square root operations are faster than (floating point) divide operations. However, if we use a look-up table to bypass the square root computations (§ D.1), the $4SSEDT$ is definitely faster than any chamfer-a-b that requires floating point divisions.

to directly recover the complete boundary from the skeleton, thereby yielding the *Inverse Grassfire Transform* [25].

D.2 The Distance Surface

In this section we briefly give definitions and properties related to the notion of a *distance surface*. See [69] for proofs and more extensive details.

We call a distance surface a surface $z = \phi(x, y)$ such that ϕ is a solution of the following differential equation:[13]

$$(\nabla\phi)^2 = \left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2 = 1\,. \tag{D.8}$$

Let $\mathcal{C}$ be the *base set* (or object contour) on which $\phi(x, y) = 0$. An associated set at $\phi(x, y) = q$, where q is a constant, consist in planar curves $\mathcal{C}_q$ which are parallel to $\mathcal{C}$ at a signed distance q from $\mathcal{C}$. The orthogonal trajectories (*i.e.*, the directions of maximal gradient of $z = \phi$) of these level curves are straight lines. These straight lines emerge from the base set $\mathcal{C}$. Therefore, $z = \phi$ is a *ruled surface* generated by such a set of lines.[14] These lines make an angle $\frac{\pi}{4}$ with the x-y plane (*i.e.*, the slope of a tangent to these lines is always of magnitude one). Furthermore, $z = \phi$ is also a *developable surface* (*i.e.*, it can be rolled out flat onto a plane) which implies that any of its regular point has a null *Gaussian curvature* [37]. This is easily understood if we note that, at any regular point of $z = \phi$, one of the *principal directions* is always along an orthogonal trajectory of a level curve $\mathcal{C}_q$ (*i.e.*, a straight line). Therefore the corresponding *principal curvature* is null.

From equation (D.8) we have directly that the gradient magnitude of $z = \phi$ is constant and equal to one at all regular points (*i.e.*, $|\nabla\phi| = 1$). Furthermore, the directions in which this gradient magnitude of $z = \phi$ is maximal and equal to one are, by definition, the directions of the orthogonal trajectories of the level curves $\mathcal{C}_q$. Therefore, $z = \phi$ varies with the Euclidean distance along these orthogonal trajectories (*i.e.*, straight lines of unit speed). In other words, the maximal slope amplitude of $z = \phi$ is always along these orthogonal trajectories (direction of maximal gradient). At singular points (*i.e.*, non-regular points where $\frac{\partial\phi}{\partial x}$ and $\frac{\partial\phi}{\partial y}$ are undefined), $z = \phi$ is the radius of curvature of $\mathcal{C}$ (*i.e.*, the center of a maximal inscribed circle).[15] At these points the orthogonal trajectories emerging from $\mathcal{C}$ intersect. We call these points of intersection *ridge points* of the distance surface $z = \phi$. In the following subsection we demonstrate that the slope along successive ridge points is

[13] This equation is known as the *eikonal equation* in geometrical optics [68].

[14] The projection of these lines onto the x-y plane correspond to the "pannormals" of Blum (Chapter 5, § 5.4.1).

[15] This is not necessarily true: a medial symmetry depends of two or more contacts irrespective of the local curvature at each such contact point (erratum, February 2003).

always smaller in magnitude than the maximal slope at regular points of $z = \phi$, this being a consequence of the fact that ridge points do not belong to the minimal path, that is, a path of minimum distance, of any other point [86].

D.2.1 Ridge Points of the Distance Surface

Let us define a *symmetric axis segment* to be a finite length 3-D curve (opened and non-planar) which trace consists of a connected set of ridge points. The trace of this curve is constrained to be on the distance surface $z = \phi(x, y)$. Since the orthogonal trajectories of ϕ are of unit speed, the slope magnitude along a tangent to a symmetric axis segment is necessarily less than one. This follows from the distance constraint imposed on ridge points (*i.e.*, being part of $z = \phi$). For example, consider two ridge points R_1 and R_2 separated by a distance $\overline{l_A}$ (arc length) along the symmetric axis segment (Figure D.3). By definition, R_1 and R_2 are at a minimal distance d_1 and d_2, respectively, from the base set $\mathcal{C}$. Let us assume that R_2 is farther away from $\mathcal{C}$ than R_1; that is:

$$d_2 > d_1 \ , or \ d_2 = d_1 + \Delta d \, . \tag{D.9}$$

At R_2, because of the minimal distance constraint, we have the following:

$$d_2 < d_1 + \overline{l_A} \, . \tag{D.10}$$

Therefore, combining equations (D.9) and (D.10), we get:

$$\Delta d < \overline{l_A} \, . \tag{D.11}$$

But the average slope magnitude, M_A, along the ridge from R_1 to R_2 is simply:

$$M_A = \frac{\phi(R_2) - \phi(R_1)}{\overline{l_A}} = \frac{\Delta d}{\overline{l_A}} \, . \tag{D.12}$$

Therefore, equations (D.11) and (D.12) imply that M_A is necessarily lesser than one. The same result holds in the limit as R_2 is chosen to be closer and closer to R_1; that is, as $\overline{l_A}$ tends to zero,[16]

$$m_A = \lim_{\overline{l_A} \to 0} \frac{\Delta d(\overline{l_A})}{\overline{l_A}} < 1 \, , \tag{D.13}$$

where m_A is the slope magnitude at any ridge point, and $\Delta d(\overline{l_A})$ is equal to the difference between d_2 and d_1 (equation (D.9)) but varies as $\overline{l_A}$ tends to zero. Therefore, m_A is taken along a tangent to the symmetric axis and is always smaller than one.

[16] If, as $R_2 \to R_1$, d_2 becomes smaller than d_1 (*i.e.*, ϕ is nonmonotonic along the ridge from R_2 to R_1), we then simply reverse the roles of R_2 and R_1 and our result still holds.

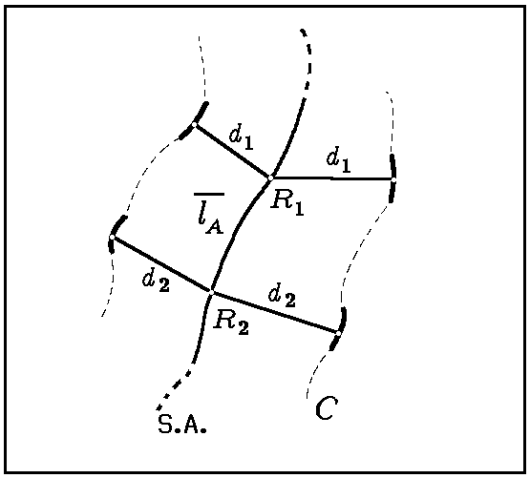


Figure D.3: The slope magnitude of the symmetric axis (S.A.). The figure shows two symmetry points (or ridge points) R_1 and R_2 and their associated distances from the base set $\mathcal{C}$, d_1 and d_2, respectively (only segments of the arbitrary base set $\mathcal{C}$ are shown). Two lines of minimal distance from $\mathcal{C}$ are shown to converge at R_1 and R_2, respectively (corresponding to the intersecting orthogonal trajectories emerging from $\mathcal{C}$; see text). $\overline{l_A}$ is the length between R_1 and R_2 along the ridge (or S.A.).

D.3 The Snake Model and the Distance Surface

In this last section we show how we can improve the computations when applying the snake model to a distance surface seen as a potential surface (Chapter 5, § 5.3.1) by taking advantage of the properties of such a surface obtained with an EDT.

- The slope of the distance surface is implicitly normalized (in magnitude). At regular points it is bounded by one, while at singular points (or ridge points), if evaluated over a discrete neighborhood, it will always be smaller than one (*i.e.*, the steepest slope corresponds to an angle of $\frac{\pi}{4}$).[17] Therefore, we avoid the "slope normalization step" that was required in the general case (Chapter 2, § 2.7).

- We can use a signed EDT algorithm to our advantage to directly evaluate the slope at any regular points of the distance surface. We employ the three entries of the extended image vector $\mathbf{L}_+$ (equation (D.3)) which give us the $\overline{X}$ ($L_{\overline{x}}$) and $\overline{Y}$ ($L_{\overline{y}}$)

[17] Note that this is not directly related to the property of the slope of the symmetric axis of being smaller than one as given by equation (D.13). For example, we can evaluate the slope at a ridge point by looking at the first differences in both $\overline{X}$ and $\overline{Y}$ directions. Because the distance surface folds at this point into two *local distance surface patches* (*i.e.*, patches of the distance surface made of regular points only) the slope is averaged over both patches and is always less than unity.

directions (both signed) and the distance amplitude (L^2). The directional slopes in the $\overline{X}$ and $\overline{Y}$ directions ($\phi_{\overline{x}}$ and $\phi_{\overline{y}}$) are then simply evaluated as follows:

$$\phi_{\overline{x}} = \frac{L_{\overline{x}}}{\sqrt{L^2}}\,,\ \ \phi_{\overline{y}} = \frac{L_{\overline{y}}}{\sqrt{L^2}}\,.$$

Obviously these equations are not valid at ridge points (where $\phi_{\overline{x}}$ and $\phi_{\overline{y}}$ are undefined). Nevertheless, we can still evaluate $\phi_{\overline{x}}$ and $\phi_{\overline{y}}$ and observe variations from expected behavior if these points were regular points. For example, we can observe how the computed orientation of the slope ($\theta =$ angle given by $(L_{\overline{x}}, L_{\overline{y}})$) varies over the iterations (when simulating the grassfire propagation). If we are in a region consisting of regular points only, the orientation will change smoothly (ideally along an orthogonal trajectory or pannormal). However, in the case of a neighborhood centered at a ridge point, the orientation θ will change abruptly (corresponding to the intersection of pannormals). When such a case is detected, we can backtrack (*i.e.*, reset a snaxel to the previous position) and evaluate the directional slopes on the basis of finite differences.

- The spatial sampling of snaxels (*i.e.*, the number of snaxels per arc length unit) along a snake segment can be efficiently updated by observing how the convergence of pannormals[18] influences the snaxels' inter-distance, $\Delta\overline{s}$. Snaxels should follow the directions imposed by pannormals when evaluating $\phi_{\overline{x}}$ and $\phi_{\overline{y}}$. Therefore, for each snake segment, we propose the following two heuristics to reset snaxel sampling. If the average $\Delta\overline{s}$ (averaged over all snaxels of a snake segment) becomes lower than a desired limit, then the number of snaxels is reduced. This may be the case of a snake segment initialized on a convex contour segment (*e.g.*, circular arc). Otherwise, if the average $\Delta\overline{s}$ becomes larger than a desired limit, the number of snaxels is increased. This may be the case of a snake segment initialized on a concave contour segment. Furthermore, this resampling procedure permits us to keep the stiffness parameter $\omega_1(\overline{s})$ constant thereby simplifying the snake computations (Chapter 2, § 2.7).

[18] Three cases are possible: the pannormals converge (*i.e.*, they emerge from a convex contour segment), they remain parallel (*i.e.*, they emerge from a straight contour segment), or they diverge (*i.e.*, they emerge from a concave contour segment).

Bibliography

[1] S. M. Ali and R. E. Burge. A new algorithm for extracting the interior of bounded regions based on chain coding. *Computer Vision, Graphics, and Image Processing*, 43:256–264, 1988.

[2] A. A. Amini, S. Tehrani, and T. E. Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *Proceedings of the* 2^{nd} *International Conference on Computer Vision*, pages 95–99, Tampa, Florida, U.S.A., December 1988. IEEE Computer Society Press.

[3] A. A. Amini, T. E. Weymouth, and R. C. Jain. Using dynamic programming for solving variational problems in vision. Technical Report CSE-TR-05-88, University of Michigan, Electrical Engineering & Computer Science Department, Ann Arbor, Michigan, U.S.A., May 1988.

[4] C. Arcelli. Pattern thinning by contour tracing. *Computer Graphics and Image Processing*, 17:130–144, 1981.

[5] C. Arcelli, L. P. Cordella, and S. Leviadi. From local maxima to connected skeletons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(2):134–143, March 1981.

[6] C. Arcelli and G. Sanniti di Baja. On the sequential approach to medial line transformation. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-8(2):139–144, 1978.

[7] C. Arcelli and G. Sanniti di Baja. A width-independent fast thinning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(4):463–474, July 1985.

[8] C. Arcelli and G. Sanniti di Baja. Computing Voronoi diagrams in digital pictures. *Pattern Recognition Letters*, 4(5):383–390, October 1986.

[9] C. Arcelli and G. Sanniti di Baja. Finding multiple pixels. In S. Leviadi V. Cantoni and G. Musso, editors, *Image Analysis and Processing*, pages 137–144. Plenum Press, New York, U.S.A., 1986.

[10] C. Arcelli and G. Sanniti di Baja. On the simplicity of digital curves and contours. In *Proceedings of the* 8^{th} *International Conference on Pattern Recognition*, pages 283–285, Paris, France, October 1986. IEEE Computer Society Press.

[11] C. Arcelli and G. Sanniti di Baja. A contour characterization for multiply connected figures. *Pattern Recognition Letters*, 6:245–249, September 1987.

[12] C. Arcelli and G. Sanniti di Baja. Finding local maxima in a pseudo-Euclidean distance transform. *Computer Vision, Graphics, and Image Processing*, 43:361–367, 1988.

[13] C. Arcelli and G. Sanniti di Baja. A one-pass two-operations process to detect the skeletal pixels on the 4-distance transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(4):411–414, April 1989.

[14] H. Asada and M. Brady. The curvature primal sketch. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):2–14, January 1986.

[15] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3):183–193, 1954.

[16] F. Attneave. Criteria for a tenable theory of form perception. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 56–67. M.I.T. Press, Cambridge, Mass., U.S.A., 1967. Proceedings of a Symposium Held in Boston, November 1964.

[17] E. Bell, D. Levinstone, S. Sher, L. Marek, C. Merrill, I. Young, and M. Eden. An interactive computer system for the analysis of cell lineages. *Journal of Histology and Cytology*, 27(1):458–462, 1979.

[18] A. Benson and D. J. Evans. A normalized algorithm for the solution of positive definite symmetric quindiagonal systems of linear equations. *ACM Transactions on Mathematical Software*, 3(1):96–103, March 1977.

[19] A. Blake and A. Zisserman. *Visual Reconstruction*. The MIT Press, Cambridge, Mass., U.S.A., 1987.

[20] A. Blake, A. Zisserman, and A. V. Papoulias. Weak continuity constraints generate uniform scale-space descriptions of plane curves. In *Proceedings of the 7th European Conference on Artificial Intelligence*, Brighton, England, 1986.

[21] C. Blakemore and R. Over. Curvature detectors in human vision. *Perception*, 3:3–7, 1974.

[22] H. Blum. An associative machine for dealing with the visual field and some of its biological implications. In E. E. Bernard and M. R. Kare, editors, *Biological Prototypes and Synthetic Systems*, volume 1, pages 244–260, New York, U.S.A., 1962. Plenum Press. Proceedings of the 2^{nd} Annual Bionics Symposium, held at Cornell University, 1961.

[23] H. Blum. A machine for performing visual recognition by use of antenna-propagation concepts. *Institute of Radio Engineers, Wescon Convention Record*, 6, part 4, session 6.4, August 1962. Western Electronics Show and Convention, held in Los Angeles, Cal., U.S.A.

[24] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. M.I.T. Press, Cambridge, Mass., U.S.A., 1967. Proceedings of a Symposium Held in Boston, November 1964.

[25] H. Blum. Biological shape and visual science (part I). *Journal of Theoretical Biology*, 38:205–287, 1973.

[26] H. Blum and R. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978.

[27] F. L. Bookstein. The line skeleton. *Computer Graphics and Image Processing*, 11:123–137, 1979.

[28] G. Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27(3):321–345, September 1984.

[29] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, June 1986.

[30] G. Borgefors. Time: Time and memory efficient distance transformations for parallel and pyramid machines. FOA Report (ISSN 0347-3708) C 30531-3.4, Swedish Defense Research Establishment, Dept. of Information Technology, Linköping, Sweden, May 1989.

[31] G. Borgefors and G. Sanniti di Baja. Skeletonizing the distance transform on the hexagonal grid. In *Proceedings of the* 9^{th} *International Conference on Pattern Recognition*, volume 1, pages 504–507, Rome, Italy, November 1988. IEEE Computer Society Press.

[32] G. Bouligand. *Introduction à La Géométrie Infinitésimale Directe*. Vuibert, Paris, France, 1932.

[33] T. E. Boult, S. D. Fenster, and T. O'Donnell. Reinterpreting physically–motivated modeling. In *Proc. of ARPA Image Understanding Workshop*, volume 2, pages 1375–1390, Monterey, CA, U.S.A., November 1994.

[34] A. Boyarsky and P. B. Noble. A marker chain characterization of human neutrophil locomotion under neutral and chemotactic conditions. *Canadian Journal of Physiology and Pharmacology*, 55:1–6, 1977.

[35] M. Brady. Computational approaches to image understanding. *Computing Surveys*, 14(1):3–71, March 1982.

[36] M. Brady and H. Asada. Smoothed local symmetries and their implementation. *The International Journal of Robotics Research*, 3(3):36–61, Fall 1984.

[37] I. N. Bronshtein and K. A. Semendyayev. *Handbook of Mathematics*. Van Nostrand Reinhold, New York, U.S.A., 1985. K. A. Hirsch, ed.; 3^{rd} edition, english version.

[38] P. J. Burt. Fast filter transforms for image processing. *Computer Graphics and Image Processing*, 16:20–51, 1981.

[39] P. J. Burt. Fast algorithms for estimating local image properties. *Computer Vision, Graphics and Image Processing*, 21:368–382, 1983.

[40] P. J. Burt. The pyramid as a structure for efficient computations. In A. Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 6–35. Springler-Verlag, 1984.

[41] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540, April 1983.

[42] R. T. Chin, H.-K. Wan, D. L. Stover, and R. D. Iverson. A one-pass thinning algorithm and its parallel implementation. *Computer Vision, Graphics and Image Processing*, 40:30–40, 1987.

[43] L. P. Cordella and G. Dettori. An $O(N)$ algorithm for polygonal approximation. *Pattern Recognition Letters*, 5(3):93–97, March 1985.

[44] H. Cox. Distance transformations in binary images. Project report for the course 304-529a, Image processing and communications, taught by m. d. levine, McGill University, Electrical Engineering Department, Montréal, QC, Canada, December 1986.

[45] J. L. Crowley and R. M. Stern. Fast computation of the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(2):212–222, March 1984.

[46] P. E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[47] C. David. From discrete tangent fields to global curves: Dynamic coverings. Master's thesis, McGill University, Electrical Engineering Department, Montréal, QC, Canada, 1989.

[48] C. David and S. Zucker. Potentials, valleys, and dynamic global coverings. McRCIM Technical Report CIM-89-1, McGill University, Electrical Engineering Department, Montréal, QC, Canada, March 1989.

[49] E. R. Davies and A. P. N. Plummer. Thinning algorithms: A critique and a new methodology. *Pattern Recognition*, 14(1):53–63, 1981.

[50] A. de Boisfleury-Chevance, B. Rapp, and H. Gruler. Locomotion of white blood cells: A biophysical analysis. *Blood Cells*, 15:315–333, 1989.

[51] R. Desimone and L. G. Ungerleider. Neural mechanisms of visual processing in monkeys. In F. Bollers and J. Grafman, editors, *Handbook of Neuropsychology*, volume 2, pages 267–299 (Chapter 14). Elsevier Science Publishers B.V. (Biomedical Division), 1989.

[52] A. R. Dill, M. D. Levine, and P. B. Noble. Multiple resolution skeletons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):495–504, July 1987.

[53] M. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, New Jersey, U.S.A., 1976.

[54] A. Dobbins, S. W. Zucker, and M. Cynader. Endstopped neurons in the visual cortex as a substrate for calculating curvature. *Nature*, 329:438–441, October 1987.

[55] L. Dorst. Pseudo-Euclidean skeletons. In *Proceedings of the 8^{th} International Conference on Pattern Recognition*, pages 286–288, Paris, France, October 1986. IEEE Computer Society Press.

[56] L. Dorst and P. W. Verbeek. The constrained distance transformation: A pseudo-Euclidean, recursive implementation of the Lee-algorithm. In I.T. Young *et al.*, editor, *Signal Processing III: Theories and Applications*, pages 917–920 (M.21). Elsevier Science Publishers B.V. (North-Holland), 1986.

[57] A. Favre and H. Keller. Parallel syntactic thinning by recoding of binary pictures. *Computer Vision, Graphics and Image Processing*, 23:99–112, 1983.

[58] F. P. Ferrie. Experiments in tracking the morphologies of proliferating cell cultures by automatic picture processing. Master's thesis, McGill University, Electrical Engineering Department, Montréal, QC, Canada, September 1979.

[59] F. P. Ferrie, J. Lagarde, and P. Whaite. Towards sensor-derived models of objects. In *Proceedings of "Vision Interface '89"*, pages 166–174, London, OT, Canada, June 1989. Canadian Image Processing and Pattern Recognition Society.

[60] F. P. Ferrie, M. D. Levine, and S. W. Zucker. Cell tracking: A modeling and minimization approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(3):277–291, May 1982.

[61] M. M. Fleck. Local rotational symmetries. In *Proceedings of the 1986 Conference on Computer Vision and Pattern Recognition*, pages 332–337, Miami, Florida, U.S.A., June 1986. IEEE Computer Society Press.

[62] M. J. Forray. *Variational Calculus in Science and Engineering*. McGraw-Hill, New York, U.S.A., 1968.

[63] H. Freeman. On the classification of line drawing data. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Form*, pages 408–412. M.I.T. Press, Cambridge, Mass., U.S.A., 1967. Proceedings of a Symposium Held in Boston, November 1964.

[64] H. Freeman. Shape description via the use of critical points. *Pattern Recognition*, 10:159–166, 1978.

[65] H. Freeman and L. Davis. A corner-finding algorithm for chain-coded curves. *IEEE Transactions on Computers*, C-26:297–303, March 1977.

[66] P. Fua and Y. Leclerc. Model driven edge detection. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1016–1021, Washington, DC, U.S.A., April 1988. To be also published in *Machine Vision and Applications*.

[67] C. Garbay. Image structure representation and processing: A discussion of some segmentation methods in cytology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):140–146, March 1986.

[68] H. Goldstein. *Classical Mechanics*. Addison-Wesley, 2^{nd} edition, Reading, Mass., U.S.A., 1980.

[69] A. W. Goodman. A partial differential equation and parallel plane curves. *American Mathematical Monthly*, 71:257–264, March 1964.

[70] M.-H. Han, D. Jang, and J. Foster. Identification of cornerpoints of two-dimensional images using a line search method. *Pattern Recognition*, 22(1):13–20, 1989.

[71] P. Hanks, editor. *The Collins Dictionnary of the English Language*. Collins, 2^{nd} edition, London, England, 1986.

[72] C. J. Hilditch. Linear skeletons from square cupboards. *Machine Intelligence*, 4:403–420, 1969.

[73] C. J. Hilditch. Comparison of thinning algorithms on a parallel processor. *Image and Vision Computing*, 1(3):115–132, August 1983.

[74] S.-B. Ho and C. R. Dyer. Shape smoothing using medial axis transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):512–520, July 1986.

[75] D. D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[76] R. Hummel. The scale-space formulation of pyramid data structures. In L. Uhr, editor, *Parallel Computer Vision*, pages 107–123. Academic Press, New York, U.S.A., 1987.

[77] D. A. H. Jacobs, editor. *The State of the Art in Numerical Analysis*. Academic Press, London, England, 1977.

[78] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proceedings of the* 1^{st} *International Conference on Computer Vision*, pages 259–268, London, England, June 1987. IEEE Computer Society Press.

[79] F. Klein and O. Kübler. Euclidean distance transformation and model-guided image interpretation. *Pattern Recognition Letters*, 5(1):19–30, January 1987.

[80] J. C. Klein and J. Serra. The texture analyzer. *Journal of Microscopy*, 95, pt. 2:349–356, April 1972.

[81] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[82] J. C. Kotelly. A mathematical model of Blum's theory of pattern recognition. Research report AFCRL-63-164, Air Force Cambridge Research Laboratories, Bedford, Mass., U.S.A., April 1963.

[83] J. M. Lackie. *Cell Movement and Cell Behaviour*. Allen and Unwin, London, England, 1986.

[84] Y. Leclerc and P. Fua. Finding object boundaries using guided gradient ascent. In *Proceedings of the DARPA Image Understanding Workshop*, pages 888–891, Los Angeles, Cal., U.S.A., February 1987.

[85] D. T. Lee. Medial axis transformation of a planar shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):363–369, July 1982.

[86] G. Levi and U. Montanari. A grey-weighted skeleton. *Information and Control*, 17(1):62–91, August 1970.

[87] M. D. Levine. *Vision in Man and Machine*. Computer Engineering Series. McGraw-Hill, New York, U.S.A., 1985.

[88] M. D. Levine, P. B. Noble, and Y. M. Youssef. Understanding blood cell motion. *Computer Vision, Graphics and Image Processing*, 21:58–84, 1983.

[89] M. D. Levine, Y. M. Youssef, P. B. Noble, and A. Boyarsky. The quantification of blood cell motion by a method of automatic digital picture processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(5):444–450, September 1980.

[90] F. Leymarie and M. D. Levine. Curvature morphology. McRCIM Technical Report CIM-88-26, McGill University, Electrical Engineering Department, Montréal, QC, Canada, December 1988.

[91] F. Leymarie and M. D. Levine. Curvature morphology. In *Proceedings of "Vision Interface '89"*, pages 102–109, London, OT, Canada, June 1989. Canadian Image Processing and Pattern Recognition Society.

[92] F. Leymarie and M. D. Levine. New method for shape description based on an active contour model. In *Proceedings of the SPIE "Visual Communications and Image Processing '89" Conference*, volume 1199, part 1, pages 390–401, Philadelphia, Penn., U.S.A., November 1989.

[93] F. Leymarie and M. D. Levine. Shape features using curvature morphology. In *Proceedings of the SPIE "Intelligent Robots and Computer Vision VIII: Algorithms and Techniques" Conference*, volume 1192, part 2, pages 536–547, Philadelphia, Penn., U.S.A., November 1989.

[94] F. Leymarie and M. D. Levine. Snakes and skeletons. McRCIM Technical Report CIM-89-3, McGill University, Electrical Engineering Department, Montréal, QC, Canada, January 1989.

[95] F. Leymarie and M. D. Levine. Skeletons from snakes. In V. Cantoni et al., editors, *Progress in Image Analysis and Processing*, pages 186–193. World Scientific, Singapore, 1990. Proceedings of the 5^{th} International Conference on Image Analysis and Processing, held in Positano, Italy, 20–22 September, 1989.

[96] F. Leymarie and M. D. Levine. Fast raster scan distance propagation on the discrete rectangular lattice. *Computer Vision, Graphics and Image Processing – Image Uunderstanding*, 55(1):84–94, 1992.

[97] F. Leymarie and M. D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):56–75, 1992.

[98] F. Leymarie and M. D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–634, June 1993.

[99] M. Leyton. Symmetry-curvature duality. *Computer Vision, Graphics and Image Processing*, 38:327–341, 1987.

[100] M. Leyton. A process grammar for shape. *Artificial Intelligence Journal*, 34(2):213–247, March 1988.

[101] N. K. Link and S. W. Zucker. Corner detection in curvilinear dot grouping. *Biological Cybernetics*, 59:246–256, 1988.

[102] D. Marr. *Vision*. W. H. Freeman, San Francisco, Cal., U.S.A., 1982.

[103] M. P. Martínez-Pérez, J. Jiménez, and J. Navalón. A thinning algorithm based on contours. *Computer Vision, Graphics and Image Processing*, 39:186–201, 1987.

[104] G. Matheron. Examples of topological properties of skeletons. In J. Serra, editor, *Image Analysis and Mathematical Morphology; Volume 2: Theoretical Advances*, pages 217–238 (Chapter 11). Academic Press, London, England, 1988.

[105] J. W. McKee and J. K. Aggarwal. Computer recognition of partial views of curved objects. *IEEE Transactions on Computers*, C-26(8):790–800, August 1977.

[106] G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic B-splines. *Computer Vision, Graphics, and Image Processing*, 39(3):267–278, 1987.

[107] P. Meer, E. S. Baugher, and A. Rosenfeld. Extraction of trend lines and extrema from multiscale curves. *Pattern Recognition*, 21(3):217–226, 1988.

[108] F. Mokhtarian and A. Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):34–43, January 1986.

[109] F. Mokhtarian and A. Mackworth. The renormalized curvature scale space and the evolution properties of planar curves. In *Proceedings of the 1988 Conference on Computer Vision and Pattern Recognition*, Ann Arbor, Michigan, U.S.A., June 1988. IEEE Computer Society Press.

[110] U. Montanari. A method for obtaining skeletons using a quasi-Euclidean distance. *Journal of the Association for Computing Machinery*, 15(4):600–624, October 1968.

[111] U. Montanari. Continuous skeletons from digitized images. *Journal of the Association for Computing Machinery*, 16(4):534–549, October 1969.

[112] A. Montanvert. Medial line: Graph representation and shape description. In *Proceedings of the* 8^{th} *International Conference on Pattern Recognition*, pages 430–432, Paris, France, October 1986. IEEE Computer Society Press.

[113] Th. Motzkin. Sur quelques propriétés caractéristiques des ensembles bornés non convexes. *Atti Della Accademia Nazionale Dei Lincei*, 21:773–779, 1935.

[114] Th. Motzkin. Sur quelques propriétés caractéristiques des ensembles convexes. *Atti Della Accademia Nazionale Dei Lincei*, 21:562–567, 1935.

[115] P. B. Noble and M. D. Levine. *Computer-Assisted Analyses of Cell Locomotion and Chemotaxis*. CRC Press, Boca Raton, Florida, U.S.A., 1986.

[116] L. O'Gorman. An analysis of feature detectability from curvature estimation. In *Proceedings of the 1988 Conference on Computer Vision and Pattern Recognition*, pages 235–240, Ann Arbor, Michigan, U.S.A., June 1988. IEEE Computer Society Press.

[117] L. O'Gorman. Curvilinear feature detection from curvature estimation. In *Proceedings of the* 9^{th} *International Conference on Pattern Recognition*, volume 1, pages 1116–1119, Rome, Italy, November 1988. IEEE Computer Society Press.

[118] N. Okabe, J.-I. Toriwaki, and T. Fukumura. Paths and distance functions on three-dimensional digitized pictures. *Pattern Recognition Letters*, 1:205–212, 1983.

[119] W. R. Oliver. Cell locomotion as shape change. *Blood Cells*, 15:334–342, 1989.

[120] P. Parent and S. W. Zucker. Trace inference, curvature consistency and curve detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(8):823–839, August 1989.

[121] T. Pavlidis. Algorithms for shape analysis of contours and waveforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(4):301–312, July 1980.

[122] T. Pavlidis. A thinning algorithm for discrete binary images. *Computer Graphics and Image Processing*, 13:142–157, 1980.

[123] T. Pavlidis. An asynchronous thinning algorithm. *Computer Graphics and Image Processing*, 20:133–157, 1982.

[124] T. Pavlidis and S. L. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, C-23(8):860–870, August 1974.

[125] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. In *Proceedings of the 1987 Workshop on Computer Vision*, pages 16–22, Miami Beach, Florida, U.S.A., November 1987. IEEE Computer Society Press.

[126] J. G. Proakis and D. G. Manolakis. *Introduction to Digital Signal Processing*. Macmillan, New York, U.S.A., 1988.

[127] I. Ragnemalm. The Euclidean distance transform and its implementation on SIMD architectures. In *Proceedings of the* 6^{th} *Scandinavian Conference on Image Analysis*, pages 379–384, Oulu, Finland, June 1989.

[128] V. S. Ramachandran. Interactions between motion, color and form: The utilitarian theory of perception. In Colin Blakemore, editor, *Visual Coding and Efficiency*. Cambridge University Press, Cambridge, England, 1987.

[129] W. S. Ramsey. Analysis of individual leukocyte behavior during chemotaxis. *Experimental Cell Research*, 70:129–139, 1972.

[130] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume 2, 2^{nd} edition. Academic Press, New York, U.S.A., 1982.

[131] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *Journal of the Association for Computing Machinery*, 13(4):471–494, October 1966.

[132] A. Rosenfeld and J. L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[133] A. Rosenfeld and Weszka. An improved method of angle detection on digital curves. *IEEE Transactions on Computers*, C-24:940–941, 1975.

[134] G. L. Scott. The alternative snake – and other animals. In J.-O. Eklundh, editor, *The 1987 Stockholm Workshop on Computational Vision*, Stockholm, Sweden, 1987.

[135] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, England, 1982.

[136] B. Shapiro, J. Pisa, and J. Sklansky. Skeleton generation from *x,y* boundary sequences. *Computer Graphics and Image Processing*, 15:136–153, 1981.

[137] L. H. Staib and J. S. Duncan. Parametrically deformable contour models. In *Proceedings of the 1989 Conference on Computer Vision and Pattern Recognition*, pages 98–103, San Diego, Cal., U.S.A., June 1989. IEEE Computer Society Press.

[138] S. R. Sternberg. Grayscale morphology. *Computer Vision, Graphics and Image Processing*, 35:333–355, 1986.

[139] G. Y. Tang and B. Lien. Region filling with the use of the discrete Green theorem. *Computer Vision, Graphics and Image Processing*, 42:297–305, 1988.

[140] C.-H. Teh and R. T. Chin. On the detection of dominant points on digital curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(8):859–872, August 1989.

[141] D. Terzopoulos. The role of constraints and discontinuities in visible-surface reconstruction. In *Proceedings of the* 8^{th} *International Joint Conference on Artificial Intelligence*, volume 2, pages 1073–1077, Karlsruhe, West Germany, August 1983.

[142] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424, July 1986.

[143] D. Terzopoulos. Elastically deformable models. In *Proceedings of the SISGRAPH '87*, Anaheim, California, U.S.A., July 1987.

[144] D. Terzopoulos. Matching deformable models to images: Direct and iterative solutions. In *Topical Meeting on Machine Vision, Technical Digest Series*, volume 12, pages 164–167. Optical Society of America, Washington, D.C., U.S.A., 1987.

[145] D. Terzopoulos. On matching deformable models to images. In *Topical Meeting on Machine Vision, Technical Digest Series*, volume 12, pages 160–163. Optical Society of America, Washington, D.C., U.S.A., 1987. Presented at the Optical Society of America Machine Vision Topical Meeting, March 18-20, 1987, Incline Village, Nevada, U.S.A..

[146] D. Terzopoulos. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-10(4):417–438, July 1988.

[147] D. Terzopoulos, A. Witkin, and M. Kass. Stereo matching as constrained optimization using scale continuation methods. In *Proceedings of the SPIE Conference on Optical and Digital Pattern Recognition*, Los Angeles, Cal., U.S.A., January 1987.

[148] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123, August 1988.

[149] Y. F. Tsao and K. S. Fu. Stochastic skeleton modeling of objects. *Computer Vision, Graphics and Image Processing*, 25:348–370, 1984.

[150] A. M. Vossepoel. A note on "Distance transformations in digital images". *Computer Vision, Graphics and Image Processing*, 43:88–97, 1988.

[151] K. Wall and P.-E. Danielsson. A fast sequential method for polygonal approximation of digitized curves. *Computer Vision, Graphics and Image Processing*, 28:220–227, 1984.

[152] W. M. Wells III. Efficient synthesis of Gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):234–239, March 1986.

[153] J. S. Wiejak, H. Buxton, and B. F. Buxton. Convolution with separable masks for early image processing. *Computer Vision, Graphics, and Image Processing*, 32:279–290, 1985.

[154] H. R. Wilson and J. R. Bergen. A four mechanisms model for spatial vision. *Vision Research*, 19:19–32, 1979.

[155] A. P. Witkin. Scale-space filtering. In *Proceedings of the* 8^{th} *International Joint Conference on Artificial Intelligence*, pages 1019–1022, Karlsruhe, West Germany, August 1983.

[156] A. P. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. In *Proceedings of the* 5^{th} *National Conference on Artificial Intelligence*, pages 714–719, Philadelphia, Penn., U.S.A., 1986.

[157] Y. Xia. Skeletonization via the realization of the fire front's propagation and extinction in binary shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(10):1076–1086, October 1989.

[158] W. Xu and C. Wang. CGT: A fast thinning algorithm implemented on a sequential computer. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17(5):847–851, September 1987.

[159] H. Yamada. Complete Euclidean distance transformation by parallel operation. In *Proceedings of the* 7^{th} *International Conference on Pattern Recognition*, pages 336–338, Montréal, QC, Canada, July 1984. IEEE Computer Society Press.

[160] Q.-Z. Ye. The signed Euclidean distance transform and its applications. In *Proceedings of the* 9^{th} *International Conference on Pattern Recognition*, volume 1, pages 495–499, Rome, Italy, November 1988. IEEE Computer Society Press.

[161] Y. M. Youssef. *Quantification and Characterization of the Motion and Shape of a Moving Cell*. PhD thesis, McGill University, Electrical Engineering Department, Montréal, QC, Canada, May 1982.

[162] A. Yuille and M. Leyton. 3-D symmetry-curvature duality theorems. In *Proceedings of the* 1^{st} *International Conference on Computer Vision*, pages 721–726, London, England, June 1987. IEEE Computer Society Press.

[163] S. H. Zigmond, H. I. Levitsky, and B. J. Kreel. Cell polarity: An examination of its behavioral expression and its consequences for polymorphonuclear leukocyte chemotaxis. *Journal of Cell Biology*, 89:585–592, 1981.

[164] S. W. Zucker, C. David, A. Dobbins, and L. Iverson. The organization of curve detection: Coarse tangent fields and fine spline coverings. In *Proceedings of the* 2^{nd} *International Conference on Computer Vision*, pages 568–577, Tampa, Florida, U.S.A., December 1988. IEEE Computer Society Press.