UCL
Université
catholique
de Louvain

LABORATOIRE DE TELECOMMUNICATIONS ET TELEDETECTION

B - 1348 Louvain-la-Neuve Belgique

DISTANCE TRANSFORMATIONS: FAST ALGORITHMS AND APPLICATIONS TO MEDICAL IMAGE PROCESSING

Olivier CUISENAIRE

Thèse présentée en vue de l'obtention du grade de
Docteur en Sciences Appliquées

Jury composé de

Benoît Macq (UCL/TELE) - *Promoteur*
Auguste Laloux (UCL/TELE) - *Examinateur*
Roger Demeure (UCL/RDGN) - *Examinateur*
Christian Michel (UCL/TOPO) - *Examinateur*
Jan Cornelis (VUB - Brussel) - *Examinateur*
Ferran Marquès (UPC - Barcelona) - *Examinateur*
Charles Trullemans (UCL/DICE) - *Président*

Octobre 1999

Foreword

This book presents the results of my Ph.D. thesis, on which I worked for four years at the Communications and Remote Sensing Laboratory (TELE) of the UCL. This work was funded by a grant from the "Fonds pour la Formation à la Recherche dans l'Industrie et l'Agriculture" (FRIA).

For our laboratory, it represents the second Ph.D. thesis in the field of medical image processing, after Jean-Philippe Thiran's pioneering work, published in July 1997. My first thanks go to Jean-Philippe for starting this activity in TELE and opening the track for me and others.

The present study of distance transformations started as a side activity to the main research track of our medical image analysis group: the study of non-rigid registration methods for matching brain scans with anatomical atlases. Over the years, this side activity has taken more and more of my time as I came to realize the richness of the problem. I am profoundly grateful to Benoît Macq, my supervisor, for giving me the freedom to do so. Benoît has also been the one who convinced me to join the laboratory in 1995, and has been a constant support through the years.

In TELE, I have had the chance to collaborate very closely with Pierre Charbonnier, then with Matthieu Ferrant. Together we explored fascinating topics such as the template-based segmentation of brain structures using a brain atlas and active surfaces, or the volumetric finite element modeling of brain deformations. Working with Pierre and Matthieu has been both an inspiring and joyful experience. I must also thank all the other members of TELE for the great social and intellectual atmosphere that they contribute to create in our laboratory.

Beyond TELE, I have had the opportunity to interact with many other teams within the UCL, in particular with the Positron Tomography Laboratory (TOPO), the Neural Rehabilitation Engineering Laboratory (GREN), the Neuro-physiology Laboratory (NEFY) and the Radiology Unit at St-Luc Hospital (RDGN). I could not possibly list all the people in those teams, and therefore I will restrict myself to two, with all due excuses to the others.

Christian Michel (TOPO) has been a constant support ever since he co-supervised my master thesis in 1995. Although he is probably one of the hardest-working researchers at the UCL, he is always remarkably available for any scientific discussion. He has also provided me with many of the images I worked upon.

Eduardo Romero (GREN) has been the physician with whom I collaborated most closely. His enthusiasm to develop the automatic morphometry of nerve cross-sections has been the trigger that led to the variety of applications that are presented in this work. He has also provided me with all the data sets needed for chapter 4, and has made significant contributions to the validation part of that chapter.

Beyond the UCL, I wish to thank the people of the Surgical Planning Laboratory at the Brigham and Women's Hospital, Harvard Medical School, Boston, MA. They have welcomed me in their amazing research environment in March, July and August 1999. Ron Kikinis suggested that I apply my algorithms to virtual endoscopy. Simon Warfield supervised my work last summer and first suggested that I apply my knowledge of distance transformations to k-NN classification. My thanks go to them and many others at the SPL.

My thanks also go to Gunilla Borgefors, whose articles on chamfer distances are probably the most referred to in the field. Although we never met in person, she sent me an very nice encouraging email a few years ago, after reading one of my first papers. Similarly, Max Viergever was the chairman of the oral session where I presented my first paper at SPIE Medical Imaging 1996. In both cases, their generous encouragement was a great source of motivation for the apprentice researcher I was.

Of course, I must also thank the members of my jury for the in-depth

reading of this text and the various corrections they suggested.

Finally, I cannot fail to thank my parents, my family and my friends for their support throughout these years. In particular, I wish to thank Zsuzsanna, Karam, Cristi, Ovidiu, Nikos, Michelangelo, Laurent, Giuseppe, Paolo, Jonas, Jari, Guy, Yannis and Marit, friends from all over Europe that worked with me in the Summer Program and Educational Committees of the Board of European Students of Technology (BEST). I will always remember the time I spent with them as some of the best of my life.

Olivier Cuisenaire
October 1999

Contents

General Overview

Medical image processing is a demanding domain, both in terms of CPU and memory requirements. The volume of data to be processed is often large (a typical MRI dataset requires 10 MBytes) and many processing tools are only useful to the physician if they are available as real-time applications, i.e. if they run in a few seconds at most. Of course, a large part of these demands are - and will be - handled by the development of more powerful hardware. On the other hand, when faced with non-linear computational complexity, the development of improved algorithms is obviously the best solution. Distance transformations, a powerful image analysis tool used in a number of problems such as image registration, requires such improvements.

A **distance map** is an image where the value of each pixel is the distance from this pixel to the nearest pixel belonging to a given set or object. A **distance transformation** (DT) is an algorithm that computes a distance map from a binary image representing this set of pixels. This definition is *global* in the sense that it requires finding the minimum on a set of distances computed between all image pixels and all object pixels. Therefore, a direct application of the definition usually leads to an unacceptable computational complexity. Numerous algorithms have been proposed to localize this definition of distance to the nearest pixel and allow a faster DT computation, but up to now, none of them combines both exactness and linear complexity.

Numerous applications of distance transformations to image analysis and pattern recognition have been reported and those related to medical image processing are explored in what follows.

Chapter 1 introduces a few basic concepts, a typical application of distance transformations in pattern recognition and the key challenges in

producing a DT algorithm.

Chapter 2 contains an exhaustive critical review of published algorithms. The strong and weak points of the most popular ones are discussed and the core principles for our original algorithms are derived.

Chapters 3, 5, 6, 8 and 10 present original distance transformation algorithms. Each of those chapters is organized in a somewhat similar fashion. First we describe the algorithm. Then we evaluate its computational complexity and compare it to the state of the art. **Chapter 4, 7, 9 and 11** each present an application to a particular problem in medical image processing, using the algorithm developed in the previous chapter.

Ideally, the description of any medical image processing problem should include a medical justification of the need for an automated processing, a complete review of the state of the art in the field, a detailed description of the proposed processing method, and an evaluation of the accuracy of the results and their medical significance. Because of both time and space constraints in this thesis, such an exhaustive work will only be presented for the application in chapter 4, while the other applications will be described more briefly.

Chapter 3 describes a new exact Euclidean distance transformation using ordered propagation. It is based on a variation of Ragnelmam's [127] approximate Euclidean DT. We analyze the error patterns for approximate Euclidean DT using finite masks, and we derive a rule defining, for any pixel location, the size of the neighborhood that guarantees the exactness of the DT. This algorithm is particularly well-suited to implement mathematical morphology operations, which are examined in details.

In Chapter 4, we apply the algorithm of chapter 3 to the segmentation of neuronal fibers from microscopic images of the sciatic nerve. In particular, it is used to determine the thickness of the myelin sheath surrounding the center of the fiber. This study was carried out in collaboration with the Neural Rehabilitation Engineering Laboratory, UCL.

Chapter 5 proposes another exact Euclidean distance transformation, based on the explicit computation of the Voronoi division of the image.

Possible error locations are detected at the corners of the Voronoi polygons and corrected if needed. This algorithm is shown to be the fastest exact EDT to date. It approaches the theoretical optimal complexity, a CPU time proportional to the number of pixels on which the distance is computed.

Chapter 6 investigates how the algorithms of chapters 3 and 5 can be extended to 3 dimensional images. It shows the limitations of both approaches and proposes an hybrid algorithm mixing the method of chapter 5 and Saito's.

In **Chapter 7**, the 3D Euclidean DT is applied to the registration of MR images of the brain where the matching criterion is the distance between the surfaces of similar objects (skin, cortex, ventricular system, ...) in both images. Examples are shown, from projects with the Neuro-physiology Laboratory, UCL, and with the Positron Tomography Laboratory, UCL.

Chapter 8 discusses an extension of the distance transformation concept: geodesic distances on non-convex domains. Because geodesic distances are based on the notion of paths, a trade-off has to be introduced between the accuracy with which straight lines are represented and the way curves of the domain are followed. It is shown that, whatever the trade-off chosen, there is an efficient implementation of the geodesic DT by propagation.

By back-tracking the geodesic distance propagation, one can find the shortest path between a target and a starting point. In **chapter 9**, this is used to plan the optimal path for the camera movements in virtual endoscopy, a work done in collaboration with the Surgical Planning Laboratory, Harvard Medical School, Boston.

Chapter 10 extends the Euclidean distance transformation from finding the nearest object pixel to finding the k nearest object pixels. It is shown that this can be done with a complexity increasing linearly with k.

In **Chapter 11**, the k-DT is used as a fast implementation of the k Nearest Neighbors (k-NN) classification between different tissue types in multi-modal MR imaging. This is illustrated through the classifica-

tion of multiple sclerosis lesions from T1-T2 images, provided by the Radiology unit, St-Luc Hospital, UCL, via the Positron Tomography Laboratory, UCL.

Finally, a general **conclusion** is drawn. It reviews the main contributions of the thesis, its applications and explores some new domains in which their applications could also be useful. Ultimately, the publications related to this thesis are briefly reviewed.

Chapter 1

Introduction

This chapter introduces and illustrates the basic concepts of distance transformations. It describes a typical application and addresses some key issues for their implementation.

1.1 Basic concepts

The aim of a distance transformation is to compute the **distance from a point to an object**, i.e. to a set of pixels. As illustrated at figure 1.1 (left), the distance from point p to the object is the smallest distance from p to any point of the object. In other words, it is the distance from p to the nearest point q belonging to the object.

If one wants to know the distance from p to the object, one can apply the above definition, although it requires a large amount of computations when the object contains many points. When one needs to know this distance in several locations p, it is often faster to simply look it up in a **distance map** (figure 1.1), i.e. an image where the distance to the object has been pre-computed in all locations.

The **distance transformation** (DT) is the operation that computes the distance map from the binary image representing the object. The key point of this thesis is that it is possible - although not necessarily easy - to implement distance transformations efficiently.

There are several ways to define distances. In figure 1.2, the distance from p to the object is the distance from p to q. That means that the

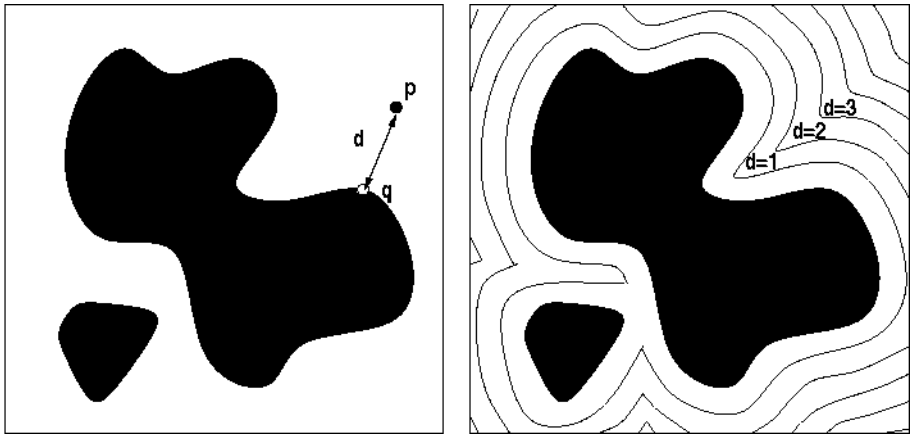

Figure 1.1: *Left:* distance from a point to an object. *Right:* distance map

value of the distance map D in p, or $D(p)$, is 5 when distances are computed with the usual Euclidean metric.

On the other hand, one could consider another **type of metric**. For instance, the city-block distance is defined as the shortest path with only vertical and horizontal steps. With that metric, $D(p)$ is the distance along the dashed line, i.e. $D(p) = |3| + |4| = 7$.

Another extension is what people call the "signed" distance. Instead of a simple scalar value, the **signed distance transformation** computes the relative location of p from its nearest object pixel q. In figure 1.2, the signed DT gives $SD(p) = p - q = (3, -4)$. It is of course possible to compute the value of the unsigned distance from the value of the signed distance, but not the opposite.

The **nearest neighbor transformation** computes, at every location p, the nearest pixel q in the object. In figure 1.2, we have $NN(p) = q = (11, 7)$. Computing the nearest neighbor or the signed distance transformations is of course equivalent, with $SD(p) = p - NN(p)$.

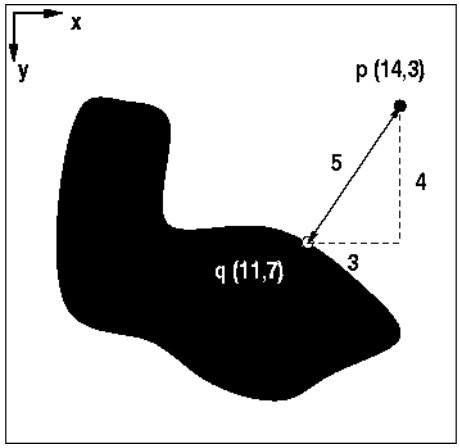

Figure 1.2: Unsigned DT, signed DT and Nearest neighbor transformation.

1.2 A typical application

The best known application of distance transformations comes from pattern recognition. It consists of looking for a specific object in a binary image including objects of various shapes, positions, orientations, ... It is often referred to as "chamfer matching", because it was first used with the chamfer DT, an approximation of the Euclidean metric.

Let us consider the example of figure 1.3. In the binary image, we are looking for the letter T. First, we compute the distance transformation and produce the distance map in the upper right corner. The pattern we are looking for - the T shape - is then moved over the relief defined by the distance map. Under the action of gravity, the pattern slides over the relief until it reaches the lowest possible altitude. If this altitude is zero or close to zero, we have found a matching pattern in the image.

More formally, the matching criterion is the correlation of the searched pattern with the distance map. The pattern is located where this correlation reaches an absolute minimum.

Theoretically, a more simple matching criterion could be used: the correlation of the pattern with the original binary image. Indeed, this criterion also has an absolute minimum at the correct location. Never-

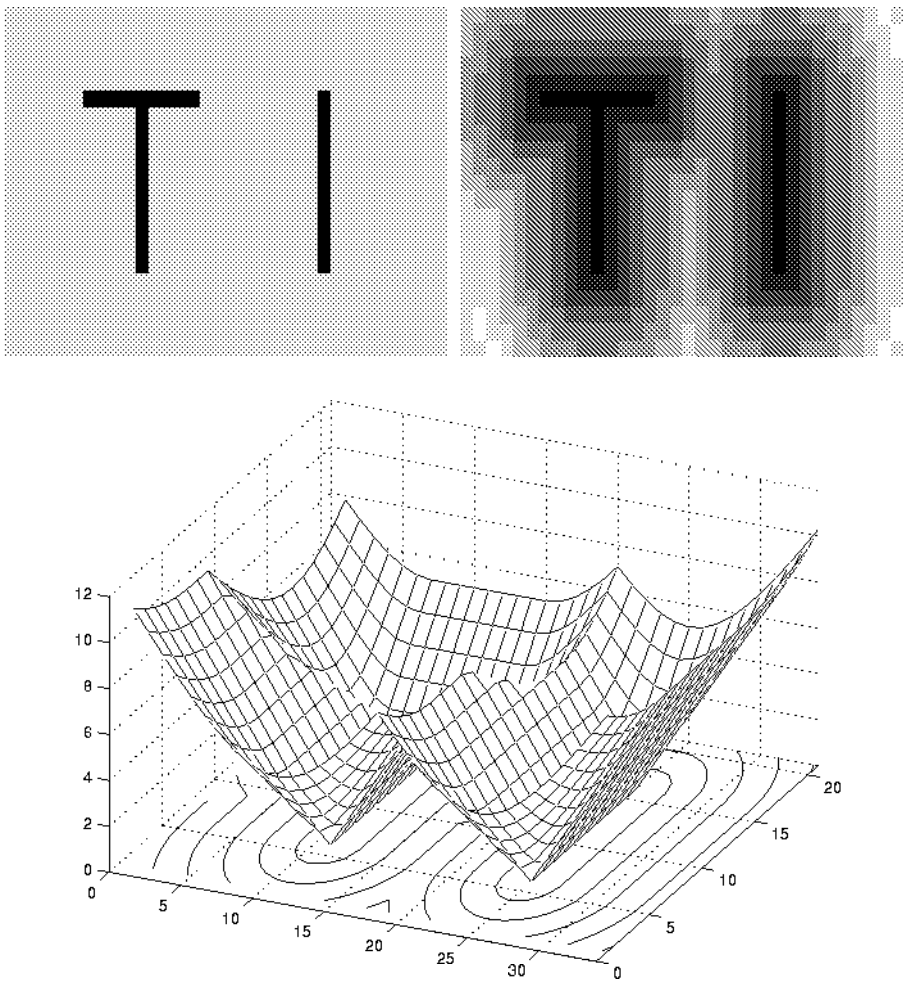

Figure 1.3: Chamfer matching: *Top-left:* original image. *Top-right* distance map. *Bottom:* distance map seen as a relief.

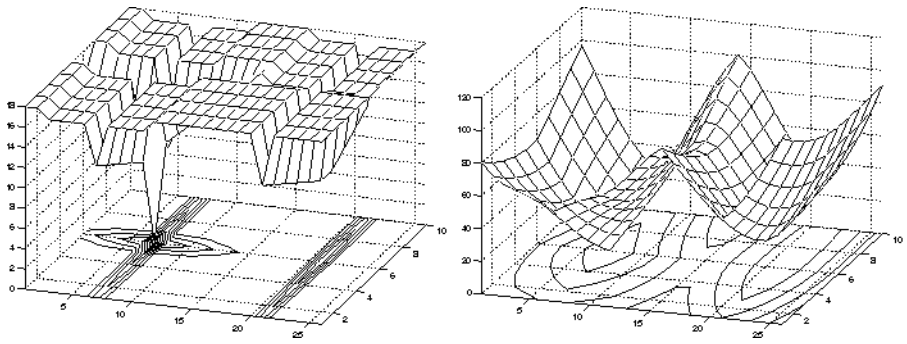

Figure 1.4: Matching criterion for the T shape. *Left:* convolution with the original image *Right:* Convolution with the distance map

theless, the use of the distance transformation brings a major practical improvement, illustrated at figure 1.4. The distance-based criterion has smooth and wide minima, which allows the use of fast minimization algorithms. On the other hand, the simple criterion only has very abrupt and narrow minima, requiring the complete set of possible positions to be searched.

There are many other types of applications of distance transformations. We review some in section 2.5, and present 4 medical imaging applications in chapters 4, 7, 9 and 11. The application of chapter 6 - the registration of medical images - is closely linked to chamfer matching.

1.3 Implementation issues

Let us now consider how distance transformations can be computed. A direct application - for every point - of the definition of the distance between a point and an object is obviously not practical. On the other hand, we know that the distances vary smoothly in the distance map, so that it must be possible to deduce the value of the map in one pixel from the values of the map around it. That is the fundamental idea used in all DT algorithms.

The natural, human way to proceed would be the following: starting from the object, we first determine the distance for the points that are in its neighborhood, then to these neighbors' neighbors and so on, as il-

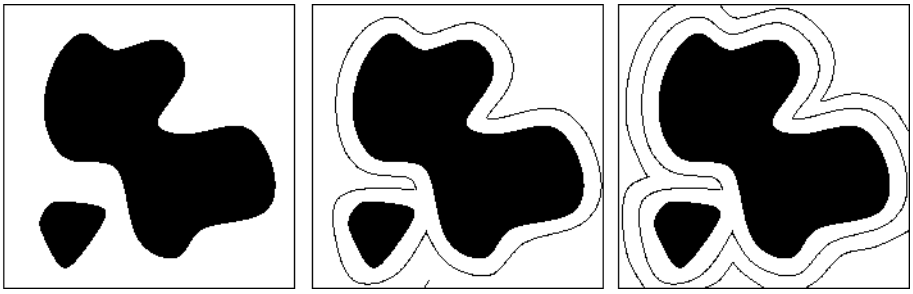

Figure 1.5: DT by propagation. Original image - after first step - after second step

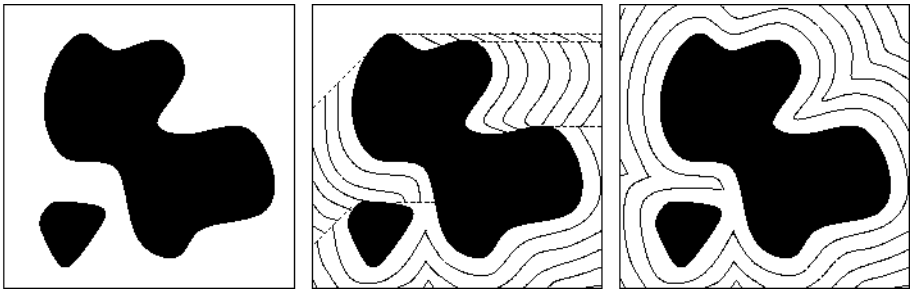

Figure 1.6: DT by raster scanning. Original image - after forward scan - after backward scan

lustrated at figure 1.5 for the two first steps of the propagation. Several algorithms reviewed in the next chapter (sections 2.2.2, 2.3.2 and 2.4.1), as well as those of chapters 3, 6, 8 and 10 try to mimic this human approach.

Generally speaking, computers are not gifted at mimicking humans. In particular, the above propagation process is not an easy thing to implement. Instead, several algorithms rely on the natural way for computers to scan the image: raster scanning. The image is processed row by row from the top-right pixel to the bottom-left pixel. Distance transformations by raster scanning require at least a forward scan (from top-right to bottom-left) and a backward scan (from bottom-left to top-right), as illustrated at figure 1.6. Algorithms of this type are reviewed in sections 2.2.1, 2.2.2 and 2.3.3. Also, the algorithm of chapter 5 uses raster scanning.

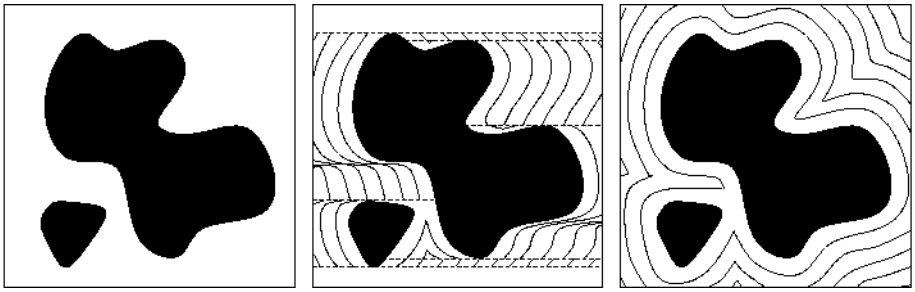

Figure 1.7: DT by independent scanning. Original image - after row-scan - after column-scan.

Finally, another computer-friendly method to scan the image is sometimes used. As seen at figure 1.7, the rows of the image are first considered independently from each other. Then, the image is considered column by column. Such algorithms are reviewed in sections 2.3.4 and 2.3.5.

Beside the order in which the image is scanned, the other key issue in implementing a distance transformation is the type of information that is propagated. On a continuous plane, as suggested by our examples so far, the distance itself would be a sufficient information. Some of the algorithms, reviewed at section 2.2.1, apply this simple idea to the discrete image grid. It produces distance transformations with approximate (non-Euclidean) metrics, called chamfer metrics.

Alternatively, one can propagate the signed distance, as in section 2.2.2. This allows the generation of distance maps that are quasi Euclidean. Surprisingly, these maps are not guaranteed to be exact in all locations. While on a continuous plane, the nearest object pixel of a point can always be deduced from the nearest object pixels of the points in its immediate neighborhood, this is not always true on a discrete grid.

This is the key reason why producing distance transformation on digital images is a complex problem. As shown at section 2.3, many solutions have been proposed, but none is optimal.

1.4 Aims of this thesis

The aims of this thesis are manifold. First, it proposes faster algorithms to produce exact Euclidean distance maps. Secondly, it explores extensions of the distance transformation concepts such as the geodesic distances of chapter 8 or the k-distance of chapter 10. Thirdly, it illustrates those methods by applying them to solve practical problems in medical image processing. Ultimately, it intends to present a global view on the powerful image analysis tools that are distance transformations, the methods to implement them and the cases where they can be successfully applied.

Chapter 2

A review of distance transformations

This chapter presents the framework of this thesis in more details. First, we define the main concepts and notations that will be used in this text. Secondly, we review most DT algorithms that were proposed in the literature. Those can be divided according to several criteria such as their accuracy, the order in which they scan the image, their computational complexity, etc. We choose to present them by order of increasing accuracy, from the coarser metrics introduced by Rosenfeld [132] to the exact Euclidean metric achieved by the latest algorithms. Thirdly we present extended DT concepts, such as the geodesic DT where the image domain is non-convex. Fourthly, we review the main applications of distance transformations, within and without the medical image processing framework. Finally, we discuss the main strong and weak points in the various algorithms and analyze which elements can be used to develop improved DT algorithms.

2.1 Definitions.

From a binary image I made of an object O and its background O', a distance transformation [132] makes an image, the distance map D (Fig. 2.1), in which the value of any pixel is the distance from this pixel to the object O, i.e. the distance to the nearest pixel of O.

$$D(p) = min\{dist_M(p,q), q \in O\} \tag{2.1}$$

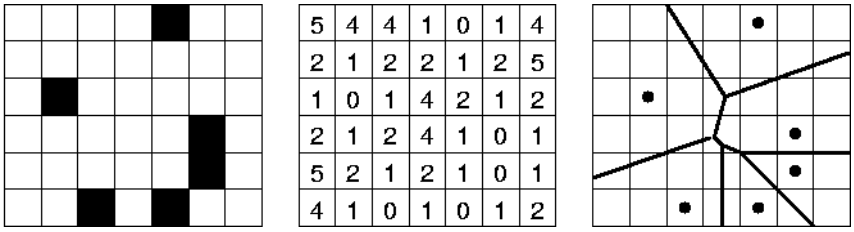

Figure 2.1: Example of a distance transformation using the $dist_E$ metric. *Left*: original image. *Center*: distance map. *Right*: Voronoi diagram.

The following notations are used in this text. Letters p, q, r are used for pixels, with indexes p_i where needed. In the original image, those pixels belong either to the object O or the background O' of the image. The coordinates of pixel p are (p_x, p_y). $dist_M(p, q)$ is the distance between pixels p and q using the metric M. The following metrics are considered:

$$dist_4(p,q) = |p_x - q_x| + |p_y - q_y| \tag{2.2}$$

$$dist_8(p,q) = \max\{|p_x - q_x|, |p_y - q_y|\} \tag{2.3}$$

$$\begin{aligned} dist_{cha(A:B)}(p,q) &= A.\max\{|p_x - q_x|, |p_y - q_y|\} + \\ &\quad (B - A).\min\{|p_x - q_x|, |p_y - q_y|\} \end{aligned} \tag{2.4}$$

$$dist_e(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{2.5}$$

$$dist_E(p,q) = (p_x - q_x)^2 + (p_y - q_y)^2 \tag{2.6}$$

They are called the city block (2.2), chess board (2.3), chamfer (2.4), Euclidean (2.5) and squared Euclidean metric (2.6), respectively. In what follows, when we speak of a Euclidean distance transformation, we usually mean a DT computed using the $dist_E$ metric. Indeed, it is equivalent to $dist_e$ but can be computed using integer operations only since pixels are located on integer locations.

In the above definitions, the only relevant information is of course the relative location of pixels p and q, i.e. $dp = p - q$. Therefore we often used the simpler notation

$$dist_M(dp) = dist_M(p, q) \tag{2.7}$$

For instance, we have

$$dist_E(dp) = dp_x^2 + dp_y^2 \tag{2.8}$$

An important related concept is that of the Voronoi diagram. It divides the plane into tiles surrounding each object pixel. $VP(p)$ is the Voronoi Polygon surrounding a pixel p of the object. For every pixel $p \in O$, $VP(p)$ is the part of image I that satisfies

$$VP(p) = \{q \in I \mid \forall r \in O\,,\, dist_M(p, q) \leq dist_M(q, r)\} \tag{2.9}$$

Finally, image processing operations are usually defined locally, i.e. pixels are only influenced by their neighborhood. We write $N(p)$ the set of neighbors of p, i.e.

$$N(p) = \{q = p + n \mid n \in N\} \tag{2.10}$$

$N = N((0,0))$ is called a neighborhood. The neighborhoods considered here are balls, i.e. neighborhoods such that

$$N = B_d = \{n \mid dist_M(n, (0,0)) < d\} \tag{2.11}$$

with some metric M. Of particular interest are the 4-direct and 8-direct neighborhoods defined as

$$\begin{aligned} N_4 &= \{n \mid dist_4(n, (0,0)) = 1\} \\ &= \{(1,0), (-1,0), (0,1), (0,-1)\} \end{aligned} \tag{2.12}$$

$$\begin{aligned} N_8 &= \{n \mid dist_8(n, (0,0)) = 1\} \\ &= N_4 \cup \{(1,1), (1,-1), (-1,1), (-1,-1)\} \end{aligned} \tag{2.13}$$

2.2 Approximate distance transformations.

As defined in equation 2.1, distance transformations are *global* transformations. Indeed, a direct application of this definition requires to consider all object pixels in order to compute the value of the DT for any non-object pixel. The computational complexity is proportional to

the product of the numbers of pixels in O and O'.

In image processing, one usually tries to consider *local* transformations, i.e. transformations for which the value of a pixel depends only on the pixels belonging to its neighborhood. The simplest method to localize the definition of eq. 2.1 is to assume that the DT at a pixel can be deduced from the values at its neighbors. This is strictly true for the city-block, chess-board and chamfer metrics, but not for the Euclidean one. It leads to a class of algorithms that Borgefors [10] called chamfering. Unfortunately, chamfer metrics are only coarse approximations of the Euclidean distance. Chamfer DTs produce systematic errors in the directions not covered by the chamfer masks. This is studied in section 2.2.1.

In order to define a local Euclidean DT algorithm, Danielsson [37] proposes to assume that the nearest object pixel (NOP) is a local property. The NOP of a background pixel is the same as that of one its neighbors. With the Euclidean metric, this assumption is true on a continuous plane, but sometimes incorrect on a discrete grid, with particular object pixels configurations. This leads to a quasi Euclidean DT where non-systematic errors occur on some locations of the image only. This is studied in section 2.2.2.

2.2.1 Chamfering.

Chamfer distance transformations are relying on the assumption that it is possible to deduce the value of the distance at a pixel from the value of the distance at its neighbors. This assumption is correct for *regular* metrics [131], i.e. metrics for which

For all p, q such that $dist_M(p,q) \leq 2$, there exists an r different from p and q such that $dist_M(p,q) = dist_M(p,r) + dist_M(r,q)$.

This property is true for the city-block (2.2), chess-board (2.3) and chamfer (2.4) metrics - among others - but not for the Euclidean one when pixels p, q, r are restricted to locations on an integer grid.

Chamfer DTs were developed and studied by Rosenfeld and Pfaltz [132, 133], Montanari [109], Barrow [3], Borgefors [10, 12], Piper and Granum

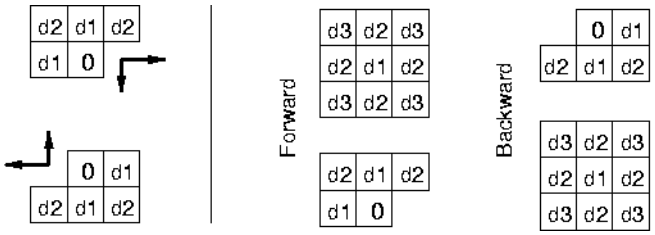

Figure 2.2: Masks used by chamfer DT algorithms, in 2 (left) and 3 (right) dimensions

[122], Forchhammer [56], Verwer, Verbeek and Dekker [167], Sharaiha and Christofides [145] and Coquin and Bolon [22].

Rosenfeld [132, 133] proposes raster scanning (see below) and independent scanning (see section 2.3.4) algorithms for the distance transformation using the $dist_4$ and $dist_8$ metrics. **Montanari** [109] investigates distances defined as the length of the shortest path between two pixels. If the path is restricted to steps between 4-direct neighbors, this is equivalent to the $dist_4$ metric. With 8-direct neighborhoods, it defines a $dist_{cha(1:\sqrt{2})}$ metric. Montanari proves that the shortest path between two points consists of at most two types of steps, those with orientations closest to the Euclidean vector between those points. This property is central to the later publications by Borgefors [10] and Ragnelmam [127]. **Barrow** [3] uses a DT similar to Montanari's, replacing $(1 : \sqrt{2})$ by $(2 : 3)$, an integer approximation.

In [10], **Borgefors** reviews a number of metrics in 2 and 3 dimensions. Chamfer distance transformations are produced in two raster scans over the image, using the masks of Figure 2.2. In the forward scan, the mask starts in the upper left corner of the picture, moves from left to right and from top to bottom. In the backward scan, it starts in the lower right corner, moves from right to left and from bottom to top. The local distances, d_1 and d_2, in the mask pixels are added to the pixel values in the distance map and the new value of the zero pixel is the minimum of the five sums.

This algorithm can produce distance transformation with several metrics. For instance, for $d_1 = 1$ and $d_2 = \infty$, we have the city block metric; for $d_1 = d_2 = 1$, the chess board metric, for $d_1 = 1$ and $d_2 = \sqrt{2}$,

Montanari's metric and for $d_1 = 2$ and $d_2 = 3$, Barrow's approximation. Borgefors computes the optimal values for d_1 and d_2 in order to minimize the maximal difference between the chamfer metric and the Euclidean one. Assuming that $d_2 < 2\,d_1$ (otherwise the diagonal direction is never used), the distance between two pixels is

$$dist_{cha(d_1:d_2)}(p,q) = m_2.d_2 + (m_1 - m_2).d_1 \tag{2.14}$$

where $m_1 = |q_x - p_x|$ and $m_2 = |q_y - p_y|$, using Borgefors' notations, and $m_1 \geq m_2$ (otherwise, m_1 and m_2 change places in (2.14)). The difference between the Euclidean and chamfer distance is thus

$$\sqrt{m_1^2 + m_2^2} - m_2.d_2 - (m_1 - m_2).d_1 \tag{2.15}$$

If we set $d_1 = 1$ and $d_2 = d < 2$, the optimal d can be found by minimizing the maximum of (2.15). The maximum occurs either for $m_2 = 0$, $m_2 = m_1$ or the value of m_2 where the differential is 0. For $m_2 = 0$, (2.15) is zero. For $m_2 = m_1$, (2.15) becomes

$$(\sqrt{2} - d)M \tag{2.16}$$

where $M = m_1 = m_2$. The differential of (2.15) with respect to m_2 is zero for $m_2 = ((d-1)/\sqrt{2d - d^2}).m_1$. Substituting this value in (2.15), we find

$$(\sqrt{2d - d^2} - 1)M \tag{2.17}$$

where $M = m_1$. The minimum of the absolute maximum of 2.16 and 2.17 happens when they cross each other at $d = 1/\sqrt{2} + \sqrt{\sqrt{2} - 1} \approx 1.351$. The upper limit for (2.15) is then $(1/\sqrt{2} - \sqrt{\sqrt{2} - 1}).M \approx 0.06M$. Slightly better results could actually be obtained by removing the constraint that $d_1 = 1$, as in [166], but this is beyond the scope of the present review.

For computational efficiency, floating point operations are not desirable, and therefore Borgefors suggests to use the sub–optimal integer approximation $d_1 = 3$ and $d_2 = 4$, then to divide the resulting DT by 3. In this case, the upper limit for the difference between the Euclidean and chamfer metric becomes $\sqrt{2} - 4/3 \approx 0.08M$. This is obviously much better than the limits found for the city-block and chess-board metrics,

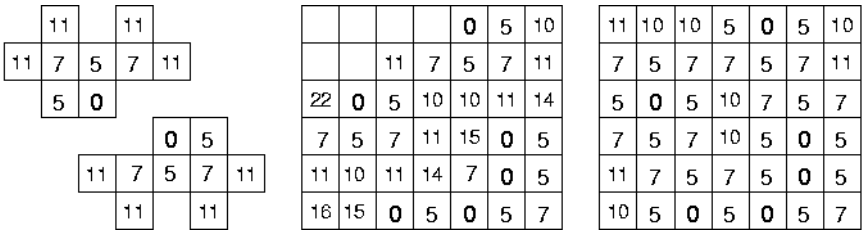

Figure 2.3: The chamfer (5:7:11) DT. *Left*: forward and backward masks. *Center*: result after forward scan applied on the original image of figure 2.1. *Right*: final result.

$(\sqrt{2}-2)M \approx -0.59M$ and $(\sqrt{2}-1)M \approx 0.41M$, respectively.

In 3 dimensions, the optimal values are $d_1 = 1$, $d_2 = (\sqrt{3}+1+\sqrt{2\sqrt{3}-2})/3 \approx 1.314$ and $d_3 = (2\sqrt{3}-1+2\sqrt{2\sqrt{3}-2})/3 \approx 1.628$, which gives $((\sqrt{3}+1-2\sqrt{2\sqrt{3}-2})/3)M \approx 0.10M$ for the upper limit of the absolute difference between the metrics. Once again, it is often better to consider a sub–optimal integer approximation with $d_1 = 3$, $d_2 = 4$ and $d_3 = 5$. The upper limit of the difference with the Euclidean metric is then $(\sqrt{7}/3-1)M \approx -0.12M$.

In [12], **Borgefors** extends the chamfer masks to 5×5 and 7×7. She computes the optimal values for the local distances and evaluates several integer approximations for 3×3 to 7×7 masks. She recommends using (3:4) and (5:7:11) approximations for 3×3 and 5×5 masks, but finds no significant interest in using 7×7 masks. The gain in precision becomes negligible compared to the additional cost of using larger masks. With chamfer (5:7:11), the upper limit for the absolute difference between this metric and the Euclidean one is reduced to $0.02M$. The chamfer (5:7:11) DT is illustrated at Figure 2.3.

Piper and **Granum** [122] and **Verwer** et al. [167] propose to reach the same results with a different type of scanning. Instead of two raster scans, they use a propagation from the object pixels to the rest of the image. This proves to be more efficient on non-convex image domains, and will be further explained in section 2.4.1.

Sharaiha and **Christofides** [145] propose a graph theoretic approach to

chamfer distance transformation. They represent the image as a graph where pixels are represented as vertices and adjacency relations as arcs. The cost associated with each arc is equal to the equivalent local distance in the chamfer masks. The vertices associated to the object pixels are called root vertices. The Chamfer DT problem is then equivalent to the shortest path forest problem in graph theory. They solve this using a variation of the algorithms proposed by Moore [110] and Dial [39]. Practically, this is similar to Piper's approach [122], with a more complex data representation.

In [56], **Forchhammer** shows that, within a limited distance, the true Euclidean distance can be deduced from its chamfer approximation. In particular, using the 3×3 mask with (3:4) local distances, the exact EDT can be deduced from the chamfer distances for $D(p) < \sqrt{17}$, but not beyond.

Finally, **Bolon** [9] and **Coquin** [22] extend the chamfer metrics to anisotropic grids. The optimal mask coefficients are found by minimizing the maximum of the difference between the anisotropic chamfer metric and the Euclidean along a circle. The complete solution is given for 5×5 anisotropic masks, but can be applied to other masks as well.

2.2.2 Vector propagation.

In order to better approximate the Euclidean distance transformation, **Danielsson** [37] proposes to propagate more information than just the distance. Instead, he uses a two-component descriptor: $|p_x - q_x|, |p_y - q_y|$ (also written $|dp_x|, |dp_y|$), the absolute values of the relative coordinates of the nearest object pixel. The Euclidean distance can of course be deduced from this information using (2.5).

The algorithm is similar to the chamfer DT algorithm described in section 2.2.1, with one major change. During the process, the descriptors need to propagate in all directions. Unfortunately, raster scanning with masks such as those of Figure 2.2 only provides a 135^o propagation angle. Danielsson opens this angle up to 180^o by modifying the raster scanning procedure. At each step in the vertical direction, the row is scanned both from left to right and from right to left. The masks considered are then those of Figure 2.4.

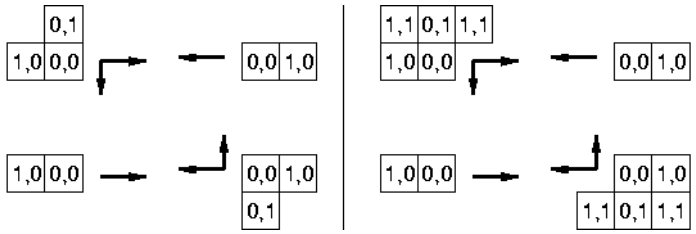

Figure 2.4: Masks used in Danielsson's 4SED (left) and 8SED (right) algorithms

On the negative side, this back and forth scanning requires more computations. This adds up to the extra computational cost of comparing descriptors - which requires the evaluation of (2.5) - instead of scalar distance values in the chamfer DT. On the positive side, it allows us to chose the smaller 4-direct neighborhood instead of the 8-direct one. Danielsson names the respective algorithms "FOUR-point Sequential Euclidean Distance mapping" or 4SED, and "EIGHT-point Sequential Euclidean Distance mapping" or 8SED.

Unfortunately, these algorithms do not always provide the exact Euclidean DT. At Figure 2.5 (left), pixel q gets $dist_e = \sqrt{3^2+0^2} = 3$ instead of $dist_e = \sqrt{2^2+2^2} = \sqrt{8}$. Indeed, 4SED only allows propagation from a pixel to its 4-direct neighbors. In this case, none of the 4-direct neighbors of the erroneous pixel had the same nearest object pixel. This particular error would have been avoided by using the 8SED algorithm instead. But even 8SED isn't completely error-free, as illustrated at Figure 2.5 (right), pixel q gets $dist_e = \sqrt{170}$ while it should have received $\sqrt{169}$ instead. Actually, whatever the size of the neighborhood considered, it will be possible to find object pixel configurations where errors do occur. This is analyzed thoroughly at section 3.2.

Nonetheless, the 4SED and 8SED are very good approximations of the Euclidean DT. First, they provide exact results for most pixels. Secondly, Danielsson proves that, for 4SED, the maximum absolute error is at most 0.29 pixel units. The maximum relative error is that illustrated at Figure 2.5, i.e. $(3-\sqrt{8})/\sqrt{8} = 6.1\%$. With 8SED, the absolute error is bounded by 0.09 pixel units and the maximum relative error is that

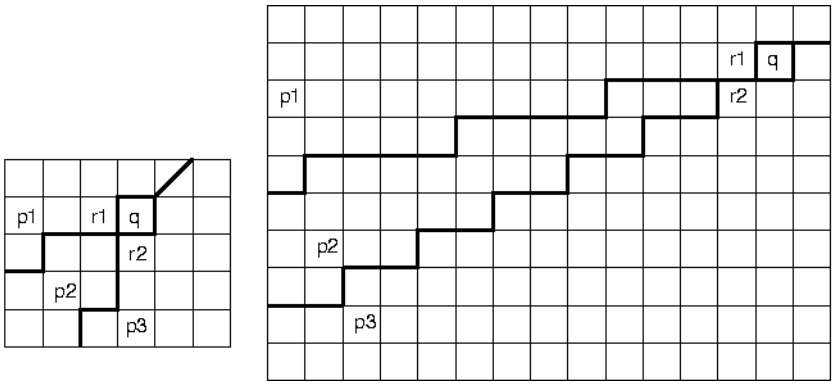

Figure 2.5: Errors made by Danielsson's 4SED (left) and 8SED (right) algorithms. Object pixel p_2 is hidden from pixel q by object pixels p_1 and p_3, that are closer to r_1 and r_2, respectively. *Left:* $q - p_1 = (3,0)$, $q - p_2 = (2,2)$, $q - p_3 = (0,3)$. *Right:* $q - p_1 = (13,1)$, $q - p_2 = (12,5)$, $q - p_3 = (11,7)$

of Figure 2.5, i.e. $(\sqrt{170} - 13)/13 = 0.3\%$. [1].

Ye [186] introduces 4SSED and 8SSED, similar algorithms producing the *signed* Euclidean distance transformations. This is done by propagating $(p_x - q_x, p_y - q_y)$ instead of their absolute values. This is achieved by replacing the masks of Figure 2.4 by signed masks, i.e. masks where the x-coordinate is negative on the left column, and y-coordinate is negative on the upper row.

Leymarie [95] shows that Danielsson's algorithm, with some minor changes, can be implemented as efficiently as chamfer DT algorithms. First, he uses the $dist_E$ metric for all comparisons made in the algorithm, so that only integer operations are needed. Secondly, he stores explicitly 3 values for each pixel, i.e. the 2 relative coordinates (dp_x, dp_y) of the nearest object pixels and the value of the $dist_E$ distance. With this information, the distance computations are simplified. Knowing the value of $dist_E(dp)$, we have

$$dist_E(dp + (\pm 1, 0)) \quad = \quad (dp_x \pm 1)^2 + dp_y^2$$

[1] In [37], Danielsson overlooks the existence of this error and announces the maximal relative error to be 0.15%

$$
\begin{aligned}
& = dist_E(dp) \pm 2.dp_x + 1 \\
dist_E(dp + (0, \pm 1)) & = dp_x^2 + (dp_y \pm 1)^2 \\
& = dist_E(dp) \pm 2.dp_y + 1 \\
dist_E(dp + (\pm 1, \pm 1)) & = (dp_x \pm 1)^2 + (dp_y \pm 1)^2 \\
& = dist_E(dp) \pm 2.(dp_x + dp_y \pm 1) \\
dist_E(dp + (\pm 1, \mp 1)) & = (dp_x \pm 1)^2 + (dp_y \pm 1)^2 \\
& = dist_E(dp) \pm 2.(dp_x - dp_y \pm 1) \quad (2.18)
\end{aligned}
$$

This reduces the computational cost of a distance to one addition, one shift ($\times 2$) and one increment, instead of two multiplications, one addition and one square root operation in (2.5). Therefore, the computational complexity of the 4SED and 8SED algorithms becomes comparable to that of the chamfer approximations.

Ragnelmam [127, 126] adapts the works of Piper [122] and Verwer [167] to the Euclidean DT. Instead of raster scanning, he uses ordered propagation (a.k.a. contour-processing) where the image pixels are considered by increasing distance values. In [126], this is done by bucket sorting with as many buckets as possible distance values. In [127], he uses ordered propagation by thresholding, where all pixels in the propagation front are stored in the same dynamic list, but are only propagated if their value is below the current distance.

In contrast with raster scanning algorithms, the propagation DT uses a "write" formalism instead of a "read" formalism. In the raster scanning method, the basic operation is the updating of the value one pixel based on the information coming from several neighbors. In the propagation method, the basic operation is the updating of all neighboring pixels based on the value of the one propagated pixel.

The additional computational cost of the dynamic data structure needed to implement ordered propagation is compensated by two factors. First, using Danielsson's back and forth raster scanning, many pixels are *updated* several times, up to 4 times. Ideally, pixels only need to be updated once, when they get their final value. Ordered propagation approaches this behaviour because pixels are only *propagated* once. Secondly, the information from a pixel only needs to be propagated in the direction of increasing distances and not backward, which reduces the size of the neighborhoods used. Adapting Montanari's theorem [109] (section 2.2.1)

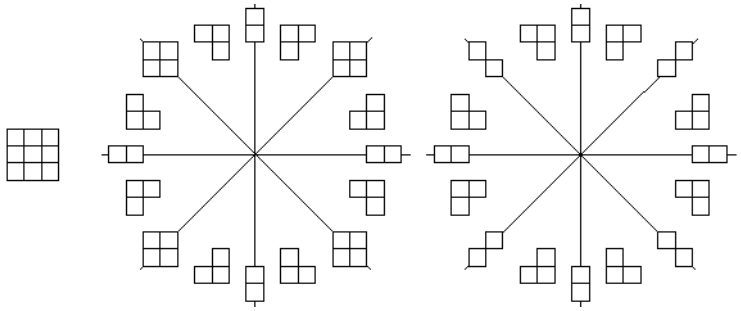

Figure 2.6: Directional neighborhoods used by Ragnelmam's ordered propagation algorithm. *Left:* for $D(p) = 0$. *Center:* for $D(p) = 1$. *Right:* for $D(p) > 1$.

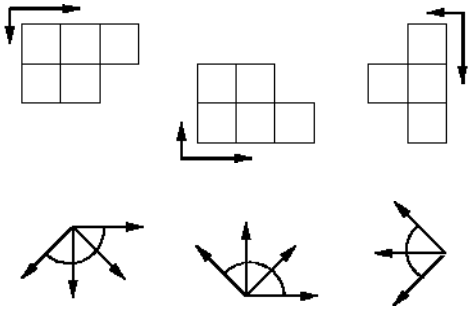

Figure 2.7: Ragnelmam's 3-scan 8SSED. *Top*: masks used. Bottom: supported propagation directions.

to Euclidean metrics, Ragnelmam shows that, apart from the two first iterations, only one or two neighbors need to be considered for propagation, as illustrated at Figure 2.6.

In [128], **Ragnelmam** shows that the 8SSED algorithm can also be implemented with separable raster scans, i.e. without the back and forth scan at each line suggested by Danielsson. Each scan, with a specific scanning order and a given mask, can support the propagation of information within some part of the direction space. Any SSED algorithm should include enough scans to support propagation in all directions. In 2 dimensions, it is possible to produce the approximate EDT in 3 raster scans only, with the masks and scanning orders illustrated at Figure 2.7. In three dimensions, 4 scans are sufficient with the masks of Figure 2.8. In n dimensions, algorithms with a minimal number of scans are harder

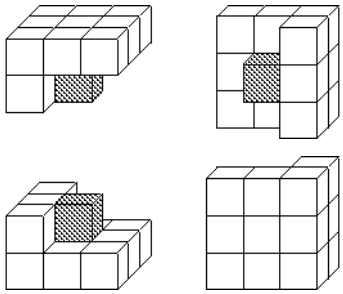

Figure 2.8: Masks for Ragnelmam's 4-scan 26SSED.

to design. Instead, he recommends to use "corner" masks, i.e. masks including half of the 2n-direct neighbors, 1 out of 2 for each direction. He shows that the approximate (2n)SSED algorithm can always be generated in 2^n raster scans over the image [2].

Finally, **Embrechts** and **Roose** [44] show how the 4SSED algorithm can be implemented efficiently on multi-processor computers. Each processor applies the 4SSED on a sub-region of the image. Before that, communications between the processors determine the list of object pixels that influence the sub-region although they are located in another one. The parallelization procedure itself does not introduce any error in Euclidean DT. The only reason why Embrechts' algorithm is approximate is that the local algorithm used by each processor is the approximate 4SSED. Therefore, this parallelization procedure will be further studied in section 2.3.5 together with other transformations based on the Voronoi diagram.

2.3 Exact Euclidean distance transformations.

Many different approaches are possible to compute exact Euclidean DT, i.e. without the errors made by the 4SSED and 8SSED algorithms. Historically, the first idea was to consider distance transformations as the result of a filtering process [185, 147, 77, 41] (sec. 2.3.1). A 3×3 mask is applied repeatedly on all image pixels until a stable solution is reached. Unfortunately, this can only be implemented efficiently on a

[2] For arbitrary dimensions, it is actually both easier and more efficient to extend Ragnelmam's ordered propagation algorithm [127] than to use his corner EDT [128]

1,1	0,1	1,1
1,0	0,0	1,0
1,1	0,1	1,1

			1,0	0,0	1,0	
1,1	0,1	1,1	1,1	0,1	1,1	
1,0	0,0	1,0		1,1	0,1	1,1
1,1	0,1	1,1		1,0	0,0	1,0
	1,1	0,1	1,1	1,0	0,0	1,0
	1,0	0,0	1,0	0,0	1,0	1,1

1,2	0,2	2,0	1,0	0,0	1,0	2,0
1,1	0,1	1,1	1,1	0,1	1,1	2,1
1,0	0,0	1,0	2,0	1,1	0,1	1,1
1,1	0,1	1,1	2,0	1,0	0,0	1,0
2,1	1,1	0,1	1,1	1,0	0,0	1,0
2,0	1,0	0,0	1,0	0,0	1,0	1,1

Figure 2.9: Yamada's EDT. *Left*: the 3x3 mask. *Center*: result after 1 step. *Right*: result after two steps.

parallel processing array.

A second approach modifies the approximate algorithms in order to incorporate the useful mechanisms of the parallel processing methods above, while keeping a reasonable computational cost. This leads to region-growing [168, 127, 42] (sec. 2.3.2) and raster scanning [112, 146] (sec. 2.3.3) algorithms.

A third approach extends an algorithm originally proposed by Rosenfeld [132] for coarser metrics, where columns and rows are scanned independently [118, 140] (sec. 2.3.4). Finally, a last approach is based on the explicit computation of the Voronoi diagram of the object pixels in the continuous plane [17, 69, 44] (sec. 2.3.5).

2.3.1 Parallel processing.

Yamada [185] combines Danielsson's 8SED masks (Fig. 2.4) into the single 3×3 mask of Figure 2.9. This masks is applied iteratively over the whole image. The $|dp_x|^t, |dp_y|^t$ descriptor of a pixel at iteration t is computed from iteration $t-1$ by adding the mask values to the descriptors of the 9 pixels covered by the mask and choosing the one that gives the minimum Euclidean distance. This process is iterated until no pixels change anymore.

Yamada proves that this algorithm is an exact Euclidean DT. Indeed, the information is propagated with 8-direct neighborhoods, and respects the order defined by the $dist_{chess}$ metric. Thus, in Figure 2.5, pixel r_1 is reached by the propagation front from p_2 at least one step earlier than by the propagation front from p_1. More generally, one can prove that

the propagation, from an object pixel p to any of the pixels q in its zone of influence, is always possible along the shortest path made of diagonal steps only, followed by vertical or horizontal steps only.

In [147], **Shih** and **Mitchell** consider distance transformations as a gray-scale mathematical morphology operation. From a binary image f where object pixels have the value 0 and non-object pixels the value $+\infty$, they compute the gray-scale distance map $g = f \ominus k$, by eroding f with a structuring element k at least as large as the maximum distance in the image. The weights of this structuring element are the negative of their distance to the center.

Handling such a large structuring element is of course inefficient. Thus, they decompose it into smaller 3×3 elements as follows:

$$k_{(2n+1\times 2n+1)} = \max\{k_{1(3\times 3)}, k_{2(5\times 5)}, \ldots, k_{n(2n+1\times 2n+1)}\} \quad (2.19)$$

$$k_{i(2i+1\times 2i+1)} = k_{i1(3\times 3)} \oplus k_{i2(3\times 3)} \oplus \ldots \oplus k_{ii(3\times 3)} \quad (2.20)$$

For instance, with $n = 2$,

$$k_{(5\times 5)} = \max\left\{ \begin{matrix} a_1 & a_0 & a_1 \\ a_0 & 0 & a_0 \\ a_1 & a_0 & a_1 \end{matrix} , \begin{matrix} b_2 & b_1 & b_0 & b_1 & b_2 \\ b_1 & x & x & x & b_1 \\ b_0 & x & x & x & b_0 \\ b_1 & x & x & x & b_1 \\ b_2 & b_1 & b_0 & b_1 & b_2 \end{matrix} \right\} \quad (2.21)$$

$$\begin{matrix} b_2 & b_1 & b_0 & b_1 & b_2 \\ b_1 & x & x & x & b_1 \\ b_0 & x & x & x & b_0 \\ b_1 & x & x & x & b_1 \\ b_2 & b_1 & b_0 & b_1 & b_2 \end{matrix} = \begin{matrix} b_2 & b_1 & b_2 \\ b_1 & x & b_1 \\ b_2 & b_1 & b_2 \end{matrix} \oplus \begin{matrix} 0 & b_0 - b_1 & 0 \\ b_0 - b_1 & x & b_0 - b_1 \\ 0 & b_0 - b_1 & 0 \end{matrix} \quad (2.22)$$

where the value of x does not matter, $a_0 = -1$, $a_1 = -\sqrt{2}$, $b_0 = -2$, $b_1 = -\sqrt{5}$ and $b_2 = -\sqrt{8}$. In all $k_{i(2i+1\times 2i+1)}$ structuring elements, only the outer weights matter. In all $k_{ij(3\times 3)}$ structuring elements, the diagonal weights are zero.

With this method, the implementation of a $k_{(2n+1\times 2n+1)}$ elements requires $n(n+1)/2$ gray scale erosions. Therefore, it can only be reasonably efficiently implemented on specialized parallel pipelined VLSI circuits such as in [1].

With **Huang** [77], **Mitchell** adapts his above method to the $dist_E$ squared Euclidean metric. The decomposition of the structuring element becomes much simpler.

$$k_{(2n+1\times 2n+1)} = k_{1(3\times 3)} \oplus k_{2(3\times 3)} \oplus ... \oplus k_{n(3\times 3)} \tag{2.23}$$

$$k_{i(3\times 3)} = \begin{matrix} -4i+2 & -2i+1 & -4i+2 \\ -2i+1 & 0 & -2i+1 \\ -4i+2 & -2i+1 & -4i+2 \end{matrix} \tag{2.24}$$

This only requires n erosions to implement a Euclidean distance transformation that is exact up to distance n. The computational cost is similar to Yamada's method.

Finally, in [41], **Eggers** proves that both Huang's and Yamada's parallel DTs can successfully be extended to anisotropic grids and to higher dimensions.

2.3.2 Sequential processing by propagation.

In the above parallel methods, a lot of computational power is wasted because, at each iteration, only a small fraction of the processed pixels actually change values. Ragnelmam [127] and Eggers [42] implement the above methods efficiently by storing the pixels in the propagation front in a dynamic list. Vincent [168] proposes an alternative approach where the propagation front is considered as chain.

In the same paper where he describes the approximate Euclidean DT by ordered propagation, **Ragnelmam** [127] proposes an efficient implementation of Yamada's parallel algorithm. First, object pixels are put in a dynamic list. Then, Yamada's mask is applied only to the pixels taken from that list. Any neighbor that changes value is itself put into the list for next iteration. Also, the updating of pixel values is delayed until the

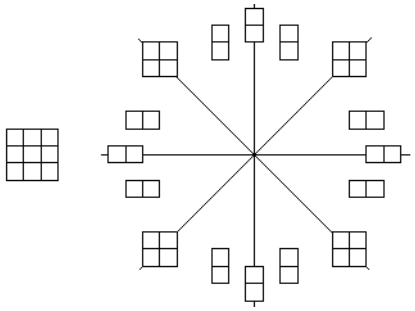

Figure 2.10: Eggers' sufficient propagation neighborhoods. *Left:* for $D(p) = 0$. *Right:* for $D(p) > 0$

next iteration, so that all pixels seem to be processed simultaneously as in purely parallel processing.

Because it only processes the pixels in the propagation front, and because it also restricts the neighbors it considers to those of Figure 2.6, this implementation is obviously much faster that Yamada's. For instance, for a single object pixel placed in the middle of an $n \times n$ image, the computational complexity becomes $o(n^2)$ instead of $o(n^3)$. Unfortunately, there is not always such a gain in complexity. In the approximate algorithm, ordered propagation guaranteed that pixels did not propagate more than once. This time, the propagation is ordered with the $dist_{chess}$ metric, but not with $dist_e$ anymore. Therefore, propagation fronts may follow each other, and some pixels may be updated many times, up to once per object pixel in the worst case configuration, a oblique line of object pixels. This brings us back to a $o(n^3)$ complexity.

Eggers [42] adapts Huang's parallel algorithm in a similar way. His implementation turns out to be even faster than Ragnelmam's for two reasons. First, Huang's masks (eq. 2.24) provide a very efficient way to compute distances, similar to Leymarie's improvement of Danielsson's algorithms. Secondly, Eggers notices that, for parallel algorithms, one only needs to support propagation along the path made of diagonal steps only, followed by horizontal or vertical steps only. Therefore, he uses the neighborhoods of Figure 2.10. He calls this sufficient propagation.

On the other hand, the propagation order is exactly the same as before, so that there is still a $o(n^3)$ complexity for worst case $n \times n$ images.

Finally, **Vincent** [168] - using his experience of the efficient implementation of mathematical morphology operators [169] - proposes to represent the borders of the object as chains, and to propagate these chains by applying dilations repeatedly. The elements of the chains remember the location of their nearest object pixels, so that the exact Euclidean distance can be computed. In order to avoid errors as in Figure 2.5, the propagation isn't stopped as soon as the propagated distance becomes larger than the distance of the pixels it reaches, but continues a little further, until it reaches its value plus one. This idea is similar to Mullikin's in next section.

This approach is obviously tempting because the structure of the chains should prevent non-desired multiple propagation to occur and keep the computational complexity low. Unfortunately, the method requires the chains to be broken at every non-convex part of the object border. And the worst-case images for propagation DTs, i.e. a sloping line or an empty disk, are made essentially or entirely of non-convex borders, so that there is no real gain in using chains instead of a simple list of pixels to represent the propagation front.

2.3.3 Sequential processing by raster scanning.

In the above section, Ragnelmam's exact EDT was presented as an efficient implementation of Yamada's parallel algorithm. Alternatively, it could have been presented as an alteration of the approximate EDT by propagation from the same paper [127]. Similarly, Mullikin [112] and Shih [146] propose to produce exact Euclidean DT by modifying slightly Danielsson's raster scanning approximate algorithm.

Mullikin [112] proposes to process the image twice. First, he applies Danielsson's 4SED algorithm, which can leave a few errors. Typically, he finds about 2% of errors for a 200×200 empty disk image. Secondly, he corrects those errors by applying a modified version of 4SED where the vectors are remembered and propagated if they are less than ε longer than the distance at the current pixel. This requires to store, for each pixel p, the list of all object pixels at a distance between $D(p)$ and $D(p) + \varepsilon$. He calls this algorithm εVDT, where VDT stands for Vector Distance Transformation.

Mullikin shows that, with $\varepsilon = \sqrt{N}/N$ in N dimensions, εVDT imple-

ments the exact Euclidean DT. Actually, he finds the following relations between ε and the maximum error

$$E_{max} = (1 - \frac{\sqrt{N}}{N})(1 - \varepsilon\sqrt{N}) \quad (2.25)$$

$$\varepsilon = \frac{N - \sqrt{N} - N.E_{max}}{\sqrt{N}.(N - \sqrt{N})} \quad (2.26)$$

where N is the number of dimensions. From eq. 2.26, he finds the adequate ε to ensure a given maximum acceptable error. This provides us with a number of trade-offs between εVDT(0) where only exact ties are stored in the lists and the exact εVDT(1), where all near ties up to $\varepsilon = \sqrt{N}/N$ are kept.

Of course, the increased accuracy has a computational cost. εVDT(0) appears to be 10 times slower than 4SED. εVDT(1) is itself approximately 10 times slower than εVDT(0) for the test images Mullikin considers. Besides, it has a higher complexity. For $n \times n$ images, 4SED and εVDT(0) are $o(n^2)$, but εVDT(1) is between $o(n^2)$ and $o(n^3)$, depending on the image.

Shih and **Liu** [146] also propose to process the image twice, first with a variant of Danielsson's 8SED algorithm, then to detect and correct possible errors. They pre-compute the possible relative locations $(p_x - q_x, p_y - q_y)$ for which an error could occur, and find out that those are extremely rare. For 100×100 images, errors can only occur for relative locations (12,5), (40,12), (48,10), (54,12), (60,13), (70,12), (72,12), (80,14) and (98,14) with $0 \leq p_y - q_y \leq p_x - q_x \leq 100$, and similar results for the other 8 angular sectors. This means 72 possible relative error locations out of 200×200, i.e. an error ratio of only 0.18%.

Thanks to this low error ratio, Shih suggests to detect possible errors by checking the distance values found by the approximate DT against a look-up table made from the above list. The very few possible error locations are then submitted to additional computations. The computational cost of the extra detection and correction step is negligible.

Unfortunately, this apparently optimal solution is flawed. In [26], we show that the possible relative locations for errors made by the 8SED

algorithm are far more numerous than computed by Shih. Actually, the possible error ratio is close to 20% for 100×100 images, and climbs to more than 50% for images larger than 400×400. This makes the look-up table approach unpractical for error detection.

It should also be noted that the approximate Euclidean DT algorithm of section 2 of [146] does not work. Indeed, the 4-scan algorithm uses twice the same two scanning orders, while Ragnelmam [128] showed that any raster scanning algorithm for EDT should use at least 3 different scanning orders to support propagation in all directions.

2.3.4 Independent scanning

In his founding article [132], Rosenfeld proposes an alternative approach to the generation of the city-block ($dist_{CB}$) distance transformation. The N-dimensional problem is decomposed into N 1-dimensional sub-problems. In 2D, the distance from the nearest pixel is first computed in each row independently. Then, these values are used to compute the distances in each columns. With the $dist_{CB}$ metric, as defined in eq. 2.2, both sub-problems have an identical solution: applying the [1 0 1] mask back and forth. Paglieroni [118] and Saito and Toriwaki [140] show that a similar approach can be used with the Euclidean metric.

Paglieroni [118] develops a "unified" DT algorithm, valid for any metric M that satisfies

$$\begin{aligned} dist_M(p,q) &= f(|p_x - q_x|, |p_y - q_y|) \\ |dp_{1x}| < |dp_{2x}| &\Rightarrow f(|dp_{1x}|, |dp_y|) \leq f(|dp_{2x}|, |dp_y|) \\ |dp_{1y}| < |dp_{2y}| &\Rightarrow f(|dp_x|, |dp_{1y}|) \leq f(|dp_x|, |dp_{2y}|) \end{aligned} \tag{2.27}$$

which is true for all the metrics defined in section 2.1, including the Euclidean one. The "row" scan goes back-and-forth across each row and determines the "nearest object pixel within the row" (NOPwR). The key to the algorithm of course relies on the up-and-down column scan.

From equations 2.27, one can deduce that if the 2D nearest object pixel from pixel $p(p_x, p_y)$ is pixel $q(q_x, q_y)$, then q is the NOPwR of pixel (p_x, q_y), in the same column as p and the same row as q. Therefore,

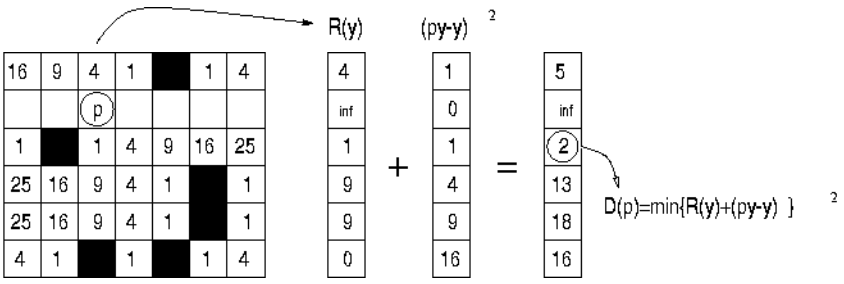

Figure 2.11: Independent scanning. *Left*: After left-right scan *Right*: up-and-down scan, finding the value for $D(p)$ by scanning column p_x.

checking all NOPwRs in column p_x guarantees to find the 2D nearest pixel for p. Paglieroni implements this check with an up-and-down scan over each column. He applies a few simple tests to restrict the number of NOPwRs to consider for each pixel.

Saito and **Toriwaki** [140] particularize Paglieroni's algorithm to the Euclidean metric, in order to restrict further the number of rows to consider for each pixel in the column scans, and therefore reduce the overall computational cost.

Let us consider the column p_x after the "row" scan. Pixel (p_x, y) in row y of that column contains the value $R(y) = (p_x - q_x(y))^2$ where $q(y)$ is the NOPwR of (p_x, y). During the up-scan along the column, we want to know to which pixels (p_x, p_y) we should propagate the information from (p_x, y). Saito first notices that if $R(y) \geq R(y+1)$, then the information from row y should not be propagated upwards since $dist_E(p, q(y)) = R(y) + (p_y - y)^2 > dist_E(p, q(y+1) = R(y+1) + (p_y - y - 1)^2$ for any $p(p_x, p_y)$ with $p_y \geq y + 1$. On the other hand, if $R(y) < R(y+1)$, then the information from row y should propagate up to $y_{max} = y + (R(y+1) - R(y) - 1)/2$, the value for which $dist_E(p, q(y)) > dist_E(p, q(y+1))$. Therefore, in the up-scan part of Saito's algorithm, pixel (p_x, y) is propagated to $(p_x, y+i)$ until either $R(y) \geq R(y+i)$ or y_{max} is reached. The down-scan proceeds similarly.

2.3.5 Voronoi transformation

Recently, a new class of DT algorithms was proposed. It considers distance transformation as a sub-product of the generation of the Voronoi diagram in the continuous plane. Obviously, the knowledge of the Voronoi diagram, i.e. the knowledge of what is the nearest object pixel for any point in the image plane, is sufficient for a straightforward computation of the DT.

On one hand, working in the continuous plane ensures that the Voronoi Polygons are indeed connected sets, so that none of the problems encountered at section 2.2.2 occur. On the other hand, computers are not particularly well-suited for working on continuous (non-discretized) problems. The key to the following algorithms [17, 69, 44] is that one can efficiently compute the intersection of the Voronoi diagram with a row or a column of the image. This could be seen as a half-way discretization, along one axis and not the other. Because they work row by row, the following algorithms also have similarities with those of the preceding section.

Breu, **Gil**, **Kirkpatrick** and **Werman** [17] first show that, because the image pixels are restricted to a limited-size array, and because they are structured in rows and columns, the Voronoi diagram of the centers of object pixels can theoretically be computed in linear time, i.e. $o(mn)$ for an $n \times m$ image. Then, they show how this can be done row by row, by computing the intersection of any row with the Voronoi diagram. Once again, this is performed in two steps, with an up- and down-scan of the row. Let us consider the up-scan for instance.

During the up-scan, we compute the intersection of every row R with the Voronoi diagram of the object pixels located under R, i.e $p(p_x, p_y) \in O$ with $p_y \leq R$. This intersection is represented as L_R, the list of object pixels p for which $VP(p)$ intersects row R. The method is based on two observations.

- for pixels p and q in the sub-image below row R, if $p_x = q_x$ and $p_y > q_y$, then $VP(q)$ does not intersect row R.
- for pixels p and q in the sub-image below row R, if $p_x < q_x$, then $VP(p)$ ($VP(q)$) lies to the left (right) of the mid-perpendicular between p and q.

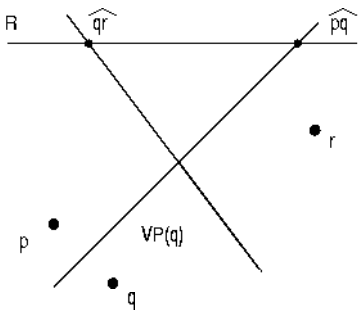

Figure 2.12: Voronoi Polygon $VP(q)$ does not intersect line R if $\widehat{pq}_x \geq \widehat{qr}_x$

From the first observation, we conclude that L_R contains at most one object pixel per column. The object pixels $(p_x, p_y) \in O$ with $p_y \leq R$ and $p_y \geq y \forall (p_x, y) \in O$ are the only candidates C_R to belong to L_R. This list of candidates C_R is easily produced from the candidates C_{R-1} and the list of object pixels in row R.

Using the second observation, we can then trim the list of candidates C_R of pixels q for which $VP(q)$ does not intersect row R. At Figure 2.12, $VP(q)$ does not reach row R because p and r hide it. The second observation tells us that, with $p_x < q_x < r_x$, this will happen if $\widehat{pq}_x \geq \widehat{qr}_x$ where $\widehat{pq}$ ($\widehat{qr}$) is the intersection of the mid-perpendicular of pq (qr) with row R, i.e. $\widehat{pq}_y = R$ and

$$\widehat{pq}_x = \frac{q_x^2 - p_x^2 - 2.R.(q_y - p_y) + q_y^2 - p_y^2}{2.(q_x - p_x)} \tag{2.28}$$

and a similar expression for $\widehat{qr}$. Therefore, pixel $q \in C_R$ will not belong to L_R only if there exists $p, r \in C(R)$ such that $p_x < q_x < r_x$ and $\widehat{pq}_x \geq \widehat{qr}_x$, i.e.

$$\begin{aligned} &(r_x - q_x).(q_x^2 - p_x^2 - 2.R.(q_y - p_y) + q_y^2 - p_y^2) \\ &\geq (p_x - q_x).(q_x^2 - r_x^2 - 2.R.(q_y - r_y) + q_y^2 - r_y^2) \end{aligned} \tag{2.29}$$

The procedure to produce the list L_R from the list C_R is then the following. Pixels from C_R are added one by one to L_R. Before actually adding a pixel r to L_R, equation (2.29) is checked with q the last pixel in L_R and p the preceding one. The last pixel of L_R is removed iteratively

until (2.29) is not satisfied anymore or until L_R only contains one pixel. Then only, r is added at the end of L_R.

Because each pixel is added or removed at most once in the process, creating L_R from C_R is a linear complexity operation. Allocating each pixel along the row to the correct VP knowing L_R is then also a linear-complexity operation, because VPs along row R and object pixels in L_R are ordered in the same way. Therefore, Breu's DT has global linear complexity.

Guan and **Ma** [69] adapt the above algorithm to another data representation. Instead of using L_R that contains the list of object pixels whose VP intersect row R, they consider an equivalent list of segments that partition row R between the VPs. The main improvement of the method is that they use L_{R-1} to produce L_R instead of using the row-to-row redundancy only when producing C_R. This reduces the computational time significantly for sparse object images.

For an $m \times n$ image with t object pixels, this method has a complexity in $\alpha.O(mn) + \beta.O(\sqrt{mnt})$. The $O(mn)$ term corresponds to the computation of the DT from the segments lists. The $O(\sqrt{mnt})$ term corresponds to the creation of the segments lists. Because of the complexity of the computations involved (typically 2.29), this second term is dominant unless the object image is extremely sparse. For some images he considers, this step can be up to 30 times slower than the $O(mn)$ step.

Finally, as announced in section 2.2.2, **Embrechts** and **Roose** [44] use similar techniques to implement Danielsson's 4SED algorithm on multiprocessors machines. The image is divided into sub-images, one per processor. In order to be able to apply 4SED, each processor needs to know two things. First the location of the object pixels inside its own sub-image, secondly the object pixels from other sub-images that influence it, i.e. the intersection of the border of its sub-image with the Voronoi diagram of the whole object image.

The intersections of the Voronoi diagram of the object pixels in the sub-image with the sub-image borders are first computed by each processor using a technique somewhat similar to Breu's. From these, the intersections of the complete Voronoi diagram with the sub-image borders are

computed, using a number of merging and splitting rules. This merging and splitting is the only step requiring communication between the processors.

Embrechts and Roose find that it is possible to reach a good parallel efficiency (typically 80%) as soon as the image size is sufficiently large, i.e. as soon as the cost of the Voronoi diagram computations on the sub-image borders becomes negligible compared to the 4SED algorithm.

2.4 Extended concepts

In the previous section, we described a number of different methods that were proposed to solve the same complex problem: computing the distance from any pixel in an image to the nearest object pixel. In contrast, several authors explored variations on the definition of this problem. In this section, we present three such variations.

First, we consider geodesic distances constrained by restricted domains [122, 167]. Distances are only defined and computed on a part of the image, that can be either convex or non-convex. Secondly, we present the k-distance transformation [175], that computes the k distances to the k nearest object pixels. The usual DT is of course a particular case of k-DT with $k = 1$. Finally, we present distances defined on gray-scale images [139, 5, 170, 108, 152, 156]. On such images, a great variety of metrics can be defined. We present some of those and an efficient general purpose algorithm to compute the related DTs.

2.4.1 Geodesic distances

The geodesic distance between two pixels p and q is defined as the length of the shortest path from p to q. Suppose $P = \{p_1, p_2, ..., p_n\}$ is a path in a connected domain between pixels p_1 and p_n, i.e. p_i and p_{i+1} are connected neighbors for $i \in \{1, 2, ..., n-1\}$ and p_i belong to the domain for all i. The path length $l(P)$ is defined as

$$l(P) = \sum_{i=1}^{n-1} d_N(p_i, p_{i+1}) \tag{2.30}$$

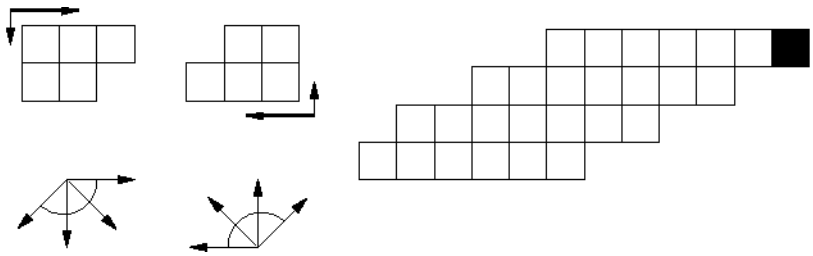

Figure 2.13: *Left*: chamfer masks and the supported propagation directions. *Right*: A convex domain on which the two raster-scan chamfer DT algorithm does not compute the distance transformation

the sum of the neighbor distances d_N between adjacent points in the path. For instance, the city-block metric $dist_{CB}$ defined on the rectangular image domain is a geodesic distance where only 4-direct neighbors are connected and $d_N = 1$. Similarly, the chess-board metric $dist_{chess}$ considers that 8-direct neighbors are connected and $d_N = 1$. The chamfer metric $dist_{cha(3:4)}$ considers 8-direct neighbors to be connected, $d_N = 3$ between 4-direct neighbors, and $d_N = 4$ between diagonal neighbors. Finally, the chamfer 5-7-11 considers that all neighbors within a 5×5 neighborhood are connected, with $d_N = 5, 7$ or 11. On the other hand, with the Euclidean metric $dist_e$, there is a direct path between any two pixels in the image, which makes the problem non-local and renders it so much more difficult to compute.

In the previous sections, distances are always defined for every pixel of the image, i.e. on a rectangular domain. In contrast, **Piper** and **Granum** [122] study geodesic distances defined on any connected domain, both convex and non-convex ones. On convex domains, they show that the usual two raster scans algorithm of section 2.2.1 does not function anymore. As illustrated at Figure 2.13, the chamfer masks do not support propagation in every direction, but only in two 135^o angles.

The surprising part should actually be that this algorithm does work on rectangular image domains. The reason for this is that, in a geodesic DT, one only needs to support the propagation along one of the paths of minimal length to get the correct distance value. For any pixel located between the up-right and right directions from their nearest object pixel, the two raster scan chamfer algorithm does provide one (and only

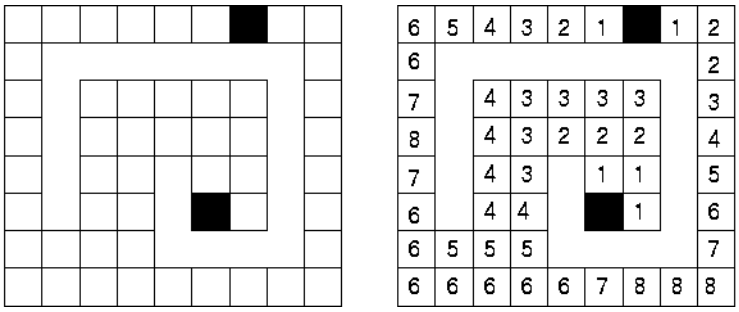

Figure 2.14: *Left:* a non-convex domain. *Right:* distance transformation using the geodesic equivalent of the $dist_{chess}$ metric.

one) such path, made of right steps only, followed by up-right step only. For arbitrary convex domains such as that of Figure 2.13, this particular path does not always belong to the domain, and another minimal-length path should be used instead, which is impossible given the available supported propagation directions.

There are several solutions to this problem. First of all, one could apply the two-scan algorithm iteratively until no pixel changes value anymore. Unfortunately, there is no way to predict how many iterations should be used. Instead, they propose to use 4 different raster-scanning orders with either 3 or 4 neighbors in each mask, or only 3 different raster-scans and masks similar to those used by Ragnelmam [128] at Figure 2.7.

For non-convex domains (Figure 2.14), even 4 raster scans algorithms do not always reach the whole domain, and would need to be applied iteratively. Instead, Piper and Granum suggest to use propagation algorithms, starting from the object pixels, then considering their neighbors, their neighbors' neighbors, ... They propose two simple implementations of such algorithms, recursive propagation which is depth-first, and ordered propagation which is breadth-first. The ordered propagation algorithm they propose uses the neighboring order, i.e. the geodesic chess-board distance order. This is suboptimal when another metric is used.

Verwer, Verbeek and and **Dekker** [167] propose an efficient algorithm for ordered propagation. The main idea is to scan the pixels in the domain in the order defined by the metric used for the DT. With

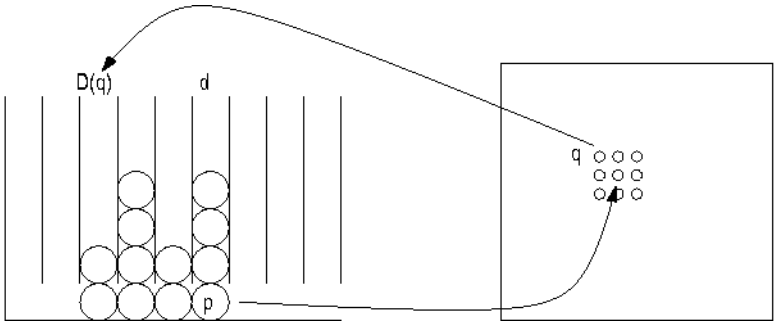

Figure 2.15: Bucket sorting propagation.

a geodesic metric, where the minimal path length is defined as equation (2.30) with $d_N > 0$, this guarantees that every pixel will only be propagated once. Therefore the computational complexity of the DT is optimal, i.e. strictly proportional to the number of pixels.

Scanning the pixels in the metric order is made possible by sorting the pixels in the propagation front before propagating them. With the metrics we consider, there is only a finite number of possible distance values. Then, bucket sorting has an optimal $O(n)$ computational complexity where n is the number of elements to sort.

Practically, there is one bucket $bucket(d)$ per possible distance value d. Initially, every object pixel is put into $bucket(0)$, all other buckets are empty. $D(p)$ is set to "0" for every object pixel and to "∞" for every non-object pixel in the domain. In the propagation phase, illustrated at Figure 2.15, pixels are taken from the bucket with the smallest value d for which $bucket(d)$ is not empty. For every neighbor q of the current pixel p, we suppose that $D(q) = D(p) + d_N(p, q)$, i.e. that the minimal-length path (eq 2.30) from the object to q goes through p. If this leads to a smaller $D(q)$ than currently stored, its value is updated and q is inserted in $bucket(D(q))$. The algorithm stops when all buckets are empty.

The implementation of the above algorithm requires special care. Because one cannot know a priori the size of the buckets, the memory for those has to be dynamically allocated. If this is done with classical dynamic lists, with one pixel per element, there is at least one memory allocation operation per pixel in the image. Dynamic memory alloca-

tion then becomes the major factor in the computational cost. Instead, memory should be allocated by chunks, with a number c of pixels stored in each chunk. Verwer analyzes the influence of the chunk size c on the memory requirements and processing time. He concludes that any chunk size between 15 and 100 is satisfactory. With $c < 15$, too much processing time is wasted in memory allocation. With $c > 100$, too much memory is wasted in partially filled chunks. This upper limit is of course dependent on the image size.

Also, during the propagation, all buckets are never used simultaneously. Actually, when processing $bucket(d)$, only buckets between d and $d + max(d_N)$ can contain pixels. Therefore, Verwer proposes to reuse the buckets circularly.

Provided it is carefully implemented, Verwer's algorithm appears to be optimal for geodesic distances defined with path lengths as in (2.30).

2.4.2 k-distance transformations

Independently from the research on distance transformations, people defined nearest-neighbor and k-nearest-neighbors (k-NN) rules for multichannel data classification [24, 23, 71, 63, 59, 60, 4, 82]. Given a training set consisting of N prototype patterns (vectors in D dimensions where D is the number of channels), and the corresponding correct classification of each prototype into one of C classes, a pattern of unknown class is classified as class c if most of the k-closest prototypes are from class c.

Warfield [175] shows that - with a large number N of prototypes and a large number F of patterns to classify - a k-distance transformation (k-DT) is a very efficient implementation of k-NN classification, provided each channel is or can be quantized into a reasonable number of discrete values. First, a synthetic D-dimensional image is created, where each dimension corresponds to one channel. The bounds of the image are set so that they cover all possible patterns. Object pixels corresponding to the prototype patterns are inserted and the signed k-DT is computed over the image. The k-distance map can then be used as a look-up table to find the k-nearest neighbors of the patterns to classify.

With 2 channels, i.e. in 2D, he proposes to do this by adapting Borgefors' chamfer(3:4) algorithm [10]. In D-dimensions ($D > 2$), he adapts Ragnelmam's corner EDT algorithm [128], that requires 2^D scans using corner masks containing D neighbors each. Both algorithms were described in sections 2.2.1 and 2.2.2, respectively. They scan the image several times with different masks. At their core, the value of a pixel is updated as the minimum of its old value and the values computed from those of its neighbors that belong to the current mask.

Instead, Warfield uses a unique identifier for every prototype pattern (object pixel). At the core of the algorithm, he gathers into a single list the k identifiers of the current pixel and those of its masked neighbors. Then, he sorts the list so that he can select the k nearest patterns from it.

With 2^D scans using masks of D neighbors, this algorithm requires $O(2^D F(D+1)k)$ distance computations, where F is the number of patterns to classify, i.e. the number of points of the D-dimensional synthetic image. Sorting of the lists of $(D+1)k$ nearest patterns for each pixel requires $O(2^D F(D+1)k.\log((D+1)k))$ comparisons. Warfield compares this to the k-NN algorithms proposed by Friedman [59] and shows that k-DT is at least an order of magnitude faster for the type of applications he considers.

2.4.3 Distance transformations on gray-scale images

If one associates a gray-level image $G(p)$ to the original binary image containing the object O, it is then possible to define a large variety of gray-level distance functions. Most of those are defined as modified geodesic distances, i.e. the distance between pixels p and q is the minimum of the lengths of the paths $P : (p = p_1, p_2, ..., p_n = q)$ linking p and q where the path length $l(P)$ is defined as

$$l(P) = \sum_{i=1}^{n-1} f(d_N(p_i, p_{i+1}), G(p_i), G(p_{i+1})) \tag{2.31}$$

where the choice of the function f allows to define many different distances. The purpose of this section is by no means to present an exhaustive review of gray-level distance functions, but merely to illustrate the diversity of their possible definitions.

Rutovitz [138], in 1968, introduced the idea of the gray-weighted distance transformation, in which the gray-level is associated with height and the gray-weighted distance is less along paths with lower gray-level values. In [139] **Rutovitz** again defines the fall-distance, where the only permitted paths are those with strictly decreasing gray-values. The set of points reached by such decreasing paths is known as the fall-set. **Vossepoel, Smeulders** and **Van der Broek** [174] propose a queueing scheme to compute fall-distances, somewhat similar to **Piper** and **Granum**'s [122], who show that their own ordered propagation algorithm can be applied to gray-level distance functions.

Levi and **Montanari** [94] introduce the gray-weighted medial axis transform (GRAYMAT), where the length of a path is computed as the sum of all gray-levels along the path. Preteux [125, 124] defines the topographical distance, where $d_N(p_i, p_{i+1})$ corresponds to the slope between the two pixels. Finally, **Toivanen** [156] presents the distance transform on curved space (DTOCS) and the weighted distance transform on curved space (WDTOCS), where $d_N(p_i, p_{i+1}) = |G(p_i) - G(p_{i+1})| + 1$ and $d_N(p_i, p_{i+1}) = \sqrt{(G(p_i) - G(p_{i+1}))^2 + 1}$, respectively.

A common feature to all of the above definitions is that, in contrast with distances from binary images, the influence zone of an object pixel is usually not convex. Therefore, the above papers that implement the DT with raster scanning have to do so iteratively. On the other hand, since those distances are all defined as geodesic distances - i.e. as the minimum length of the paths joining the pixels - we know that the optimal implementation of all of the above distance functions is the uniform cost algorithm of **Verwer, Verbeek** and **Dekker** [167], described in section 2.4.1. The only restrictions are that function f in (2.31) should be positive defined, and that it should be quantified, so that bucket indexes can be computed from the distance values.

To illustrate this, let us consider the well-known watershed transform introduced by **Beucher** and **Lantuéjoul** [5], and for which **Vincent** and **Soille** [170] proposed an efficient implementation by immersion. Intuitively, the watershed transform is defined as the division of a relief into its catchment basins. A catchment basin is associated with every local minimum of the relief. It consists of locations such that a water drop falling on that location would flow downstream towards that min-

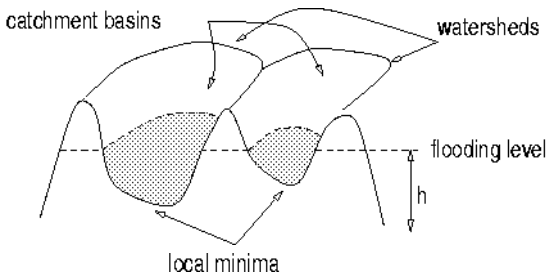

Figure 2.16: Minima, catchment basins, watersheds, dams and flooding level

imum. In image processing, the gray-scale level of a pixel, or a function of it, is considered as an altitude, so that the image constitutes the relief.

An alternative, more algorithmic, definition of watersheds relies on the "flooding" analogy. We consider that holes are pierced at every local minimum of the image, and that the relief is slowly immersed into water. Starting from the minima of lowest altitude, the water progressively floods the different catchment basins. Whenever waters coming from two different basins meet, a dam is constructed to keep those waters separated. After the whole image is flooded, dams separate the catchment basins completely. They are the watershed lines.

Vincent implements this by first sorting all pixels in the image by increasing gray-level. Essentially, he performs bucket-sorting in two steps. First he determines the frequency of each gray-level value, which allows him to allocate the needed memory for each list of pixels. Then, he scans the image a second time and inserts the pixels in the list corresponding to their gray-level value.

Next comes the flooding step, illustrated at Figure 2.17. Supposing that the flooding has reached level h, the pixels of level $h+1$ have to be divided between the catchment basins of level h and the new basins corresponding to the local minima at level $h+1$. First, the pixels at level $h+1$ that are neighbors to pixels from a catchment basin at level h are put into a FIFO (first in first out) queue. Then, the catchment basins are extended by propagation under the constraint of considering only pixels at level $h+1$. When several catchments basins of level h are connected at level $h+1$, this procedure separates the resulting basins

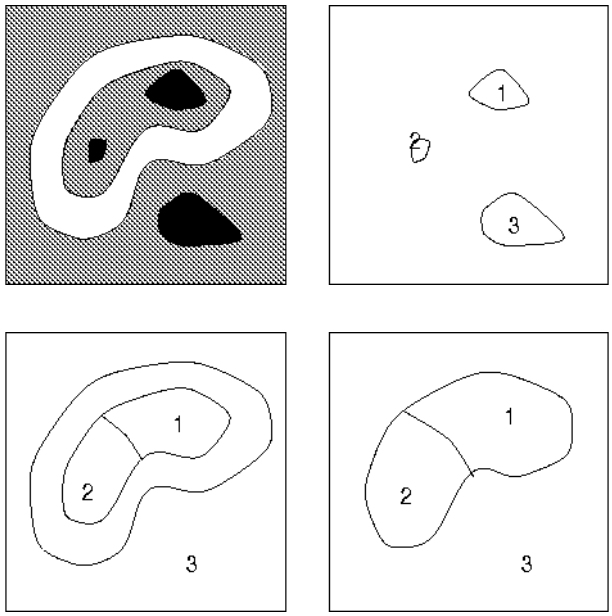

Figure 2.17: Watershed segmentation of a 3-level image by Vincent's algorithm. *Upper left:* Original image *Upper right:* flooding reaches level 1, 3 minima are detected. *Lower left:* flooding reaches level 2 *Lower right:* flooding reaches level 3.

at level $h+1$ along the geodesic skeleton by influence zone (SKIZ). The pixels at level $h+1$ that are not reached by one of the catchment basins must be local minima, and become the seeds of new basins.

Alternatively, **Meyer** [106, 107, 108] shows that the watershed transform can also be expressed in terms of a topographical distance function. If points p and q belong to a line of steepest slope between p and q ($G(q) > G(p)$), the topographical distance is defined as $dist_{topo}(p,q) = G(q) - G(p)$. The catchment basins are then the equivalent of the Voronoi division of the image, where object pixels are the local minima and the distance considered is $dist_{topo}$.

Meyer implements this computation by using hierarchical waiting queues, which allows him to process the image by increasing topographical distance values. This technique was later adapted by **Thiran**, **Warscotte** and **Macq** [152] to implement a variation of the watershed segmentation.

Clearly, both Vincent's [170] and Meyer's [108] algorithms can be seen as adaptation of the optimal bucket sorting algorithm of Verwer, Verbeek and Dekker [167].

2.5 Applications

In this section we present of few applications of distance transformations, within and without the medical image processing domain. This intends to illustrate their usefulness, but should by no means be considered as an exhaustive review of DT applications. Distance transformations are probably too generic a tool anyway for such an exhaustive review to be possible.

Borgefors [11, 13] proposes to use distance transformations for *pattern matching*. In order to find an object in an image, she computes the chamfer DT from the edges of that image. The object is located at the minimum of the correlation between the edges of the object and the resulting distance map.

Paglieroni [118] studies the correlation properties of the Euclidean DT. He shows applications for pattern matching and for stereo vision. In *stereo vision*, one aims to find equivalent features in the right and left images. The distance between the locations of a feature in both images reveals its depth, i.e. its distance from the camera. The conjugate of a point in the right image is constrained, by camera parameters and geometry, to lie on a line in the right image. Its exact position must be determined by contextual information. For this, Paglieroni uses edge matching, i.e. he applies an edge detector on both images, then computes the Euclidean DT from both edge images. The matching criterion is a composition of the correlation of an edge image with the DT of the other, and vice-versa.

Mangin, **Froin**, **Bloch**, **Bendriem** and **Lopez-Krahe** [101] apply a similar method for the *registration of 3D medical images* of different modalities, i.e. Positron Emission Tomography (PET) and Magnetic Resonance Imaging (MRI).

Ragnelmam [126], **Paglieroni** [118] and **Huang** and **Mitchell** [77] use distance transformations in order to implement *mathematical morphology operators*. In mathematical morphology, the dilation of an object O by a structural element B is defined as

$$O \oplus B = \{p + q \mid p \in O\ ,\ q \in B\} \tag{2.32}$$

In the common case where the structural element is a ball, i.e.

$$B = \{q \mid dist_M(q, (0.0)) \leq d_{\max}\} \tag{2.33}$$

the dilation can be efficiently implemented by thresholding the DT, i.e.

$$O \oplus B = \{p \mid DT(p) \leq d_{\max}\} \tag{2.34}$$

In [126], Ragnelmam shows that his distance transformation by propagation is particularly well-suited to implement the morphological dilation since the DT can be stopped as soon as $d_{\max}$ is reached.

Danielsson [37], **Niblack**, **Gibbons** and **Capson** [114] and **Ge** and **Fitzpatrick** [64], among many others, show that the DT can be used to produce another useful morphological operator, the *skeleton*. Blum [6, 7] defines the skeleton of an object as the locus where fire-fronts started from its edges meet. He shows that an equivalent definition is simply the locus of the centers of maximal disks contained in the object.

As illustrated at figure 2.18, **Danielsson** [37] finds an approximate skeleton with a local test on the Euclidean DT of the object's edges. A pixel belongs to the skeleton if the largest inscribed disk centered on it is included in none of the largest inscribed disks centered on one of its neighbors. The set of pixels he finds is redundant, both because some disks can be included inside disks centered on non-neighboring pixels, and because some disks can be included in the union of several disks. **Niblack** [114] uses chamfer DTs on which he detects local maxima, saddle points, and the steepest uphill paths from them. **Ge** [64] uses the Euclidean DT, detects the exact set of centers of maximal disks by performing additional tests on the set found by Danielsson's method. Then, he links the centers of maximal disks in order to produce connected skeletons. The resulting skeletons have all desirable properties: they have the same connectivity as the figure, they are well-centered, they are insensitive to rotation and they allow the exact reconstruction

Figure 2.18: Skeleton of an object generated using Danielsson's method. *Left:* Original object *Center:* Distance transformation *Right:* Skeleton

of the original object.

Euclidean skeletons have a large variety of applications. For instance, in [15], **Bourland** uses a 3D skeleton computed from the Euclidean DT to plan the optimal location of the shots in a radio-surgical treatment. The optimal locations are either the end-points or cross-points of the skeleton of the targeted tumor.

Borgefors [14] uses the *center of maximal disks* as an efficient tool for shape representation.

In his paper about watershed segmentation, **Vincent** [170] suggests that overlapping objects can be separated by computing the watershed transformation on the negative DT of the edge of the compound object (Figure 2.19). **Thiran** and **Macq** [153] obtain a similar result as they find the centers of the nuclei in images of cancerous tissues by applying the *mathematical morphology ultimate erosion operator*. Obviously, the local minima of the DT and the ultimate erosion are identical operators.

In [168], **Vincent** proposes several other applications, such as computing *Delaunay triangulations*, *Gabriel graphs* [61], *relative neighborhood graphs* [159], ...

Saito [140] uses the Euclidean DT to compute the 3D *Voronoi diagram*. Working on 3D microscopic images of a human liver, he proves that the shortest path from a portal vein to an hepatic vein is almost constant whatever point of the volume is considered. He also shows that the por-

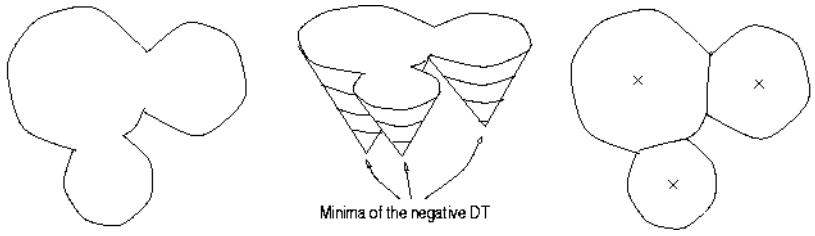

Figure 2.19: Separation of overlapping objects by applying the watershed segmentation to the distance transformation of the edges of the compound object

tal veins lie on or near the borders of the Voronoi diagram derived from the hepatic veins.

Starovoitov [149] and **Warfield** [175] use distance transformation for the analysis of multi-dimensional data-sets. Starovoitov defines an *unsupervised clustering* technique. Warfield uses the k-DT as a look-up table for *k-NN classification*, as discussed in section 2.4.2.

The main application of the geodesic distance transformation was described by **Verbeek**, **Dorst**, **Verwer** and **Groen** [165] and **Lengyel**, **Reichert**, **Donald** and **Greenberg** [93]. By backtracking the distance propagation, one can *find the shortest path* between any point and a single object pixel, the source of the propagation (Figure 2.20). In particular, this can be used to control the motion of a robot. The geodesic DT is computed in the robot's multi-dimensional state space, where each dimension corresponds to a possible movement of the robot (translation, rotation, ...) and the propagation domain is restricted to possible robot locations.

In [85], **Jolesz**, **Lorensen**, **Shinmoto**, **Atsumi**, **Nakajima**, **Kavanaugh**, **Saiviroonporn**, **Seltzer**, **Silverman**, **Phillips** and **Kikinis** apply Lengyel's method to generate the camera movements in an interactive virtual endoscopy application.

This concludes our short review of possible DT applications. Among them, several belonged to the field of medical image processing: those of Mangin [101], Bourland [15], Thiran [153], Saito [140], Warfield [175] and Jolesz [85].

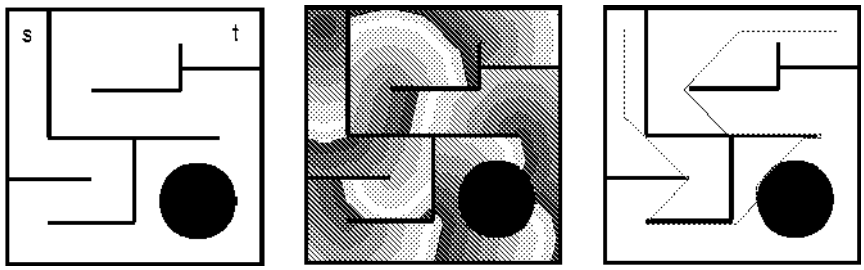

Figure 2.20: Shortest path computation. *Left:* Original mask, source and target locations. *Center:* Geodesic distance from s, constrained by the mask. *Right:* Shortest Path between s and t, obtained by backtracking the propagation of the Geodesic DT.

2.6 Discussion.

Today, distance transformation is a mature domain, in the sense that, for most problems, optimal algorithmic solutions are known. The theoretical optimal complexity of a DT is obviously the complexity of the distance image it generates. For an $n \times n$ image, that means $O(n^2)$ for usual DTs and $O(k.n^2)$ for the k-DT. The aims of this discussion are

- to review the algorithms of sections 2.2 and 2.3 in the light of the possible applications of section 2.5.
- to discuss which algorithms are optimal and which problems still require algorithmic improvements.
- to stress the strong and weak points in the existing solutions, in order to suggest how such improvements can be achieved.

The generation, in two raster scans, of the approximate **chamfer DT** obviously has an optimal $O(n^2)$ algorithmic complexity. The best values for the chamfer masks are known, as well as good integer approximations. Chamfer DT can also be adapted to anisotropic grids, although it requires to compute the optimal mask values for every new anisotropy factor. The relative error is less than 8% for chamfer (3:4) masks and 2% for chamfer (5:7:11) masks.

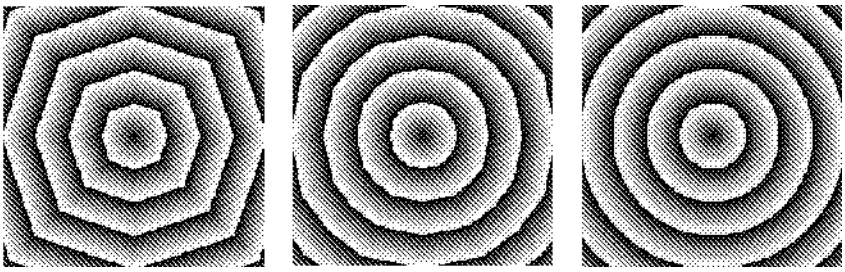

Figure 2.21: Disks generated with the chamfer(3:4), chamfer(5:7:11) and Euclidean metric

On the negative side, one cannot improve the precision of the chamfer DT unless one uses much larger masks, which requires an unacceptable additional computational cost. Several applications may suffer from this. For instance, when computing centers of maximal disks, it is important that the shape of the disks be as circular as possible, which is only coarsely approximated by chamfer DT, as illustrated at figure 2.21. Similarly, skeletons generated from a chamfer DT do not have a good rotational invariance.

Also, for pattern matching applications, faster convergence can usually be reached if one can use the gradient of the DT in the maximization of the correlation. As illustrated at figure 2.22, the possible directions for the gradient of the chamfer DT are restricted to the directions available in the mask, while the full range of directions would be possible with the Euclidean DT.

Finally, chamfer DT doesn't scale well to higher dimensions. While it can still be generated in two scans over the image, it requires to use masks that contain $3^D - 1$ neighbors in D dimensions. In conclusion, chamfer DT should only be used for applications in 2 or 3 dimensions, where the precision of the DT is not a crucial parameter.

The **approximate Euclidean DT** algorithms of section 2.2.2 also have an optimal $O(n^2)$ algorithmic complexity, both for the raster scanning and for the ordered propagation implementations. Implemented carefully, their computational cost is actually nearly as low as that of chamfer DT, with a much better precision. Errors only occur in a few locations while most of the distances are computed exactly.

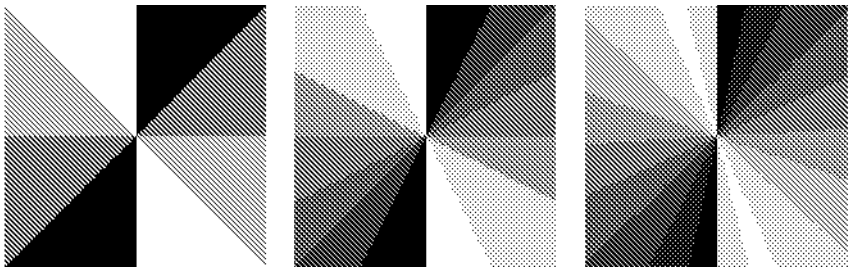

Figure 2.22: Direction of the gradient of the distance in the images of Figure 2.21. *Left:* chamfer(3:4). *Center:* chamfer(5:7:11). *Right:* Euclidean.

Also, because it can be computed using only direct neighbors and not the diagonal ones, it scales quite well to higher dimensions. In D dimensions, one only needs to consider $2.D$ neighbors. Unfortunately, the raster scanning algorithm then requires 2^D scans. Instead, one should use the ordered propagation algorithm as soon as $D > 2$. This algorithm is optimal.

On the negative side, these algorithms still don't provide a perfect Euclidean DT. For some applications, the non-systematic nature of the errors can cause problems. For instance, when one implements the morphological dilation, using (2.34) and the 8SED algorithm, of object $O = \{p_1, p_2, p_3\}$ in figure 2.5 by a ball B of size 13, pixel q is mistakenly not included in $O \oplus B$. If one then applies a morphological erosion with the same ball - i.e the dilation of $(O \oplus B)^c$, the complement of $O \oplus B$ - the error is not repeated because there is no unfortunate arrangement of object pixels in $(O \oplus B)^c$. Therefore, the closing $O \bullet B = (O \oplus B) \ominus B$ does not include pixel p_2. This contradicts a fundamental property of openings and closings in mathematical morphology, that $O \circ B \subseteq O \subseteq O \bullet B$. For similar reasons, the exact reconstruction of the original object is not guaranteed from skeletons generated using an approximate Euclidean DT.

The precision of the **exact Euclidean DT** is of course perfect. Unfortunately, there is still no optimal (in terms of computation time) algorithm to produce it, which restricts its use in practical applications. Let us consider the five families of algorithms described in section 2.3.

First, the parallel EDT is obviously unpractical on general purpose computers. It requires as many scans over the image as the largest distance in the image, i.e. has a $O(n^3)$ complexity.

Among the exact EDT by propagation, Eggers' [42] algorithm is the most efficient. Unfortunately, in order to be exact, it can not use the optimal ordered propagation, but instead the natural propagation order of 3×3 neighborhoods. That can lead to multiple updates by successive propagation fronts. In the worst case - an oblique line with a $22,5^o$ inclination - it has a $O(n^3)$ complexity too. Practically, the computational cost is highly image-dependent, sometimes as low as that of approximate EDT, sometimes orders of magnitude slower.

The only raster-scan exact EDT - Mullikin's [112] ϵVDT(1) - has a complexity between $O(n^2)$ and $O(n^3)$ and is several orders of magnitude slower than the approximate EDTs.

Among the independent scans algorithms, Saito's [140] algorithm is the most efficient. The exact computational complexity of the method isn't assessed, but experiments show that it is more than $O(n^2)$.

Finally, the algorithms explicitly based on the computation of the Voronoi diagram of the object pixels do have an optimal $O(n^2)$ complexity. Unfortunately, the computations involved are so complex that the practical computational cost ends up being much larger than the cost of approximate EDT, Eggers' or Saito's algorithms.

In conclusion, none of these approaches leads to an algorithm that is both fast and asymptotically optimal. Fast algorithms, such as the approximate ones, Eggers' or Saito's, require that only a few simple operations be performed on each pixel. On the other hand, asymptotically optimal algorithms require to explicitly take into account the continuous nature of the Voronoi diagram. In chapters 3, 5 and 6, we show how both approaches can be combined efficiently.

The optimal algorithm to compute a **Geodesic DT**, constrained by a non-convex domain, was presented by Verwer [167]. The ordered propagation by bucket sorting has a $O(n^2)$ complexity, and the cost of the

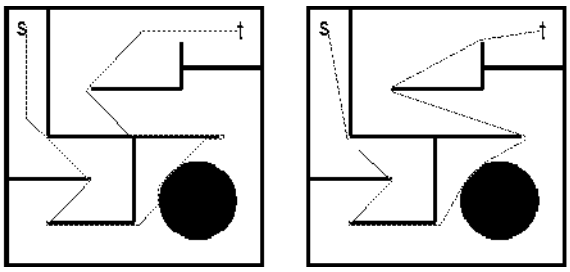

Figure 2.23: Shortest path through a maze. *Left:* computed with a chamfer geodesic DT *Right:* on a continuous plane.

dynamic data structure it requires can be minimized with a careful implementation.

On the other hand, the discrete geodesic DT is only a poor approximation of its continuous equivalent. In particular, the shortest paths it generates are restricted to steps along the directions available in the mask used during the propagation. For instance, for the commonly used chamfer(3:4) geodesic DT, paths are restricted to horizontal, vertical and 45^o diagonal steps, as illustrated at figure 2.23.

In chapter 8, we show how a quasi-Euclidean geodesic DT can be generated, using balls of any chosen size in the propagation process, while keeping the optimal computational cost of the propagation algorithm.

Finally, the **k-DT algorithm** proposed by Warfield [175] is obviously non optimal, it requires $O(2^D n^D (D+1)k)$ distance computations and $O(2^D n^D (D+1)k.\log((D+1)k))$ comparisons on a $n \times n \times ... \times n$ (D times) image. It is also not an exact DT, since it is based on Ragnelmam's approximate EDT [128], but this does not matter much in the k-NN classification application.

In chapter 10, we show that k-DT can be implemented optimally, i.e with $O(n^D.k)$ distance computations and $O(n^D.D.k)$ comparisons.

Chapter 3

Euclidean distance transformation by propagation

In this chapter, we explore a first approach to the improvement of exact Euclidean distance transformation algorithms: improving the EDT by propagation. In order to keep the computational cost low, one has to order the propagation according to the metric order. This potentially produces errors in the resulting EDT. The properties of those errors are analyzed, in particular their dependence on the size of the neighborhood used during the propagation. Thanks to these properties, we show how an exact EDT can be produced by using neighborhoods of increasing size, applied on restricted sets of pixels. The computational cost and complexity of the algorithm is then evaluated. It shows a significant improvement over the methods proposed by Saito and Eggers. In order to illustrate potential applications, we show that it can be used to implement mathematical morphology operators on binary images.

3.1 Propagation with a single neighborhood

In a propagation algorithm, the pixels should ideally be updated only once, when they receive their final value. In other words, the propagation from one object pixel should be restricted to the pixels of its tile in the Voronoi diagram. In order to achieve this, the propagation order should be the same as the metric order. In other words, the pixels in the

propagation front should be sorted by increasing distance value before being propagated. As shown by Verwer [167] for constrained chamfer DT and Ragnelmam [127] for Euclidean DT, this can be accomplished by bucket sorting of the pixels in the propagation front.

Instead of using a single list, pixels to be propagated are stored in a number of buckets, one for each possible $dist_E(p,q)$ value. Indeed, $dist_E(p,q)$ is an integer if p and q are located on an integer grid. Thus, it is an appropriate choice for the index d for the buckets, that are processed by increasing index values. For each pixel p in the propagation front, we store its coordinates (p_x, p_y) and its coordinates $(dp_x, dp_y) = (p_x - q_x, p_y - q_y)$ relatively to the nearest pixel q of the object O. This gives the following algorithm, which we call "**E**uclidean **D**istance **T**ransformation by **P**ropagation using a **S**ingle **N**eighborhood", or EDT-PSN.

Algorithm 1 *EDT by Propagation with a Single Neighborhood*

Input: A binary image I with an object O in it.
Output: An image D such that $D(p) \approx \min_{q \in O}\{dist_E(p,q)\}$

```
for all p ∈ I do {Initialization}
  if p ∈ O then
    D(p) ← 0
    put (p, (0,0)) in bucket(0)
  else
    D(p) ← M {M is an upper bound for D(p)}
  end if
end for
d ← 0;

repeat {Main loop}
  for all (p, dp) in bucket(d) do
    for all n ∈ N do
      D_new ← dist_E(dp + n)
      if D_new < D(p + n) then
        D(p + n) ← D_new
        put (p + n, dp + n) in bucket(D_new)
      end if
    end for
```

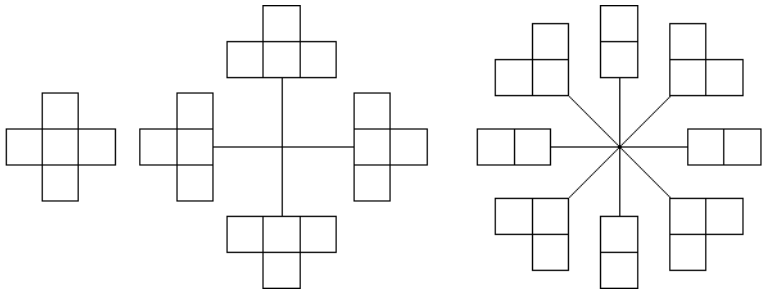

Figure 3.1: Directed neighbors to consider when using 4-direct neighborhoods

end for
$d \leftarrow d + 1$
until all buckets are empty

If N is the 4-direct neighborhood, we have $dist_E(dp+n) \leq (\sqrt{d}+1)^2 = d + 2.\sqrt{d} + 1$. Thus, the termination condition - that all buckets are empty - is true when the last $(2\sqrt{d}+1)$ buckets are empty. With the 8-direct neighborhood, the whole bucket structure is empty as soon as the last $2\sqrt{2d}+2$ buckets are empty.

As pointed our earlier [167], an efficient implementation of the buckets' data structure requires a memory allocation in chunks for the dynamic lists. In what follows, we use chunks of 16 elements, but a large range of values is acceptable, as discussed at section 2.4.1.

Also, special care should be given to the efficient computation of $dist_E(dp+n)$. Leymarie [95] recommends to use eq. (2.18), which only requires to use additions, one shift and one increment. Instead, we routinely use lookup tables for $sq[\pm i] = i^2$, so that $dist_E(dp+n) = sq[dp_x + n_x] + sq[dp_y + n_y]$. This has a similar computational cost and is more general when n belongs to a large neighborhood, as we use later.

Finally, as shown by Ragnelmam [127], at most 2 neighbors need to be considered in the propagation phase, as soon as $d > 1$. For the 8-direct connectivity, this means the neighbors of figure 2.6. The neighbors for the 4-direct connectivity are found in figure 3.1.

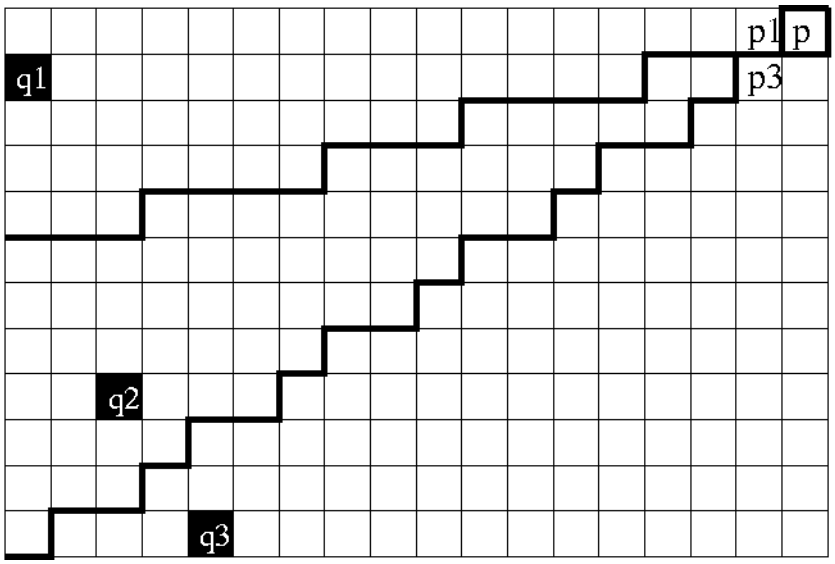

Figure 3.2: A typical error made by an approximate EDT using the 3×3 neighborhood. Object pixel q_2 is hidden from pixel p by object pixels q_1 and q_3, that are closer to p_1 and p_3, respectively.

3.2 Errors in approximate EDT

Unfortunately, as pointed out at section 2.2.2, the above algorithm does not guarantee an exact Euclidean DT. In order to be able to detect and correct those errors, let us analyze a few of their properties.

3.2.1 Errors with a 3×3 neighborhood

Let us first consider that the approximate EDT was produced using a 3×3 (8-direct) neighborhood, such as in Danielsson's 8SED [37], Leymarie's [95] and Ragnelmam's [127] algorithms. Either in raster scanning or by increasing distance order, these algorithms propagate the information from sources - the object pixels - to the rest of the image. As illustrated at figure 3.2, errors occur when the source of a pixel differs from the sources of all its 8-direct neighbors. $p(0,0)$ is closer to $q_2(x_2, y_2)$ than $q_1(x_1, y_1)$ or $q_3(x_3, y_3)$, but pixels $p_1(1,0)$ and $p_3(1,1)$ are not.

In general, since q_2 is the source of p, we have

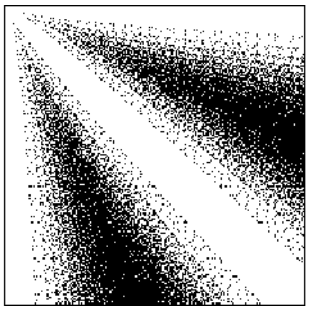

Figure 3.3: Relative locations (dx, dy) for which it is possible that an error occurs, with $0 \leq dx \leq 200$ and $0 \leq dy \leq 200$.

$$x_2^2 + y_2^2 < x_1^2 + y_1^2 \tag{3.1}$$

$$x_2^2 + y_2^2 < x_3^2 + y_3^2 \tag{3.2}$$

Similarly, since q_1 is the source of p_1,

$$(x_1 - 1)^2 + y_1^2 \leq (x_2 - 1)^2 + y_2^2 \tag{3.3}$$

and q_3 the source of p_3,

$$(x_3 - 1)^2 + (y_3 - 1)^2 \leq (x_2 - 1)^2 + (y_2 - 1)^2 \tag{3.4}$$

In [146], Shih restricts these conditions further by setting implicitly $x_1 = x_2 + 1$ and $x_3 = x_2 - 1$, and considers strict inequalities for eq. 3.3 and 3.4. Therefore he misses many possible errors, including that of Figure 3.2, with $q_1(17, 1)$, $q_2(15, 8)$ and $q_3(13, 11)$.

Instead, we find the integer solutions of equations 3.1 to 3.4 by exhaustive search. In order to know if an error could occur at the relative location (x_2, y_2), we check eq. 3.3 for all integer couples (x_1, y_1) with

$$0 \leq x_1 \leq \sqrt{x_2^2 + y_2^2 + 1} \tag{3.5}$$

$$\sqrt{x_2^2 + y_2^2 + 1 - x_1^2} \leq y_1 \leq \sqrt{x_2^2 + y_2^2 + 1 - x_1^2} + 1 \tag{3.6}$$

Equation (3.6) guarantees that (3.1) is fulfilled. If we can find such a couple (x_1, y_1) satisfying (3.3), and if we find another couple (x_3, y_3)

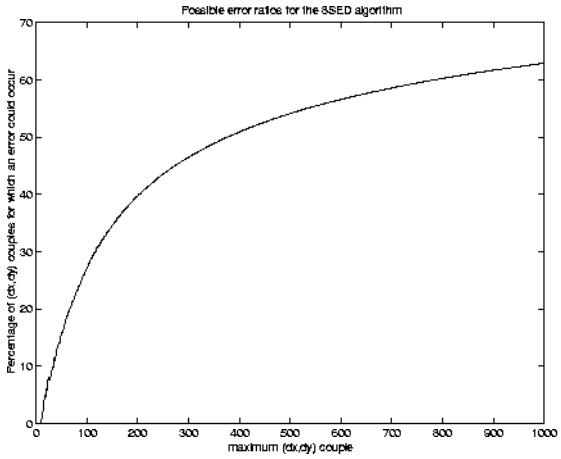

Figure 3.4: Possible errors ratio for various image sizes

satisfying (3.2) and (3.4) in a similar way, then it is possible that errors do occur for the relative location (x_2, y_2).

Applying this exhaustive search for all couples (x_2, y_2) within a 200×200 range, we find the result of Figure 3.3, where black pixels correspond to possible error locations. Defining the possible errors ratio as the percentage of possible error locations among all locations, we obtain the values of Figure 3.4, i.e. 20% for a 100×100 image, and more than 50% for images larger than 400×400. Obviously, possible locations are not uncommon. In particular, it means that an "error locations lookup table" approach to detect errors, such as suggested by Shih [146], is not practical.

3.2.2 Influence of the neighborhood size

Let us now consider any neighborhood N. Similarly to the previous section, an error may occur at a relative location (dp_x, dp_y) if, for every neighbor $n \in N$, one can find (dq_x, dq_y) such that

$$dq_x^2 + dq_y^2 > dp_x^2 + dp_y^2 \tag{3.7}$$

$$(dq_x - n_x)^2 + (dq_y - n_y)^2 \leq (dp_x - n_x)^2 + (dp_y - n_y)^2 \tag{3.8}$$

Once again, the possible errors relative locations can be determined by exhaustive search, for any neighborhood N. In particular, we are interested in the shortest distance for which an error can occur when computing the approximate DT with a neighborhood N. This is determined as follows.

Algorithm 2 *Compute the closest error location* (dp_x, dp_y) *for a neighborhood* N

Input: A neighborhood N.
Output: dp_{err} and $D_{err} = dist_E(dp_{err})$, the smallest error location

```
D_err ← ∞
dp_x ← 1
repeat
  for dp_y = 0 → dp_x do
    D ← dp_x^2 + dp_y^2
    if D < D_err then
      for all n ∈ N do
        test(n) ← FALSE
      end for
      for dq_x = 0 → (int)√(D+1) + 1 do
        dq_y = (int) √(D + 1 − dq_x^2)
        for all n ∈ N do
          if dist_E(dq − n) ≤ dist_E(dp − n) then
            test(n) ← TRUE
          end if
        end for
      end for
      if test(n)∀n ∈ N then
        D_err ← D
        dp_err ← dp
      end if
    end if
  end for
  dp_x ← dp_x + 1
until dp_x^2 > D_err
```

The results for the 4-direct neighborhood and several $N \times N$ neighborhoods are found in table 3.1. In the right column of this table, we also find the relative location of the nearest pixel with the same source as the

Neighborhood	Smallest error		Non-propagating pixel	
	(dp_x, dp_y)	$dist_E$	(dp_x, dp_y)	$dist_E$
4-direct	(2,2)	8	(1,1)	2
3×3	(12,5)	169	(10,4)	116
5×5	(25,7)	674	(22,6)	520
7×7	(48,10)	2404	(44,9)	2017
9×9	(72,12)	5328	(67,11)	4610
11×11	(108,15)	11889	(102,14)	10600
13×13	(143,17)	20738	(136,16)	18752
15×15	(192,20)	37264	(184,19)	34217
17×17	(238,22)	57128	(229,21)	52882
19×19	(300,25)	90525	(290,24)	84676
21×21	(357,27)	128178	(346,26)	120392
23×23	(420,29)	177241	(408,28)	167248
25×25	(500,32)	251024	(487,31)	238130
27×27	(574,34)	330632	(560,33)	314689
29×29	(667,37)	446258	(652,36)	426400
31×31	(728,40)	591424	(752,39)	567027

Table 3.1: Errors closest to (0,0) for the 4-direct and a number of $N \times N$ neighborhoods. The first two lines correspond to the errors illustrated at Figure 2.5

pixel where the error occurs. Using this table, we know the size of the neighborhood to consider to produce an exact EDT up to a given distance. For instance, if we want it to be correct up to $dist_E(dp) = 1000$, we find in table 1 that 1000 is between 674 and 2404. Therefore, a 7×7 neighborhood is large enough to ensure an exact result, but a 5×5 neighborhood might be too small.

Unfortunately, increasing the neighborhood size to produce an exact EDT soon leads to a prohibitive computational cost when the image size increases.

3.2.3 Influence of the propagation process

Instead of considering all possible relative error locations as in the previous section, we now restrict ourselves to the errors occurring for the particular image on which algorithm 1 is performed. Intuition tells us that errors only happen near pixels that do not propagate.

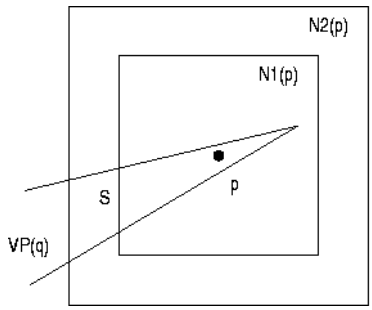

Figure 3.5: $S = VP(q) \bigcap (N2(p) \backslash N1(p))$

More formally, let N_1 and N_2 be two neighborhoods such that $N_1 \subset N_2$. Let D_1 and D_2 be the resulting signed distance maps generated using N_1 and N_2 respectively. If there is a pixel p such that $D_1(p) \neq D_2(p)$, i.e. $D_1(p)$ is inexact and $D_1(p) > D_2(p)$, then there is a pixel $r \in N_2(p)$ such that either $D_1(r) \neq D_2(r)$, either $D_1(r)$ was not propagated using N_1.

To prove this, let us first consider that pixels belonging to a Voronoi polygon $VP(q)$ can only propagate to other locations belonging to $VP(q)$ with the same q, which is what ordered propagation aims to achieve. Let us consider q such that $p \in VP(q)$, i.e. let q be the nearest object pixel to p. Let us consider the set $S = VP(q) \bigcap (N2(p) \backslash N1(p))$ (Figure 3.5). We know this set is not empty since at least one pixel propagated to p using N_2 but none did using N_1. Let us suppose that all pixels $r \in S$ have a correct value and were propagated using N_1. Since propagation implies a strict increase of the distance value, at least one pixel $r \in S$ must propagate to a pixel $r' \notin S$, i.e. a pixel $r \in VP(q) \bigcap N1(p)$. This is impossible since r' would then have been propagated to p using N_1, and $D_1(p)$ would then be correct.

In practice, the ordered propagation of algorithm 1 allows propagation to locations that are either in the same VP or in the neighborhood of this VP. Nevertheless, the above proof remains correct since propagated pixels located in the wrong VP are corrected before being propagated, and do not propagate themselves. Q.E.D.

3.3 Propagation with multiple neighborhoods

From the above properties, we can design a new exact EDT algorithm by propagation, using multiple neighborhoods. A fast approximate DT is first produced using the smallest neighborhood possible. Then, non-propagating pixels are further processed using larger neighborhoods, whose size are determined from table 3.1 to ensure that the resulting DT is error free.

3.3.1 The PMN algorithm

The "**E**uclidean **D**istance **T**ransformation by **P**ropagation using **M**ultiple **N**eighborhoods", or EDT-PMN works as follows:

Algorithm 3 *EDT by Propagation with Multiple Neighborhoods*

Input: A binary image I with an object O in it.
Output: An image D such that $D(p) = \min_{q \in O}\{dist_E(p,q)\}$

```
for all p ∈ I do {Initialization}
  if p ∈ O then
    D(p) ← 0
    put (p,(0,0)) in bucket(0)
  else
    D(p) ← M {M is an upper bound for D(p)}
  end if
end for
d ← 0;

repeat {First loop}
  for all (p,dp) in bucket(d) do
    propagated ← FALSE
    for all n ∈ N do
      D_new ← dist_E(dp + n)
      if D_new < D(p + n) then
        D(p + n) ← D_new
        put (p + n, dp + n) in bucket(D_new)
        propagated ← TRUE
      end if
    end for
```

```
      if propagate = FALSE then
        put (p, dp) in buffer
      end if
    end for
    bucket(d) ← buffer
    d ← d + 1
  until the last 2√d + 1 buckets were empty

  d_max ← d
  for d = 0 → d_max do {Second loop}
    N ← N_min(d) {Smallest needed neighborhood, from table 3.1}
    for all (p, dp) in bucket(d) do
      for all n ∈ N do
        D_new ← dist_E(dp + n)
        if D_new < D(p + n) then
          D(p + n) ← D_new
          put (p + n, dp + n) in bucket(D_new)
        end if
      end for
    end for
  end for
```

In this algorithm, pixels p that do not propagate in the first propagation loop remain in the bucket structure. This is accomplished by storing them in a temporary *buffer* list, that becomes $bucket(d)$ after all pixels in $bucket(d)$ were processed. During the second propagation, the size of the neighborhoods $N_{\min}(d)$ increases as the right-most column of table 3.1. For $2 \leq d < 116$, $N_{\min}(d)$ is the 3×3 neighborhood. For $116 \leq d < 520$, it is the 5×5 neighborhood, and so on.

3.3.2 Oriented neighborhoods

A final improvement to the algorithm is possible using the the following observation.

For a pixel p, whose relative location from the nearest object pixel is (dp_x, dp_y), the only neighbors (n_x, n_y) that need to be considered are those "in the same direction" as (dp_x, dp_y). More precisely, if we consider dp_x and dp_y positive without loss of generality, the neighbors (n_x, n_y) to consider are such that

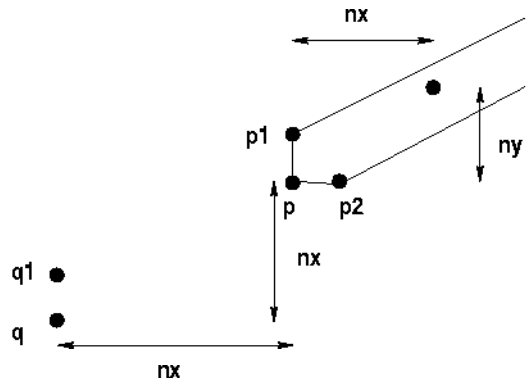

Figure 3.6: pixel p only needs to be propagated to neighbors within the grey area

$$(n_x - 1).\frac{dp_y}{dp_x} < n_y < 1 + n_x.\frac{dp_y}{dp_x} \tag{3.9}$$

To prove this, let us consider figure 3.6. We wish to determine which neighbors (n_x, n_y) should be considered when propagating pixel p, whose location relative to the nearest object pixel q is (dp_x, dp_y). Let us first consider the upper bound. If q is also the nearest object pixel for $p_1 = p + (0, 1)$, the upper bound for neighbors of p_1 is higher than for those of p, which guarantees no neighbor will be missed.

On the other hand, we consider that $q_1 \neq q$ is the nearest object pixel of p_1. The limit between the tiles $VP(q)$ and $VP(q_1)$ is on the mid-perpendicular of q_1q. This mid-perpendicular would be a good upper bound for neighbors of p, since the information from q only needs to be propagated to pixels belonging to $VP(q)$. This mid-perpendicular itself is bounded by the second inequality of (3.9). Indeed, it intersects pp_1 between p and p_1, thus below p_1. Also, its angle must be lower than dp_y/dp_x. Otherwise, it would cross q_1q between q and q_1, which is impossible for the mid-perpendicular of q_1q with q_1 on an integer location. The proof for the lower bound is similar.

Applying this property reduces the computational cost of applying a $n \times n$ neighborhood from $O(n^2)$ to $O(n)$. The "Euclidean Distance Transformation by Propagation using Multiple Oriented Neighborhoods", or EDT-PMON uses the same initialization and first propagation as al-

gorithm 3. The second propagation works as follows:

Algorithm 4 *EDT by Propagation with Multiple Oriented Neighborhoods*

```
for d = 0 → d_max do {Second loop}
  for all (p, dp) in bucket(d) do
    if 0 < dp_y < dp_x then {First octant}
      sta ← 0
      end ← 1 + step_end
      for n_x = 1 → n_min(d) do
        for n_y = sta → end do
          D_new ← dist_E(dp + n)
          if D_new < D(p + n) then
            D(p + n) ← D_new
            put (p + n, dp + n) in bucket(D_new)
          end if
        end for
        sta ← sta + dp_y/dp_x
        end ← end + dp_y/(dp_x − 1)
      end for
    else
      ... {The other 7 octants are treated similarly}
    end if
  end for
end for
```

One should notice that $dp_y/(dp_x + 1)$ is added to *end* instead of adding dp_y/dp_x as suggested by the upper bound of (3.9). Indeed, in algorithm 1, the information from object pixel q may propagate one step out of $VP(q)$ before it stops propagating. Using $dp_y/(dp_x + 1)$ relaxes the upper bound on n_y and addresses this problem.

3.4 Computational Complexity

Comparing the complexity and computational costs of DT algorithms is a complex task. For $n \times n$ images, approximate algorithms such as Danielsson's [37], the chamfer DT [10], Leymarie's [95] or Ragnelmam's

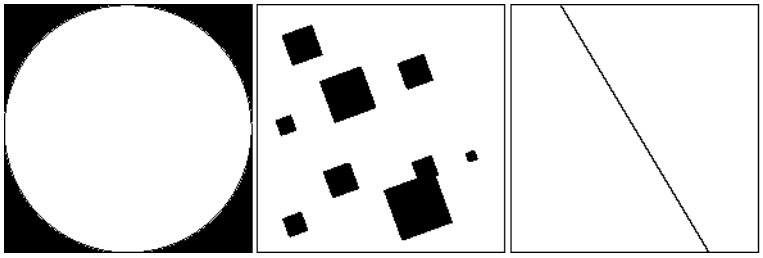

Figure 3.7: Test images where object pixels are black and the DT is computed at every white pixel. The size of the "circle" image varies from 200×200 to 2000×2000. The orientation of the other two images varies from 0 to 90^o.

[127, 128] have a fixed $O(n^2)$ cost regardless of the image content. Yamada's parallel algorithm [185] has a $O(d.n^3)$ cost, proportional to the maximum distance d found in the image. More complex exact algorithms such as Ragnelmam's [127], Eggers' [42] and Saito's [140] have costs that are highly dependent of the image content and can vary between $O(n^2)$ and $O(n^3)$. For those, experiments give a better knowledge of their complexity than theoretical considerations.

Among the methods described in chapter 2, those of Eggers and Saito appear to be the fastest exact Euclidean DTs. They are used here as the benchmark against which our algorithms are compared. Three versions of our algorithms are considered in this comparison: the approximate PSN and the exact PMN and PMON algorithms. In the PMN algorithm, only the fourth of the neighborhoods N in the same direction as (dp_x, dp_y) is used.

The choice of images on which the tests are performed is subjective and may dramatically affect the results. We perform 3 tests, illustrated at Figure 3.7. Test 1, suggested by Saito, is an image filled with an empty disk, whose size varies from 200×200 to 2000×2000. Test 2, suggested by Eggers, is made of random squares covering 15% of the image in total, with an orientation varying between 0^o and 90^o. Test 3, our suggestion, is the worst case scenario foreseen by Eggers and Ragnelmam for propagation DTs: a straight line across the image with an orientation varying between 0^o and 90^o. Images for tests 2 and 3 are 1024×1024 pixels large.

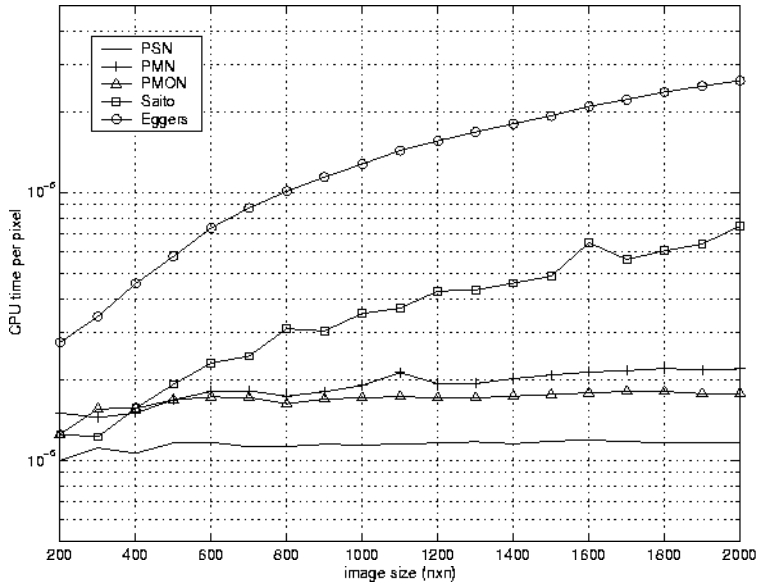

Figure 3.8: Test1: Saito's empty circle image. Note the logarithmic scale for the CPU times.

The algorithms were implemented in C on a Pentium II computer running at 233 MHz. We compare the CPU time required by each algorithm. Results are shown at Figures figure 3.8 to figure 3.10 for tests 1 to 3, respectively.

Test 1 (figure 3.8) illustrates the CPU time per pixel for images of increasing size. An optimal $O(n^2)$ algorithm should therefore give a constant result. It is of course the case of the approximate PSN algorithm. PMN and PMON display a very slight increase with the image size, but the relative cost compared to PSN remains less then a factor 2, even for 2000×2000 images. PMON is faster than PMN as soon as the image is larger than 400×400. Saito's and Eggers' algorithms both have a computational cost per pixel increasing with the image size. Actually, their complexity is $O(n^3)$ for $n \times n$ images. Nevertheless, Saito's algorithm is as fast as PMN or PMON for small images ($n \leq 400$).

Test 2 gives more complex results. All algorithms work faster on images where the squares are either oriented at 0^o, 45^o or 90^o, and slower for orientation near to $22,5^o$ and 67.5^o. This effect is more accentuated for

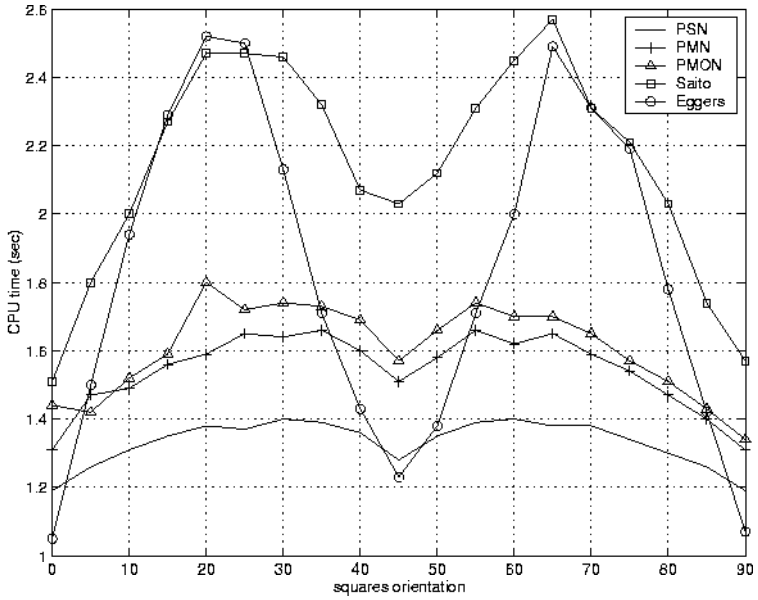

Figure 3.9: Test2: Eggers' random squares

Eggers' and Saito's. In average, PSN is the fastest, requiring $1.33sec.$ PMN follows (1.54), then PMON (1.60), Eggers (1.82) and Saito (2.14), but these are not significant differences.

Test 3 shows the worst-case scenario for the propagation methods of Ragnelmam and Eggers, for which they are known to have a $O(n^3)$ complexity. As in test 1, the disparity between CPU costs is such that the results are here displayed using a logarithmic scale. Eggers' DT performs very poorly for any orientation but the horizontal, vertical and 45 diagonal. Saito's DT performs reasonably well for orientations between 0 and 45, but very poorly for orientations around 60. In average, Eggers and Saito are respectively 14 and 7.5 times slower than PSN. In the worst orientation, Eggers and Saito are respectively 25 and 28 times slower than PSN. On the other hand, PMN and PMON only show a limited influence from the orientation. In average, they are 1.45 and 1.3 times slower than PSN. Even in the worst case, they are still less than twice slower than PSN. In contrast with the results of test 2, PMON out-performs PMN for all orientations.

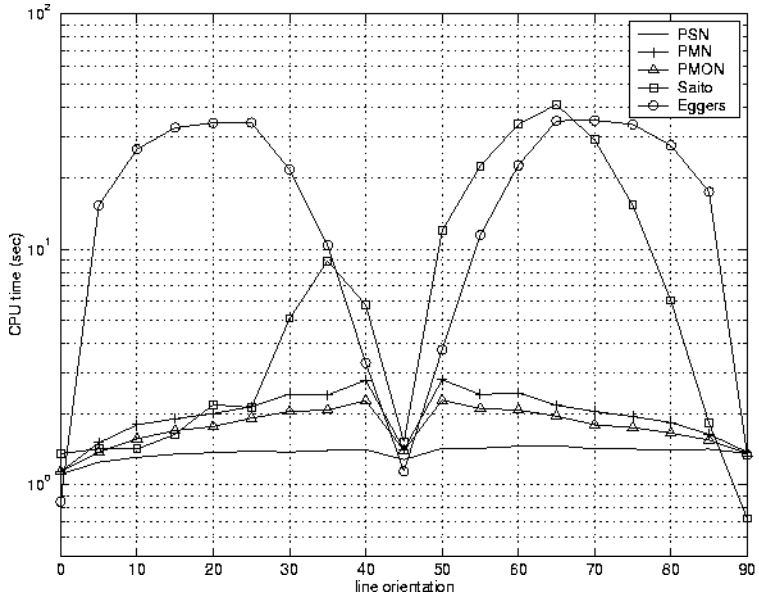

Figure 3.10: Test3: the worst case straight line.

In Figure 3.10, let us also note that Saito's algorithm is the only one that has a non-symetric behaviour around 45^o. This can easily be understood if one considers that all other algorithms treat all axis in the same way, while Saito's chooses to process the image first along one axis, then along the other. This choice can have a major influence on the computational cost. Unfortunately this is data dependent, which precludes the choice of an ordering of the axis that would be optimal in all cases. This issue is further studied for 3D data in chapter 6.

In conclusion, both PMN and PMON seem to have a $O(n^2)$ complexity. Even in the worst case, the cost required to produce the exact EDT is less than double the cost of the approximate PSN. PMON out-performs PMN when the DT includes very large distance values, but its complexity is not justified for smaller images[1]. Both algorithms are major improvements over the methods of Eggers and Saito.

[1]This remark may be hardware dependent. In particular, PMON should out-perform PMN in all cases on a SUN Sparc workstation, where the relative cost of the floating point operations is lower.

3.5 Using the Euclidean DT to implement mathematical morphology

3.5.1 Mathematical morphology operators

Dilation and erosion are the basic operators of mathematical morphology [143, 144]. The dilation of a set of points X by a structural element B is written $X \oplus B$ and is defined as

$$X \oplus B = \{ x + b | x \in X \text{and } b \in B \} \tag{3.10}$$

The erosion, written $X \ominus B$, is the dual of dilation, i.e. the complement of a dilation performed on the complement set of X. Other morphological operators can be derived by combining dilation and erosion. For instance, the opening and closing are defined as

$$X \circ B = (X \ominus B) \oplus B \tag{3.11}$$
$$X \bullet B = (X \oplus B) \ominus B \tag{3.12}$$

Symmetrical and circular structural elements (SE) play a central role in mathematical morphology in the continuous plane, because they provide an isotropic treatment of the image. On the other hand, for digital images, circular SE are rarely used because other shapes are easier and faster to implement.

3.5.2 Fast implementations

Fast implementations of the dilation operator usually rely on a property of this operator. For instance, dilation is an associative operation, i.e.

$$X \oplus (B \oplus B') = (X \oplus B) \oplus B' \tag{3.13}$$

Some structural elements can be decomposed into simpler elements, as illustrated at figure 3.11. Applying the dilations with the smaller elements iteratively instead of the large SE at once reduces the complexity of the dilation. The dilation of an $n \times n$ image by a SE of radius d has a $O(d.n^2)$ complexity. In addition, mono-dimension dilation can be performed in $O(n^2)$, so that the square SE's complexity is also $O(n^2)$. Unfortunately, the square and diamond SE are very poor approximations of a circle. A common improvement is to use a combination of

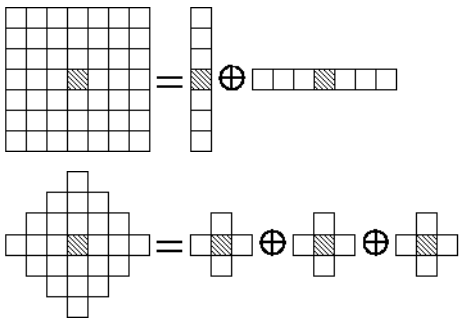

Figure 3.11: The square and diamond structuring elements are separable.

both, which leads to an octagonal SE.

Another property that can be used to implement the dilation efficiently is

$$X \oplus B = X \bigcup (\delta(X) \oplus B) \tag{3.14}$$

where $\delta(X)$ is the edge of X. This is valid for any connected SE that contains (0,0). If l is the length of the contour of X, then the computational complexity is reduced to $o(l.d)$ for a SE of radius d, plus a small $O(n^2)$ term to determine the pixels belonging to $\delta(X)$.

By combining the properties of equations (3.13) and (3.14), we have the basis for contour-processing algorithms for decomposable SE as in **Van Vliet** and **Verweer** [173], whose complexity is further reduced to $O(l_{\max}.d)$ where $l_{\max}$ is the maximal size of the contour during the iterations with elementary SE. Finally, **Vincent** [169] proposes an algorithm using both contours $\delta(X)$ and $\delta(B)$, of the set X and the structural element. This algorithm has a complexity proportional to the product of the number of pixels in each contour, which means $O(l.d)$ for a circular element of size d. It can also be expressed as $O(A)$ where A is the cardinal of $(X \oplus B) \backslash X$ where $\backslash$ is the set difference, i.e. $X \backslash Y = X \bigcap Y^c$.

A third approach applies to structuring elements B that are balls, that are defined as

$$B = \{b | dist_M(b, (0,0)) \leq d\} \tag{3.15}$$

then, the dilation by B can also be expressed as the threshold of a distance transformation, i.e.

$$X \oplus B = \{x | DT(x) \leq d\} \tag{3.16}$$

where DT(x) is the distance transformation from object X. The square and diamond SE can be considered as balls for distances defined using the $dist_4$ and $dist_8$ metrics respectively. Since the corresponding DT algorithms are of a $O(n^2)$ complexity, so are the dilations.

Ragnelmam [126] proposes to combine contour processing and DT thresholding. He uses an algorithm similar to PSN, and stops it as soon as d is reached. The computational complexity is of course $O(A)$ where A is the cardinal of $(X \oplus B)\backslash X$, i.e. the set of points reached by the propagation. Unfortunately, PSN or Ragnelmam's variant of it are approximate EDTs. As mentioned in chapter 2, the non-systematic errors of these algorithms can lead to situations where

$$X \not\subseteq (X \bullet B) \tag{3.17}$$

which contradicts the axioms of mathematical morphology.

3.5.3 Morphological dilation using PMN

The above problem can of course be solved by using an exact EDT by propagation algorithm, such as PMN. A further improvement can be obtained by merging the two steps into one. This way, we can perform the threshold dynamically when each point is considered in the propagation. The algorithm is then written

Algorithm 5 *Mathematical morphology dilation by a variant of PMN*

Input: an image I with object X in it. A ball B_d.
Output: an image D where $D(p) = 0$ if $p \in X \oplus B_d$ and $D(p) = d^2 + 1$ otherwise

for all $p \in I$ **do** {Initialization}
  if $p \in X$ **then**
    $D(p) \leftarrow 0$
    if $p \in \delta(X)$ **then**
      put $(p, (O, O))$ in $bucket(0)$

```
      end if
    else
      D(p) ← d² + 1
    end if
  end for

  i ← 0
  for i = 0 → d² do {Main loop}
    N ← N_min(i) {taken from table 3.1}
    for all (p, dp) in bucket(i) do
      D(p) ← 0
      propagated ← FALSE
      for all n ∈ {(0,1),(0,−1),(1,0),(−1,0)} do
        D_new ← dist_E(dp + n)
        if D_new < D(p + n) then
          D(p + n) ← D_new
          put (p + n, dp + n) in bucket(D_new)
          propagated ← TRUE
        end if
      end for
      if propagated = FALSE then
        for all n ∈ N do
          D_new ← dist_E(dp + n)
          if D_new < D(p + n) then
            D(p + n) ← D_new
            put (p + n, dp + n) in bucket(D_new)
          end if
        end for
      end if
    end for
  end for
```

This implementation supposes that we know the size d of the ball B_d that we use as structuring element. The algorithm can of course be easily modified to allow the decision on the size d to be made during the dilation. The algorithm is fully progressive in the sense that it generates all dilations $X \oplus B_{d'}$ for every size $0 < d' \leq d$, during the process of creating $X \oplus B_d$.

3.5.4 Discussion

With a computational complexity proportional to the size of $(X \oplus B) \setminus X$, this algorithm performs as well as, but not significantly better than Vincent's [169]. Both algorithms out-perform any other method, either in precision or in cost.

Both algorithms have additional capabilities that can make them more interesting for a particular application: On one hand, Vincent's algorithm can also be used to perform dilations with structural elements of arbitrary shape. On the other hand, this algorithm is limited to Euclidean balls, but can be stopped at any distance d and provide $X \oplus B_d$. For instance, it can easily perform dilations with elements B_d of increasing size, until some criterion is met, as we do in the next chapter.

Chapter 4

Application: morphometry of nerve cross-sections

In this chapter, we illustrate the use of the EDT by propagation algorithm with a first medical imaging application, the automatic morphometry of nerve cross-sections. In this application, the EDT is used as a criterion to separate connected neuronal fibers and evaluate the myelin sheath thickness around the axons. First we describe the nature of the images to be treated and of the entities to detect. Then, we present the full segmentation procedure. Finally, we present typical results and study their validity.

4.1 Introduction

In this introduction, we briefly describe the anatomy of the nervous system and the nature of the cells we intend to observe. Then, we present a few usual approaches to nerve morphometry. Finally, we describe the methodology used to acquire the images for this study.

4.1.1 Anatomy of the nervous system

The basic structural and functional unit of the nervous system is the nerve cell or neuron, illustrated at figure 4.1.a [1] . All neurons have a cell body which contains the usual cellular organelles common to all cells in the body. Most nerve cells have processes called dendrites, which act like

[1] The illustrations found in this section are taken from [45]. This online anatomy course has also largely inspired the text of the section

antennae, in that they receive input to the cell. Most neurons also have a single long process called an axon, which is capable of transmitting a pulse of electricity from the cell body to some distant target either in the brain or the periphery. These axons may be quite long, up to a meter or more for the axons connecting the spinal cord to the foot. Axons usually break up into terminal branches near their target. These branches end in swellings which make a specialized contact with the target cell. If the target cell is another neuron, the swelling is called a bouton, and the specialized contact a synapse. If the target is a muscle fiber, the bouton is a motor endplate and the synapse is a neuro-muscular junction.

The nervous system also contains cells which are not neurons and which do not directly participate in the task of sending and receiving electrical signals. These supporting cells are called glia. We are particularly interested in those that form myelin sheaths around axons in the central and peripheral nervous systems.

Indeed, axons are generally not naked, as in figure 4.1.a. Rather, they are wrapped into an insulating material called myelin. The presence of a myelin sheath around an axon increases the velocity at which it conducts a nerve impulse down its length. The myelin sheath (see figure 4.1.b) is formed by flattened out cells that wrap themselves jelly-roll style around the axon. In the central nervous system the cells that form the myelin sheath are called oligodendroglia; in the peripheral nervous system they are simply called axon sheath cells. Figure 4.1.c is a highly schematic drawing of a flattened axon sheath cell. To the right of it is a cross-section of a myelinated axon, on which one can see how the axon sheath cell wraps around the axon.

The sheath itself is essentially composed of flattened cell membrane, with all of the cytoplasm squeezed out except in the outermost layer. The major component of a cell membrane is the phospholipid bilayer. With many layers of membrane stacked on top of one another, it has a fatty appearance due to the presence of this phospholipid. Myelinated axons therefore have a glistening white appearance in the central and peripheral nervous systems, and are referred to as white matter. Areas containing mainly cell bodies tend to lack myelin and are referred to as gray matter. The terminology for both the central and the peripheral nervous system is found at figure 4.1.d.

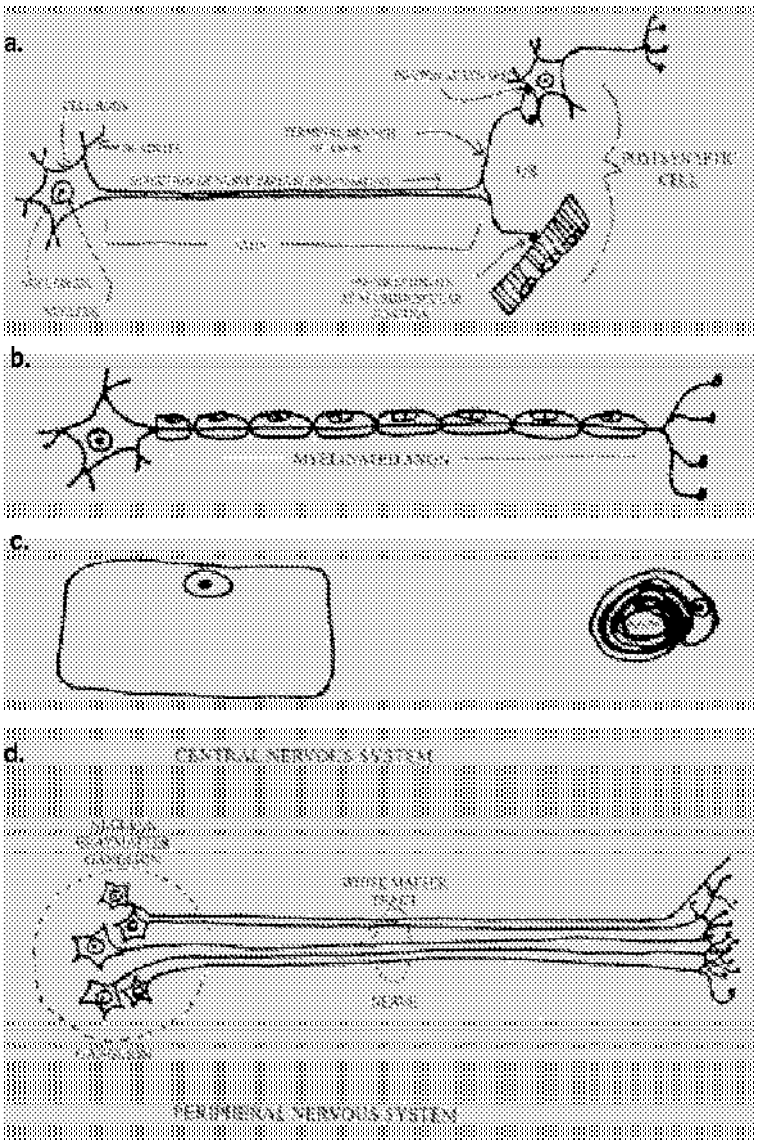

Figure 4.1: The neurons. **a.** a neuron. **b.** a myelinated fiber. **c.** a myelin sheath cell. **d.** terminology for the central and peripheral nervous systems.

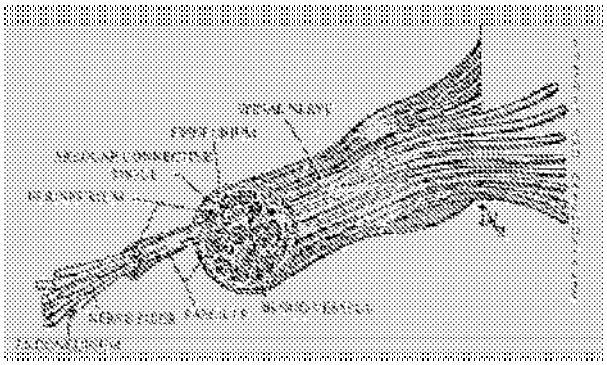

Figure 4.2: A nerve

A nerve is a bundle of axons traveling together in the periphery. If the nerve contains sensory axons only, it is called a sensory nerve. If it contains motor axons (going to muscles) only, it is called a motor nerve. Most nerves in the body contain both sensory and motor axons and are therefore called mixed nerves.

Most axons in any nerve are myelinated, which gives nerves their glistening white appearance. In addition, there are some connective tissue elements associated with nerves (see figure 4.2): individual axons are enveloped in a connective tissue wrapping called endoneurium. Bundles, or fascicles, of axons are wrapped in a connective tissue covering called perineurium. The nerve as a whole is enveloped in a connective tissue sheath called the epineurium.

4.1.2 Nerve morphometry

Morphometric studies of nerves or fiber tracts involve information about alterations in nerve bundle size, number or size of the axons. They have been shown to be of great value in detecting developmental or pathological abnormalities [70, 25, 116, 78, 40]. They have also been broadly used in experimental nerve research [120, 18].

Most techniques used for estimating nerve and fiber parameters are based on highly time consuming manual measures. For instance, in the study below, the sciatic nerve contains approximately 15000 myeli-

nated fibers, included in 85 images of 1850×1234 pixels, or 500 MBytes of raw data. Therefore, the information is usually sampled by selecting only a few of the images in the nerve cross-section. Unfortunately, the large variability in fiber distribution according to function, i.e. sensitive or motor nerves, or to the specie specificity, precludes the selection of a sampling pattern that is reasonably representative of the nerve. **Torch** [158] estimates that sampling schemes involving less than 50% of the images provide an unreliable measure of myelinated fiber distribution.

Alternatively, an automatic image analysis tool solves the problem by allowing to examine all the available material. Algorithms are usually divided in two steps. First, the image is analyzed with a local operator that classifies pixels between the various tissues types. For this stage, **Jain** [79], **Garbay** [62] or **Thiran** [153] rely on thresholding, sometimes preceded by filtering. Secondly, the image is analyzed at the structural level using a variety of tools such as region growing segmentation [79], grouping of edge elements [62], or mathematical morphology [153]. Unfortunately, none of these methods can handle multi-part objects such as axons surrounded by the myelin sheath.

Alternatively, **Amini** [2], **Fok** [55] or **Elmoatoz** [43] rely on active contour models, or snakes, to handle both local and structural analysis in one step. After detecting candidates through a global tool such as the Hough transform, each region of interest is processed individually with an explicit active contour model evolving towards the real contours of the cell. Unfortunately, such methods tend to be too computationally expensive for the large data sets required by a full study. Also, it is unclear whether any of these models could handle the large size variability encountered in nerve fibers.

4.1.3 Image acquisition

In order to visualize the myelinated fibers in the microscopic images of the nerve cross-sections, one applies the following operations.

4.1.3.1 Animals and tissue preparation

The images used in this study were provided by Eduardo Romero, from the Neural Rehabilitation Engineering Laboratory of the UCL. The sciatic nerves were obtained from one female cat ($3250gr$). The cat was

anaesthetized with Sodium pentobarbital ($30mg/kg$ I.M), the abdominal aorta was canalized with a catheter $14G$, between the renal branches and iliac bifurcation and perfused with a solution of phosphate-buffered paraformaldehyde. The sciatic nerve was dissected from $2cm$ distal from the spinal cord to the popliteal fossa, after the tibial and peroneal bifurcation. A $3cm$ segment was excised where the sural branche was coming out from the nerve (i.e. $4cm$ proximal to the sciatic nerve bifurcation in a tibial and a peroneal branches). Nerves were stored in Karnovsky fixative (2.0% paraformaldehyde and 2.5% glutaraldehyde in $0.1M$ Sodium cacodylate buffer solution) for 24 hours, postfixed in 1% Osmium tetroxide during 4 hours and embedded in epon $LaddLX-112$. Semi-thin nerve cross sections ($1\mu m$) were cut on a Reichter Ultracut microtome (Reichter, Wien, Austria) and stained with toluidine blue.

4.1.3.2 Photography

Photomicrographs were obtained using a color filter system in a Zeiss microscope with an 40× PLANAPO oil immersion objective and a 10× ocular. A total of 85 photographs represented about 98% of the entire nerve. Images were then digitized by a system Nikon $25-1000$ software with a size of 1850×1234 pixels. Each photograph was enlarged by the digitizing process (×3.3) to give a final magnification of ×1695. A microscale ($0.01mm$, Wild, Switzerland) was processed in the same way for scaling. The pixel size was found to be $0.1135\mu m$.

4.2 Segmentation procedure

When a nerve section is correctly preserved, fixed and stained, it shows the myelin appearing darker in the images (see figure 4.3). Therefore, we can define nerve fibers in our images as objects where a clear region is surrounded by a dark myelin sheath of constant thickness. Beyond this basic definition, let us review a few properties that can be used to differentiate fibers from other structures.

Fibers have a round or ellipsoidal shape, but it is quite frequent to observe some kind of deformation [141, 129]. A shape parameter, such as $perimeter^2/surface$, for the axon and myelin sheath - can be used as a helpful criterion, but only in a loose fashion.

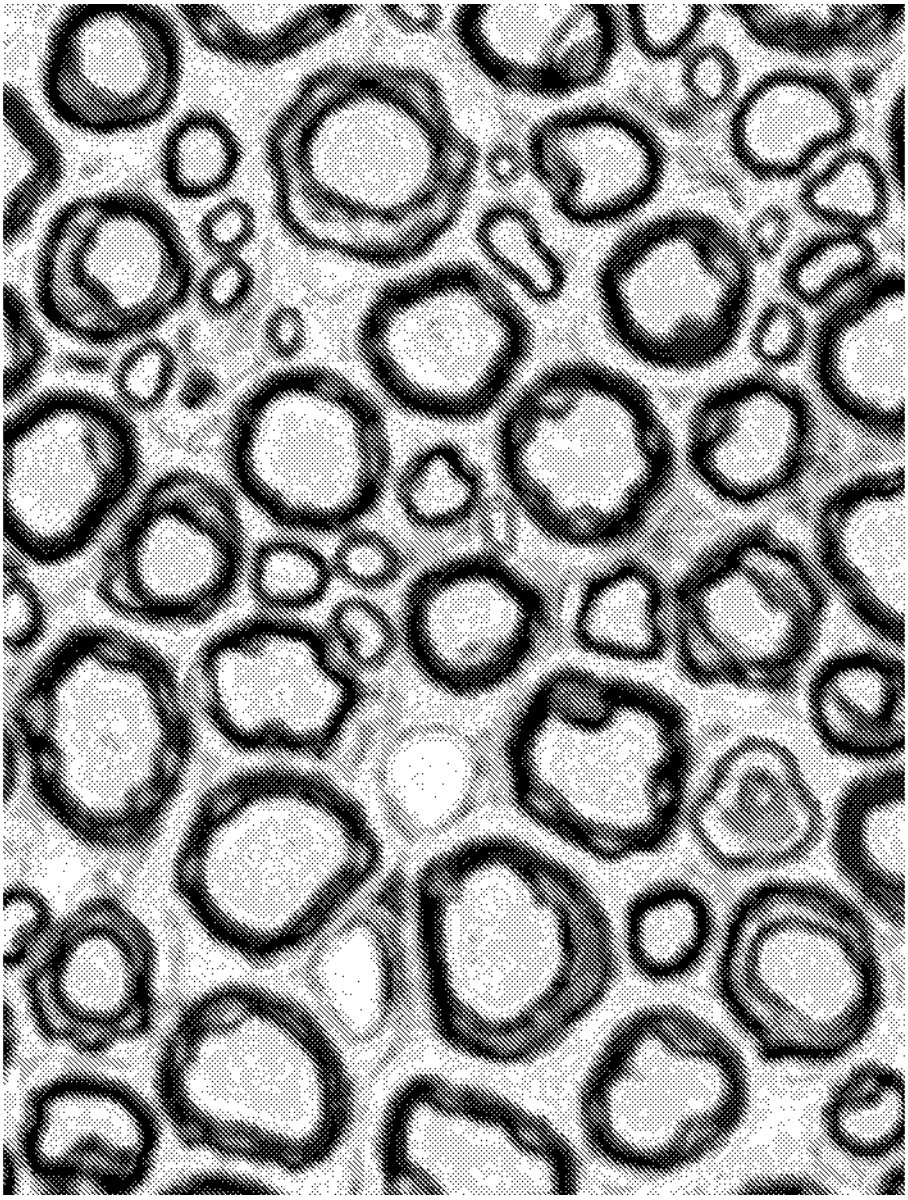

Figure 4.3: Part of a typical image in the nerve cross-section after tissue preparation, staining and digitization

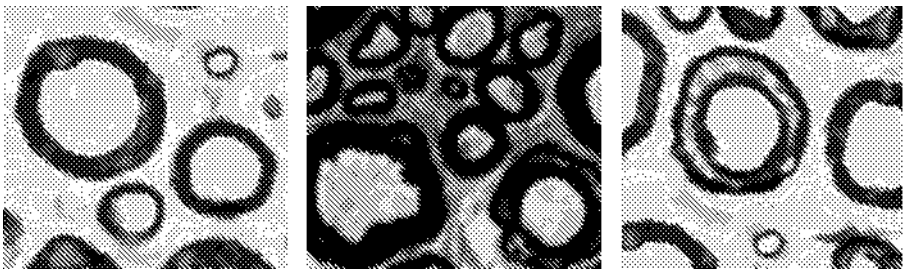

Figure 4.4: Typical irregularities in fibers. *Left:* size can vary from 2 to $20\mu m$ (diameters). *Center:* densely packed axons are connected. *Right:* bad fixation and coloration leaves bright rings in the myelin sheath.

Rushton [137] established that the ratio d/D, between the diameter of the axon and that of the whole fiber, is close to 0.6, a value that optimizes the nerve impulse transmission in the myelinated axon.

Unfortunately, the fibers also present a number of highly variable features that can hinder the efficiency of detection algorithms (see figure 4.4). For instance, for mixed nerves containing both sensitive and motor axons, the diameter of the fibers can vary between $1\mu m$ (the light microscope resolution limit) and $20\mu m$. Another difficulty comes from the spatial distribution of the fibers: they can be either isolated or densely packed together, which can makes their separation a crucial problem. Finally, fixation and coloration problems can create bright spots within the myelin sheaths, multiple rings, etc.

The method we propose is divided in five steps. First, pixels are classified into myelin (black) or non-myelin (white) pixels according to their luminance. Secondly, the resulting binary image is filtered with connected morphological operators, using rules derived from the above description. Axon candidates are identified in the equivalent zonal graph. Thirdly, the thickness of the myelin sheath around each axon is evaluated, which discards inappropriate candidates and separates adjacent fibers. Fourth, additional morphological criteria are used to detect and discard false fibers. Finally, oblique cuts are detected and a geometrical correction is performed when needed.

4.2.1 Pixel classification

As pointed out by **Trier** [160], locally adaptive threshold methods are more robust than global ones. In our case, this is particularly important since tissue processing and dye preparation often lead to inhomogeneous staining. This is observed as a smooth variation of the average luminance over the image. Thus, the threshold level for a pixel is chosen on the basis of the histogram of a subimage around it, typically a square of $25 \times 25 \mu m$ that contains a couple of axons. To maintain a low computational cost, the histogram analysis is only performed for a few locations of the window, and the threshold levels are bi-linearly interpolated in between.

The subimages contain three types of tissue: the myelin, the endoneurium and the axons. This corresponds to 2 or 3 lobes in the histogram, depending on whether the endoneurium presents any degree of coloration differentiating it from the axons. We use a simple heuristic to select a threshold level between the first two lobes. The 15^{th} and 50^{th} percentile in the histogram of the grey levels are considered typical values of the first and second lobes. The mean of these two values is taken as the threshold level.

4.2.2 Connected operators filtering

The resulting binary image presents a number of artifacts that are best expressed and handled in terms of regions and their properties. This can be formalized using connected morphological operators, as described by **Heijmans** [72].

The binary image is considered as a partition $P(X)$ of the set X of pixels into black and white regions. As illustrated in Figure 4.5, the zonal graph of the image is the graph that takes the regions of $P(X)$ as vertices and whose arcs represent the adjacency of the regions. The graph also specifies the color of the regions it represents. Given two partitions P and P' of the image, P is coarser than P' if $P' \subseteq P$. A morphological operator ψ is called connected if the resulting partition $P(\psi(X))$ is coarser than $P(X)$, for any set X. In other words, connected zones are either left untouched or changed altogether. In the common case where connectivity is based on adjacency, connected operators can be described and implemented by re-coloring and merging vertices in the zonal graph.

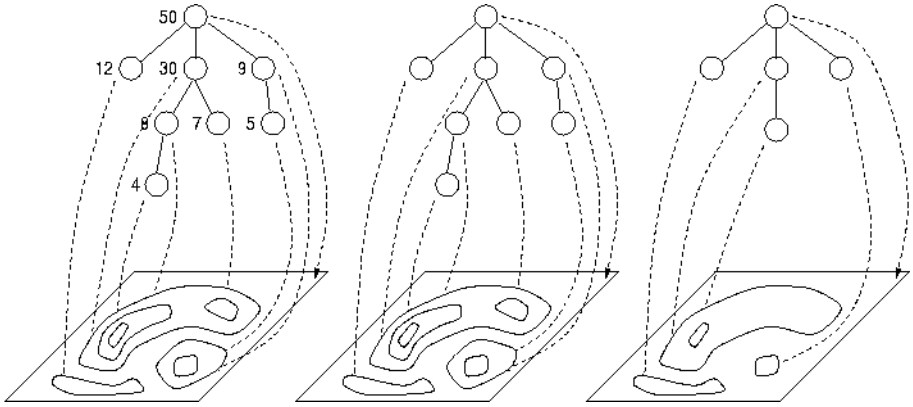

Figure 4.5: The area operator flips zones with an area of less than 10 in the original image (*left*). It can be seen as a re-coloring (*center*) and merging (*right*) of vertices in the zonal graph.

A well know connected morphological operator is the morphological opening by reconstruction, where objects that are too small to contain the structural element of the original erosion are deleted, while the other objects are left unchanged. More complex criteria can be of course defined, either considering each zone separately (it is then called a grain operator) or considering the relationships between zones and their neighbors. We use both hereafter.

Different connectivities yield different zonal graphs. In our case, we use 8-adjacency for foreground pixels and 4-adjacency for background pixels. This defines a topology similar to the continuous case, and in particular the zonal graph is then a tree, i.e. a graph without cycles. The following connected operators are applied:

- Noise in the original image creates small mis-labeled areas in the binary image. Those are removed by applying the area operator of figure 4.5 for all areas smaller than the smallest axons ($0.5\mu m^2$). Unfortunately, this operator is not stable. Applied iteratively it can fail to converge and oscillate between two solutions. Thus, we restrict its action to the leaves of the zonal tree, i.e. the regions with only one neighbor.
- Fibers have a bright center surrounded by a black ring i.e. a black

region with two neighboring white ones. Thus, all black leaves in the zonal tree do not represent a useful feature and are removed.

- Fixation and coloration problems can separate the myelin sheath in two parts (see figure 4.4). It appears as a white ring surrounded by two black rings. These white rings should be re-colored in black. White rings are detected because the gravity center of the zone is located outside of it. Two cases are possible: either the ring is open and it is a leaf of the zonal graph, it is then merged with its only neighbor; either the ring is closed and has 2 neighbors in the zonal graph, the three vertices of the graph are then merged together.

After this filtering, axon candidates are identified as white leaves in the zonal tree satisfying both a size criterion ($1\mu m < d < 12\mu m$) and a shape criterion ensuring the compactness and approximate circularity of the axon, depending on the $perimeter^2/area$ ratio.

4.2.3 Myelin sheath thickness evaluation

Unless fibers are very sparse, some of them are connected in the binary image. In the zonal tree, this corresponds to several white leaves that share the same neighboring black zone. This section deals with the division of this black zone into sub-regions that are either myelin sheaths surrounding axon candidates or artifacts to be merged with the background.

For instance, let us consider the example of figure 4.6. The black zone includes 9 white leaves, numbered $1-7, x, y$. Leaves x and y were discarded at the previous stage, because they lack circularity to be proper axon candidates. Among the 7 axon candidates, areas 1 to 6 are true axons while area 7 is an artifact.

Let us first consider a single white area. We evaluate the thickness of the myelin sheath around it as follows: we define X_d as the set of pixels at a distance d of a set X of pixels

$$X_d = (X \oplus B_{d+1}) \backslash (X \oplus B_d) \tag{4.1}$$

with B_d a ball of size d, $\oplus$ the morphological dilation and $\backslash$ the set difference. We define the thickness of the myelin sheath around a white

Figure 4.6: Axon separation by distance transform. From left to right: original image; result of the connected operators filtering; distance map corresponding to the dilation process; detected fibers.

area X as the smallest distance d for which there are more white than black pixels in X_d.

This is very efficiently implemented using the algorithm of section 3.5 for the morphological dilation. In particular, the set X_d is composed of the pixels present within the buckets structure when $bucket(d)$ is being processed. Therefore, the termination criterion can be computed for all values of d during the dilation, and the propagation process can be stopped as soon as needed.

Let us now consider all the axon candidates that are leaves of the same black area. We apply the previous procedure to each candidate, by order of decreasing size. In figure 4.6, this separates fibers numbered from 1 to 5. For area number 6, the propagation process reaches pixels that were previously considered as belonging to the myelin sheath around area number 2. These pixels are re-labeled as belonging to the sheath around the axon they are closer to. The resulting edge between the two fibers corresponds either to the thickness of the smallest fiber, or to the iso-distance line between the two axons. Area number 7 is entirely included inside the myelin sheath surrounding area 1. Therefore, it must be an artifact and it is discarded.

4.2.4 False positive detection

The above detection procedure can produce two kinds of errors: missed detection when a fiber is not found and false positive when an image feature is wrongly considered to be a fiber. False positive is considered a worse problem because it is most likely to introduce bias in size distribution statistics, as most false positives are of a small size. Missed

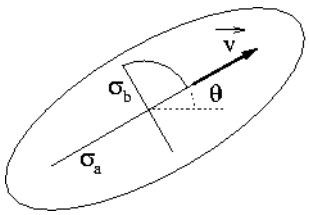

Figure 4.7: Obliquity parameters of a fiber

detection is only detrimental if its rate is size dependent, which does not appear to be the case.

In order to minimize the number of false positives, we apply three additional tests on the detected fibers. First, each individual fiber should have a d/D ratio close to 0.6. Secondly, fibers should be mostly surrounded by endoneurium, not by other fibers. This removes common false positives located in the space between 3 neighboring axons. Finally, isolated features are discarded.

4.2.5 Correction of obliquity

Even in expert hands a perfect transversal cut is almost impossible, and a certain degree of obliquity always remains. In that case, most fibers appear as ovals instead of disks. Fiber orientation provides an estimation of the obliquity. It can be evaluated by inspecting the principal axes of the fibers (see figure 4.7). If the long axes are globally aligned, this global alignment corresponds to the orientation of the cut. The ratio between the average length of the long and short axis denotes the angle of the cut.

Practically, we define the obliquity vector $\vec{v_i}$ for the ith fibre as follows:

$$\vec{v_i} = |v_i| e^{j\theta_i} \tag{4.2}$$

where θ_i is the angle of the longest axis with the horizontal and $|v_i|$ is

$$|v_i| = \frac{\sigma_{ai} - \sigma_{bi}}{\sigma_{bi}} \tag{4.3}$$

with σ_{ai} and σ_{bi} the lengths of the long and short principal axis of fiber i, respectively. The average obliquity over the N fibers of the image is

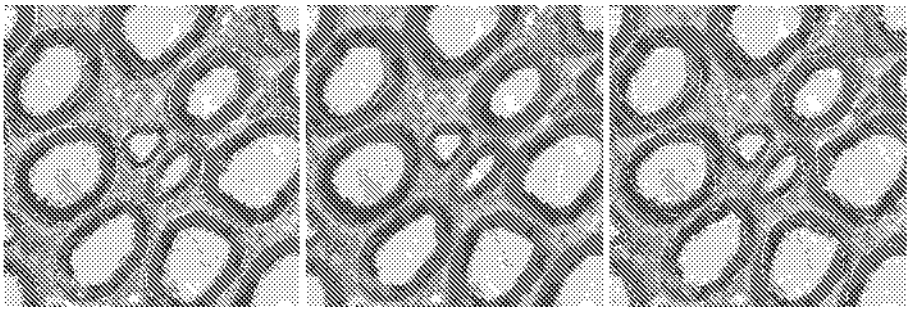

Figure 4.8: Correction of obliquity. From left to right: a) Fibers found on an oblique section; b) Principal axes of fibers c) Oblique-corrected fibers

$$\vec{v_{mean}} = |v_{mean}|e^{j\theta_{mean}} \tag{4.4}$$

$$\text{where} \quad |v_{mean}|e^{j2\theta_{mean}} = \sum_{i=1}^{N} \omega_i |v_i| e^{j2\theta_i} \tag{4.5}$$

with weighting factors ω_i. In practice, we use $\omega_i = \sigma_{bi}$, because larger fibers provide a more reliable estimate of the obliquity. In order to correct the obliquity of the cut, all fibers are contracted along the direction θ_{mean} by a factor $\frac{1}{(1+|v_{mean}|)}$, as illustrated at figure 4.8.

4.3 Experimental results

The accuracy of the method is assessed in three different ways. First the false positive and missed detection ratios are determined. Secondly the fiber size distribution is compared to that found by a manual procedure. Finally, the bias introduced by the automatic procedure is compared to the bias due to an arbitrary sampling.

4.3.1 Detection ratios

First, we measure the false positive and missed detection rates on a set of 30 images resulting in a total of more than 5000 fibers, i.e. on half of the images for one fascicle of a nerve. Detected fibers are visually evaluated by super-imposing the edges of the detected fibers on the original

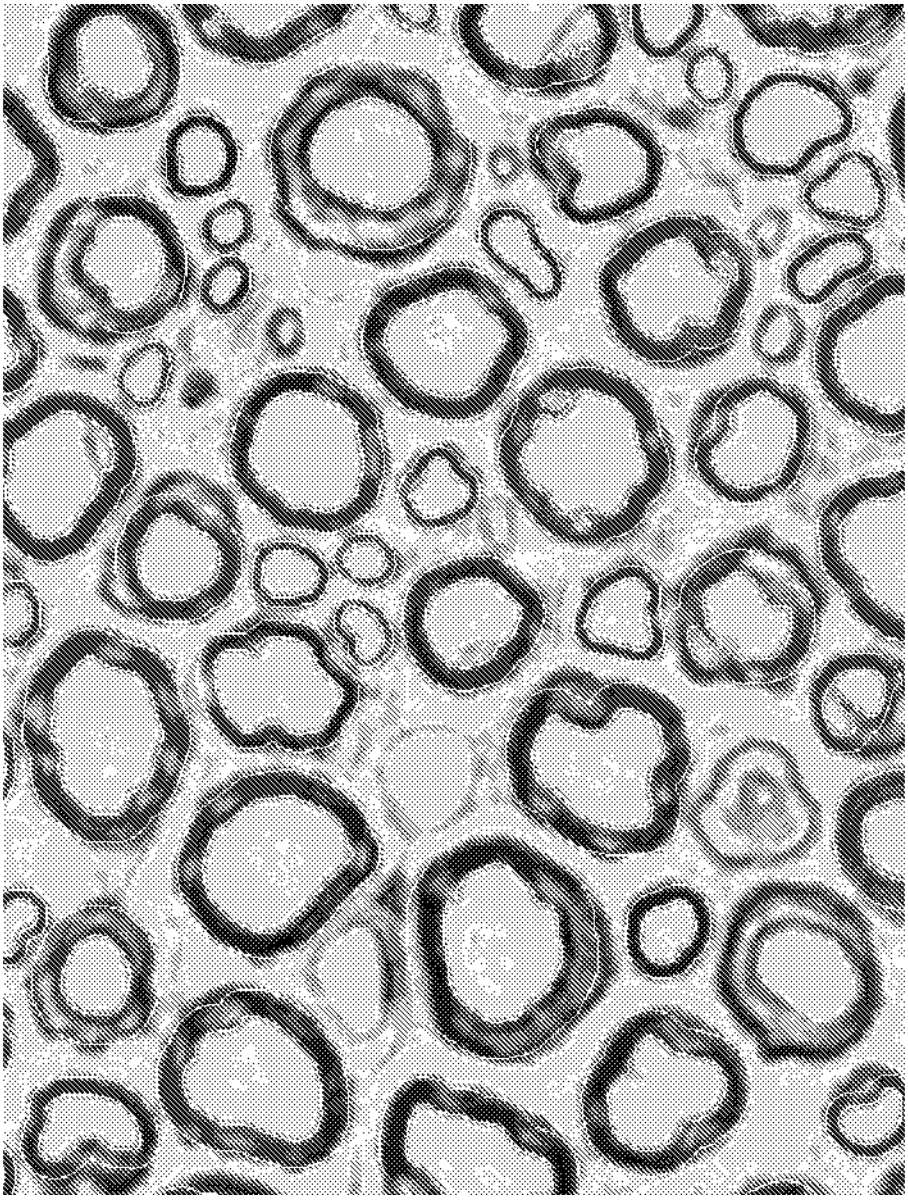

Figure 4.9: Detected fibers overlaid upon the image of figure 4.3

image, as illustrated in Figure 4.9.

For each image, false positives and missed fibers are manually counted.

Image ID	Number of fibers	Detected entities	Missed fibers	False positives
1	75	59	16	0
2	143	127	18	2
3	223	181	43	1
4	128	122	9	3
5	171	166	8	3
6	185	173	18	6
7	186	162	26	2
8	194	186	15	7
9	135	126	12	3
10	178	154	30	6
11	230	210	25	5
12	212	181	36	5
13	188	162	28	2
14	189	172	23	6
15	189	172	23	6
16	197	172	31	6
17	195	175	25	5
18	214	196	23	5
19	249	232	26	9
20	247	224	25	2
21	98	95	7	4
22	229	202	36	9
23	233	202	36	9
24	229	206	27	4
25	200	183	21	4
26	204	189	23	8
27	178	165	17	4
28	203	178	29	4
29	190	179	17	6
30	210	187	25	2
Total	5596	5035	691	130

Table 4.1: Detection results for the set of 30 images

The number of detected entities is determined by the program itself, and the true number of fibers is computed by implementing the correction. The results of this test are shown in Table 4.1. The average false positive rate is 2.5% and the missed detection rate is 11%. This is acceptable for a complete histological study, in which traditionally, information is summarized as histograms of fiber size distribution.

4.3.2 Comparison with the manual procedure

For one nerve, 9 images were selected at different nerve locations. Using standard software (NIH image for Macintosh), contour fibers were manually drawn and measured. A total number of 1936 fibers were found by the manual and 1899 by the automatic method. In figure 4.10, the manual and automatic histograms are depicted ($bin = 0.5\mu m$) in the upper left panel. The fit between both histograms H_1, H_2 made of n bins is evaluated with the χ^2 test, i.e.

$$\chi^2 = \sum_{b=1}^{n} \frac{(H_1(b) - H_2(b))^2}{H_1(b) + H_2(b)} \tag{4.6}$$

This value and the number of bins used determine the goodness of the fit. For the complete histogram, there is no significant difference ($p > 0.05$; $n = 26$).

In order to get finer results, the histogram is split into its two lobes. The first lobe shows a negligible difference confirmed by the χ^2 test ($p > 0.1$; $n = 10$). Interestingly, when the second lobe of the manual histogram is left shifted by $0.5\mu m$, the fitting is highly improved ($p > 0.5$; $n = 16$). This indicates a slight underestimation of the size of the large fibers by the automatic method.

4.3.3 Comparison with an arbitrary sampling

The set of measures (9 images) is arbitrarily split into two sub-groups, containing 960 and 976 fibers respectively. Figure 4.11 shows the fiber distributions found for each sub-group with the manual and automatic procedures.

The two peaks in the 4 histograms are at the same place (around $5\mu m$ and 11μ) and their pattern is equivalently biphasic. However, histogram

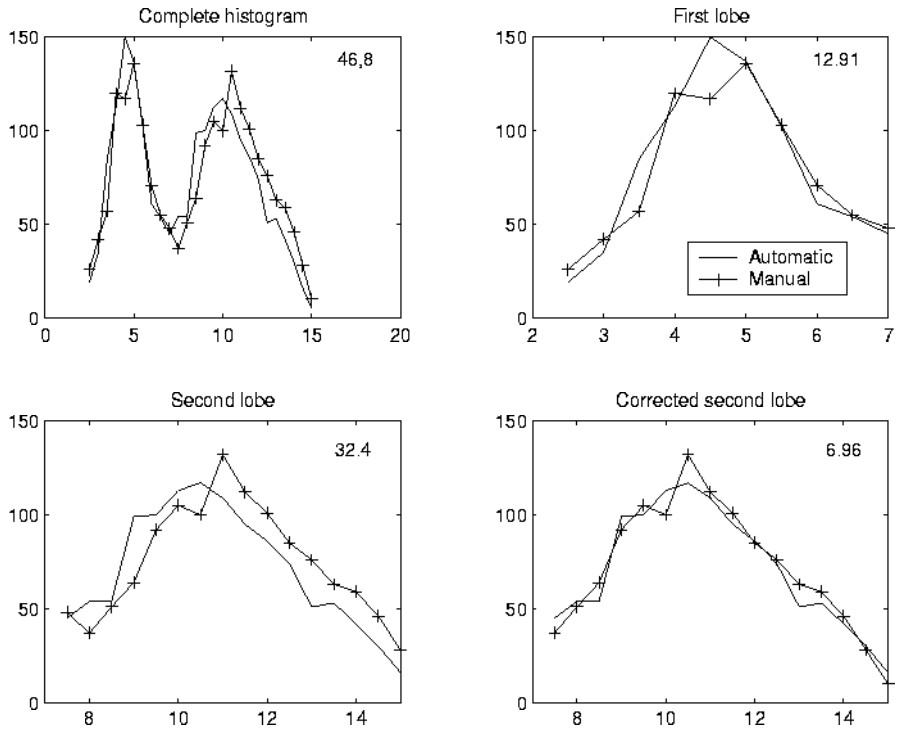

Figure 4.10: Comparison of fiber size distribution found by the manual and automatic procedures. χ^2 values are placed on the upper right corner for each histogram. The bin size is $0.5\mu m$.

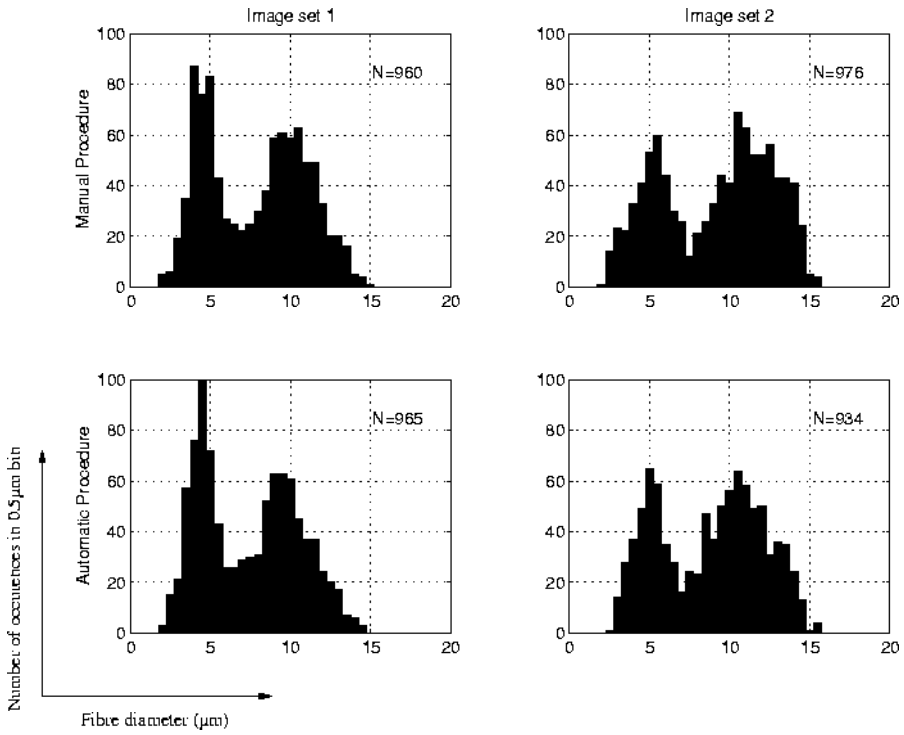

Figure 4.11: Comparison of fiber distributions for automatic (*above*) and manual (*below*) measures for data sets 1 (*left*) and 2 (*right*) ($bin = 0.5\mu m$)

shape is different for the two groups and the χ^2 statistical test, performed on manual normalized histograms, shows significant differences for them ($p < 0.0001$). In contrast, the automatic histograms correspond very closely to their manual counterparts ($p > 0.01$ and $p > 0.05$).

This last test shows that, even if it does introduce some bias, the automatic procedure is much less detrimental than the arbitrary choice of a limited number of images in the nerve cross-section, a critical decision in the manual morphometric study.

4.4 Discussion

In the above section, we show that the automatic procedure gives results that are very similar to the manual procedure for a given set of images. Because of the following considerations, we may even claim that the automatic procedure can be more accurate than the manual one, because it is able to process the entirety of the available data set.

Manual procedures use a uniform sampling scheme in order to maintain an equal representation of all locations within the nerve cross section. **Mayhew** [105] or **Fiola** [50] consider 10% of the entire nerve surface as the optimal area to examine. On the other hand, **Torch** [158] shows that myelin fibers are not randomly distributed within nerves and concludes that it would be necessary to perform measures on at least 50% of the nerve bundle in order to keep an acceptable representation of the fiber populations. This non-random distribution may also be increased by pathological conditions where the fiber loss is either focal, as it has been described for diphtheritic polyneuritis, amyloidosis, leprosy and primary nerve tumors (**Fisher** [51], **Rukaniva** [136], **Simpson** [148] and **Rudge** [135]) or multi-focal as in diabetic neuropathy (**Thomas** [154]).

The computational cost of the method is a critical parameter due to the considerable amount of data to be processed to study a complete cross-section of a nerve. Most of the processing is performed on the zonal graph, not on the image itself. Because this graph is orders of magnitude smaller than the image, the cost of the connected operators filtering is negligible. The main computational costs lie in the thresholding step, in the generation of the zonal graph and in the evaluation of the myelin sheath's thickness. Globally, the method requires less than

one minute on a Pentium II, 233MHz computer to process a 1850×1234 pixels image. This means between one and two hours to process a complete cross-section.

In conclusion, the above procedure is a fast and accurate method for the morphometry of nerve cross-section.

Chapter 5

Signed Euclidean DT with error detection and correction.

In this chapter, we consider signed distance transformations. We show that the approximate signed distance maps contains sufficient information to allow the detection and correction of errors, notwithstanding the method used to produce them.

First, we consider a few properties of the Voronoi diagram of a discrete set of points, both on a continuous plane or on a discrete grid. Errors in signed EDT always occur near the corners of the Voronoi polygons, which can easily be detected and corrected. The algorithm's computational complexity is evaluated.

This time, the error detection paradigm can be extended to 3 dimensions. We evaluate the difficulty of a 3D exact EDT by error correction and compare a few alternatives.

5.1 Signed EDT and Voronoi diagrams

As defined in chapter 2, the signed distance from an object and the Voronoi diagram of the object pixels are equivalent concepts. In particular, we have

$$\forall p \in I,\ p \in VP(p - SD(p)) \tag{5.1}$$

In order to understand the behaviour of Signed EDT algorithms, it is interesting to consider a few properties of the Voronoi diagram, both on the continuous plane and on a discrete grid. Let us start with the continuous case.

A first property of the continuous Voronoi diagram is that its tiles are connected sets. An even stricter property says that any tile VP(q) in the Voronoi diagram is *star-shaped*, i.e.

$$p \in VP(q) \Rightarrow \forall\ 0 \leq \alpha \leq 1\ ,\ p' = \alpha.p + (1-\alpha).q \in VP(q) \qquad (5.2)$$

This is proved ab absurdo. Consider that there is a $p' \notin VP(q)$. Then, there exists $q' \in O$ such that

$$dist_e(p', q') < dist_e(p', q) \qquad (5.3)$$

Let us then consider the distance between our point p and this object pixel q'. By the triangular inequality, we have

$$dist_e(p, q') \leq dist_e(p, p') + dist_e(p', q) \qquad (5.4)$$

by combining (5.3) and (5.4), we have

$$dist_e(p, q') < dist_e(p, p') + dist_e(p', q) = dist_e(p, q) \qquad (5.5)$$

with the equality a consequence of the alignment of p, p' and q. Obviously, with $dist_e(p, q') < dist_e(p, q)$, p cannot be part of $VP(q)$, which contradicts our assumptions. **QED**.

While Voronoi diagrams based on any metric that satisfies the triangular inequality are star-shaped, continuous Voronoi diagrams based on the Euclidean metric have an even stronger property: they are convex, i.e.

$$p_1, p_2 \in VP(q) \Rightarrow \forall\, 0 \leq \alpha \leq 1,\ p' = \alpha.p_1 + (1-\alpha).p_2\ \in VP(q) \quad (5.6)$$

Once again this can be proved ab absurdo. Let us consider that there is a $p' \notin VP(q)$, i.e. that $p' \in VP(q')$ with $q \neq q'$. The plane is divided in two by the mid-perpendicular of qq', and each half-plane corresponds to the zones of influence of q and q'. $p' \notin VP(q)$ implies that this mid-perpendicular intersects $p'q$ between p' and q. On the other hand,

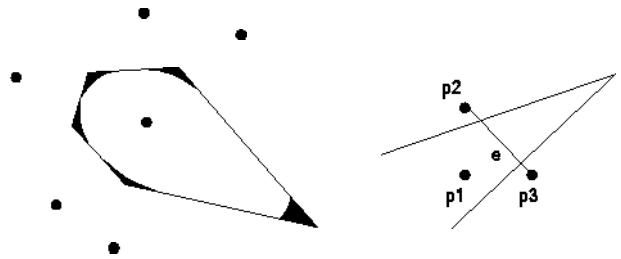

Figure 5.1: Corners of a Voronoi Polygon.

$p_1 \in VP(q)$ and $p_2 \in VP(q)$ imply that it intersects $p'q$ beyond p_1p_2. Both propositions contradict each other. **QED.**

A last property says that the tiles of the Voronoi diagram of a discrete set of point are polygons, because it is constructed from the lines that divide the plane between the pixels two by two. This is why we usually refer to the tiles of the Voronoi diagram as Voronoi Polygons (VP).

Let us now consider what happens for a digital image, i.e. for the Voronoi diagrams defined on discrete grids. In the exact digital Voronoi diagram, the value of each pixel is strictly equal to its value in the underlying continuous plane. Unfortunately, the tiles in such a diagram are not always connected anymore, as illustrated in chapter 2, section 2.2.2. This leads to errors in the computation of the signed DT with limited size neighborhoods.

Interestingly, errors only occur in the corners of the Voronoi polygons, where the corners of a polygon are defined as the locus of the segments smaller than $\sqrt{2}$ that join two edges of the polygon. As illustrated at figure 5.1, errors in the signed DT only occur when the VP is disconnected on the digital grid, i.e. when the VP can fit between points of the grid. And the maximum length between two neighboring points of the grid is $\sqrt{2}$.

To every corner of the continuous Voronoi diagram corresponds a corner of its digital approximation generated by an approximate signed EDT algorithm. Those corner pixels share the following property. Corner pixels and their two direct neighbors in the propagation direction have 3 different nearest object pixels. This property can easily be checked

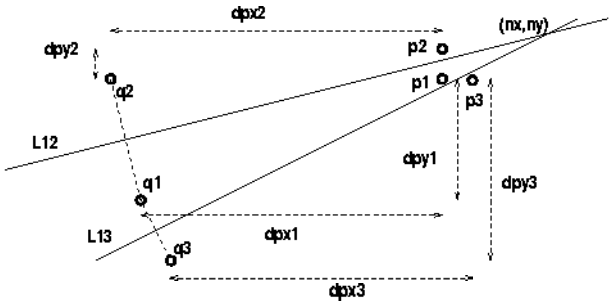

Figure 5.2: Neighbors to consider during error correction.

and provides a fast method to detect corners.

5.2 Error correction

Once we know where the corners of the digital Voronoi diagram are, we can correct all errors done by the approximate signed Euclidean DT algorithm by checking if the real corner of the VP contains any pixel.

In figure 5.2, p_1 is a corner pixel, neighbored by $p_2 = p_1 + (0,1)$ and $p_3 = p_1 + (1,0)$. The nearest object pixels of p_1, p_2 and p_3 are q_1, q_2 and q_3, respectively. The signed distances are $SD(p_1) = (dp_{x1}, dp_{y1})$, $SD(p_2) = (dp_{x2}, dp_{y2})$ and $SD(p_3) = (dp_{x3}, dp_{y3})$. With that knowledge, we can actually compute the limits of the corner of the continuous Voronoi diagram analytically, since lines L_{12} and L_{13} are the mid-perpendicular of q_1q_2 and q_1q_3 respectively.

Using pixel p_1 as the center of our coordinate system, we have

$$L_{12} : n_y \leq \alpha_{12}.n_x + \beta_{12} \tag{5.7}$$
$$L_{13} : n_y \geq \alpha_{13}.n_x + \beta_{13} \tag{5.8}$$

L_{12} is the mid-perpendicular of q_1q_2, so that

$$\alpha_{12} = \frac{dp_{x2} - dp_{x1}}{dp_{y1} - dp_{y2} + 1} \tag{5.9}$$
$$\beta_{12} = \frac{1}{2}.(\alpha_{12}.(dp_{x1} + dp_{x2}) - dp_{y1} - dp_{y2} + 1) \tag{5.10}$$

and L_{13} is the mid-perpendicular of q_1q_3, so that

$$\alpha_{13} = \frac{dp_{x3} - dp_{x1} - 1}{dp_{y1} - dp_{y3}} \tag{5.11}$$

$$\beta_{13} = \frac{1}{2}.(\alpha_{13}.(dp_{x1} + dp_{x3-1}) - dp_{y1} - dp_{y3}) \tag{5.12}$$

Finally, we are interested in the location of the true corner of the Voronoi diagram, i.e. the intersection of L_{12} and L_{13}. The true corner, (c_x, c_y), gives us the maximum values of (n_x, n_y) to consider. In particular, for n_x, there is no need to go further than

$$n_{x\max} = c_x = \frac{\beta_{12} - \beta_{13}}{\alpha_{13} - \alpha_{12}} \tag{5.13}$$

Equations (5.9), (5.11) and (5.13) require some more attention. In (5.9), a singularity could occur if $dp_{y1} = dp_{y2} - 1$. Fortunately this never happens with $q_1 \neq q_2$. In (5.11), there is a singularity if $dp_{y1} = dp_{y3}$. When this happens, it means that L_{13} is a vertical line, for which $\alpha_{13} = \infty$ is an appropriate slope. Finally, in (5.13), a singularity occurs when $\alpha_{13} = \alpha_{12}$. This means that L_{12} and L_{13} are parallels, which only occurs for $\alpha_{13} = \alpha_{12} = \pm 1$. In that case, the pixels in the diagonal direction should be tested until the first one that does not change value.

5.3 CSED Algorithm

Using the above considerations, we can design the following algorithm. First, we apply any approximate signed Euclidean DT algorithm. For instance, let us consider a signed version of Danielsson's raster scanning algorithm [37].

Algorithm 6 *4-neighbors Sequential Signed Euclidean Distance transformation (4SSED).*

Input: an $M \times N$ image I containing an object O.
Output: the signed distance transformation $SD(p) = p - q$ where $q \in O$ and $dist_e(p, q) \leq dist_e(p, q') \; \forall \; q' \in O$.

```
for all p ∈ I do {initialization}
  if p ∈ O then
    SD(p) ← (0,0)
```

```
    else
      SD(p) ← (∞, ∞)
    end if
  end for

  for p_y = 0 → N − 1 do {forward scan}
    for p_x = 0 → M − 1 do
      test(p, (−1, 0))
      test(p, (0, −1))
    end for
    for p_x = M − 1 → 0 do
      test(p, (1, 0))
    end for
  end for
  for p_y = N − 1 → 0 do {backward scan}
    for p_x = M − 1 → 0 do
      test(p, (1, 0))
      test(p, (0, 1))
    end for
    for p_x = 0 → M − 1 do
      test(p, (−1, 0))
    end for
  end for

  procedure test(p, n)
    if p + n ∈ I then
      if dist_E(SD(p + n) − n) < dist_E(SD(p)) then
        SD(p) → SD(p + n) − n
      end if
    end if
  end procedure
```

As before, a special attention should be taken to the efficient computation of $dist_E(dp)$, either using Leymarie's formulae (2.18) or lookup tables to compute the squares of integers.

In the second step, we apply the corner detection and error correction using the principles of the previous sections, with a slight modification. Indeed, for corner pixels with $dist_E(SD(p)) < 116$, we know from chapter 3 that a 3×3 neighborhood is always sufficient to ensure a correct

distance transformation. It is then more computationally efficient to only test the diagonal pixels rather than to make the complex floating point operations of equations (5.9) to (5.13). The algorithm goes as follows.

Algorithm 7 *Corner detection and error correction in a signed Euclidean DT.*

Input: An approximate signed Euclidean distance map SD.
Output: An exact signed Euclidean distance map SD.

```
for all p ∈ I do {corner detection}
  n1 ← (sgn(SD(p)_x), 0)
  if SD(p + n1) − SD(p) ≠ n1 then
    n2 ← (0, sgn(SD(p)_y))
    if SD(p + n2) − SD(p) ≠ n2 then
      if SD(p + n2) − SD(p + n1) ≠ n2 − n1 then
        correct(p, n1, n2)
      end if
    end if
  end if
end for

procedure correct(p, n1, n2) {error correction}
  if dist_e(p) < 116 then
    test2diagonal(p, n1, n2)
  else
    compute α12, α13, β12, β13, n_x max
    if α12 = α13 then
      testdiagonal(p, n1, n2)
    else
      for n_x = 0 → n_x max do
        for n_y = α13.n_x + β13 → α12.n_x + β12 do
          testwrite(p, n)
        end for
      end for
    end if
  end if
end procedure
```

```
procedure testdiagonal(p, n1, n2)
  i ← 0
  repeat
    testwrite(p, i.(n1 + n2))
    i ← i + 1
  until testwrite(p, i.(n1 + n2)) did not modify SD
end procedure

procedure testwrite(p, n)
  if p + n ∈ I then
    if dist_E(SD(p) + n) < dist_E(SD(p + n)) then
      SD(p + n) → SD(p) − n
    end if
  end if
end procedure
```

One can notice that, in contrast with the 4SSED algorithm, the correction step uses a "write" formalism in the *test*() procedure. With the "read" formalism, $test(p, n)$ can modify the value of $SD(p)$. With the "write" formalism, it can modify the value of $SD(p + n)$.

5.4 Computational Complexity

In order to evaluate the computational cost and complexity of this new algorithm, we apply the same tests as in chapter 3, i.e. the test images of figure 3.7. The result of these experiments is found in figures 5.3 to 5.5.

In this section, we compare five different algorithms. Three of those are PSN, PMN and PMON, the propagation algorithms of chapter 3. The other two are the approximate and exact algorithms of this chapter. The approximate one is called 4-neighbors Sequential Signed Euclidean Distance transformation, or 4SSED. The exact one is a correction on 4SSED, and we call it 4SSED+.

In all three tests, 4SSED performs marginally better than the other approximate algorithm, PSN. There is no difference between them for test 1, the empty disk image. For tests 2 and 3, 4SSED does not suffer from the slight orientation dependence that affects PSN.

The exact 4SSED+ algorithm is faster than the exact PMN and PMON

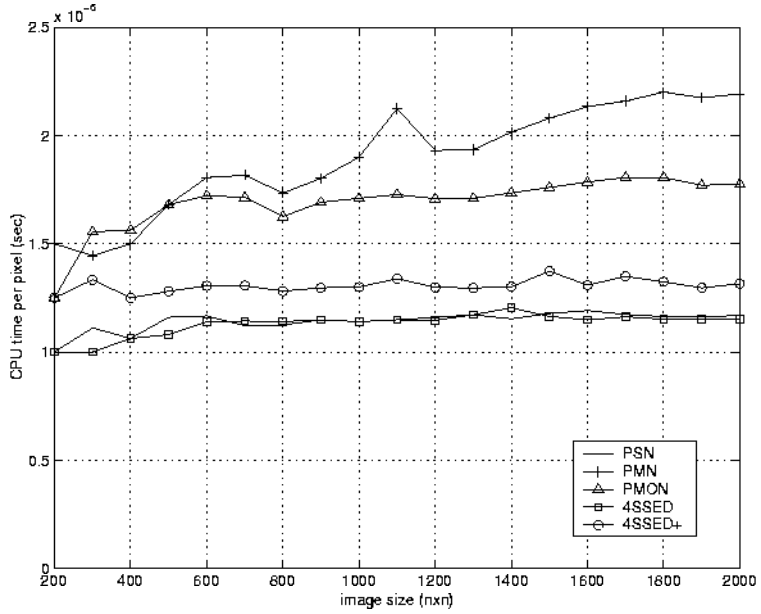

Figure 5.3: Test1: Empty circle image.

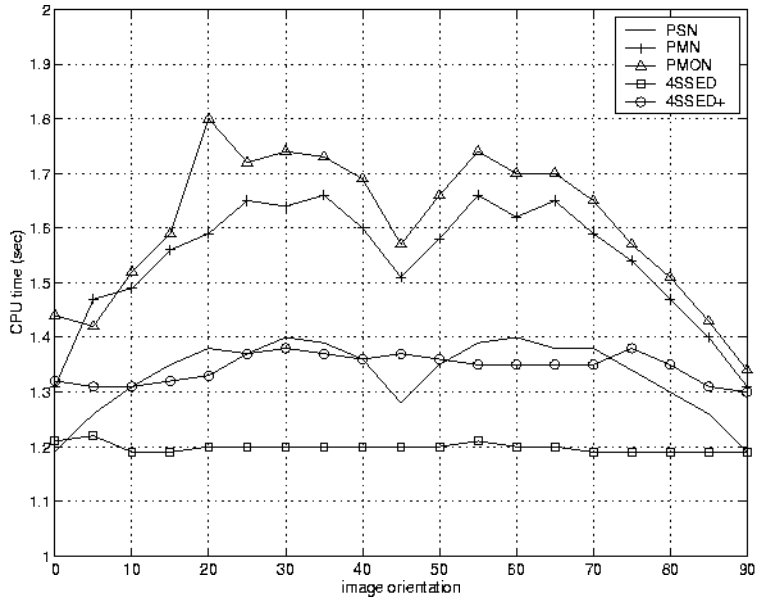

Figure 5.4: Test2: Random squares .

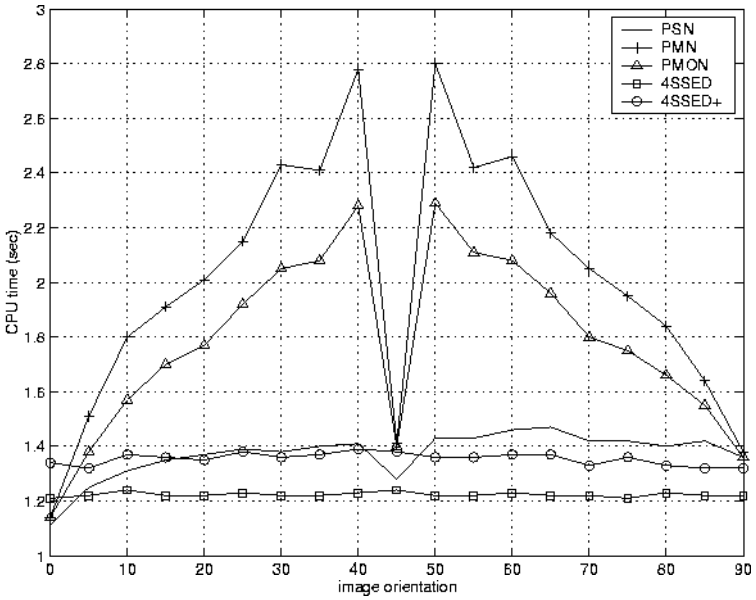

Figure 5.5: Test3: Straight line.

algorithms in all cases. Actually, for tests 2 and 3, it is even as fast as the approximate PSN. Also, 4SSED+ is nearly unaffected by the image orientation. Indeed, during the "corner detection and error correction" step, most of the processing power is spent on the constant time corner detection. Corners typically represent less than 2% of the image pixels, which keeps the correction cost low.

To conclude, 4SSED+ is both faster and less orientation dependent than its propagation counterparts. In average, it is only 15 to 20% slower than 4SSED. This makes it arguably the optimal exact signed Euclidean DT algorithm.

Chapter 6

Euclidean DT in 3 dimensions

In this chapter, we explore how the algorithms of chapters 3 and 5 can be extended to 3 dimensions. First, we remind the algorithms that can be used to produce approximate Euclidean DT in 3 dimensions. Secondly, we consider the error detection and correction methods of chapter 3 and 5 and discuss the possibility of extending them to 3 dimensions. Thirdly, we evaluate the computational complexity of algorithms that would use this approach, and show that it can not compete with Saito's method. Alternatively, we propose a new hybrid method that combines our optimal 2D distance transformation algorithms with Saito's along the third axis. We show that this is the best available algorithm for large data sets, especially when one considers anisotropic volumes.

6.1 Extending the approximate EDT to 3D

In order to produce approximate 3D Euclidean DT, Danielsson [37] and Leymarie [95] propose a complex 6 scans algorithm. It includes a forward and a backward scan over the whole image, plus opposite scans at the plane and line level. In total, it requires 12 comparisons per pixel with direct neighbors and 36 for the $3 \times 3 \times 3$ neighborhood.

In [128], Ragnelmam proposes two algorithms with independent raster scans. The corner EDT can be extended to any dimension. It uses 2^D scans involving D direct neighbors in D dimensions. In 3D, it requires to perform 24 comparisons per pixel. Alternatively, he proposes a 4 scan

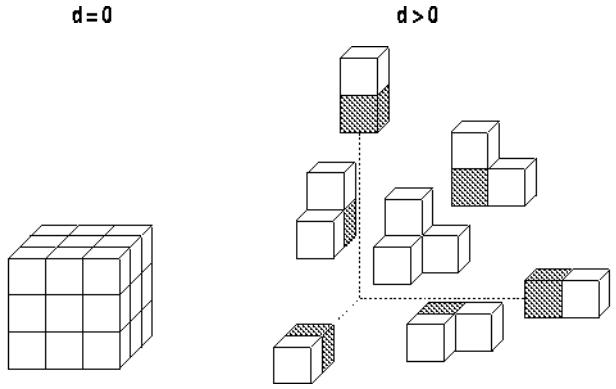

Figure 6.1: Neighborhood used for the 3D ordered propagation algorithm. *Left:* $D(p) = 0$ *Right:* $D(p) \geq 1$ for one eighth of the directions space.

algorithm with the neighborhoods of figure 2.8. It requires 48 comparisons per pixel.

Finally, it is possible to extend the PSN algorithm of chapter 3 to 3 dimensions. The only modification required is that 3D neighborhoods should be used. Instead of those of figure 3.1, we use those of figure 6.1. For object pixels, the complete 3×3 neighborhood is checked [1]. For other pixels, only direct neighbors n such that in the same direction as dp are considered. This algorithm requires approximately 3 comparisons per pixel.

6.2 Possible error detection and correction methods in 3D

The error detection and correction methods of chapters 3 and 5 are summarized as in table 6.1. Actually, other algorithms combining the methods of both chapters could also be designed. One could detect corners of the Voronoi polygons in a signed EDT, and then use neighborhoods from table 3.1 to propagate them further. Alternatively, one could detect non propagating pixels in a signed version of PSN, and then

[1] Actually, to avoid unnecessary computations, it is only used for pixel p in the border of the object O, i.e. such that $p \in O$ and $\exists q = p + n \in O'$ with a direct neighbor n

	Chapter 3	Chapter 5
Detection	non-propagating pixels in the PSN algorithm	Corners of the digital Voronoi Polygons.
Correction	use of neighborhoods of increasing sizes	explicit computation of the true VP corner

Table 6.1: The error detection and correction phases of the algorithms of chapter 3 and 5.

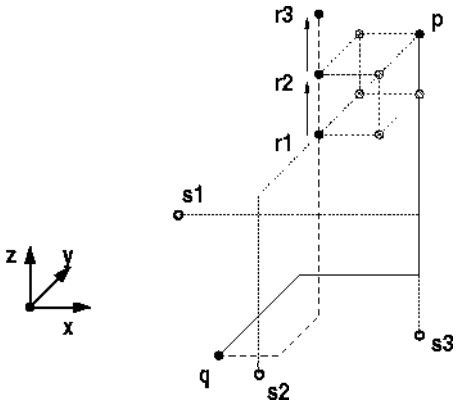

Figure 6.2: Why PMN's detection criterion does not work in 3D: $p(0,0,0)$ is closer to $q(4,4,8)$ than to $s_1(8,0,6)$, $s_2(0,8,6)$ and $s_3(0,0,10)$. Still, pixels $r_1(3,3,7)$ and $r_2(3,3,8) \in VP(q) \bigcap N(p)$ propagate.

compute the corner of the continuous VP.

Unfortunately, not all of these methods can be extended to 3 dimensions. For instance, there can be errors in the 3D EDT while none of the neighboring voxels fails to propagate, as illustrated at figure 6.2 where there is an error in p while both r_1 and r_2 propagate. Hence, the detection method of chapter 3 cannot be used.

Similarly, computing the exact corner of the continuous VP from the 3 direct neighbors in the propagation direction does not necessary provide a finite volume. It makes the error correction method of chapter 5 impractical.

Neighborhood	Smallest error		Non-propagating pixel	
	(dp_x, dp_y, dp_z)	$dist_E$	(dp_x, dp_y, dp_z)	$dist_E$
$3 \times 3 \times 3$	(6,3,2)	49	(4,2,2)	24
$5 \times 5 \times 5$	(12,4,3)	169	(9,2,3)	94
$7 \times 7 \times 7$	(24,5,5)	626	(20,4,4)	432
$9 \times 9 \times 9$	(24,9,4)	673	(19,7,3)	419
$11 \times 11 \times 11$	(37,11,5)	1515	(31,9,4)	1058
$13 \times 13 \times 13$	(45,12,5)	2194	(38,10,4)	1560
$15 \times 15 \times 15$	(52,12,5)	2873	(44,10,4)	2052
$17 \times 17 \times 17$	(60,12,5)	3769	(51,10,4)	2717

Table 6.2: Errors closest to (0,0,0) for a number of $N \times N \times N$ neighborhoods.

On the other hand, the corner detection of chapter 5 and the multiple neighborhoods correction of chapter 3 can be extended to 3D. For corner detection, one should be aware that there are two types of corners. Either the three neighbors in the propagation direction belong to different tiles of the Voronoi diagram than the current voxel. We call this a 3-corner. Either only two neighbors out of three belong to different tiles. We call this a 2-corner. As illustrated at figure 6.2, considering 2-corner such as r_2 is a needed to detect errors such as that in p.

Similarly to chapter 3, one can compute the limits for which a neighborhood of a given size guarantees that no error occurs in the Euclidean DT. These limits are found in table 6.2. The middle column gives the error for which the distance is the smallest. The right column contains the voxel closest to the error and that belongs to the same tile of the Voronoi diagram.

6.3 Limitations to the 3D error detection and correction methods

Unfortunately, although it is theoretically possible, the generation of an exact Euclidean DT in 3D using VP corner detection and multiple neighborhoods correction suffers from two major drawbacks. On one hand, corners are not as uncommon in 3D as in 2D. For instance, for the sphere

test images used in the next section, there are typically only 1 or 2% of 3-corners, but sometimes as many as 20% of 2-corners. This makes the correction phase a non-negligible part of the computational cost. On the other hand, the use of large 3D neighborhoods guarantees exact EDT up to much smaller values than in 2D. For instance, the $17 \times 17 \times 17$ neighborhood only guarantees an exact EDT up to $dist_E = 3769$, while the 2D 17×17 was sufficient up to $dist_E = 57128$.

Because of these drawbacks, we first assess the feasibility of the 3D error detection and correction approach by studying the computational complexity of a prototype algorithm, that computes an approximate EDT using the full error detection, and an error detection limited to $3 \times 3 \times 3$ neighborhoods.

We compare the computational costs and complexities of 5 algorithms. These algorithms are

- Ragnelmam's corner EDT, that uses 8 scans with 3 direct neighbors each.
- The 3D version of the PSN algorithm, with the neighborhoods of figure 6.1.
- The same algorithm used to produce a signed DT, as required by the error detection method. We call it signed-PSN
- Our prototype algorithm, where 3-corners are further checked with the $3 \times 3 \times 3$ neighborhood and 2-corners are further checked with a 3×3 neighborhood in the corner plane. We call this $3 \times 3 \times 3$ PMN.
- Saito's exact Euclidean DT algorithm.

In order to compare these algorithms, we use two classes of test images. In test 1, non-object pixels are included in one eighth of a sphere centered in $(0, 0, 0)$, with a radius equal to the image size. This image size varies from $32 \times 32 \times 32$ to $256 \times 256 \times 256$. In test 2, the object consists of a plane whose orientation varies from 0^o to 90^o. The image size is $200 \times 200 \times 200$. The algorithms were implemented in C and executed on a SUN Sparc Ultra 1 workstation.

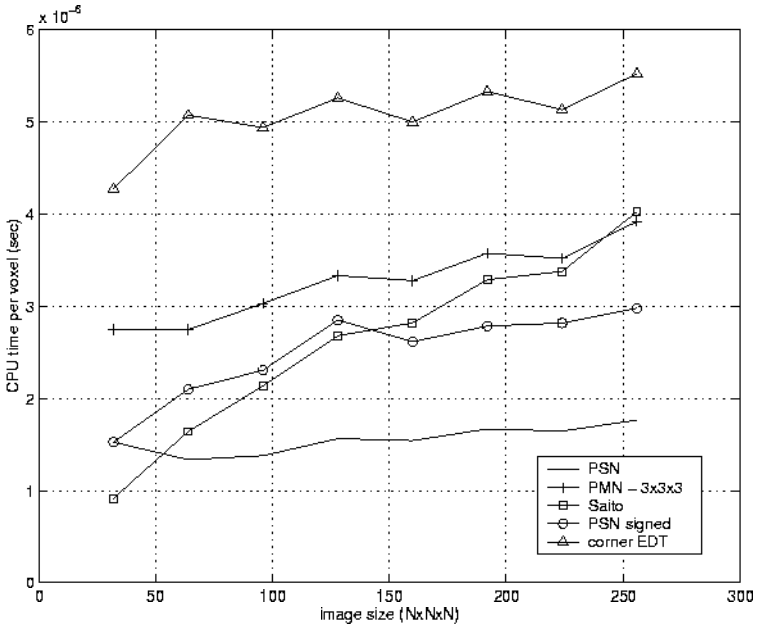

Figure 6.3: Test1: Empty sphere image.

Figure 6.3 shows the computational cost per pixel for test 1. PSN is the fastest in all cases but for very small images. Its cost is only marginally affected by the image size. The corner EDT, which gives a similar approximation of the EDT, is the slowest in all cases. This illustrates the weakness of raster scanning methods in higher dimensions.

The signed PSN has a computational cost significantly higher than the unsigned one. It also has much larger memory requirements. The hybrid $3 \times 3 \times 3$ PMN requires an approximately constant additional cost compared to the signed PSN. It corresponds mostly to the cost of the detection phase since corrections are strictly restricted within the $3 \times 3 \times 3$ neighborhood.

Finally, Saito's algorithm appears to be quite efficient in 3D. For images smaller than $50 \times 50 \times 50$, it is the fastest algorithm of all. For images smaller than $150 \times 150 \times 150$, only the unsigned PSN is faster. Even for images as large as $256 \times 256 \times 256$, its computational cost is similar to the $3 \times 3 \times 3$ PMN, only twice slower than PSN.

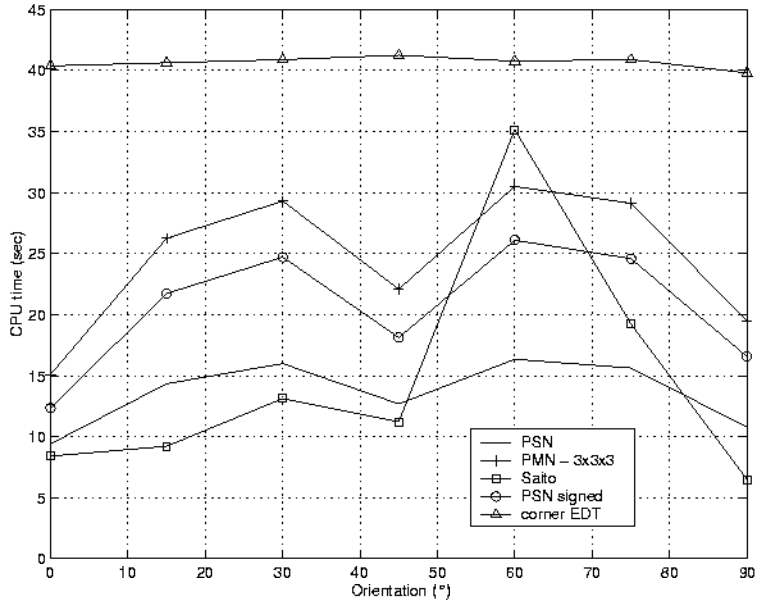

Figure 6.4: Test2: Oriented plane.

On the other hand, its complexity appears to be higher than that of the approximate algorithms. Actually, one can show that Saito's algorithm has a $O(n^4)$ complexity for $n \times n \times n$ images. More generally, it has a $O(D.n^{D+1})$ complexity in D dimensions, for an $n \times ... \times n$ image. Indeed, in Saito's algorithm, the scans in each direction are independent. In one direction, scanning one line of n voxels has a $O(n^2)$ complexity, and there are n^{D-1} such lines, so that the computational cost of a full scan in one direction is $O(n^{D+1})$.

The results of test 2 are found in figure 6.4. The approximate algorithms are ordered in the same way as for test 1. The PSN-based algorithms are somewhat orientation dependent, in contrast with the slower corner EDT. Saito's algorithm performs better than all others in most orientations, but worse for orientations close to 60^o.

6.4 Hybrid algorithm, combining 4SSED+ and Saito's methods

For small images, Saito's algorithm is extremely efficient, even outperforming approximate EDT algorithms in many cases. For numerous applications, 3D images indeed have small dimensions, because large dimensions would lead to enormous datasets, that would be impractical to acquire, store and process. A typical PET scan only has a $128 \times 128 \times S$ resolution. A typical MR image would be $256 \times 256 \times S$, where S is the number of slices (typically 100).

For larger images, Saito's method suffers from a $O(n^4)$ complexity, but it appears that the error detection and correction methods can not provide better results. This makes the Euclidean DT an expensive tool to process CT scans, for instance, where the resolution can easily reach $512 \times 512 \times S$. Also, one may expect that the resolution of other modalities, such as MRI, will increase in the future. Already today, non medical applications such as remote sensing or 3D microscopy provide datasets where the size of each slice is much larger, even though the number of slices may be smaller.

Fortunately, it is possible to improve Saito's 3D EDT by noticing that it works axis by axis. In particular, it means that a 2D EDT is computed in each slice before the algorithm considers the 3rd (inter-slice) axis. Therefore, it is possible to replace the computations along the two first axis by our optimal 2D EDT of chapter 5 applied on all slices. We call this the hybrid algorithm.

The gains in computational cost of this change are illustrated at figure 6.5. Saito's and the hybrid algorithm are compared using the test 1 image, i.e. a 3D image containing the 8^{th} of a sphere, with a size $N \times N \times N$, with N varying from 100 to 600. PSN is also shown as a reference.

Both for Saito's and the hybrid algorithm, the cost of the slice by slice 2D EDT stage is shown as a dashed line. The additional cost for the computations along the 3^{rd} (inter-slice) axis - the distance between the dashed and plain lines - is of course identical in both cases. Surprisingly, this additional cost is only very lightly dependent on the image

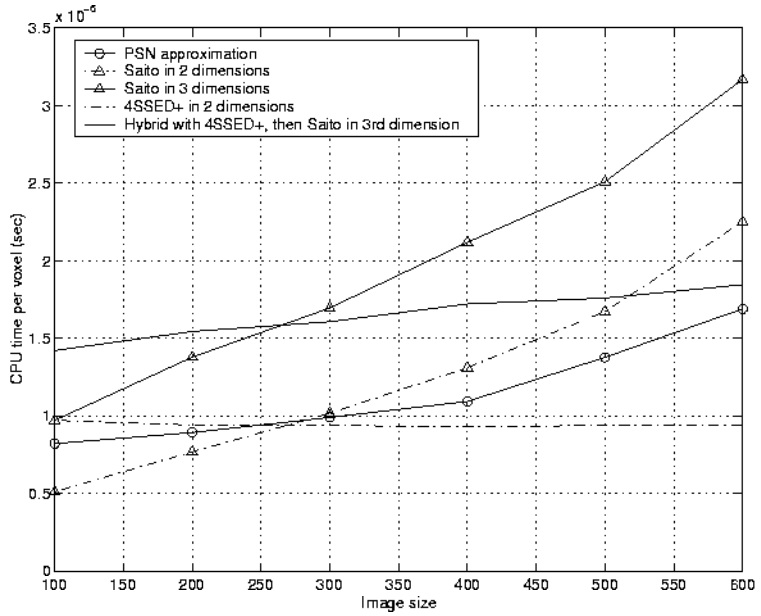

Figure 6.5: Computational complexity of Saito's and the hybrid algorithm on 3D isotropic data.

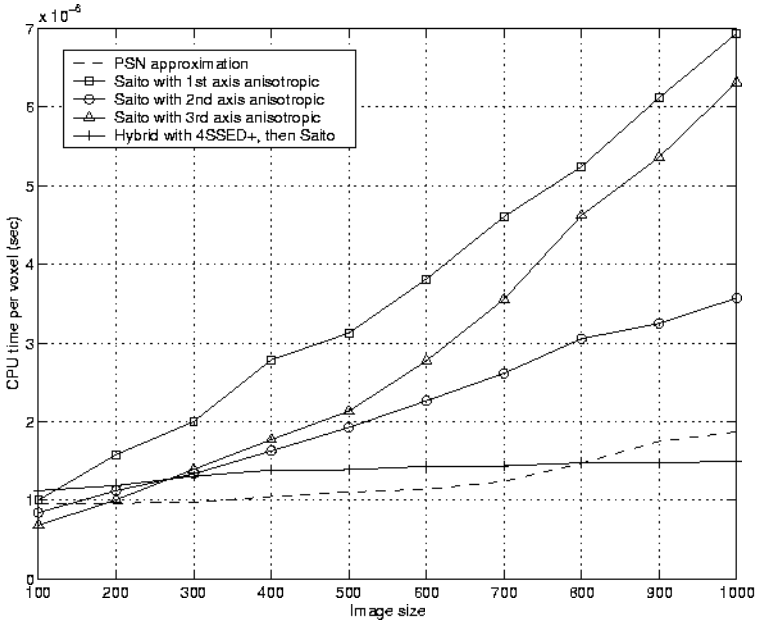

Figure 6.6: Computational complexity of Saito's and the hybrid algorithm on 3D anisotropic data (voxel size is $1 \times 1 \times 4$

size, so that the hybrid method nearly has a $O(n^3)$ complexity. Practically[2], the hybrid algorithm is faster than Saito's for images larger than $260 \times 260 \times 260$.

When on considers practical applications, it appears that 3D images are very seldom cubic as considered until now. Instead, they are usually made of a relatively small number of high-resolution slices with a lower inter-slice resolution. We are therefore interested in how the above results adapt to anisotropic data.

Figure 6.6 considers the same cubic volume as before (the 8^{th} of a sphere) sampled with $1 \times 1 \times 4$ voxels. Therefore, there are 4 times less slices than pixels along a column or a row of a slice. For instance, an image

[2]This limit is of course dependent on the content of the image. In particular, it would be possible to find pathological images - such as lines oriented at 60^O in all slices - where the limit is much lower. For other images - such as a single object pixel in the middle - Saito's algorithm is always the fastest. Nevertheless, the "sphere" image gives us a good indication of where the limit will be in most practical cases.

size of 600 means a $600 \times 600 \times 150$ image. In such a case, the complexity of Saito's algorithm depends on the order in which the axis are considered, i.e. whether the smaller anisotropic axis is considered first, second or third. Obviously, considering it in second place is the best choice. On the other hand, the hybrid method appears to be nearly insensitive to the image size, which makes it asymptotically optimal. Practically, the hybrid method is once again the fastest as soon as the image size exceeds 260. Let us remind that this corresponds to a volume of data that is 4 times smaller than at figure 6.5.

Finally, let us notice that the hybrid algorithm is not significantly slower than PSN[3], the fastest and coarsest approximate EDT. This is especially true for anisotropic data.

[3] At figure 6.6, PSN's computational cost increases for images sizes over 700. This contradicts the theory that says that PSN has a fixed cost per pixel. We postulate this experimental increase is linked to the size of the dynamic data structure that holds the pixels in the propagation front. For small images, the buckets' structure is small enough to hold entirely within the cache memory. For large images, it has to be stored in the main RAM, which slows down the whole process.

Chapter 7

Application: registration of MR images

This chapter illustrates the use of the 3D Euclidean distance transformation in two registration applications. First, we introduce the need for data fusion and registration in medical imaging, and briefly review the main approaches to the problem. Then, in the first application, we show how to find the best rigid-transformation from an MR image to the physical space. In the second application, a computerized brain atlas is warped into a patient's MRI using a parameterized second degree transformation.

7.1 Introduction

In medical imaging, it is often needed to use several complementary sources of information, such as scans taken at different times, scans from different modalities, template anatomies, ... Registration is the process of finding the proper spatial relation of a data-set in reference to another. It makes it possible to super-impose the information from the different sources and enables their combined interpretation.

7.1.1 Applications

Data fusion and registration are such important and widely used tools in medical imaging that it is not possible here to make an exhaustive review of their applications. Thus, we only present a few illustrative ones.

Registering time-series data from the same patient, in one or more modalities, allows to evaluate the progress of a disease. For instance, **Ettinger** [48] applies it to the follow-up of multiple-sclerosis patients. **Wong** [183] uses it in epilepsy diagnosis.

Registering images from the same patient in different modalities allow their simultaneous interpretation. For instance, **Kapouleas** [88] or **Mangin** [101] register functional Positron Emission Tomography (PET) with Magnetic Resonance (MR) images where the soft tissues' anatomy can be seen.

Registering medical images to the physical world permits their use in image-guided surgery. For instance, **Davey** [38] uses it for neuro-surgery planning; **Herring** [75] for image-guided surgery of the spine; **Wasserman** [178] for radio-therapy treatment planning.

Finally, registering images to a template anatomy can be useful for comparison purposes, to provide anatomical a priori to automatic segmentation procedures, ... For instance, **Talairach** [151] proposed a standard coordinate system for the brain. **Kikinis** [46] developed a digital brain atlas used - among others - by **Warfield** [176] in model-driven segmentation.

7.1.2 State of the Art

The variety of published registration methods can be presented in many different ways. Possible classification criteria include the type of transformation (rigid, affine, polynomial, fluid, ...) or the type of matching criterion (visual, point-based, surface-based, volume-based, ...). Here, we present them according to the type of interaction they require from the user.

7.1.2.1 Methods using fiducial markers

When extreme precision is required, such as in the case of image-based surgery, the registration can be based on external fiducial markers, visible in both imaging modalities, implanted on the patient. **Mandava** [100] and **Maurer** [102, 103] present such methods where the exact location of the markers is precisely extracted from both scans, and the best

transformation is the one that minimizes the misfit between the pairs of corresponding markers. The precision achievable by this approach is evaluated by **Fitzpatrick** [52].

Unfortunately, the use of markers is both time-consuming in the image acquisition phase and sometimes painful for the patient, which restricts its use to some specific applications, such as stereotactic surgery.

7.1.2.2 Manual retrospective methods

Retrospective methods do not require external markers, but use the anatomy itself to search for the best transformation.

The simplest methods rely on the user to manually define the parameters of the registration transformation. For instance, **Kapouleas** [88] registers MRI and PET scans of the brain by first identifying the inter-hemispheric plane, then interactively adjusting the remaining two translational and one rotational parameters of the rigid transform, while the operator visualizes the edges of the MRI overlaid on the PET data. A similar approach is used by **Bohm** [8] to iteratively select the parameters of a second degree polynomial transform that registers a computerized brain atlas to PET or MRI.

Alternatively, point-based methods require the user to select corresponding points in both images, and automatically compute the best transformation based on those pairs of corresponding points. **Pietrzyk** [121] proposes an iterative method based on the repeated selection of one or two such pairs. **Hill** [76] and **Henri** [74] select a set of 4 to 26 corresponding points, and compute the transformation that minimizes the global misfit on these.

Manual methods are of course time-consuming and their accuracy is limited to the precision with which the operator can designate matching features.

7.1.2.3 Automatic retrospective methods

Manual (human-based) methods rely on a small set of highly semantic information such as the exact location of specific anatomical features in both images. Automatic (computer-based) methods compensate the

lesser semantic ability of computers by increasing the amount of data used in the matching criterion.

Surface-based method, such as those proposed by **Pellizari** [119], **Jiang** [80, 81], **Mangin** [101], **Lemoine** [92] or **Hemler** [73], are divided in two steps. First, similar features are segmented from both images. Then, the best transformation is defined as minimizing the distance between those features.

Volume-based methods use the complete dataset into account in the matching criterion. Such methods were proposed by **Woods** [184], **van den Elsen** [162, 161], **Maintz** [99], **Studholme** [150], **Collignon**, **Maes** [21, 98] and **Wells** [47]. The central point of these methods is how they define the similarity between pixels from modalities where similar gray-levels do not necessary correspond to similar tissue types. For instance, Van den Elsen [161] correlates "ridgeness" images to register MRI and CT scans. Wells [47] and Maes [98] maximize the mutual information between images from any two modalities.

7.1.3 Discussion

Registration has been the subject of several important reviews and comparative studies, the first of which was made by **Van den Elsen** [163] in 1993.

Zuk [189] compared manual and automatic retrospective methods. He found the automatic surface-based methods to be the fastest and automatic volume-based methods the most accurate, provided the initial position of the scans to register is close enough to converge to the correct minimum.

West and most other researchers in the field performed a major comparative study [181, 182], using fiducial markers to define the gold standard transformation. These results were also used in a later study comparing surface-based and volume based methods [179, 180]. It concluded that both approaches can give satisfactory results, but that volume-based techniques tend to be more accurate, especially in the rotational component of the rigid transformation.

On the other hand, there are cases where only surface-based registration

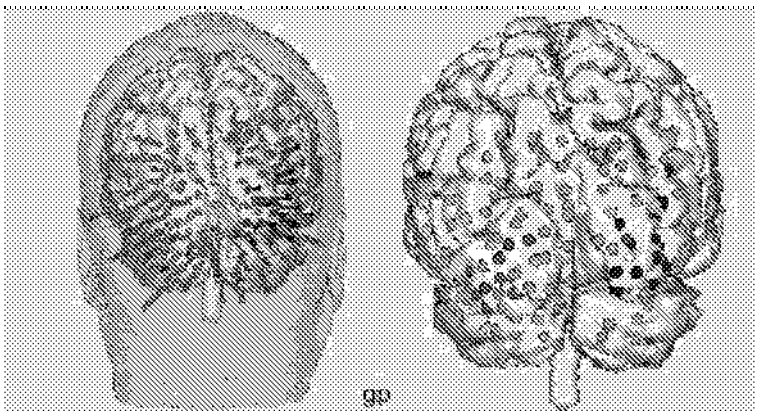

Figure 7.1: Transcranial magnetic stimulation of the visual cortex, from Potts [123].

is possible, because volumetric data is not available in one of the information sources to register. For instance, **Herring** [75] registers CT images to the physical space for image-guided spine surgery. The physical surface is acquired using a 3DSL consisting of a probe and an Optotrak system. The image surface is generated by an iso-surfacing algorithm on a tetrahedral decomposition of the volume data. **Grimson** [68] proposes a surface-based registration for frameless stereotaxy, image-guided surgery. **Maurer** [104] uses a combination of a point-based and surface-based methods to register head CT images to the physical space.

In what follows, we present two such applications, where the choice of a surface-based matching criterion is driven by the lack of volumetric data in one of the modalities.

7.2 Localization of transcranial magnetic stimulation

Transcranial magnetic stimulation (TMS) consists of applying a focal magnetic field on specific parts of the brain in order to induce motor or sensitive responses. TMS has been extensively used in research in order to map the brain functions, for instance by **Brasil-Neto** [16] for the motor cortex. It has also been used for therapeutic purposes, such as the treatment of depression by **George** [66].

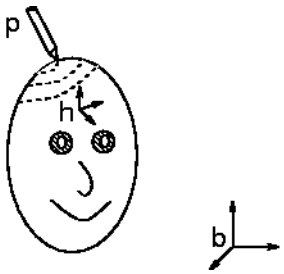

Figure 7.2: Experimental setting.

Initial TMS studies used cranial landmarks to define the locations of the stimulation, which does not take into account the individual variability in cortical morphology. Recently, **Potts** [123] has used a 3D optical tracking system to register TMS with the individual's MRI. This allows to locate the stimulation precisely on both the scalp and the cortical surface, as illustrated at figure 7.1 for a study of the inhibition of the visual stimulus.

7.2.1 Registration method

The experimental setting used in our method is illustrated at figure 7.2. It is based on a Isotrack 3D localization device. It uses two probes, "p" and "h", whose location and orientation relatively to the base station "b" is known at all times. Probe "h" is attached to the forehead of the patient and provides a local coordinate system for the head. Probe "p" is first used to digitize the scalp surface for the registration, then to localize the magnet during the stimulation.

The first step of the registration process is the segmentation of the scalp from the MR image. For this, the image is thresholded and median-filtered. Then, the background of the image is found as the largest connected component below the threshold level. The scalp is the edge of the background.

The surface of the head in the physical space is defined as the set S of points p digitized by the "p" probe and transformed into the "h"

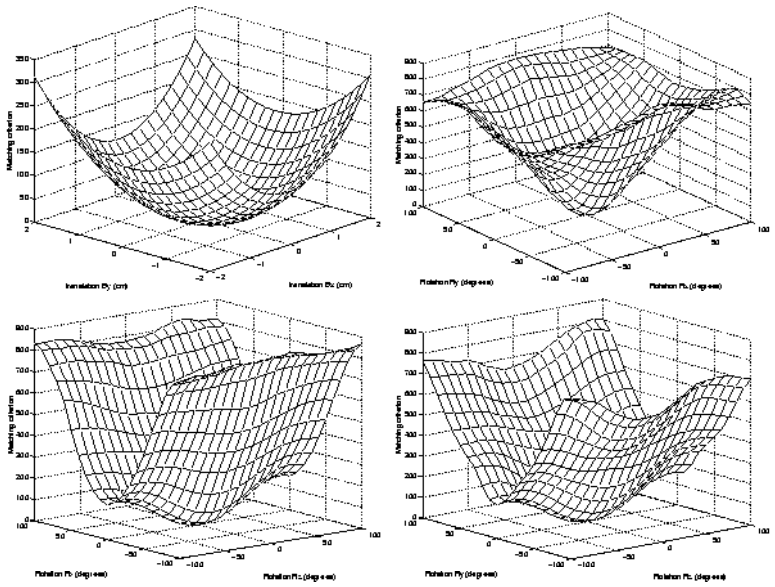

Figure 7.3: Matching criterion as a function of the translation or rotation errors.

coordinate system. The registration matching criterion is defined as the mean square distance between this set of points and the MRI-derived scalp surface O. Hence, the best transformation T from the set $\Re$ of rigid transformations is

$$T = \arg\min_{T \in \Re} \{ \sum_{p \in S} dist_E(T(p), O) \} \tag{7.1}$$

In this equation, $dist_E(q, O)$ is of course pre-computed using a 3D Euclidean distance transformation. Because $T(p)$ is not necessarily located on the integer grid, the value of $dist_E(T(p), O)$ is tri-linearly interpolated from the distance map.

Figure 7.3 studies the evolution of the matching criterion's value as a function of the mis-registration. It is a smooth function of the transformation parameters, so that we use a simple gradient-based minimization algorithm to find the optimal translation and rotation parameters.

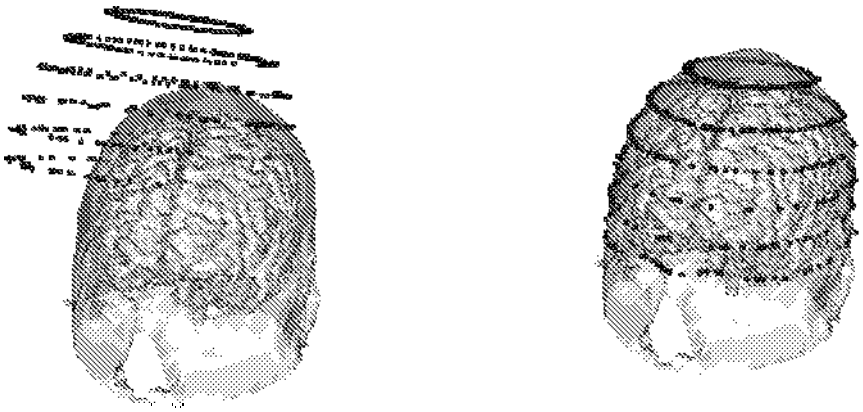

Figure 7.4: MRI to physical space registration. *Left:* unregistered *Right:* registered.

7.2.2 Results

The study of figure 7.3 uses synthetic data. The physical surface S is generated from the MRI scalp, then submitted to a known geometric transformation. Using this method, we know the exact transformation that the algorithm should find, which allows us to evaluate its accuracy.

A typical registration result is illustrated at figure 7.4. We applied a variety of rigid transformations to the synthetic data, including translations up to 5 cm and rotations from -20^o to $+20^o$ around each axis. In all cases, the registration procedure recovers the exact inverse transformation with sub-pixel accuracy.

Practically, for non-synthetic data, the accuracy of the registration is limited by the MR image voxel size and the discrete nature of the scalp surface.

7.3 Registration of MR images with a Computerized Brain Atlas

The Computerized Brain Atlas (CBA) database [8] is based on anatomical information obtained from a cryosectionned brain. It provides a meshed description of the surfaces of most brain structures, including sulci and gyri, Brodmann cytoarchitectonic areas and basal ganglia. This atlas is adjusted to individual brain images which supplies templates for sulci labelling and delineation of the Brodmann areas.

In [155], matching the CBA with an individual MRI data set is done manually by choosing the optimum parameters for a number of elementary transformations that are combined into a general 3D second degree transformation. The optimization criterion is the visual matching of the CBA objects, mainly the cortical surface and the ventricular system, super-imposed on the scan. Although the accuracy of the method has been demonstrated [142], it suffers from being both time-consuming and operator dependent.

7.3.1 Registration method

We propose to adapt the above method so that it becomes fully automated. To achieve this, we define the matching criterion as the distance between the cortical and ventricular system surfaces and the equivalent structures in the CBA database. The CBA surface is used as the reference surface from which the distance transformation is generated. The MRI surface is used as the mobile surface, on which the transformation is applied in order to fit the reference surface.

First, we segment the cortex from the MRI, using a variant of the directional watershed transform described by **Warscotte** [177]. The resulting object is simplified using a mathematical morphology closing that merges the sulci with the cortex.

The set of possible transformations $\overrightarrow{p} = (p_x, p_y, p_z) \rightarrow T(\overrightarrow{p})$ is defined using N basis functions f_j and $3.N$ parameters α_{ij}.

$$T_i(\overrightarrow{p}) = \sum_{j=0}^{N-1} \alpha_{ij}.f_j(\overrightarrow{p}) \tag{7.2}$$

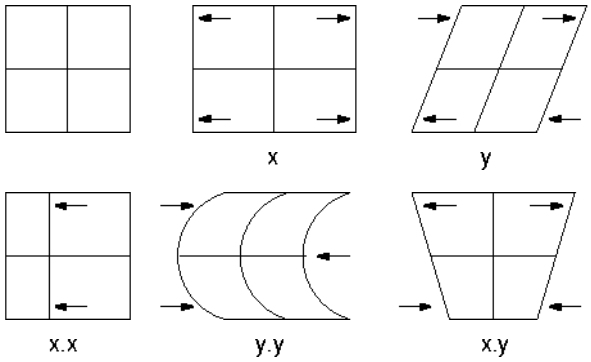

Figure 7.5: Set of elementary first and second degree transformations in the direction of the x-axis.

with $i \in \{x, y, z\}$. The affine transform is represented with $N = 4$, $f_j(\overrightarrow{p}) = 1$, p_x, p_y, p_z, and 12 coefficients α_{ij}. The 3D polynomial second degree transform uses $N = 10$, $f_j(\overrightarrow{p}) = 1$, p_x, p_y, p_z, p_x^2, p_y^2, p_z^2, $p_x p_y$, $p_x p_z$, $p_y p_z$, and 30 coefficients α_{ij}. The effects of some of these elementary transformations in 2D are illustrated at figure 7.5.

The matching itself is performed in two steps. First, the best affine transform is found using the cortical surface only in the matching criterion. Then, the second degree coefficients are optimized using both the cortical surface and the ventricular system as matching criterion. In both cases, the minimization of the criterion is performed using a gradient-descent algorithm in the $3N$-dimensional parameter space, after ortho-normalizing the functions f_j relatively to the mobile surface.

7.3.2 Results

The effect of the transformations found by the automatic registration procedure are illustrated at figure 7.6. The cortical surface and ventricular systems from the CBA appear in grey, the structures from the MR image in black. The grid allows us to visualize the deformations.

The ultimate aim of computerized atlases is to achieve a 3D representation of all identifiable anatomical structures in the individual brain. As shown in figure 7.7, this objective is well reached by the present work which allows to delineate unambiguously the sylvian fissure (sl),

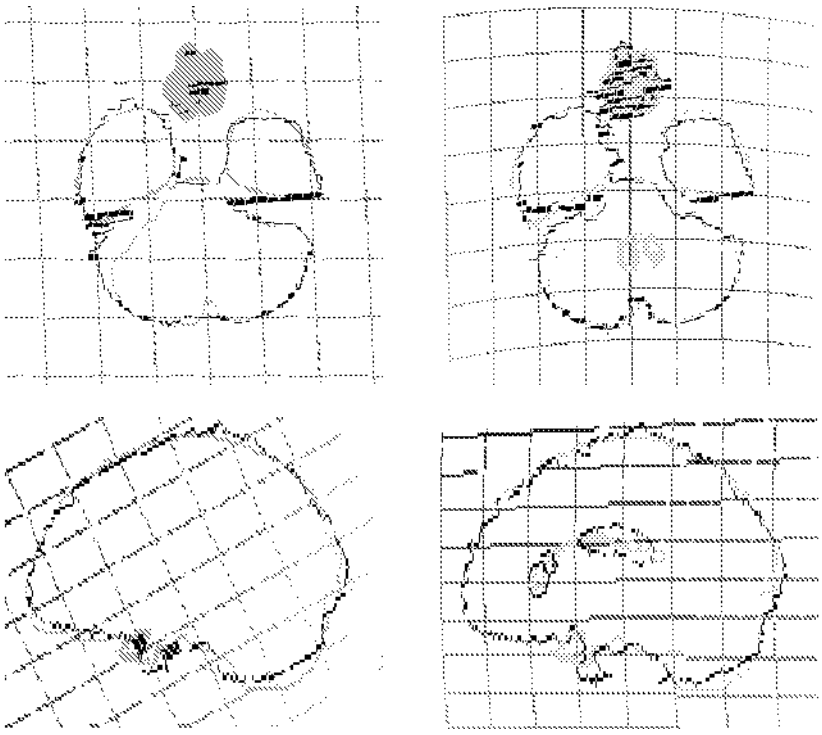

Figure 7.6: Effect of the registration transformations. *Left:* Affine transformation. *Right:* Additional second degree components. *Up:* Axial cut *Down:* Sagital cut.

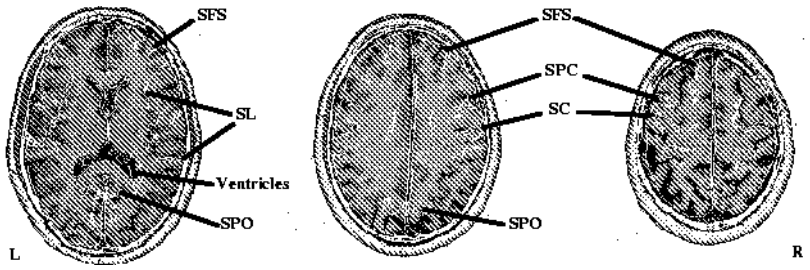

Figure 7.7: Some CBA structures overlaid over the MRI: Cortical surface, ventricular system and a few sulci.

the superior frontal (ssf), the precentral (spc), the central (sc) and the parieto-occipital (spo) sulci. As clearly seen on the top slice, the template for the left superior frontal sulcus does not match with the one of the subject despite the perfect matching of that sulcus on the contralateral hemisphere. Inter-hemispheric variations in the topography of the sulcal patterns represents an intrinsic limitation in this approach.

Chapter 8

Geodesic Distance Transformation

In this chapter we extend the works of Piper and Granum [122] and Verwer et al. [167] on geodesic distances. First, we generalize the definition of geodesic distances. Secondly, we propose two algorithms to compute the new geodesic DT. Thirdly we evaluate how accurately these algorithms approximate the Euclidean metric. Finally, we study their computational complexity.

8.1 Geodesic metrics

As defined in chapter 2, the geodesic distance between two pixels p and q is the length of the shortest path from p to q. Suppose $P = \{p_1, p_2, ..., p_n\}$ is a path between pixels p_1 and p_n, i.e. p_i and p_{i+1} are connected neighbors for $i \in \{1, 2, ..., n-1\}$ and p_i belong to the domain for all i. The path length $l(P)$ is defined as

$$l(P) = \sum_{i=1}^{n-1} d_N(p_i, p_{i+1}) \tag{8.1}$$

the sum of the neighbor distances d_N between adjacent points in the path. A particular geodesic metric is defined by which neighbors are considered to be connected and the values of d_N for each pair of connected neighbors.

The simplest metric is the geodesic version of the city-block distance, used in [122, 167, 93]. It is defined as

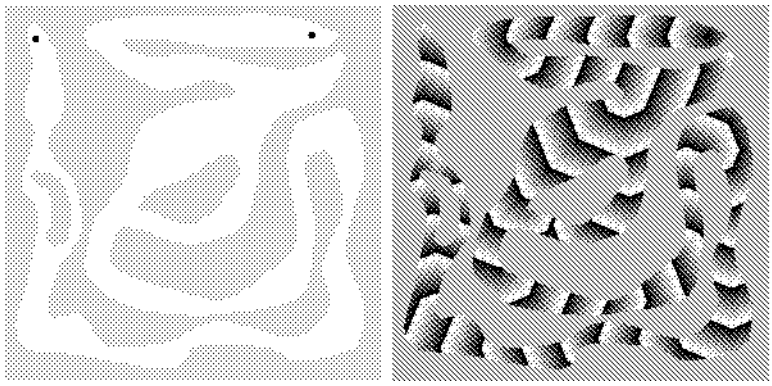

Figure 8.1: *Left:* Domain and object - *Right:* Geodesic DT with a 3-4 metric.

$$\begin{aligned} d_N(p,q) &= 0 \quad \text{if} \quad p = q \\ d_N(p,q) &= 1 \quad \text{if} \quad dist_e(p,q) = 1 \\ d_N(p,q) &= \infty \quad \text{if} \quad dist_e(p,q) > 1 \end{aligned} \tag{8.2}$$

In order to get better approximations of the Euclidean metric, geodesic chamfer metrics were also used. Piper [122] uses the 3-4 chamfer metric, Verwer [167] considers a 5-7 metric, and Verbeek [165] uses a 5×5 neighborhood with $1-\sqrt{2}-\sqrt{5}$ weights. For instance, for the 3-4 chamfer metric, this gives

$$\begin{aligned} d_N(p,q) &= 0 \quad \text{if} \quad p = q \\ d_N(p,q) &= 3 \quad \text{if} \quad dist_e(p,q) = 1 \\ d_N(p,q) &= 4 \quad \text{if} \quad dist_e(p,q) = \sqrt{2} \\ d_N(p,q) &= \infty \quad \text{if} \quad dist_e(p,q) > \sqrt{2} \end{aligned} \tag{8.3}$$

Such a geodesic DT is illustrated at figure 8.1. We compute the geodesic distance from the dot in the upper right corner, under the constraint that the domain is restricted to the white pixels. The distances are shown using a cyclic grayscale colormap so that the iso-distance curves are visible. Obviously this DT is only a coarse approximation of the geodesic Euclidean DT defined on a continuous domain.

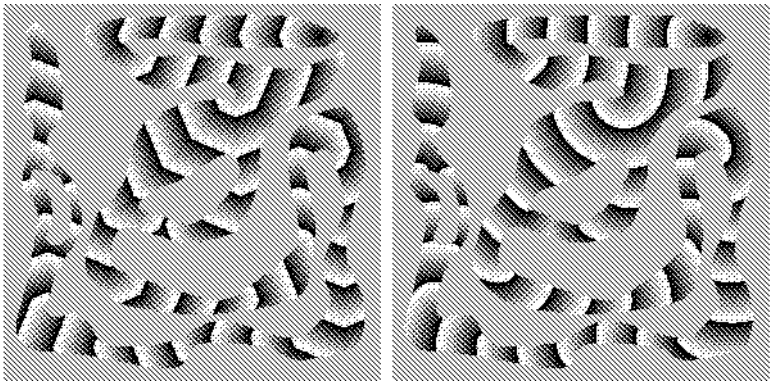

Figure 8.2: *Left:* Geodesic DT, ball size= $\sqrt{2}$ - *Right:* Geodesic DT, ball size= 6

We propose to use a more general definition of the geodesic DT. Given a ball B_d of radius d, the local distances d_N in (8.1) are defined as

$$d_N(p,q) = dist_e(p,q) \tag{8.4}$$

if $dist_e(p,q) \leq d$ and there is a path $R = \{r_1, r_2, ..., r_m\}$ such that $p = r_1$, $q = r_m$, $dist_e(r_j, r_{j+1}) = 1$, $\forall\ 1 \leq j \leq m-1$ and $dist_e(p, r_j) \leq d$, $\forall\ 1 \leq j \leq m$. Otherwise,

$$d_N(p,q) = \infty \tag{8.5}$$

We call this the B_d-geodesic DT. This means that there is a direct path between a pixel p and any pixel q in the neighborhood defined by the ball B_d, provided there is a path made of direct neighbors only and entirely included in $B_d(p)$, that connects p and q. This definition allows a large variety of trade-offs according to the size d. A larger d approximates the Euclidean DT better in obstacle-less regions. A smaller d follows the obstacles' shape better.

This definition is a generalization of those used in [122, 167, 165, 93]. In particular, $d = 1$ corresponds to the city-block metric, $d = \sqrt{2}$ is similar to the chamfer 3-4 metric and $d = \sqrt{5}$ corresponds to the chamfer $1 - \sqrt{2} - \sqrt{5}$ metric used by Verbeek [165]. In figure 8.2, distance maps using $d = \sqrt{2}$ and $d = 6$ are shown.

8.2 Geodesic DT algorithms

In chapter 2, we saw that the most efficient algorithm to implement usual geodesic distance transformations was that of Verwer, Verbeek and Dekker [167]. It scans the pixels in order of increasing distance by bucket sorting the pixels in the propagation front.

Unfortunately, there are two major hindrances to the use of this algorithm to implement the B_d-geodesic DT. First of all, the neighborhoods to consider during the propagation would have the size of B_d, which is prohibitive for any non-trivial d. For instance, $d = 6$ would require to consider 112 neighbors for each pixel. Secondly, the B_d-geodesic distances are real-valued. This makes them unsuitable as buckets' indexes.

8.2.1 Bucket sorting algorithm

In order to define an efficient implementation of the B_d-geodesic DT, we restrict the class of paths under consideration to the paths $\{p_1, p_2, ..., p_m\}$ such that

$$\begin{aligned} \forall\, 1 \leq i < m-1\,, \quad & d_N(p_i, p_{i+1}) \leq d \\ & \exists n \in N_4 \mid d_N(p_i, p_i + n) > d \\ \text{for } i = m\,, \quad & 1 \leq d_N(p_{i-1}, p_i) \leq d \end{aligned} \tag{8.6}$$

That is, all steps but the last one are approximately of length d. This slightly reduces the accuracy of the EDT approximation by restricting the path directions to those of the edge of B_d, instead of the complete ball. The accuracy of the EDT approximation is studied in section 8.3.

On the other hand, it makes the implementation of the DT much easier. For all p such that $D(p) \leq d$, the DT can be computed similarly to the PSN algorithm in chapter 3. For $d < D(p) \leq 2.d$, the distances can be computed by propagating from the edge of $O \oplus B_d$, i.e. $\delta(O \oplus B_d)$. For those pixels, $D(p)$ is defined as

$$D(p) = \min_{p' \in \delta(O \oplus B_d)} \{D(p') + dist_e(p, p')\} \tag{8.7}$$

Similarly, for $2.d < D(p) \leq 3.d$, we propagate from the edge of $O \oplus B_d \oplus$

B_d, and so on. More generally, the value of $D(p)$ for a pixel p can be split into two terms:

$$\begin{aligned} D(p) &= \sum_{i=1}^{m-1} d_N(p_i, p_{i+1}) \quad \text{with } p_m = p \\ &= \sum_{i=1}^{m-2} d_N(p_i, p_{i+1}) + d_N(p_m - 1, p_m) \\ &= D(p_{m-1}) + d_N(p_m - 1, p_m) \end{aligned} \tag{8.8}$$

where $p_{m-1} \in \delta(O \oplus B_d \oplus ... \oplus B_d)$ (m-1 times) and $p \in O \oplus B_d \oplus ... \oplus B_d$ (m times).

In the following algorithm, we remember two values for each pixel in the propagation front. d_{path} corresponds to the first term of (8.8) and is the length of the path from p_1 to p_{m-1}. The vector dp is the relative location of p from p_{m-1}. It corresponds to the second term of (8.8).

The algorithm proceeds step by step, first computing $D(p)$ in $O \oplus B$, then in $(O \oplus B) \oplus B$, ... Within each step, the propagation is ordered by bucket sorting on $dist_E(dp)$. When $dist_e(dp) > d$, the pixel is saved in a temporary buffer to be used in the following step.

Algorithm 8 *B_d-geodesic DT by bucket sorting.*

Input: an image I including object O and a domain M, the size d of the ball B_d defining the metric.
Output: a distance map D, with geodesic distance from O constrained by M

for all $p \in I$ **do** {Initialization}
  if $p \in O$ **then**
    $D(p) \leftarrow 0$
    put $p, (0,0), 0$ in $bucket(0)$
  else
    $D(p) \leftarrow \infty$
  end if
end for

```
repeat {Main Loop}
  repeat {dilate by B_d}
    for i = 0 → d do
      process(bucket(i))
    end for
  until bucket(i) is empty ∀ 0 ≤ i ≤ d
  bucket(1) ← buffer
until bucket(1) is empty

procedure process(list)
  for all p, dp, d_path in list do
    if D(p) = d_path + dist_e(dp) then
      for all n ∈ N_4 do
        if p + n ∈ M then
          d_new ← dist_e(dp + n)
          D_new ← d_path + d_new
          if D_new < D(p + n) then
            D(p) ← D_new
            if d_new > d then
              put p + n, n, D(p) in buffer
            else
              put p + n, dp + n, d_path in bucket(dist_E(dp + n))
            end if
          end if
        end if
      end for
    end if
  end for
end procedure
```

8.2.2 Circular propagation algorithm

Unfortunately, the above algorithm isn't always optimal, as illustrated in section 8.4. The reason for this is that the order used in the bucket sorting propagation, $dist_E(dp)$, may be different from the metric order $D(p)$. Indeed, the value of d_{path} isn't exactly $m.d$ after m dilations by B_d, but may vary because of the discrete shape of B_d.

Therefore, it is more efficient to replace the bucket sorting propagation by a variant of Ragnelmam's circular propagation algorithm [127]. In

this method, all pixels in the propagation front are stored in a single list, but are only propagated if the value of their distance is lower than a given threshold. If not, they are left in the list for later processing. After all pixels in the list have been considered for propagation with a given threshold, this threshold is incremented and the list is processed again.

For the B_d-geodesic DT, things are a little more complex. Indeed, in a non-convex domain, $dist_e(dp+n)$ isn't always larger than $dist_e(dp)$. In particular, when turning around a corner of the domain, it may decrease locally. A single list circular propagation could then create multiple propagation fronts. To handle this, we use two lists. $list1$ contains all pixels for which $D(p) \leq t$ and $list2$ those for which $D(p) > t$. The threshold t is only incremented once $list1$ is empty. The algorithm goes as follows.

Algorithm 9 *B_d-geodesic DT by circular propagation.*

Input: an image I including and object O and a domain M, the size d of the ball B_d defining the metric.
Output: a distance map D, with geodesic distance from O constrained by M

```
for all p ∈ I do {Initialization}
  if p ∈ O then
    D(p) ← 0
    put p, (0,0), 0 in list1
  else
    D(p) ← ∞
  end if
end for

t ← 0
while list1 is not empty do {Main loop}
  for all p, dp, d_path in list1 do
    if D(p) = d_path + dist_e(dp) then
      if D(p) ≤ t then
        for all n ∈ N_4 do
          if p + n ∈ M then
            d_new ← dist_e(dp + n)
```

```
                D_new ← d_path + d_new
                if D_new < D(p+n) then
                  if d_new > d then
                    d_path ← D(p)
                    dp ← (0,0)
                  end if
                  if D_new ≤ t then
                    put p+n, dp+n, d_path in list1
                  else
                    put p+n, dp+n, d_path in list2
                  end if
                end if
              end if
            end for
          else
            put p, dp, d_path in list2
          end if
        end if
      end for
      swap(list1,list2)
      t ← t+1
    end while
```

8.3 Accuracy

As mentioned in section 8.1, the accuracy of a geodesic DT is a trade-off between how precisely it approximates the Euclidean DT in obstacle free domains and how closely it follows the obstacles in their corners. The larger B_d, the better it approximates the Euclidean DT, but large B_d may cut through corners of the domain.

In this section, we focus on the first part of the trade-off, i.e. how closely the B_d-geodesic DT can approximate the Euclidean DT on an obstacle free domain. Let us consider an object pixel q and domain pixel p. The difference between $dist_e(p,q)$ and $D(p)$ comes from two sources.

First, when the direction of $p-q$ is not supported by the neighbors in B_d, the distance is approximated similarly to the chamfer metrics of section 2.2.1. This creates a systematic relative error depending on the

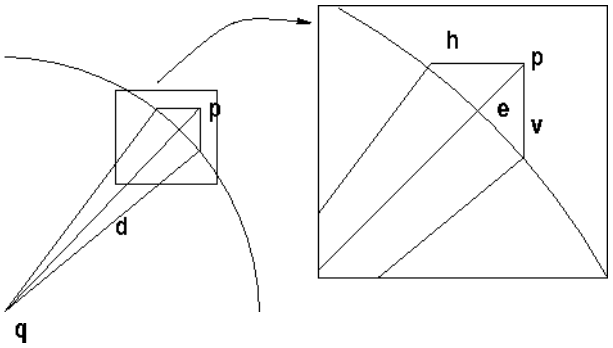

Figure 8.3: Non-Systematic error

angle made by $p - q$.

Secondly, the restrictions imposed by (8.6) introduce an additional error on the last segment of the path. This non-systematic error is null if $d_N(p_{m-1}, p_m) \approx d$, it is maximum when $d_N(p_{m-1}, p_m) = 1$. Let us try to evaluate it. In figure 8.3, the path from p to q is composed of two parts, as in (8.8). $D(p_{m-1})$ is considered as an error free distance d. $d_N(p_{m-1}, p_m)$ is the last segment of the path.

The true value of $D(p)$ is $D_{exact}(p) = dist_e(p, q) = d + e$. The computed value of $D(p)$ is $D_{comp}(p) = \min\{d + h, d + v)\}$ with $\max\{h, v\} = 1$. The error is $E = D_{comp}(p) - D_{exact}(p) = \min\{d + h, d + v\} - (d + e) = \min\{h, v\} - e$. It is maximum for a 45^o orientation, where $E = 1 - \cos(45^o) \approx 0.3$ pixel. This non-systematic error is an absolute error, and its importance fades for large distances.

In order to reduce the non-systematic error, one can use larger neighborhoods than N_4 during the propagation process. In particular, with N_8, the non-systematic error is reduced to $E = 1 - \cos(22.5^o) \approx 0.075$ pixel.

In figure 8.4, we show the systematic errors for 3 geodesic DT algorithms: the bucket sorting algorithm using N_4, and the circular (ordered) propagation algorithm both with N_4 and N_8. We notice that the use of N_8 - suggested to decrease the non-systematic error - also decreases the systematic error. Indeed, using N_8 in (8.6) increases the class of acceptable paths, and adds a few directions to $\delta(B_d)$.

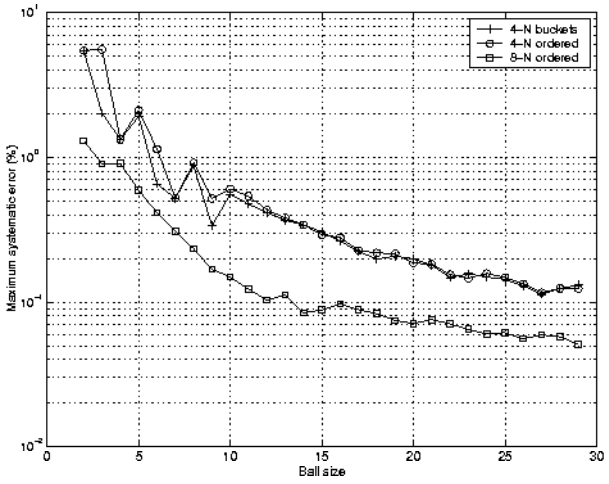

Figure 8.4: Systematic error

8.4 Computational complexity.

In order to evaluate the computational cost of the above algorithms, we apply them to the image of figure 8.1. For this test image, we compute the B_d-geodesic DT for balls of size $1 \leq d \leq 9$ using the three same algorithms as in section 8.3.

Figure 8.5 shows the computational cost of the bucket sorting and circular propagation algorithms according to the ball size d. The costs of the circular propagation algorithms are independent of the ball size d. In contrast, the bucket sorting algorithm has a cost highly dependent from the ball size. It is sometimes as fast as the others, and sometimes more than 4 times slower.

Similar results were obtained from a variety of other test images.

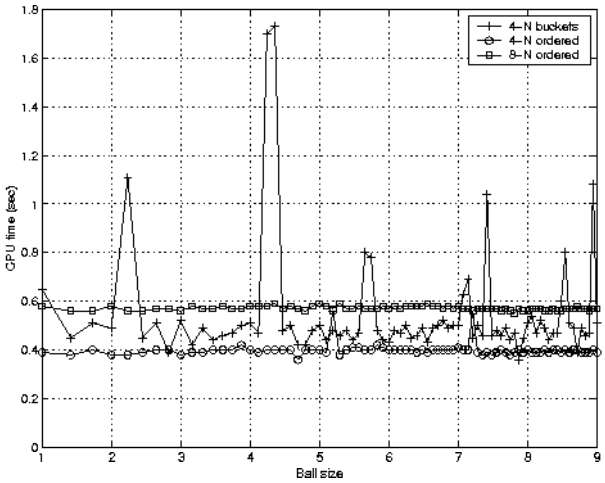

Figure 8.5: Computational complexity of the geodesic DT algorithms

Chapter 9

Application: Camera path-planning in virtual endoscopy

In this chapter, we illustrate the use of geodesic distance transformations in a virtual endoscopy application. First, we describe virtual endoscopy, its potential applications and some of the technical challenges it faces. Secondly, we show how the geodesic distance propagation can be back-tracked to provide the shortest path between two points. This is used to define the optimal path for the camera that flies through the 3D model of the organ. Thirdly, we propose a path centering technique, based on a snake model that smoothes the path while maximizing the distance from the path to edges of the model. Finally, some experimental results are presented.

9.1 Virtual Endoscopy

Over the last 5 years, virtual endoscopy has emerged as a new imaging and visualization technique to explore the human anatomy. It is probably best described by **Jolesz** et al. in [85]:

"Endoscopy is used to view the inner surfaces of hollow organs in a continuous fashion using optical, video-assisted technology. By changing the position of the endoscope, the operator can view the inside of an organ while controlling the viewing position and angle of the probe. [...]

Endoscopy yields detailed anatomy of the inner surfaces of a displayed wall segment

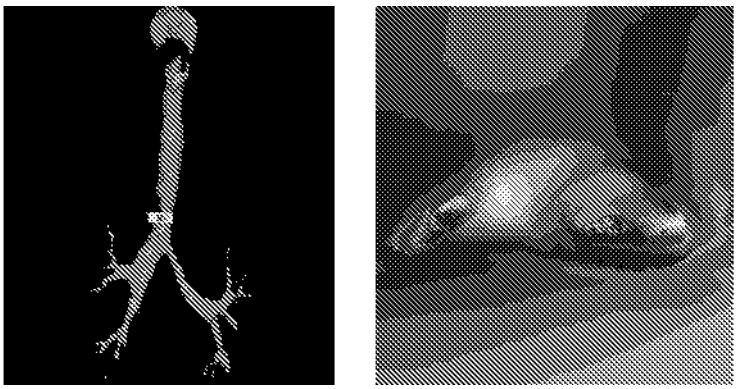

Figure 9.1: Virtual bronchoscopy. *Left:* 3d model and camera position. *Right:* endoscopic view. (image from Jolesz et al. [85])

[...] The method provides a direct, relatively high resolution views and, in most cases, obtains access through natural orifices or small incisions (i.e., laparoscopy). Nonetheless, endoscopic procedures can be uncomfortable and sedation or anesthesia may be required. Furthermore, endoscopes display only the inner surface of hollow organs and yield no information about the anatomy within or beyond the wall. This limitation prevents evaluation of the transmural extent of tumors and limits the ability to localize the lesion relative to surrounding anatomic structures.

CT and MRI provide cross-sectional images in which the inner surfaces of hollow organs are displayed at a much lower resolution than by endoscopy, but the techniques are noninvasive. Conventional slice-by-slice presentation of these data precludes contiguous viewing of the inner wall, forcing the radiologist to create a mental picture of anatomic continuity. This slice-based visual inspection is quite difficult and may be faulty, especially with highly convoluted tubular organs like small bowel or tortuous blood vessels. Traditional computer integration of cross-sectional data into 3D renderings have provided only outer surfaces of organs, which in the case of hollow or tubular structures are diagnostically less important. Despite these disadvantages, volumetric CT and MRI data can provide information not accessible by the endoscope. These important features include: information on tissue extending through and beyond organ walls, and the anatomic context of the entire volume, which permits correct localization of the lesion in relationship to adjacent anatomic structures.

Virtual endoscopic or fly-through methods [85, 84, 65, 171, 111, 172] which combine the features of endoscopic viewing and cross-sectional volumetric imaging may provide an advance in diagnosis, however, so far there are no comparative clinical data to demonstrate the advantages of virtual endoscopy. Nevertheless, virtual endoscopic presentation of image data enables the operator not only to explore the inner wall surfaces but also to navigate inside the virtual organs extracted from CT or MR images. [...]"

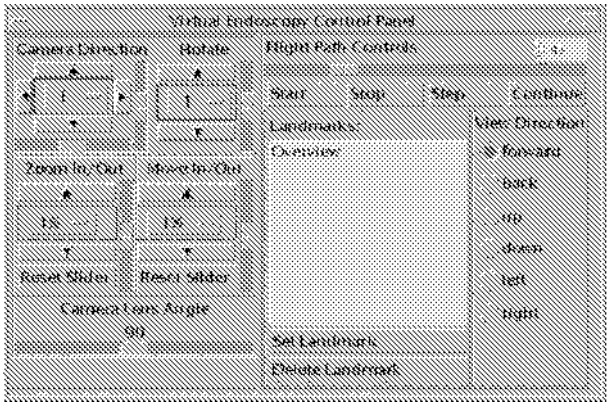

Figure 9.2: Graphical user interface for the interactive camera control. (Image from Frankenthaler et al. [57])

Because of these promising features, virtual endoscopy has been applied to many different organs, including bronchoscopy [171, 111], colonoscopy [172], pancreatoscopy [113], laryngoscopy [58], sinus endoscopy [67] or otoscopy [57, 90].

The extension of the technique to such different areas requires an appropriate choice of 3D imaging technique - Computed Tomography (CT), MR Imaging or a combination of those - , of the contrast agents, ... The image volume is then subject to a number of image processing tools in order to create a 3D model of the organ one wants to visualize. This model can then be either volume [134] or surface [97] rendered, the later allowing faster (interactive) visualization. For instance, in figure 9.1, the surface of the model was generated using the marching cubes algorithm [96].

After the 3D model of the organ has been created, the last critical step is the selection of the camera's position. **Jolesz** et al. [85] propose three techniques to guide the virtual camera through the surface models:

- *Manual camera movement*, where the mouse controls the camera position and focal point interactively. For instance, figure 9.2 shows the graphical user interface used by Frankenthaler et al. [57].

- *Key-framing*, that interpolates between user-selected key points, using cubic splines to calculate intermediate parameter values at any desired resolution. This technique was used to generate motion through open interior and exterior environments.

- *Automatic path planning*, a technique adapted from **Lengyel**'s robot path planning algorithm [93]. The camera is considered as a point robot and the walls of the organs as obstacles. The path planner labels all voxels with a distance to the terminus of the organ to be explored. Given a starting point, the path planning algorithm finds the shortest path to the goal using a steepest descent algorithm.

Automatic path planning is particularly useful for tubular organs, that present a challenge for both manual camera movement and key-framing. Indeed, manual camera movement is particularly difficult within confined spaces.

Unfortunately, the automatic path planning technique suffers from several imperfections. Lengyel's algorithm uses the city-block metric in the geodesic distance computation. This restricts the path to directions along the axis, which produces an unpleasantly jagged camera movement. In the order to compensate this effect, the path is smoothed. This can in turn introduce a new adverse effect, when the camera path crosses the object boundaries and includes positions outside of the organ.

This chapter proposes methods to improve the quality of automatic path planning. In section 9.2, we show how the geodesic DT algorithm of chapter 8 can be adapted to compute the shortest path between two points. In section 9.3, we propose a new smoothing technique based on the snake model, that searches a trade-off between the smoothness of the path and its centering in the middle of the bowel.

9.2 Computing the shortest path from the B_d-geodesic DT

In [93], the shortest path between pixels q and r is found as follows. First, the geodesic distance from q is computed. Then, the path is generated by steepest descent from r. Pixel r is the first point of the path,

then the neihgbor of r with the smallest distance value, then the neighbor's neighbor, and so on until q is reached.

Using the B_d-geodesic distance transformation in the first step, the steepest descent cannot be used to backtrack the distance propagation from r to q. Indeed, around corners of the domain, $D(p)$ may have local minima when the obstacles are smaller than B_d. Instead, we chose to store explicitely the origin of the propagation for each pixel, that is pixel p_{m-1} in eq. (8.8). The distance propagation can then be easily backtracked by iteratively looking up the origin of the current last pixel in the path. The algorithm goes as follows

Algorithm 10 *Computes the shortest path between two points.*

Input: an image I and a domain M, points $q, r \in M$, the size d of the steps in the path.
Output: The shortest path $P = \{p_1, p_2, ..., p_m\}$ with $p_1 = r$, $p_m = q$, $p_i \in M$, $\forall i$ and $d_N(p_i, p_{i+1}) \approx d$

for all $p \in I$ **do** {Initialization}
  $D(p) \leftarrow \infty$
end for
$D(q) \leftarrow 0$
put $q, (0,0), 0$ in $list1$

$t \leftarrow 0$ {Geodesic DT, where we keep track of $Origin(p)$ for each pixel}
while $list1$ is not empty **do**
  for all p, dp, d_{path} in $list1$ **do**
    if $D(p) = d_{path} + dist_e(dp)$ **then**
      if $D(p) \leq t$ **then**
        for all $n \in N_4$ **do**
          if $p + n \in M$ **then**
            $d_{new} \leftarrow dist_e(dp + n)$
            $D_{new} \leftarrow d_{path} + d_{new}$
            if $D_{new} < D(p + n)$ **then**
              $Origin(p + n) \leftarrow p - dp$ { * }
              if $d_{new} > d$ **then**
                $d_{path} \leftarrow D(p)$
                $dp \leftarrow (0,0)$
              end if

```
                if  D_new ≤ t  then
                  put p + n, dp + n, d_path in list1
                else
                  put p + n, dp + n, d_path in list2
                end if
              end if
            end if
          end for
        else
          put p, dp, d_path in list2
        end if
      end if
    end for
    swap(list1,list2)
    t ← t + 1
  end while

  p_1 ← r {Path generation: P = {p_1, p_2, ..., p_m} with p_1 = r, p_m = q }
  i ← 1
  while p_i ≠ q do
    p_{i+1} ← Origin(p_i)
    i ← i + 1
  end while
  m ← i
```

The core of the algorithm is highly similar to that of section 8.2.2, with the only additional line marked by $\{*\}$. The initialization is modified to take q as the only object pixel. The path generation step is of course new. Note that it can be used to generate paths from any pixel r to the same target q.

The result of this algorithm is illustrated at figure 9.3 and the synthetic 2D data used in figures 8.1 and 8.2.

9.3 Path centering

The path found at figure 9.3 is visibly a good approximation of the shortest path between the two points. It follows the corners of the domain when it bends and it crosses open spaces in straight lines. It does not

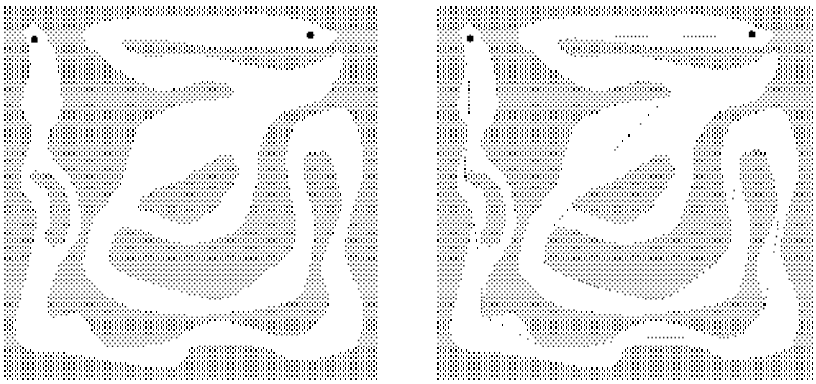

Figure 9.3: Shortest path. *Left:* Domain and points to link *Right:* path computed by the path-planning algorithm with $d = 6$.

present the jagged-ness of Lengyel's paths.

On the other hand, it is not the optimal path for our virtual endoscopy application. In colorful terms, the fly through behaves too much like a formula one to be cosy for the passengers. That is, the camera should follow a more centered path through the bowels, while remaining smooth.

This behaviour can be obtained using a variant of the snake model introduced by **Kass** et al. [89]. In this model, the path is considered as a parameterized curve - or snake - $v(s) = (x(s), y(s))^T$ with $s \in [0, 1]$. The snake evolves in order to minimize an energy defined as

$$E(v) = \int_0^1 w_1 \left|\frac{\partial v}{\partial s}\right|^2 + w_2 \left|\frac{\partial^2 v}{\partial s^2}\right| + P(v)ds \tag{9.1}$$

where w_1 and w_2 are smoothing terms and $P(v)$ is the image term. In segmentation application, $P(v)$ is typically a decreasing function of the image gradient, forcing the snake to move towards the edges in the image.

Typically, equation (9.1) is solved by deriving an evolution equation, then solving it numerically using finite differences. Practically, it means iterating

$$v^{t+1} = (I + \tau.A)^{-1}.\left(v^t + F(v)\right) \tag{9.2}$$

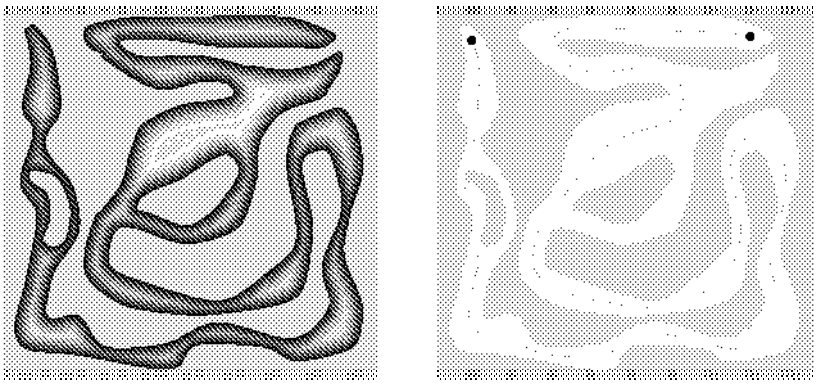

Figure 9.4: *Left:* Euclidean DT from the walls of the maze *Right:* Centered path

where I is the identity matrix, τ the chosen time step, A the smoothness matrix corresponding to the w_1 and w_2 terms and $F(v)$ the image force derived from the potential energy $P(v)$.

In order to center and smooth the path in figure 9.3, we adapt the snake model as follows. First of all, the image potential energy is used to drive the snake towards the center of the bowel. To achieve this, we use a decreasing function of the Euclidean distance from the edges of the domain, illustrated at figure 9.4. Also, the force deriving from this potential only needs to applied perpendicularly to the snake, because the parallel component would only modify the sampling of the curve and not its location. Therefore, the image force is defined as follows

$$\overrightarrow{F(v)} = \overrightarrow{G(v)} - \frac{(\vec{v}.\overrightarrow{G(v)}).\vec{v}}{|v|^2} \tag{9.3}$$

$$\text{with } \overrightarrow{G(v)} = -\nabla(-D(v)) \tag{9.4}$$

with arrows omitted over v where its vectorial nature is not important.

For the smoothing term, we use a simplified model. w_2 is set to 0 so that only the first derivative is used. Then, $(I+\tau.A)^{-1}$ is approximated by $I-\tau.A$. Although this is potentially less stable than matrix inversion, experiments show that the smoothness of the distance-based image energy is a sufficient guarantee of stability. With this simplified model, the snake iteration can be de-coupled into two sub-iterations

$$v^{t'} = v^t + F(v^t) \tag{9.5}$$

$$v^{t+1}(i) = (1-\tau).v^{t'}(i) + \frac{\tau}{2}.(v^{t'}(i-1) + v^{t'}(i+1)) \tag{9.6}$$

where $v(i)$ is the discrete representation of the curve v.

In our experiments, the snake converges in a few iterations and stabilizes itself very robustly. The resulting centered and smoothed path is illustrated at figure 9.4.

9.4 Experimental results

The method proposed in sections 9.2 and 9.3 was applied on real medical data, for virtual endoscopies of the aorta and of the colon. The aorta is a relatively smooth structure, but it contains several branches. The colon is a very complex structure, but has a simple tubular topology.

The aortascopy was performed on a structure manually segmented from the abdominal CT of figure 9.5 by doctors at the Surgical Planning Laboratory - Boston, MA - as part of their abdominal atlas. The CT image, and the atlas derived from it, is made of 114 slices of 256×256 voxels, which requires to adapt the algorithm of section 9.2 to 3-dimensional anisotropic data. This does not present any major difficulty.

The typical result generated by an algorithm such as Lengyel's is illustrated at figure 9.6. The path is composed of segments in a reduced set of directions, which makes it irregular. It touches the edges of the structure in all turns in order to keep the path as short as possible.

The results of our algorithm are found in figure 9.7. The paths show similar qualities to those computed on synthetic 2D data at figure 9.4. The path on the left is the shortest possible. It is relatively smooth thanks to the variety of possible segment directions. The path on the right was smoothed and centered.

The colonoscopy was performed using the CT scan of figure 9.8. Because it was filled with air, the colon appears as a darker structure, which makes it possible to segment using a simple threshold. The segmentation and generation of the 3D model from the CT volume were

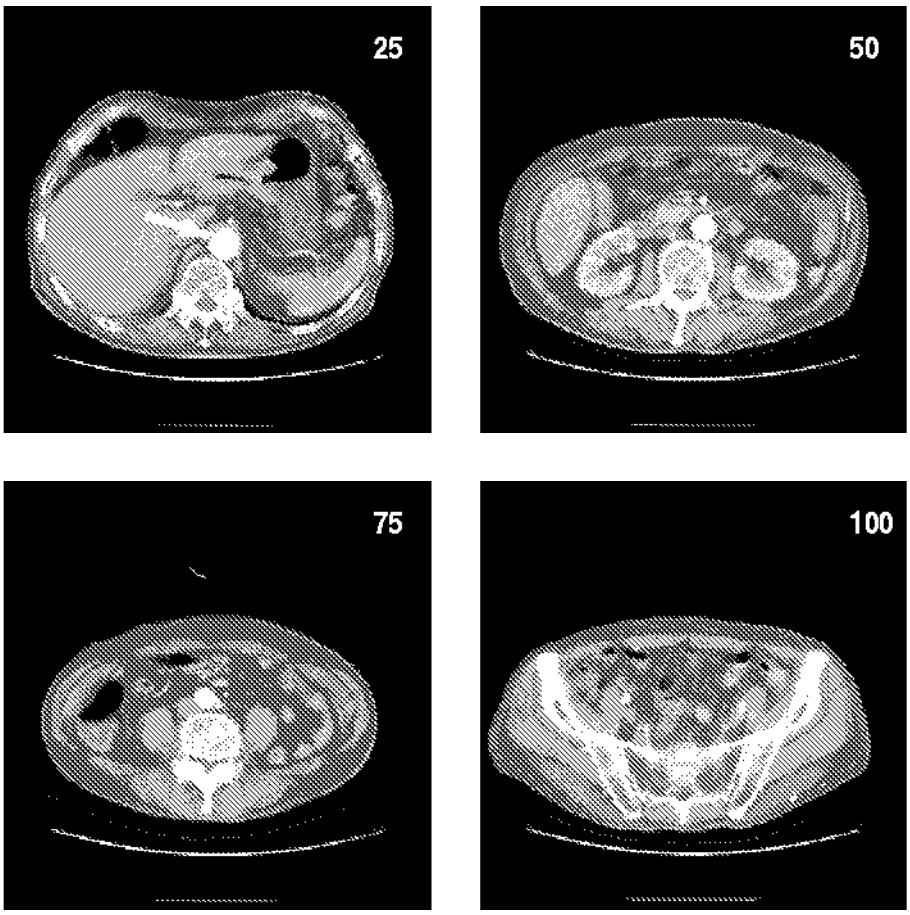

Figure 9.5: Computed Tomography of the abdomen. The aorta is shown using a white arrow in slice 25.

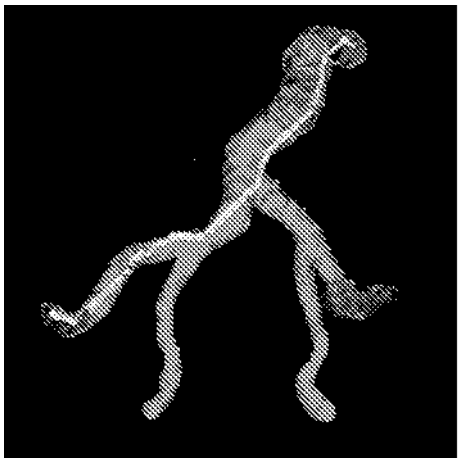

Figure 9.6: Virtual endoscopy of the aorta: camera path generated by an algorithm similar to Lengyel's

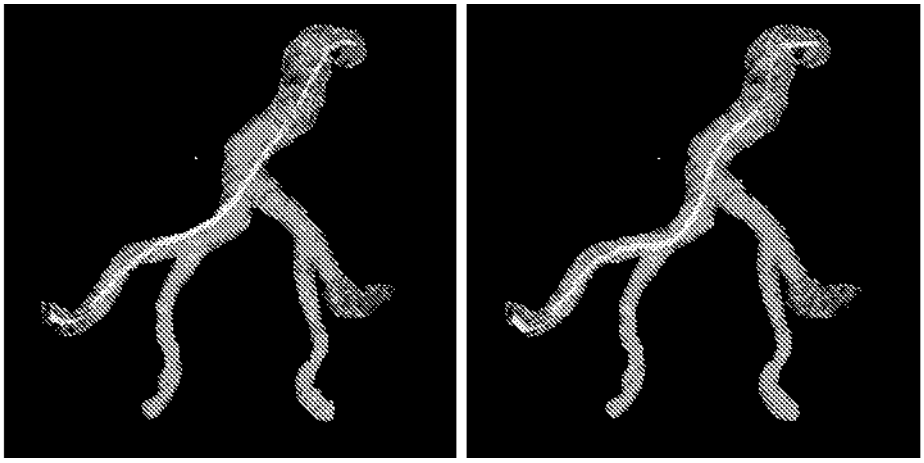

Figure 9.7: Virtual endoscopy of the aorta. *Left:* Shortest path *Right:* Path centered and smoothed by the snake energy minimization.

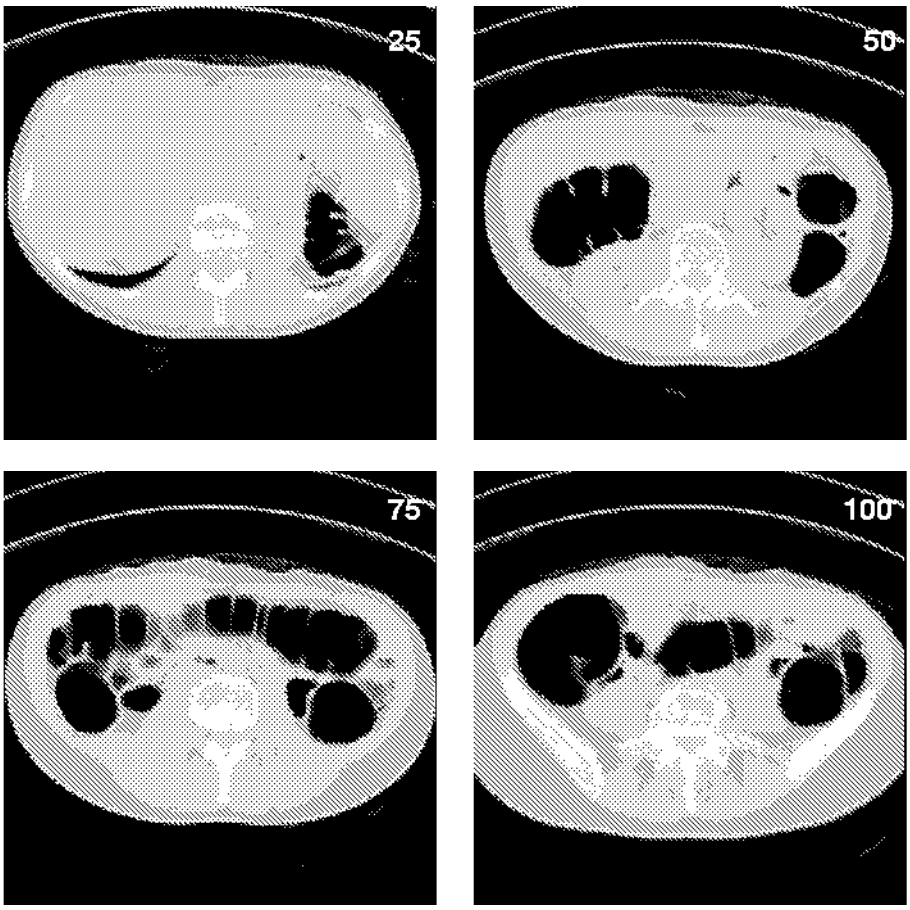

Figure 9.8: Computed Tomography of the colon.

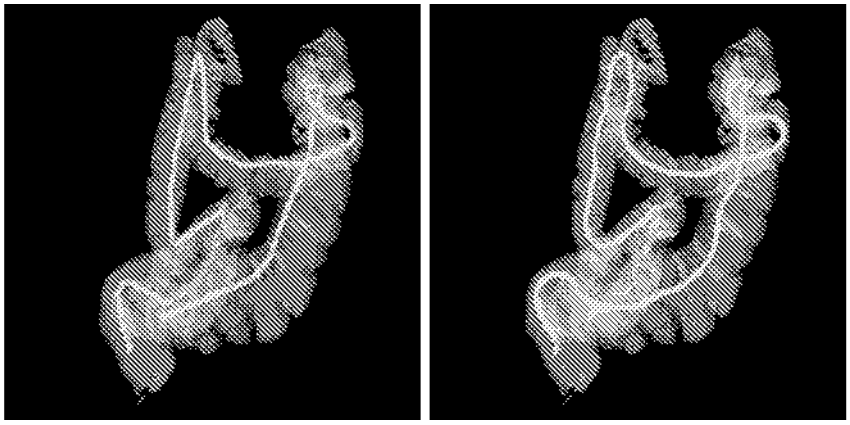

Figure 9.9: *Left:* Shortest path *Right:* Centered path flying through the colon

Figure 9.10: Endoscopic view

provided by Dr. Shigeo Okuda, of the SPL.

The paths computed by our method are shown at figure 9.9. Once again, we show both the shortest and the centered paths. This time, one can see very abrupt changes in directions on the shortest path where the colon turns. Those abrupt changes are smoothed on the centered path.

Finally, a typical colonoscopic view is illustrated at figure 9.10. Unfortunately, the paper medium does not allow us to show the dynamic visual

effect of the fly-through.

Chapter 10

k-NN classification and k-distance transformation

In this chapter, we extend Warfield's work on fast k-NN classification for multi-channel image data [175]. First, we present a short review of k-NN classification methods. Then, we describe a new k-DT algorithm by propagation. Finally, we show that it has an optimal algorithmic complexity.

10.1 Introduction

The k-Nearest Neighbors (k-NN) classification rule is a technique for non-parametric supervised pattern classification. Given the knowledge of N prototype patterns (vectors of dimension D) and their correct classification into several classes, it assigns an unclassified pattern to the class that is most heavily represented among its k nearest neighbors in the pattern space.

The first formulation of the above rule appears to have been made by **Fix** and **Hodges** [53] in 1951. They established the consistency of the rule for sequences such that $k \to \infty$ and $k/N \to 0$. In [54], they investigate the small sample performance of the rule numerically, under the assumption of normal statistics.

The $k-NN$ decision rule makes no assumption on the underlying probabilistic distribution of the samples points and of their classification. Therefore, the probability of error R of this rule must be at least as

large as the Bayes probability of error R^* - the minimum probability of error when the distribution is known. **Cover** and **Hart** [24] show that, with certain statistical assumptions, the conditional risk for the 1-NN rule is $R \leq 2.R^*$. For the k-NN rule, the risk is bounded by $(1+1/k)R^*$ [23]. Thus, when $k \to \infty$, $R \to R^*$.

If one implements the k-NN rule with a brute-force method to classify F patterns using N prototypes, it requires $F.N$ distance computations and $o(F.N.\log(N))$ comparisons. This is often unpractical with large data sets, and has led to the research of efficient algorithms.

Hart [71], **Gates** [63] and **Tomek** [157] reduce the number of prototypes to consider while trying not to affect the accuracy of the resulting classification. Hart's Condensed Nearest Neighbor (CNN) and Gates' Reduced Nearest Neighbor (RNN) rules only apply to 1-NN classification according to their authors.

Belkasim [4] proposes to make use of the natural clustering of the training data to reduce the number of prototypes to which distances must be computed. In the worst case, this partitioning of the prototype data has no benefit.

Fukunaga and **Narendra** [60] propose a branch and bound approach, later improved by **Kamgar-Parsi** and **Kanal** [87], **Niemann** and **Goppert** [115] and finally **Jiang** and **Zhang** [83]. In this approach, the prototypes are hierarchically decomposed into disjoint subsets. A powerful tree-search algorithm, the branch and bound method (see survey by **Lawler** [91]), is then applied to the resulting groups.

Friedman [59] orders the training data along the axis with the maximal sparsity for each pattern. This restricts the computations to a band around the projection of the test data onto this axis. The expected number of distance computations is reduced to $O(F.k^{1/D}.N^{1-1/D})$ with D dimensional vectors.

Finally, **Warfield** [175] considers a particular type of applications where the number of possible patterns is much smaller than the number of patterns to classify. One such application is the classification of MRI data, where patterns consists of 2-3 channels (D) of data quantified over a

small range of values, for a 3D volume including typically $1 - 6 \times 10^6$ voxels. Then, it becomes efficient to compute a lookup table for every possible pattern, then to classify the voxels by accessing the location of their values in the lookup table.

The computation of this lookup table is essentially a k-distance transformation problem, where the image is the pattern space and the object pixels are the prototype patterns. Warfield's k-DT algorithm is based on Borgefors' chamfer DT [10] in 2D and on Ragnelmam's corner EDT [128] in higher dimensions. The difference with those methods is that the k nearest patterns (object pixels) are considered, instead of 1. It goes as follow:

Algorithm 11 *Warfield's k-distance transformation*

Insert training data patterns identifiers into the map.
for all distance transform mask scans **do**
  for all pixels p in the map **do**
    Propagate the k-NN identifiers from each mask edge pixel to the center pixel p,
    Compute distance from p to each of the training patterns,
    Sort in order of increasing distance,
    Select the identifiers of the k nearest patterns
  end for
end for

The method requires $O(2^D F(D+1)k)$ distance computations and $O(F(D+1)k \log((D+1)k))$ comparisons. Warfield shows that - for this type of applications - it is an order of magnitude faster than the above k-NN methods.

10.2 The k-DT algorithm.

In terms of k-NN classification, the k-DT problem can be formulated as follows: given a set of N prototype patterns $q(l)$ labeled from $l = 0$ to $l = N - 1$, determine the labels of the k nearest prototypes $\text{kNN}(p) = \{\text{NN}_i(p)\ ,\ 0 \leq i < k\}$ for every possible pattern p in the pattern space I.

Similarly, the k-DT problem can be described in familiar image terms: given a set of N object pixels $q(l) \in O$ labeled from $l = 0$ to $l = N - 1$, determine the k nearest object pixels kNN(p) for every pixel p in the image I.

Although both formulations look similar, there is a minor difference in the fact that the "image" formulation supposes that prototypes are unique, i.e. that $q(l_1) = q(l_2) \Rightarrow l_1 = l_2$. The "pattern space" formulation does not make this assumption. There can be several identical patterns among the prototypes.

The key to our approach is to notice that in algorithm 11, a lot of computational power is wasted in two ways. First, as pointed out by Ragnelmam in [127], the raster scanning procedure propagates the information further than needed. This is especially true for pattern spaces in higher dimensions, where 2^D scans are performed. Secondly, a large part of the computational cost is due to the sorting procedure.

We propose to generate the k-DT using ordered propagation to scan the pattern space, starting from the prototype patterns, then to their neighbors, their neighbors' neighbors, ... by order of increasing distance. The ordered propagation is achieved by bucket sorting the patterns in the propagation front, similarly to the Euclidean DT algorithm of chapter 2. The benefits of the method are twofold. First the propagation of every label is restricted to the zone of influence of the pattern it represents. Secondly, it is possible, by delaying the updates of the propagated patterns, to avoid the sorting procedure completely.

For every pixel p in the propagation front, we store both its coordinates $(p_{x_1}, p_{x_2}, ..., p_{x_D})$ and the propagating label l. The propagation front is implemented as a number of buckets $bucket(index)$. A propagating label l at pixel p is stored in $bucket(dist_E(p, q(l)))$. By emptying the buckets by increasing $index$ value, we scan the image in order of increasing distances.

In addition to the kNN(p) label maps that we generate, we store three additional temporary information for each pixel p. $i_{cur}(p)$ indicates how many labels have reached p at any step of the algorithm. d_{cur} is the value of the distance from p to the prototype of the last label to have

reached p, i.e $d_{cur} = dist_E(p, q(\mathrm{NN}_{icur-1}(p)))$. If more than one label among $\mathrm{kNN}(p)$ correspond to a prototype at distance d_{cur}, then i_{dcur} stores the first i for which $d_{cur} = dist_E(p, q(\mathrm{NN}_i(p)))$.

Let us now consider that distance d has been reached, i.e all $buckets(d')$, $d' < d$ have been emptied and that the couples (p, l) in $bucket(d)$ are being processed. Label l should be added to $\mathrm{kNN}(p)$ if two conditions are fulfilled. First, there should be less than k labels in $\mathrm{kNN}(p)$ already, since all labels in there are such that $dist_E(p, q(l)) \leq d$, and therefore better candidates than l. This is easily tested by checking that $i_{cur} < k$. Secondly, label l should not belong to $\mathrm{kNN}(p)$ yet. If $d_{cur}(p) < d$, it obviously does not. Otherwise, i.e when $d_{cur}(p) = d$, all labels $\mathrm{NN}_i(p)$ with $i_{dcur} \leq i < i_{cur}$ should be checked.

Finally, when a couple (p, l) is added to $\mathrm{kNN}(p)$, label l is then propagated towards p's neighbors. Of course, only those neighbors $n \in N$ that lead to $dist_E(p+n, q(l)) > d$ need to be considered, i.e. those in the same direction as $p - q(l)$. Practically, we use the 2D-direct neighborhood, for which this test is extremely simple. For instance, in 2 dimensions, with the 4-direct neighborhood, neighbor $n = (1, 0)$ is used if $p_x - q(l)_x \geq 0$, $n = (-1, 0)$ is used if $p_x - q(l)_x \leq 0$, ...

Algorithm 12 *k-DT algorithm by bucket-sorting propagation.*

Input: N prototypes $q(l)$ with labels l, $0 \leq l < N$
Output: the sets $\mathrm{kNN}(p) = \{\mathrm{NN}_i(p),\ 0 \leq i < k\}$, $\forall p \in I$

for all $p \in I$ **do** {Initialization}
  $i_{cur}(p) \leftarrow 0$
  $d_{cur}(p) \leftarrow -1$
  $i_{dcur}(p)$ undefined
  $\mathrm{NN}_i(p)$ undefined $\forall i,\ 0 \leq i < k$
end for
for $l = 0$ to $N - 1$ **do**
  put $(q(l), l)$ in $bucket(0)$
end for
$d \leftarrow 0$

repeat {Main loop}

```
    while bucket(d) is not empty do
      get (p, l) from bucket(d)
      if i_cur(p) < k then
        if d_cur(p) < d then
          assign(p, l)
          propagate(p, l)
        else { in this case, d_cur(p) = d}
          if NN_j(p) ≠ l ∀j, i_dcur(p) ≤ j < i_cur(p) then
            assign(p, l)
            propagate(p, l)
          end if
        end if
      end if
    end while
    d ← d + 1
  until all buckets are empty

  procedure assign(p, l)
    NN_{i_cur(p)}(p) ← l
    i_cur(p) ← i_cur(p) + 1
    if d_cur(p) ≠ d then
      i_dcur(p) ← i_cur(p)
    end if
    d_cur(p) ← d
  end procedure

  procedure propagate(p, l)
    for all n ∈ N do
      if (p − q(l)).n ≥ 0 then
        put (p + n, l) in bucket(dist_E(p + n, q(l)))
      end if
    end for
  end procedure
```

As usual, the implementation of this algorithm requires a special attention. In particular, the dynamic data structure used to implement the buckets should allocate memory in chunks and not element by element. In the experiments below, we use chunks of 2048 elements each. On the other hand, the k NN_i label maps and the additional temporary information are stored statically.

Finally, the computation of $dist_E(p,q)$ is fastened by using small lookup tables for the squares of integers, i.e. $dist_E(p,q) = \sum_{i=1}^{D} \mathrm{sq}[p_{x_i} - q_{x_i}]$ with $\mathrm{sq}[n] = n^2$ for all integer n within the useful range. Of course, any other integer-value metric could be used instead of $dist_E$.

10.3 Computational complexity.

In [175], it is shown that using a k-DT to compute a lookup table is the most efficient method to perform the k-NN classification of a large data set where the number of possible different patterns is comparable of lower than the size of the data set. Thanks to this result, we can restrict the present complexity analysis to showing that the algorithm of section 10.2 has an optimal computational complexity for a k-DT.

The output of the algorithm is made of k maps covering the F patterns. The complexity of the output is therefore $\mathrm{O}(F.k)$. This is the absolute lower bound for the optimal algorithmic complexity of a k-DT. More realistically, a k-DT algorithm should at least consider the neighbors of a pixel to compute its value, which means a $\mathrm{O}(F.D.k)$ complexity in D dimensions.

In algorithm 12, the procedure $assign(p,l)$ is called exactly $F.k$ times, since as soon as $\mathrm{NN}_{icur(p)}(p)$ is assigned a value, $i_{cur}(p)$ is incremented. Because they are always called together, the procedure $propagate(p,l)$ is also called exactly $F.k$ times. In that procedure, the neighbors of p are entered in the buckets structure. Using 2D-direct neighborhoods, restricted to those in the same direction as $p - q(l)$, there are between D and $2D$ neighbors propagated for each of the $F.k$ pixels that enter $propagate(p,l)$. Thus, the total amount of elements passing through the buckets is $\mathrm{O}(F.D.k)$.

The distance $dist_E(p,q(l)$ is only computed inside the $propagate(p,l)$ procedure, in order to determine in which bucket (p,l) should go. Thus, the total amount of distance computations is exactly the same as the total amount of elements passing through the bucket structure, i.e $\mathrm{O}(F.D.k)$.

Finally, the number of comparisons performed inside the main loop for

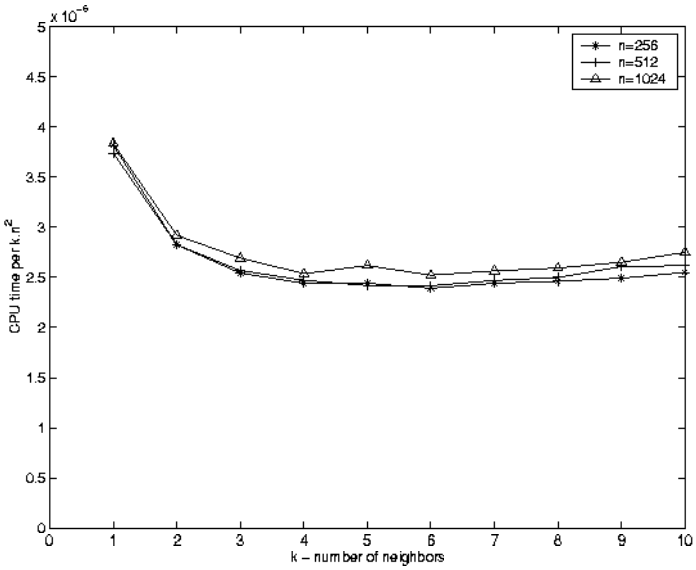

Figure 10.1: k-DT algorithmic complexity: dependence from the number of neighbors k for several image sizes

an element (p, l) taken from $bucket(d)$ is fixed, unless $d_{cur}(p) = d$. In this case, it will be compared $i_{cur}(p) - i_{dcur}(p)$ times. This number is in average very low when prototypes are unique. The total number of comparisons is then also $O(F.D.k)$.

In the worst case, with k identical prototypes at every prototype location, the average number of comparisons is close to k. It raises the number of comparisons to $O(F.D.k^2)$. This could be avoided by replacing prototypes (represented by a label l) by prototype locations (represented by a label l and a number m of occurrences in that location). Nevertheless, for practical applications, this does not appear to be needed.

In order to confirm this theoretical analysis, we ran 3 experiments on synthetical 2D data, varying the number k of nearest neighbors, the size of the $n \times n$ images and the number N of prototypes. In experiment 1 (Figure 10.1), k varies from 1 to 10, 3 values of n are considered and N is fixed to 1000. In experiment 2 (Figure 10.2), n varies from 128 to 1024, 3 values of k are considered and $N = 1000$. In experiment 3 (Figure 10.3), N varies from 100 to 1000, n is fixed to 512 and 3 values

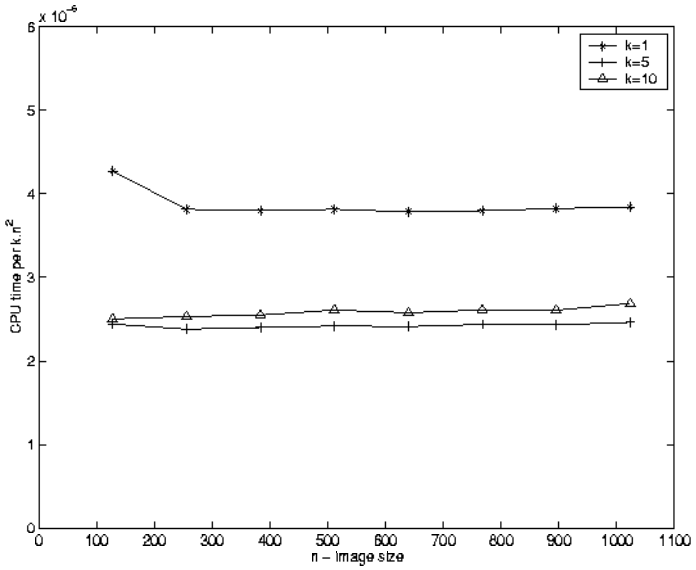

Figure 10.2: k-DT algorithmic complexity: dependence from the image size ($n \times n$) for several values of k.

of k are considered. In all experiments, the prototypes are randomly generated.

Theoretically, the computational complexity is $O(F.D.k) = o(n^2.2.k)$, so that the ratio of the needed CPU time by $k.n^2$ should be a constant. The 3 experiments were performed on a Pentium II at 233 MHz computer. CPU times were recorded and their ratio to $k.n^2$ are displayed in Figures 10.1 to 10.3.

In experiment 1 (Figure 10.1), the CPU time per k.pixel is constant for $k > 3$. For $k \leq 3$, the fixed cost of handling the additional information in i_{cur},d_{cur} and i_{dcur} is a non-negligible factor, so that the CPU time per k.pixel is slightly higher.

In experiment 2 (Figure 10.2), the image size has no influence on the CPU time per k.pixel. In experiment 3 (Figure 10.3), the number of prototypes has no influence either. In both Figures, the line for which $k = 1$ is significantly higher than the other two, as suggested by the results of experiment 1.

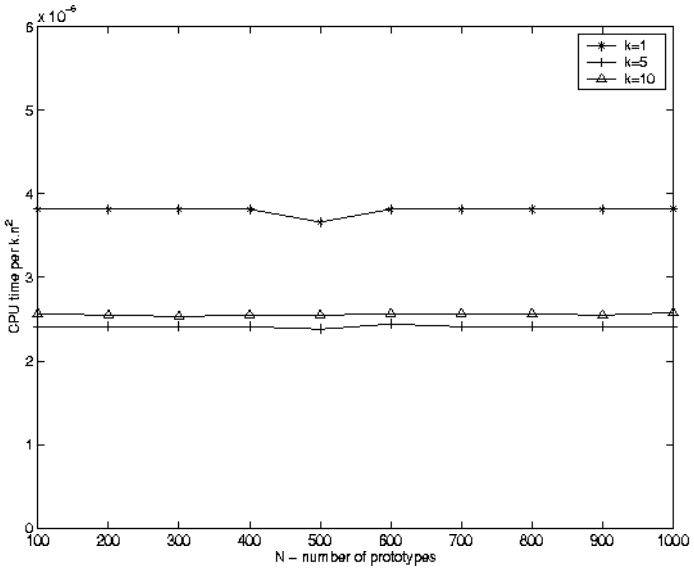

Figure 10.3: k-DT algorithmic complexity: dependence from the number N of training patterns for several values of k.

	Distance computations	Comparison operations
Cover	$F.N$	$O(F.N.\log(N))$
Friedman	$O(F.k^{1/D}.N^{1-1/D})$	$D.N.\log(N)$+$O(F.k^{1/D}.N^{1-1/D})$
Warfield	$O(2^D.F.(D+1).k)$	$O(2^D.F.(D+1).k.\log((D+1).k)$
This k-DT	$O(F.D.k)$	$O(F.D.k)$

Table 10.1: Complexity of k-NN classification algorithms, with F the number of patterns to classify, N the number of training prototypes, k the number of nearest neighbors and D the dimension of the pattern space.

Finally, let us notice that the CPU time required by the k-DT algorithm can be further reduced if, in the application considered, patterns that are too far away from all prototypes would better be not classified. In this case, the propagation can be stopped as soon as a critical distance is reached, even though the whole pattern space hasn't been labeled yet.

In table 10.1, extended from the original in [175], the performance of this k-DT algorithm is compared to the brute force algorithm of Cover

[24], and those of Friedman [59] and Warfield [175].

Chapter 11

Application: tissue classification in T1, T2 MR images.

In this chapter, we apply the k-NN rule to the supervised classification of tissues in pairs of T1- and T2-weighted MR images of the brain. First, we remind the nature of the T1 and T2 signals in MRI, and their medical significance. Then, we review a few supervised methods to classify multi-channel MR images of the brain. Finally, we illustrate this by classifying a T1-T2 pair of images of a brain with multiple sclerosis lesions.

11.1 The physics of T1- and T2-weighted MRI

Protons in a magnetic field have a microscopic magnetization, and act like toy tops that wobble as they spin. The rate of the wobbling, or precession, is the resonance or Larmor frequency. In the magnetic field of an MRI scanner, there is approximately the same number of proton nuclei aligned with the main magnetic field B_0 as counter aligned. Still, the aligned position is slightly favored because the nucleus has a lower energy in this position. This results in a net macroscopic magnetization pointing in the direction of B_0.

The exposure of the nuclei to a RF radiation (the B_1 field) at the Larmor frequency causes the nuclei in the lower energy state to jump to the higher energy state. At the macroscopic level, this causes the net magnetization to spiral away from the B_0 field. In a field of reference rotating

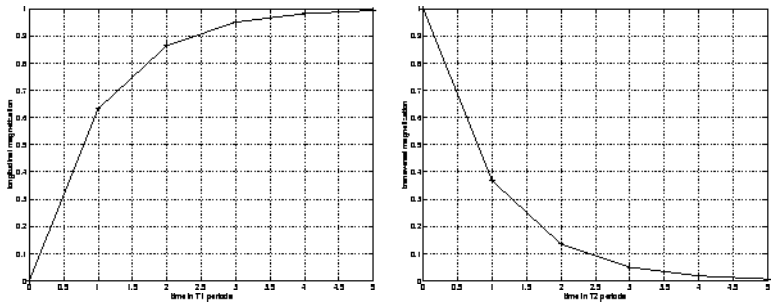

Figure 11.1: T1 and T2 relaxation

with the field, the net magnetization vector rotates from the longitudinal direction to a flip angle proportional to the duration of the RF pulse. After some time, the magnetization vector becomes perpendicular to the main field B_0. In this position, it can be detected by the MRI scanner. For angles different from 90^o, the perpendicular component of the magnetization vector can still be detected, but the signal is of course weaker.

MR Imaging is based on the observation of the relaxation that takes place after the RF pulse has stopped. The return of the excited nuclei from the high energy to the low energy state is associated with the loss of energy to the surrounding nuclei. Macroscopically, this spin-lattice or **T1** relaxation is characterized by the longitudinal return of the net magnetization to its maximum length in the direction of the magnetic field. This return is an exponential process of the form of $1 - e^{-t/T1}$ (Figure 11.1). The T1 relaxation time is the time constant of this exponential, i.e. the time needed for the magnetization to return to 63% of its original value.

Microscopically, **T2** relaxation, or spin-spin relaxation, occurs when the spins in the high and low energy state exchange energy but do not loose energy to the surrounding lattice. Macroscopically, this results in a loss of transverse magnetization. Once again, T2 relaxation is a exponential process, in the form of $e^{-t/T2}$ (Figure 11.1), and the T2 time is the time needed for the transverse magnetization to lose 63% of its value. In pure water, the T2 and T1 times are approximately identical. For biological material, the T2 time is considerably shorter than the T1 time.

By varying imaging parameters such as TR (repetition time) and TE

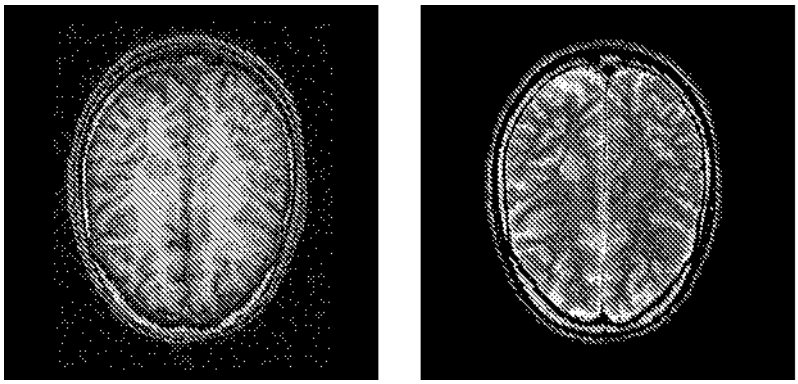

Figure 11.2: T1 and T2 MR images.

(echo time), it is possible to weight the IRM signal to produce T1-, T2- or PD-weighted (proton density) images. From a medical perspective, it means that MR Imaging can provide multiple channels to observe the same anatomy. For instance, Figure 11.2 shows T1- and T2-weighted images of the same brain. Different tissues appear differently in both images. White matter appears in a light grey in T1 and a dark grey in T2. Grey matter appears grey in both images. The Cerebro-Spinal Fluid (CSF) appears black in T1 and white in T2. The background of the image (air) appears black in both images.

11.2 T1,T2 classification

The classification of biological images into tissue components is an old problem, and many techniques have been used over the years. Classical methods range from simple thresholding to more sophisticated techniques based on local features such as the median, the variance, ... These techniques, however, do not always take advantage of the multi-dimensional nature of the MRI data, which can provide information about different tissue parameters, such as T1 and T2 relaxation time, proton density (PD), ...

On the other hand, multi-dimensional data classification had been used extensively in the area of remote sensing. **Vannier** [164], in 1985, was first to adapt those techniques to medical imaging and propose to use

the multi-spectral nature of MR images.

Supervised classification techniques all work similarly. A few samples of each tissue types are manually selected. Their values (T1,T2,...) in the pattern space are used to train a statistical classifier. The complete data set is then classified automatically.

Several such classifiers have been proposed. For instance, **Ozkan** [117] proposes to use Artificial Neural Networks (ANN), and shows that it performs better than a maximum likelihood classifier based on the assumption that the data can be modeled with multivariate normal distributions. ANN was successfully applied to the classification and detection of multiple sclerosis (MS) white matter lesions by **Zijdenbos** [187, 188], from T1-, T2- and PD weighted MRI volumes in combination with SPAMs (Statistical Probability of Anatomy Maps) for white matter, gray matter and CSF.

Kamber [86] investigates distance-to-mean-feature classifiers, Bayesian classifiers and decision-tree classifiers to detect MS lesions from multi-channel MRI and a probabilistic model of the brain.

In [19], **Clarke** shows that the k-NN rule has higher accuracy and stability for MRI data than the other common classifiers, but has a slower running time. Unfortunately, slow classification is often impractical because it limits the interactive selection of classification parameters during training. **Cline** [20] reports the effectiveness of interactive classification, where the training data is modified by a trained observer on the basis of the classification results.

Warfield [175] proposes the above mentioned k-DT algorithm for faster k-NN classification when the pattern space has a low dimensionality. In [176], he embeds k-NN classification into a template-moderated spatially varying statistical classification, as illustrated at Figure 11.3.

In conclusion, the k-NN classifier is one of the most efficient tools for tissue classification in multi-channel MRI. For an improved accuracy, it should be combined with a template description of the spatial tissue distribution in the anatomy.

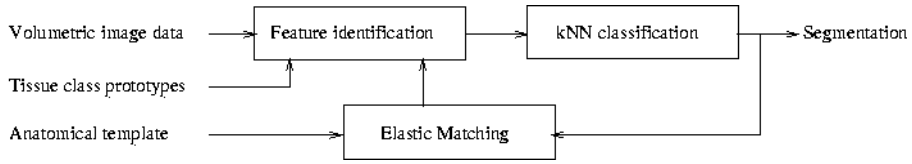

Figure 11.3: Schematic for Adaptive Template Moderated Spatially Varying Classification [176]. Initialization consists of image acquisition, tissue class prototypes selection and rigid registration of a normal anatomy to the image data. The anatomical template is converted into a set of features describing the anatomical localization with a distance transform. A segmentation based on these features and on the volumetric data is done with k-NN classification. This segmentation is then used to refine the alignment of the template anatomy with a fast elastic matching algorithm, and the process is iterated.

11.3 Results

In order to illustrate the efficiency of k-NN classification for the analysis of T1,T2 pairs of MRI images, we consider the images of Figure 11.2. The image on the left is a $256 \times 256 \times 120$ T1-weighted MRI. The image on the right is a $256 \times 256 \times 30$ T2-weighted MRI. The T1 image was registered into the coordinates of the T2 image, and resampled to an identical size. Slice number 22 out of 30 is displayed.

For each of 4 tissue types (background, CSF, grey matter, white matter), 300 training samples are manually selected, as illustrated at Figure 11.4. The T1 and T2 values are normalized as values between 0 and 511. The k-DT algorithm is then applied over the 512×512 pattern space. Every possible pattern is classified within the class with most occurrences in its k-nearest neighbors among the training samples. The resulting classification of the pattern space is shown at the bottom of Figure 11.4, with $k = 1$ and $k = 11$. The choice of this value is justified later.

Every voxel in the 2 MRI dataset is then labeled as belonging to the class found in this (T2,T1) values lookup table. The result of this process is found in Figure 11.5.

In Figure 11.4, we can see that the boundaries between the different

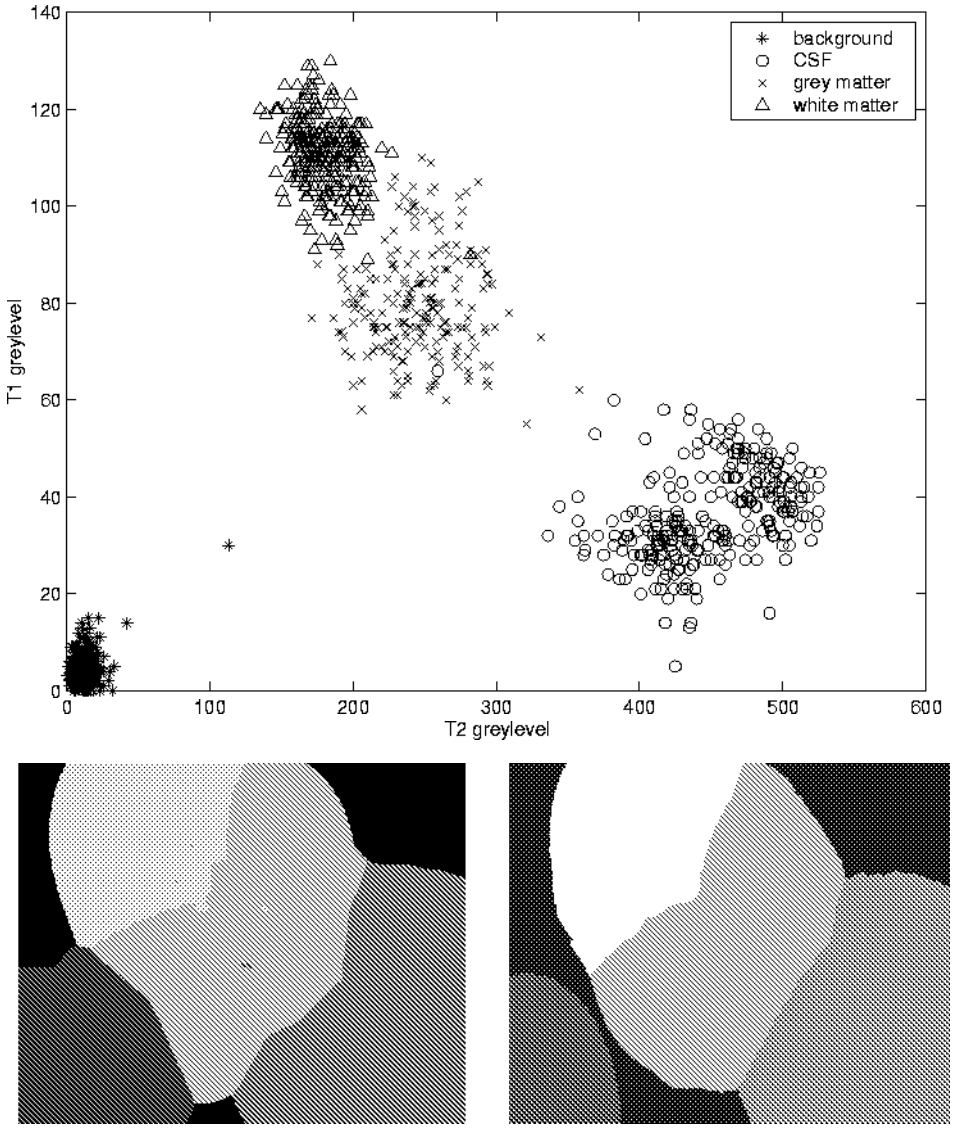

Figure 11.4: Classification in 4 classes : lookup tables. *Up:* Training samples from the 4 tissue classes. *Left:* Division of the (T2,T1) space with the NN rule ($k = 1$). *Right:* Division of the (T2,T1) plane with the k-NN rule ($k = 11$). k-DT was limited to $dist_E = 10000$, leaving some (T2,T1) couples unclassified.

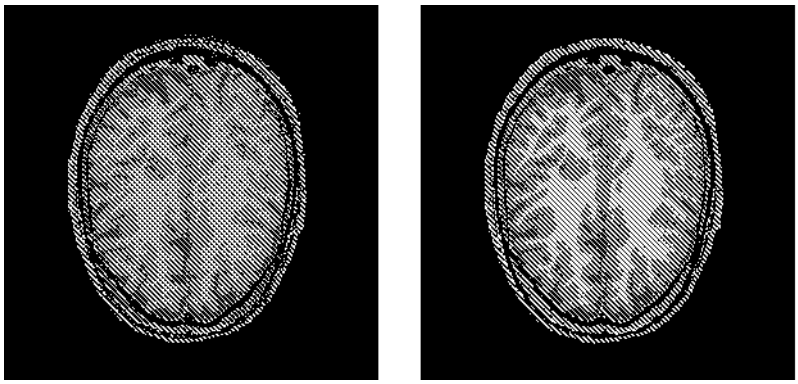

Figure 11.5: Classification in 4 tissue classes. *Left:* $k = 1$. *Right:* $k = 11$.

classes in the lookup table are much smoother with $k = 11$ than with $k = 1$. A priori, we expect the unknown probabilistic distribution of (T1,T2) values for each tissue type to be smooth functions. It is therefore reasonable to believe that they are better approximated with a higher k. In Figure 11.5, it results in a classification that is significantly less noisy with a higher k, and visually appears to be better with the use of our a priori anatomical knowledge of the brain.

In a second experiment, training samples from a fifth tissue class were added to the training data-set. This new tissue type corresponds to white matter lesions (WML) resulting from multiple sclerosis. One such lesion can be seen slightly up and left of the center of the image at figure 11.2. From a medical point of view, WML result from a loss of myelin around the axons. This turns the macroscopic appearance of WML into a tissue similar to grey matter, located where one would expect white matter.

Because the lesions in the image are small and only cover a few slices, only 60 training samples were manually selected this time. They appear as dots in Figure 11.6.

Obviously, the 5 tissue types classification appears to be a more complex task, as illustrated by the irregularity of the borders between classes in the 1-NN lookup table of figure 11.6. Indeed, lesions and grey matter appear to have very similar characteristics. Nevertheless, with $k = 11$, the borders between classes in the lookup table do appear rather regular.

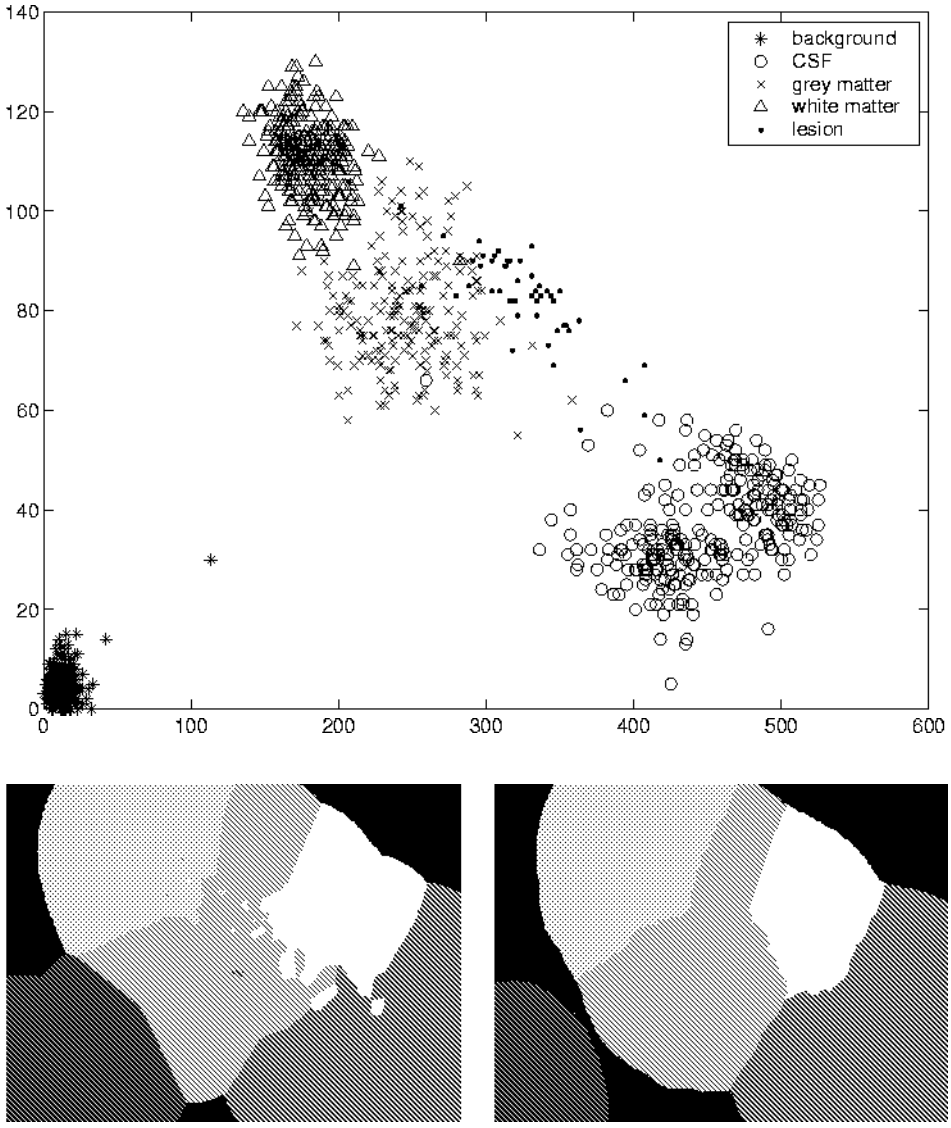

Figure 11.6: Classification in 5 classes : lookup tables. *Up:* Training samples from the 5 tissue classes. *Left:* Division of the (T2,T1) space with the NN rule ($k = 1$). *Right:* Division of the (T2,T1) plane with the k-NN rule ($k = 11$).

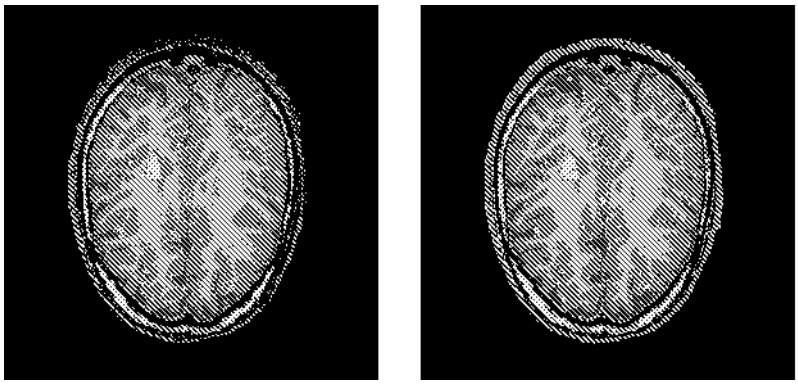

Figure 11.7: Classification in 5 tissue classes. *Left:* $k = 1$. *Right:* $k = 11$.

In Figure 11.7, the resulting classification of the voxels of the MR images is displayed. The lesion is correctly classified, but several other voxels are mis-classified as belonging to the lesion. This happens for voxels at the border between grey matter and CSF, where partial volume effects affect their values. This is of course a limit of the classification method, since the lesion class is indeed situated partly between the CSF and the grey matter classes in the lookup table. Improving this would require to take into account the spatial information and not only the gray level values.

Finally, we make a quantitative evaluation of the gain in classification accuracy made by using a large k. The training data set - 1200 and 1260 samples respectively - is uniformally divided into 20 subsets. Each subset is classified using the 19 others as training data. The error ratio is defined as the average percentage of mis-classified samples in the subsets.

The observed error ratio for $k = 1$ to 20 are displayed in figure 11.8. The jagged aspect of the curve results from the fact that most classification decisions are the result of a vote of the k nearest samples, choosing between two classes. In such a dual vote, an even number of voters make a decision of lesser quality than the same number minus 1.

The error ratio R depends on two main factors. First, the probabilistic distributions of observed (T1,T2) values for each tissue type can overlap each other, because of the noise in the images, because of inaccuracies in

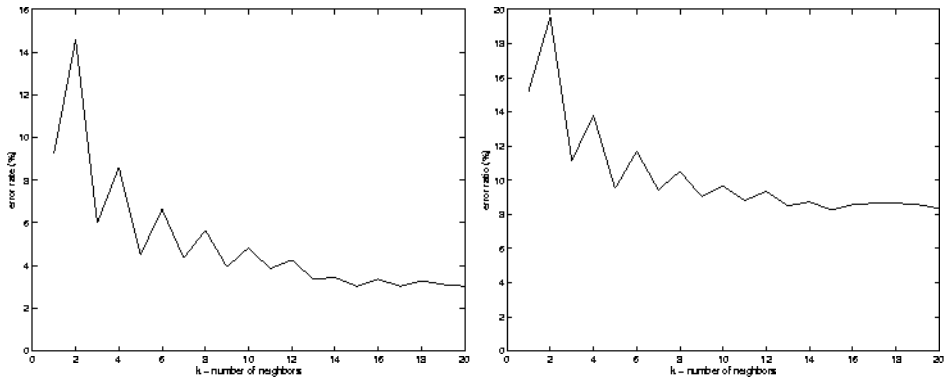

Figure 11.8: Error ratio for 4 tissues (left) and 5 tissues (right) experiments, as a function of k.

the manual selection of samples, because of the intrinsic tissue properties ... This corresponds to the Bayesian risk R^*, where the probabilistic distributions are known exactly and the best decision criterion is used. The second factor depends on the quality of the decision rule that is used, i.e. on how well we can use the available data to reach a good decision. Cover [23] showed that the error ratio associated to a k-NN decision rule was such that

$$R^* \leq R \leq (1 + \frac{1}{k})R^* \tag{11.1}$$

which corresponds to the shape of the curves in figure 11.8, notwithstanding the odd-even staggering. Practically, the value we selected for the above experiments ($k = 11$) is a suitable choice. A smaller k would not take full advantage of the available training samples. A larger k would require both additional computational time and memory to store the k-DT. Worse, (11.1) only stands if $k/N \to 0$, so that using a larger k requires to take more training samples, a time-consuming operation for the user.

Conclusion and perspectives

In this thesis, we have presented an extended review of distance transformation algorithms and their applications to medical image processing. While doing so, we have proposed a number of new algorithms. Some solve old problems faster, some address new problems by extending the distance transformation concept.

To a large extent, the problem of computing the exact Euclidean DT in two dimensions can be considered solved by the algorithms of chapters 3 and 5. They have a theoretically optimal $O(n^2)$ computational complexity on $n \times n$ images and a computational cost similar to that of approximate algorithms, which can reasonably be considered as the practical optimum.

In 3 dimensions, Saito's algorithm is the fastest for relatively small images, with a row or column size below 260. For larger datasets, the hybrid method of chapter 6 - combining slice by slice 2D EDT with the algorithm of chapter 5 and Saito's method along the inter-slice axis - performs best. Within the limits we explored, its computational cost is nearly independent of the image size.

Chapter 8 (geodesic DT) and chapter 10 (k-DT) explore extensions of the basic DT concept. Both are similar in the sense that they propose numerical rasterized solutions to problems that were traditionally considered within a continuous, analytical framework: the planning of a robot's path through a complex environment and k-NN classification of multi-spectral data.

Rasterizing other analytical problems represents a *first major perspective* for further extensions of this work. In particular, fast Delaunay triangulation and mesh generation techniques on complex data-sets could likely be addressed with modified DT algorithms.

This may appear to contradict the approach chosen by Breu [17], Guan [69] or Embrechts [44] (section 2.3.5) who find efficient solutions to the discrete EDT problem by moving back to the continuous description of the Voronoi diagram. On the contrary, it stresses the fact that such issues are best handled when one can consider both sides of the problem - continuous and digital. This both-sides approach is best illustrated by the algorithm of chapter 5, that detects corners of the digital Voronoi diagram, then analytically computes the extent of these corners in the continuous space.

Considering the geodesic and k-DT algorithms further, both appear to have an optimal computational complexity, propagating from each pixel only once (or k times). On the other hand, both algorithms are only approximate Euclidean DTs, similarly to Danielsson's [37] or the PSN algorithm. One may wish to explore ways to generate exact geodesic or k-DTs, but it should be noted that no application currently requires this.

On the other hand, a *second promising perspective* resides in the search for algorithms with a better-than-optimal complexity. This apparently impossible task could be achieved using a multi-resolution approach, in which the full resolution is only computed in the few locations where it really is needed. For instance, k-NN classification only requires an exact k-DT on the borders separating the various classes, a tiny fraction of the complete sample space.

Beside its algorithmic content, this thesis also presents a broad range of medical image processing applications. In all cases, the distance transformation is only a part of the whole image processing pipe-line. In chapter 4, myelin sheath thickness can only be evaluated after the image was thresholded and filtered with connected morphological operators. In chapter 7, surface-based registration methods require the preliminary segmentation of similar features in both images. In chapter 9, virtual endoscopy works in three steps. First the organ is segmented and a 3D model of it is computed, then the geodesic DT is used to plan the cam-

era's path, and finally the 3D model and the camera locations are used to generate the endoscopic visualization. In chapter 11, pixel classification in multi-channel MRI requires to manually pick samples of each tissue types. Alternatively, the k-NN classification can be embedded into a more global framework in which the anatomical knowledge is inserted using deformable templates.

From this brief review, it is clear that applications - particularly in the medical imaging field - is where *most perspectives* are open for further creative research. In our experience, creativity is often stimulated by practical examples. We hope that those presented in this thesis have contributed to stimulate the reader's mind.

Related publications

The work carried out in the framework of this thesis has been the subject of a number publications in international conferences and journals.

The use of multiple neighborhoods in the EDT by propagation algorithm was first presented at *ICIAP'97* [27]. In *ICIP'97* [30], it was extended to anisotropic grids and applied to the generation of morphological skeletons. The implementation of the mathematical morphology dilation with the PMN algorithm was presented in a local conference [33] and in *IPA'99* [34]. Most of **chapter 3**, including a detailed analysis of the computational complexity of the algorithms, will appear in *Computer Vision and Image Understanding* [29]. Noticeably, this paper does not distinguish the PMN and PMON algorithms.

The technical aspects of **chapter 4** - the automatic morphometry of nerve cross-sections - were presented at *SPIE Medical Imaging 99* [35]. This paper received a poster award. A paper with less emphasis on the method itself and more on its validation has been submitted to the *Journal of Neuroscience Methods* [130].

The algorithm of **chapter 5** was presented in a local conference [31] and during *ICASSP'99* [32]. An extended version has been submitted to *Pattern Recognition* [26].

The atlas-MRI registration in **chapter 7** was presented at *SPIE Medical Imaging 96* [36]. The registration was used as the initialization step of an active surface segmentation of the inner brain structures. The concept of this method was presented in *NMBIA'98* [28] and results in *SPIE Medical Imaging 99* [49].

Papers about the algorithms of **chapters 8** to **11** are under preparation.

List of Figures

Bibliography

[1] L. Abbott, R.M. Haralick, and X. Zhuang. Pipeline architectures for morphological image analysis. *Mach. Vision Appl.*, 1(1):23–40, 1988.

[2] A.A. Amini, T.E. Weymouth, and R.C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:855–867, 1990.

[3] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles, and H.C. Wolf. Parametric correspondance and chamfer matching: two techniques for image matching. In *Proc. 5th International Joint Conference on Artificial Intelligence*, pages 659–663, 1977.

[4] S. Belkasim, M. Shridhar, and M. Ahmadi. Pattern classification using an efficient knnr. *Pattern Recognition*, 25:1269–1274, 1992.

[5] S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Int. Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, 1979.

[6] H. Blum. A transformation for extracting new descriptors of shape. In W. Walthen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, 1967.

[7] H. Blum and R.N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978.

[8] C. Bohm, T. Greitz, D. Kingsley, B.M. Berggren, and L. Olsson. Adjustable computerized stereotaxic brain atlas for transmission and emission tomography. *Am J. Neuroradiology*, 4:731–733, 1983.

[9] Ph. Bolon, J.L. Vila, and T. Auzepy. Oprateur local de distance en maillage rectangulaire. In *2^{me} Colloque de Gomtrie Discrte: Fondements et Applications*, pages 45–56, 1992.

[10] G. Borgefors. Distance transformation in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27:321–145, 1984.

[11] G. Borgefors. An improved version of the chamfer matching algorithm. In *Proc. 7th Int. Conf. on Patter Recognition*, pages 1175,1177, Montreal, Canada, 1984.

[12] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986.

[13] G. Borgefors. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.

[14] G. Borgefors and I. Nystrom. Efficient shape representation by minimizing the set of centres of maximal discs/spheres. *Pattern Recognition Letters*, 18:465–472, 1997.

[15] J.D. Bourland and Q.R. Wu. Use of shape for automated, optimized 3d radiosurgical planning. In *Proc. SPIE Medical Imaging*, pages 553–558, Newport Beach, California, 1996.

[16] J.P. Brasil-Neto, L.M. McShane, P. Fuhr, and M. Hallett ad L.G. Cohen. Topographic mapping of the human motor cortex with magnetic stimulation: factors affecting accuracy and reproducibility. *Electroencephalogr Clin Neurophysiol*, 85:9–16, 1992.

[17] H. Breu, J. Gil, D. Kirkatrick, and M. Werman. Linear time euclidean distance transform algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):529–533, 1995.

[18] F. Buchnal and F. Behse. Sensory action potentials and biopsy of the sural nerve in neuropathy. Proc. of the Int. Symp. on Peripheral Neuropathies, Milan, June 1978.

[19] L.P. Clarke, R.P. Velthuizen, S. Phuphanich, J.D. Schellenberg, J.A. Arrington, and M. Silbiger. Mri: stability of three supervised segmentation techniques. *Magnetic Resonance Imaging*, 11:95–106, 1993.

[20] H.E. Cline, W.E. Lorensen, R. Kikinis, and F. Jolesz. Three-dimensional segmentation of mr images of the head using probability and connectivity. *J. Computer Assisted Tomography*, 14(6):1037–1045, 1990.

[21] A. Collignon, F. Maes, D. Delaere, D. Vandermeulen, P. Suetens, and G. Marchal. Automated multi-modality image registration based on information theory. In Y. Bizais, C. Barillot, and R. Di Paola, editors, *Information processing in medical imaging 1995*, pages 263–274, Dordrecht, The Netherlands: Kluwer, 1995.

[22] D. Coquin and Ph. Bolon. Discrete distance operator on rectangular grids. *Pattern Recognition Letters*, 16(9):911–923, 1995.

[23] T.M. Cover. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:50–55, 1968.

[24] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[25] G.B. Cragg and P.K. Thomas. Changes in conduction velocity and fibre size proximal to peripheral nerve lesions. *J. Physiol.*, 157:315, 1961.

[26] O. Cuisenaire. On the errors made by approximate euclidean distance transformation algorithms. submitted to *Pattern Recognition*.

[27] O. Cuisenaire. Region growing euclidean distance transforms. In *9th International Conference on Image Analysis and Processing (ICIAP'97)*, volume 1, pages 263–270, Florence, Italy, September 1997.

[28] O. Cuisenaire, M. Ferrant, J-Ph. Thiran, and B. Macq. Model-based segmentation and recognition of anatomical brain structures in 3d mr images. In *Proc. Noblesse Model-Based Image Analysis Workshop (NMBIA'98)*, pages 9–14, Glasgow, UK, July 1998.

[29] O. Cuisenaire and B. Macq. Fast euclidean distance transformation by propagation using multiple neighborhoods. to appear in *Computer Vision and Image Understanding*.

[30] O. Cuisenaire and B. Macq. Applications of the region growing euclidean distance transform: anisotropy and skeletons. In *International Conference on Image Processing (ICIP'97)*, volume 1, pages 200–203, Santa Barbara (CA), October 1997.

[31] O. Cuisenaire and B. Macq. Fast and exact signed euclidean distance transformation with linear complexity. In *Proc. Int. Symposium on Pattern Recognition 'In Memorian Pierre Devijver'*, pages 140–143, Brussels, Belgium, February 1999.

[32] O. Cuisenaire and B. Macq. Fast and exact signed euclidean distance transformation with linear complexity. In *Proc. IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP99)*, volume 6, pages 3293–3296, Phoenix (AZ), March 1999.

[33] O. Cuisenaire and B. Macq. Fast euclidean morphological operators using local distance transformation by propagation. In *Proc. Int. Symposium on Pattern Recognition 'In Memorian Pierre Devijver'*, pages 144–148, Brussels, Belgium, February 1999.

[34] O. Cuisenaire and B. Macq. Fast euclidean morphological operators using local distance transformation by propagation, and applications. In *Proc. 7th Int. Conf. on Image Processing and its Applications (IPA99)*, Manchester, UK, July 1999.

[35] O. Cuisenaire, E. Romero, C. Veraart, and B. Macq. Automatic segmentaion and measurement of axones in microscopic images. In *Proc. SPIE Medical Imaging 1999*, volume 3661, pages 920–929, San Diego (CA), February 1999. MI Poster Award.

[36] O. Cuisenaire, J.Ph. Thiran, Benoît Macq, Christian Michel, Anne De Volder, and Ferran Marquès. Automatic registration of 3d mr images with a computerized brain atlas. In *Proc. SPIE Medical Imaging 1996*, volume 2710, pages 438–448, Newport Beach (CA), February 1996.

[37] P.E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[38] B. Davey, R. Comeau, P. Munger, L. Pisani, D. Lacerte, A. Olivier, and T. Peters. Multimodality interactive stereoscopic image-

guided neurosurgery. In *Visualization in Biomedical Computing 94*, volume SPIE 2359, pages 526–536, 1994.

[39] R. Dial. Algorithm 360: Shortest path with topological ordering. *Communications of the ACM*, 1:632–633, 1969.

[40] P.J. Dyck, J. Karnes, P. O'Brien, H. Nukada, A. Lais, and P. Low. Spatial pattern of nerve fiber abnormality indicative of pathologic mechanism. *Am. J. of Pathology*, 117:225–238, 1984.

[41] H. Eggers. Parallel euclidean distance transformations in z^n. *Pattern Recognition Letters*, 17:751–757, 1996.

[42] H. Eggers. Two fast euclidean distance transformations in z^2 based on sufficient propagation. *Computer Vision and Image Understanding*, 69(1):106–116, 1998.

[43] A. Elmoataz, S.Schüpp, R. Cloaurd, P. Herlin, and D. Bloyet. Using active contours and mathematical morphology tools for quantification of immunohistochemichal images. *Signal Processing*, 71:215–226, 1998.

[44] H. Embrechts and D. Roose. A parallel euclidean distance transformation algorithm. *Computer Vision and Image Understanding*, 63:15–26, 1996.

[45] A. English, M.B. Luskin, R. McKeon, K. Petersen, K. Vydareny, J.R. Wilson, and S.L. Wolf. Organization of the nervous system: An introduction for students in the human anatomy course. http://www.cc.emory.edu/ANATOMY/AnatomyManual/nervous_system.html, Emory University, Atlanta, GA.

[46] Kikinis et al. A digital brain atlas for surgical planning, model driven segmentation and teaching. *IEEE Transactions on Visualization and Computer Graphics*, 2, 1996.

[47] Wells et al. Multi-modal volume registration by maximization of mutual information. *Medical Image Analysis*, 1:35–51, 1996.

[48] G.J. Ettinger, W.E.L. Grimson, T. Lozano-Perez, W.M. Wells, S.J. White, and R. Kikinis. Automatic registration for multiple sclerosis change detection. In *IEEE workshop on biomedical image analysis*, pages 297–306, Seattle, WA, 1994.

[49] M. Ferrant, O. Cuisenaire, and B. Macq. Multi-object segmentation of brain structures using a computerized brain atlas. In *Proc. SPIE Medical Imaging 1999*, volume 3661, pages 986–995, San Diego (CA), February 1999.

[50] M.R. Fiola. Peripheral nerve morphometry for daily practice. *Anal.Quant.Cytol.Histol.*, 7:299–304, 1984.

[51] C.M. Fisher and R.D. Adams. Diphtheritic polyneuritis: a pathological study. *J.Neuropath.Exp.Neurol.*, 15:243, 1956.

[52] J.M. Fitzpatrick, J.B. West, and C.R. Maurer. Predicting error in rigid-body point-based registration. *IEEE Transactions on Medical Imaging*, 17:694–702, 1998.

[53] E. Fix and J.L. Hodges. Discriminatory analysis, non-parametric discrimination. Technical report, USAF School of Aviation Medicine, Randolf Field, Tex. Project 21-49-004, Rept. 4, Contract AF41(128)-31, February 1951.

[54] E. Fix and J.L. Hodges. Discriminatory analysis, small-sample performance. Technical report, USAF School of Aviation Medicine, Randolf Field, Tex. Project 21-49-004, Rept. 11, August 1952.

[55] Y.L. Fok, J.C.K. Chan, and R.T. Chin. Automated analysis of nerve-cell images using active contour models. *IEEE Trans. on Medical Imaging*, 15:353–368, 1996.

[56] S. Forchhammer. Euclidean distances from chamfer distances for limited distances. In *6th Scandinavian Conf. on Image Analysis*, pages 393–400, Oulu, Finland, 1989.

[57] R. Frankenthaler, V.M. Moharir, R. Kikinis, P. van Kipshagen, F.A. Jolesz, C. Umans, and M.P. Fried. Virtual otoscopy. *Otolaryngologic Clinics of North America*, 31:383–391, 1998.

[58] M.P. Fried, V.M. Moharir, H. Shinmoto, A.M. Alyassin, W.E. Lorensen, L. Hsu, F.A. Jolesz, and R. Kikinis. Virtual laryngoscopy. *Annals of Otologym Rhinology and Laryngology*, 108:221–226, 1998.

[59] J.H. Friedman, F. Baskett, and L.J. Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24:1000–1006, 1975.

[60] K. Fukunaga and P. Narendra. A branch and bound algorithm for k-nearest neighbors. *IEEE Transactions on Computers*, 24:750–753, 1975.

[61] K.R. Gabriel and R.R. Sokal. A new statistical approach to geographic variations analysis. *Systematic Zoology*, 18:259–278, 1969.

[62] C. Garbay. Image structure representation and processing: a discussion of some segmentation methods in cytology. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8:140–146, 1986.

[63] G.W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18:431–433, 1972.

[64] Y. Ge and J.M. Fitzpatrick. On the generation of skeletons from discrete euclidean distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:1055–1066, 1996.

[65] B. Geiger and R. Kikinis. Simulation of endoscopy. In N. Ayache, editor, *Proceedings First International Conference on Vision, Virtual Reality, and Robotics in Medicine*, volume 905, pages 542–548, Lecture Notes in Computer Science. Verlag Springer, 1995.

[66] M.S. George, E.M. Wassermann, W.A. Williams, A. Callahan, T.A. Ketter, P. Basser, M. Hallett, and R.M. Post. Daily repetitive transcranial magnetic stimulation (rtms) improves mood in depression. *Neuroreport*, 6:1853–1856, 1995.

[67] S. Gilani, A.M. Norbash, H. Ringl, G.D. Rubin, S. Napel, and D.J. Terris. Virtual endoscopy of the paranasal sinus using perspective volume rendered helical sinus computed tomography. *Laryngoscope*, 107:25–29, 1997.

[68] W.E.L. Grimson, G.J. Ettinger, S.J. White, T. Lozano-Perez, W.M. Wells, and R. Kikinis. An automatic registration method for frameless stereotaxy, image-guided surgery and enhanced reality visualization. *IEEE Transactions on Medical Imaging*, 15:129–140, 1996.

[69] W. Guan and S. Ma. A list-processing approach to compute voronoi diagrams and the euclidean distance transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):757–761, 1998.

[70] E. Gutmann and F.K. Sanders. Recovery of fiber numbers and diameters in regeneration of peripheral nerves. *J. Physiol. (Lond.)*, 43:101–489, 1942.

[71] P.E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[72] H.J.A.M. Heijmans. Connected morphological operators for binary images. *Computer Vision and Image Understanding*, 73 (1):99–120, 1999.

[73] P.F. Hemler, T.S. Sumanaweera, P.A. van den Elsen, S. Napel, and J.R. Adler. A versatile system for multimodality image fusion. *J. Image Guided Surg.*, 1:35–45, 1995.

[74] C.J. Henri, A. Cukiert, D.L. Collins, A. Olivier, and T.M. Peters. Towards frameless stereotaxy: anatomical-vascular correlation and registration. In *Visualization in Biomedical Computing 1992*, volume SPIE 1808, pages 214–224, 1992.

[75] J.L. Herring, B.M. Dawant, C.R. Maurer, D.M. Muratore, R.L. Galloway, and J.M. Fitzpatrick. Surface-based registration of ct images to physical space for image-guided surgery of the spine: a sensitivity study. *IEEE Transactions on Medical Imaging*, 17:743–752, 1998.

[76] D.L.G. Hill and D.J. Hawkes. Medical image registration using knowledge of adjacency of anatomical structures. *Image and vision computing*, 12:173–178, 1994.

[77] C.T. Huang and O.R. Mitchell. A euclidean distance transform using greyscale morphology decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):443–448, 1994.

[78] J.M. Jacobs and S. Love. Qualitative and quantitative morphology of human sural nerve at different ages. *Brain*, 108:897–924, 1985.

[79] A.K. Jain, S.P. Smith, and E. Backer. Segmentation of muscle cell pictures: a preliminary study. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2:232–242, 1980.

[80] H. Jiang, K.S. Holton, and R.A. Robb. Image registration of multi-modality 3-d medical images by chamfer matching. In *Proc.*

Biomedical image processing and three-dimensional microscopy, volume SPIE 1660, pages 356–366, 1992.

[81] H. Jiang, R.A. Robb, and K.S. Holton. A new approach to 3-d registration of multimodality medical images by surface matching. In *Proc. Visualization in Biomedical Computing 1992*, volume 1808, pages 196–213, 1992.

[82] Q. Jiang and W. Zhang. An improved method for finding nearest neighbors. *Pattern Recognition Letters*, 14:531–535, 1993.

[83] Q. Jiang and W. Zhang. An improved method for finding nearest neighbors. *Pattern Recognition Letters*, 14:531–535, 1993.

[84] F.A. Jolesz, W.E. Lorensen, and R. Kikinis. Virtual endoscopy: three-dimensional rendering of cross-sectional images for endoluminal visualization. *Radiology*, 193:469, 1994.

[85] F.A. Jolesz, W.E. Loresen, H. Shinmoto, H. Atsumi, S. Nakajima, P. Kavanaugh, P. Saiviroonporn, S.E. Seltzer, S.G. Silverman, M. Phillips, and R. Kikinis. Interactive virtual endoscopy. *Amer. Journ. Radiology*, 169:1229–1237, 1997.

[86] M. Kamber, R. Singhal, D.L. Collins, G.S. Francis, and A.C. Evans. Model-based 3-d segmentation of multiple sclerosis lesions in magnetic resonance brain images. *IEEE Transactions on Medical Imaging*, 14(3):442–453, 1995.

[87] B. Kamgar-Parsi and L.N. Kanal. An improved branch nd bound algorithm fro computing k-nearest neighbours. *Pattern Recognition Letters*, 3:7–12, 1985.

[88] I. Kapouleas, A. Alavi, W.M. Alves, R.E. Gur, and D.W. Weiss. Registration of three-dimensiona mr and pet images of the human brain without markers. *Radiology*, 181:731–739, 1991.

[89] M. Kass, A. Witkin, and D. Terzopoulos. Snakes, active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[90] R. Kikinis, C. Umans, S. Jones, W.E. Lorensen, and F.A. Jolesz. Virtual otoscopy. In *Image Conference*, Scottsdale, AZ, June 1996.

[91] E.L. Lawler and D.E. Wood. Branch-and-bound methods. a survey. *Oper. Res.*, 149(4), 1966.

[92] D. Lemoine, D. Leigeard, E. Lussot, and C. Barillot. Multimodal registration system for the fusion of mri, ct, meg and 3d or stereotactic angiography data. In *Proc. SPIE Medical Imaging 94: Image Capture, Formatting and Display*, volume SPIE 2164, pages 46–56, 1994.

[93] J. Lengyel, J. Reichert, B.R. Donald, and D.P. Greenberg. Real-time robot motion planning using rasterizing. *Computer Graphics*, 24:327–335, 1990.

[94] G. Levi and U. Montanari. A grey-weighted skeleton. *Inform. Control*, 17:62–91, 1970.

[95] F. Leymarie and M.L. Levine. Fast raster-scan distance propagation on the discrete rectangular lattice. *CVGIP: Image Understanding*, 55:85–94, 1992.

[96] W.E. Lorensen and H.E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, pages 163–169, 1987.

[97] W.E. Lorensen, F.A. Jolesz, and R. Kikinis. The exploration of cross-sectional data with a virtual endoscope. In *Interactive technology and the new pardigm for health-care: medicine meets virtual reality III*, pages 221–230, Amsterdam, Holland: IOS Press, 1995.

[98] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens. Multimodality image registration by maximization of mutual information. *IEEE Transactions on Medical Imaging*, 16:187–198, 1997.

[99] J.B.A. Maintz, P.A. van den Elsen, and M.A. Viergever. Comparison of feature-based matching of ct and mr brain images. In N. Ayache, editor, *Proc. CVRMed '95*, pages 219–228, Berlin: Springer-Verlag, 1995.

[100] V.R. Mandava, J.M. Fitzpatrick, C.R. Maurer, R.K. Maciunas, and G.S. Allen. Registration of mutimodal volume head images vie attached markers. In *SPIE Medical Imaging 92: Image Processing*, volume SPIE 1652, pages 271–282, 1992.

[101] J.F. Mangin, V. Froin, I. Bloch, B. Bendriem, and J. Lopez-Krahe. Fast non-supervised 3d registration of pet and mr images from the brain. *Journal of Cerebral Blood Flow and Metabolism*, 14:749–792, 1994.

[102] C.R. Maurer, J.M. Fitzpatrick, R.L. Galloway, M.Y. Wang, R.J. Maciunas, and G.S. Allen. The accuracy of image-guided neurosurgery using implantable fiducial markers. In *Proc. Computed Asisted Radiology 1995*, pages 1197–1202, Berlin: Springer-Verlag, 1995.

[103] C.R. Maurer, J.M. Fitzpatrick, M.Y. Wang, R.L. Galloway, R.J. Maciunas, and G.S. Allen. Registration of head volume images using implantable fiducial markers. *IEEE Transactions on Medical Imaging*, 16:447–462, 1997.

[104] C.R. Maurer, R.J. Maciunas, and J.M. Fitzpatrick. Registration of head ct images to physical space using a weighted combination of points and surfaces. *IEEE Transactions on Medical Imaging*, 17:753–761, 1998.

[105] T.M. Mayhew, A.K. Sharma, and K.S. Bedi. Economical sampling designs for sizing fibres in peripheral nerves. *Acta Anat.*, 111:97, 1981.

[106] F. Meyer. Un algorithme optimal de ligne de partage des eaux. In *Actes du 8ème congrès AFCET*, pages 847–859, Lyon-Villeurbane, France, 1991.

[107] F. Meyer. Color image segmentation. In *Proc. 4th Internat. Conf. Image Processing Applications*, pages 523–548, Maastricht, Netherlands, 1992.

[108] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38:113–125, 1994.

[109] U. Montanari. A method for obtaining skeletons using a quasi-euclidean distance. *J. Assoc. Comp. Mach.*, 15(4):600–624, 1968.

[110] E.F. Moore. The shortest path through a maze. In *Proc. International Symposium on the Theory of Switching, part II.*, pages 285–292, Harvard University Press, Cambridge, MA, 1957.

[111] K. Mori, J. Hasegawa, and J. Toriwaki. Automated extraction and visualization of bronchus from 3d ct images of lung. In N. Ayache, editor, *Proceedings First International Conference on Vision, Virtual Reality, and Robotics in Medicine*, volume 905, pages 542–548, Lecture Notes in Computer Science. Verlag Springer, 1995.

[112] J.C. Mullikin. The vector distance transform in two and three dimensions. *CVGIP: Graphical Models and Image Processing*, 54:526–535, 1992.

[113] T. Nakagohri, F.A. Jolesz, S. Okuda, T. Asano, T. Kenmochi, O. Kainuma, Y. Tokoro, H. Aoyama, W.E. Lorensen, and R. Kikinis. Virtual pancreatoscopy of mucin-producing pancreas tumors. *Comp. Aid. Surgery*, 3:264–268, 1998.

[114] C.W. Niblack, P.B. Gibbons, and D.W. Capson. Generating skeletons and centerlines from the distance transform. *CVGIP: Graphical Models and Image Processing*, 54:420–437, 1992.

[115] H. Niemann and R. Goppert. An efficient branch-and-bound nearest neighbor classifier. *Pattern Recognition Letters*, 7:67–72, 1988.

[116] A. Ohnishi, K. Schilling, W.S. Brimijoin, E.H. Lambert, V.F. Fairbanks, and P.J. Dyck. Lead neuropathy: morphometry, nerve conduction and choline acetyltransferase transport. new finding of endoneurial edema associated with segmental demyelination. *J. Neuropathol. Exp. Neurol.*, 36:499–518, 1977.

[117] M. Ozkan, B. Dawant, and R. Maciunas. Neural-network-based segmentation of mutli-modal medical images: a comparative and prospective study. *IEEE Transactions on Medical Imaging*, 12(3):534–544, 1993.

[118] D.W. Paglieroni. Distance transforms: properties and machine vision applications. *CVGIP: Graphical Models and Image Processing*, 54:56–74, 1992.

[119] C.A. Pelizzari, G.T.Y. Chen, D.R. Spelbring, R.R. Weichselbaum, and C.T. Chen. Accurate three-dimensional registration of cp,pet and/or mr images of the brain. *J. Comput. Assist. Tomogr*, 13:20–26, 1989.

[120] J. Pick. Myelinated fibers in gray rami commnunicants. *Anat. Rec.*, 126:395–414, 1956.

[121] U. Pietrzyk, K. Herholtz, and W. Heiss. Three-dimensional alignement of functional and morphological tomograms. *J. Computer Assisted Tomography*, 14:51–59, 1990.

[122] J. Piper and E. Granum. Computing distance transformations in convex and non-convex domains. *Pattern Recognition*, 20(6):599–615, 1987.

[123] G.F. Potts, L.D. Gugino, M.E. Leventon, W.E.L. Grimson, R. Kikinis, W. Cote, E. Alexander, J. E. Anderson, G.J. Ettinger, L.S. Aglio, and M.E. Shenton. Visual hemifield mapping using transcranial magnetic stimulation coregistered with cortical surfaces derived from magnetic resonance images. *J. Clin. Neurophysiology*, 15:344–350, 1998.

[124] F. Preteux. On a distance function approach for grey-level mathematical morphology. In E.R. Dougherty, editor, *Mathematical Morphology in Image Processing*, pages 323–351, Marcel Dekker, New York, 1993.

[125] F. Preteux and N. Merlet. New concepts in mathematical morphology: the topographical distance functions. In P.D. Gardner and E.R. Dougherty, editors, *Proc. SPIE 1568*, pages 66–77, 1991.

[126] I. Ragnelmam. Fast erosion and dilation by contour processing and thresholding of distance maps. *Pattern Recognition Letters*, 13:161–166, 1992.

[127] I. Ragnelmam. Neighborhoods for distance transformations using ordered propagation. *CVGIP, Image Understanding*, 56(3):399–409, 1992.

[128] I. Ragnelmam. The euclidean distance transformation in arbitrary dimensions. *Pattern Recognition Letters*, 14:883–888, 1993.

[129] J.D. Robertson. Structural alterations in nerve fibres produced by hypotonic and hypertonic solutions. *Journal of Biophysics, Biochemistry and Cytology*, 4:349–364, 1958.

[130] E. Romero, O. Cuisenaire, J.F. Denef, J. Delbeke, B. Macq, and C. Veraart. Automatic morphometry of nerve cross sections. submitted to the *Journal of Neuroscience Methods*.

[131] A. Rosenfeld and A.C. Kak. *Digital Picture Processing*. Academic Press, New York, 1976.

[132] A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *J. Assoc. Comp. Mach.*, 13:471–494, 1966.

[133] A. Rosenfeld and J.L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1(1):33–61, 1968.

[134] G.D. Rubin, C.F. Beaulieu, and V. Argiro V. Perspective volume rendering of ct and mr images: applications for endoscopic imaging. *Radiology*, 199:321–330, 1996.

[135] P. Rudge, J. Ochoa, and R.W. Gilliatt. Acute peripheral nerve compression in the baboon. *J.Neurol.Sci.*, 23:403, 1974.

[136] J.G. Rukavina, W.D. Block, C.E. Jackson, H.F. Falls, J.H. Carey, and Curtin A.C. Primary systemic amyloidosis: a review and an experimental, genetic and clinical study of 29 cases with particular emphasis on the familial form. *Medicine, Baltimore*, 35:29, 1956.

[137] W.A.H. Rushton. A theory of the effects of fiber size in medullated nerve. *J. Physiology*, 115:101–122, 1951.

[138] D. Rutovitz. Data structures for operations on digital images. In G.C. Cheng, R.S. Ledley, D.K. Pollok, and A. Rosenfeld, editors, *Pictorial Pattern Recognition*, pages 105–133, Thomson Book, WA, 1968.

[139] D. Rutovitz. Expanding picture components to natural density boundaries by propagation methods, the notions of fall-set and fall-distance. In *Proc. 4th Int. Joint Conf. Pattern Recognition*, pages 657–664, Kyoto, Japan, 1978.

[140] T. Saito and J.I. Toriwaki. New algorithms for euclidean distance transformations of an n-dimensional digitised picture with applications. *Pattern Recognition*, 27(11):1551–1565, 1994.

[141] R.H. Schultz and U.L.I. Karlsson. Fixation of the central nervous system for electron mycroscopy by aldehyde perfusion. ii. effect of osmolarity. ph of perfusate, and fixative concentration. *J. Ultrastruct. Res.*, 12:187–206, 1965.

[142] R.J. Seitz, C. Bohm, T. Greitz, P.E. Roland, L. Eriksson, G. Blomqvist, G. Rosenqvist, and B. Nordell. Accuracy and precision of the computerized brain atlas program for localization and quantification in positron emission tomography. *Journal of Cerebral Blood Flow and Metabolism*, 10:443–457, 1990.

[143] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, New York, 1982.

[144] J. Serra. *Image Analysis and Mathematical Morphology, Vol. 2: Theoretical Advances*. Academic Press, New York, 1988.

[145] M.Y. Sharaiha and N. Christofides. A graph-theoric approach to distance transformations. *Pattern Recognition Letters*, 15:1035–1041, October 1994.

[146] F.Y.-C. Shih and J.J. Liu. Size-invariant four-scan euclidean distance transformation. *Pattern Recognition*, 31(11):1761–1998, 1998.

[147] F.Y.-C. Shih and O.R. Mitchell. A mathematical morphology approach to euclidean distance transformation. *IEEE Transactions on Image Processing*, 1(2):197–204, 1992.

[148] J.A Simpsom. Biology and disease of periheral nerves. *Br.Med.J.*, 2:709, 1964.

[149] V. Starovoitov. A clustering technique based on the distance transform. *Pattern Recognition Letters*, 17:231–239, 1996.

[150] C. Studholme, D.L.G. Hill, and D.J. Hawkes. Automated registration of trucated mr and ct datasets of the head. In *Proc. B. Machine Vision Association*, pages 27–36, 1995.

[151] J. Talairach and P. Tournoux. *Co-planar stereotaxic atlas of the human brain. 3-dimensional proportional system: an approach to cerebral imaging*. Thieme Medical Publicher, Inc., Stuttgart, New York, 1988.

[152] J.-P. Thiran, V. Warscotte, and B. Macq. A queue-based region growing algorithm for accurate segmentation of multi-dimensional digital images. *Signal Processing*, 60:1–10, 1997.

[153] J.Ph. Thiran and B. Macq. Morphological feature extraction for the classification of digital images of cancerous tissues. *IEEE Trans. on Biomedical Engineering*, 43:1011–1019, 1996.

[154] P.K. Thomas and R.G. Lascelles. Schwann-cell abnormalities in diabetec neuropathy. *Lancet*, i:1355, 1965.

[155] L. Thurfjell, C. Bohm, T. Graitz, and L. Eriksson. Transformations and algorithms in a computerized brain atlas. *IEEE Trans. on Nuclear Sciences*, 40:1187–1191, 1993.

[156] P.J. Toivanen. New geodesic distance transforms for gray scale images. *Pattern Recognition Letters*, 17:437–450, 1996.

[157] I. Tomek. Two modifications of cnn. *IEEE Trans. Syst. Man Cybernet.*, 6:769–772, 1976.

[158] S. Torch, P. Stoebner, Usson Y., Drouet D'Aubigni, and R. Saxod. There is no simple adequate sampling scheme for estimating the myelinated fibre size distribution in human peripheral nerve: a statistical ultrastructural study. *J. Neuroscience Methods*, 27:149–164, 1989.

[159] G.T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12:1324–1347, 1980.

[160] O.D. Trier and T. Taxt. Evaluation of binarization methods for document images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17:312–315, 1995.

[161] P.A. Van den Elsen, J.B.A. Maintz, E.J.D. Pol, and M.A. Viergever. Automatic registration of ct and mr brain images using correlation of geometrical features. *IEEE Transactions on Medical Imaging*, 14:384–395, 1995.

[162] P.A. van den Elsen, E.J.D. Pol, T.S. Sumanaweera, P.F. Hemler, S. Napel, and J.R. Adler. Grey value correlation techniques used for automatic matching of ct and mr brain and spine images. In *Proc. Visualization in Biomedical Computing 1994*, volume 2359, pages 227–237, 1994.

[163] P.A. van den Elsen, E.J.D. Pol, and M.A. Viergever. Medical image matching - a review with classification. *IEEE Eng. Med. Biol.*, 12:26–39, 1993.

[164] M.W. Vannier, R.L. Butterfield, D.L. Rickman, D.M. Jordan, W.A. Murphy, R.G. Lewitt, and G. Mohktar. Multi-spectral analysis magnetic resonnance images. *Radiology*, 154(1):221–224, 1985.

[165] P.W. Verbeek, L. Dorst, B.J.H. Verwer, and F.C.A. Groen. Collision avoidance and path finding through constrained distance transformation in robot state space. In *Proc. Intelligent Autonomous Systems*, pages 634–641, Amsterdam, Netherlands, 1986.

[166] B.H. Verwer. Local distances for distance transformations in two and three dimensions. *Pattern Recognition Letters*, 12:671–682, 1991.

[167] B.H. Verwer, P.W. Verbeek, and S.T. Dekker. An efficient uniform cost algorithm applied to distance transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(4):425–429, 1989.

[168] L. Vincent. Exact euclidean distance function by chain propagation. In *Computer Vision and Pattern Recognition Conference*, pages 520–525, Hawaii, June 1991.

[169] L. Vincent. Morphological transformations of binary images with arbitrary structuring elements. *Signal Processing*, 22:3–23, 1991.

[170] L. Vincent and P. Boille. Watersheds in digital spaces: an efficient algorithm based on immersion simulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:583–597, 1991.

[171] D.J. Vining, A.R. Padhani, and S. Wood. Virtual bronchoscopy: a new perspective for viewing the tracheobronchial tree. *Radiology*, 189:438, 1993.

[172] D.J. Vining, R.Y. Shifrin, and E.K Grishaw. Virtual colonoscopy. *Radiology*, 193:446, 1994.

[173] L.J. Van Vliet and B.J.H. Verwer. A countour-processing method for fast binary operations. *Pattern Recognition Letters*, 7:27–36, 1988.

[174] A.M. Vossepoel, A.W.M. Smeulders, and K. Van der Broek. Dioda: delineation and feature extraction of miscroscopical objects. *Comp. Prog. Biomedicine*, 10:231–244, 1979.

[175] S. Warfield. Fast k-NN classification for multichannel image data. *Pattern Recognition Letters*, 17:713–721, 1996.

[176] S. Warfield, M. Kraus, F. Jolez, and R. Kikinis. Adaptive template moderated spatially varying statistical classification. In *Proc. of Medical Image Computing and Computer-Assisted Intervention - MICCAI'98*, pages 431–438, Cambridge, MA, USA, 1998.

[177] V. Warscotte, J.-P. Thiran, B. Macq, C. Michel, and P. Fourez. Accurate segmentation of 3d magnetic resonance images of the head using a directional watershed transform. In *IEEE Engineers in Medicine and Biology Conference*, Montreal, September 1995.

[178] R. Wasserman, J. Rajapakse, and R. Acharya. Multimodality medical imaging for radiotherapy treatment planning. In *IEEE workshop on Biomedical image analysis*, pages 235–244, 1994.

[179] J. West, J.M. Fitzpatrick, M.Y. Wang, B.M. Dawant, C.R. Maurer, R.M. Kessler, and R.J. Maciunas. Retrospective intermodality registration techniques for images of the head: surface-based versus volume-based. In J. Troccaz, E. Grimson, and R. Mosges, editors, *Proc. CVRMed-MRCAS '97*, pages 151–160, Berlin: Springer-Verlag, 1997.

[180] J. West, J.M. Fitzpatrick, M.Y. Wang, B.M. Dawant, C.R. Maurer, R.M. Kessler, and R.J. Maciunas. Retrospective intermodality registration techniques for images of the head: surface-based versus volume-based. *IEEE Transactions on Medical Imaging*, 18:144–150, February 1999.

[181] J. West, J.M. Fitzpatrick, M.Y. Wang, B.M. Dawant, C.R. Maurer, R.M. Kessler, R.J. Maciunas, C. Barillot, D. Lemoine, A. Collignon, F. Maes, P. Suetens, D. Vandermeulen, P.A. van den Elsen, S. Napel, T.S. Sumanaweera, B. Harkness, D.L.G. Hill, C. Studholme, G. Malandain, X. Pennec, M.E. Noz, C.Q. Maguire, M. Pollack, C.A. Pellizari, A. Robb, D. Hanson, and R.P. Woods. Comparison and evaluation of retrospective intermodality image registration methods. In *Proc. SPIE Medical Imaging 96*, volume SPIE 2710, pages 332–346, 1996.

[182] J. West, J.M. Fitzpatrick, M.Y. Wang, C.R. Maurer, R.M. Kessler, R.J. Maciunas, C. Barillot, D. Lemoine, A. Collignon, F. Maes, P. Suetens, D. Vandermeulen, P.A. van den Elsen, S. Napel, T.S. Sumanaweera, B. Harkness, P.F. Hemler, D.L.G. Hill, D.J. Hawkes, C. Studholme, J.B.A. Maintz, M.A. Viergever, G. Malandain, X. Pennec, M.E. Noz, C.Q. Maguire, M. Pollack, C.A. Pellizari, A. Robb, D. Hanson, and R.P. Woods. Comparison and evaluation of retrospective intermodality image registration methods. *J. Comput. Assist. Tomogr.*, 21:554–566, 1997.

[183] S. Wong, R. Knowlton, M. Chew, and H. Huang. Integrating mutli-dimensional imaging, multi-modality registration and multimedia database for epilepsy diagnosis. In *SPIE medical imaging 1995*, page 2431, 1995.

[184] R.P. Woods, J.C. Maziotta, and S.R. Cherry. Mri-pet registation with automated algorithm. *J. Computer Assisted Tomography*, 17:536–546, 1993.

[185] H. Yamada. Complete euclidean distance transformation by parallel operation. In *7th International Conference on Pattern Recognition*, pages 336–338, Montreal, Canada, 1984.

[186] G.Z. Ye. The signed euclidean distance transform and its applications. In *9th International Conference on Pattern Recognition*, volume 1, pages 495–499, Rome, Italy, 1988.

[187] A. Zijdenbos, B. Dawant, R. Margolin, and C. Palmer. Morphometric analysis of white matter lesion in mr images: Method and validation. *IEEE Transactions on Medical Imaging*, 13(4):716–724, 1994.

[188] A. Zijdenbos, R. Forghani, and A. Evans. Automatic quantification of ms lesions in 3d mri brain data sets: validation of insect. In *Proc. of Medical Image Computing and Computer-Assisted Intervention - MICCAI'98*, pages 439–448, Cambridge, MA, USA, 1998.

[189] T.D. Zuk and M.S. Atkins. A comparison of manual and automatic methods for registering scans of the head. *IEEE Transactions on Medical Imaging*, 15:732–744, 1996.