Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Bachelor's Program in Computer Science

Bachelor's Thesis

Hamilton-Jacobi Skeletonisation in Image Processing

submitted by

Pascal Tobias Peter

on February 25, 2010

Supervisor
PD Dr. Michael Breuß

Reviewers
PD Dr. Michael Breuß
Prof. Dr. Joachim Weickert

Mathematical Image Analysis Group

Peter, Pascal Tobias
Hamilton-Jacobi Skeletonisation in Image Processing
Bachelor's Thesis in Computer Science
Saarland University
Saarbrücken, Germany
February 2010

Abstract

Shape analysis is a central problem for many practical applications in image processing and computer vision, such as shape recognition and segmentation. In order to represent shapes in a way that fits the needs of shape-related image processing methods, 2-D shapes are often not described by their outline, but by alternative shape descriptors. The medial axis transform [Blum, 1967] is a widely-used shape descriptor, which represents a shape by a thin set of lines and arcs that are centred in the shape. Due to its visual similarity to bone structures, the medial axis is also referred to as the skeleton of a shape.

One particular method for skeletonisation is the Hamilton-Jacobi approach by Siddiqi et al. [2002], which is derived from a wave propagation model using methods from classical mechanics. It identifies skeleton points as the sinks of an Euclidean distance map's gradient vector field by making use of the vector field's outward flux. In order to preserve major topological features of the original shape, the skeleton is computed by removing points sequentially from the shape, obeying homotopy preserving rules.

The primary goal of this work is to analyse and extend the Hamilton-Jacobi method, as well as to design general methods for skeleton comparisons. Those methods are used to assess the output quality of the Hamilton-Jacobi approach in relation to other skeletonisation algorithms.

Two new algorithms are proposed that are based on the Hamilton-Jacobi method and a recent maximal disc detection algorithm [Rémy and Thiel, 2005]. The new methods feature performance and exactness improvements compared to the original Hamilton-Jacobi algorithm, as well as a reduction of the dependence of output quality on input parameters. Additionally, a new thinning algorithm that does not rely on classical mechanics is introduced as a basis for comparisons of the Hamilton-Jacobi method to thinning methods with different theoretical foundations.

In order to allow in depth comparisons of skeletonisation methods, general quality criteria for discrete skeletons are introduced and used as a basis for defining quality measures. As a second method for skeleton analysis, an approach based on graph matching is proposed. Skeleton graphs are defined by taking skeleton end- and branching points as vertices. Edges represent the configurations of skeleton branches. In order to measure skeleton quality, the graphs of skeletonisation results are matched to reference graphs that represent exact skeletons.

Tests of both the newly proposed algorithms and the quality measures were conducted on CE-Shape-1, a widely used image database consisting of 1400 shapes, which was specifically designed for testing shape descriptors [Latecki et al., 2000]. Additionally, test cases for specific properties of the medial axis transform were designed and implemented.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, February 25, 2010

Pascal Tobias Peter

Acknowledgments

I thank PD Dr. Michael Breuß for providing not only an interesting and engaging topic, but also for his constant support, especially for listening to all of my ideas and problems and for supplying me with all the materials I could wish for. My thanks also go to Prof. Dr. Joachim Weickert for giving me a wide range of topics to choose from and offering a perfect working environment at the Mathematical Image Analysis Group.

Furthermore, I owe gratitude to Prof. Dennis Shasha for providing his approximate graph matcher *graphdiff* and for his kind replies to my support requests.

Additional thanks go to Sarah Diehl and Haiko Wick for their tireless and competent proofreading efforts. I am also very grateful for Sarah's valuable hints on LaTeX and her imperturbable patience in putting up with my tirades on thesis-related ideas and problems in her spare time.

Finally, I want to thank my family for their constant and unconditional support in everything I do.

Contents

Chapter 1

Introduction

Segmentation and object recognition are two fundamental problems of image processing. Those and similar tasks are closely related to the natural problem of shape vision [Blum, 1967]. For humans as well as for animals it is important to recognise different objects in the vision field and assess their importance and functionality. Based on this input, decisions can be posed, for instance if an animal recognises a predator, it might flee.

While the shape of objects is not the only information that is usually considered in a decision process based on sensory input, it remains a very important factor for the aforementioned tasks.

Intuitively, the oldest science dealing with shapes, geometry, can provide the necessary mathematical tools for shape vision. However, according to Blum [1967], classic geometry is not suited for this "biological problem of shape".

On one hand, classic geometry is based on observation and was developed with problems of physical science in mind and thus is not optimised for the biological point of view. On the other hand, there is such a vast variety of different shapes, that researchers usually have to concentrate on a small subset of shapes. The choice and interpretation of those subsets might be culturally biased. To avoid those problems, a higher degree of abstraction is needed.

It is possible to analyse shape with established geometric tools, for example by using the number of inflection points of the shape's boundary or introducing other curvature-based properties. However, in general, the different fields of geometry are either too specific (e.g. Euclidean/projective geometry) or too general (e.g. topology). A suitable shape descriptor must therefore offer a degree of abstraction that lies between those two extremes.

As a remedy for the shortcomings of classic geometry, Blum [1967] introduced the medial axis transform (MAT). The MAT was designed to provide a set of shape attributes that can be used to rate shapes based on specific relevance criteria, thus mimicking biological sensory processes. It exploits local symmetries of a shape, reducing it to a thin line that is equidistant to the shape's boundaries and is therefore also referred to as a central shape descriptor. The MAT is also called the skeleton of a shape, since for animal silhouettes it resembles simplified articulated bones.

1.1 The Grass-Fire Analogy

An intuitive definition of the MAT of 2-D shapes is based on wave propagation and can be compared to the spread of a prairie fire.

In order to define a two-dimensional skeleton, a planar shape is considered that is described by its boundary curve in $\mathbb{R}^2$. For the purpose of the analogy, the x-y-plane is regarded as a field of grass, while the boundary of the shape corresponds to the source of a spreading grass-fire. This means that the grass on the shape's boundary lines is already burning when the observation of the propagation process begins. The fire spreads uniformly in all directions, inciting new patches of grass adjacent to its front and leaves behind burned patches that cannot be inflamed anymore.

Eventually, several fronts of the spreading fire may collide. They cannot pass through each other, since the areas that were previously passed by the expanding fire are already burned. Therefore, the fronts undergo cancellation in all collision points (shocks) of two or more fronts. Those collision points form the skeleton of the shape. Depending on the shape of the object, skeleton points may appear both outside and inside of the object. Usually, the term skeleton is only used for the set of inside MAT points.

The analogy described above was used by Blum [1967] and is also referred to as Blum's grass-fire model. A formal wave propagation model based on the prairie fire analogy can be found in Section 2.5.

1.2 Applications

The medial axis provides useful mathematical properties: in combination with the distance of the skeleton points to the object boundary, the MAT is a compact description of the shape, which is equivalent to the representation by its boundary. Additionally, the skeleton is invariant under Euclidean transformations and shares major topological features with the original object. More details on this properties can be found in Section 2.4.

The aforementioned properties qualify the medial axis transform for a wide variety of both theoretical and practical applications. In this section, several examples for the use of skeletons in different fields are briefly presented.

Skeletons and related concepts were successfully applied to the task that Blum [1967] originally designed the MAT for: object recognition. Examples for this use are the FORMS framework of Zhu and Yuille [1995], which uses modified skeletons, and different methods based on shock graphs [Siddiqi et al., 1999; Sebastian et al., 2001].

Shock graphs are labeled skeletons, wherein each point of the medial axis is assigned a certain shock class based on its relation to other medial points (e.g. medial points that have a larger distance to the image boundary than their medial neighbours in an ϵ-neighbourhood belong to the same shock class). The labels form the nodes of a graph. Siddiqi et al. [1999] use shock graphs to construct tree-representations of shapes. Object recognition is achieved by comparing the trees of the corresponding shapes (see Figure 1.2). MATs are also used in optical character recognition [Lakshmi et al., 2009], which can be regarded as a special case of object recognition.

Furthermore, there are geographical applications: The Crown Registry and

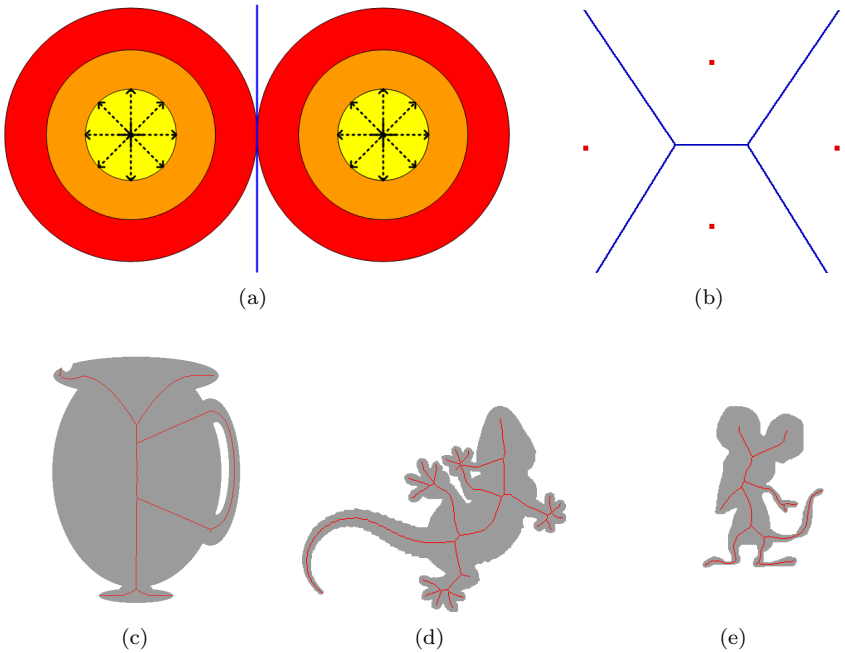


Figure 1.1: **Skeletons and the grass-fire model.** In (a), the wave propagation of the grass-fire algorithm for two points is described. The wavefronts expand in a circular fashion from each point. The first shock occurs when the circular fronts have traveled the same distance from both source points and intersect in exactly one point. The final skeleton consists of the line of points with equal distance to both propagation sources. A skeletonisation result for four points (red) is displayed in (b). The skeleton (blue) coincides with the borders of the Voronoi diagram (see Section 2.3) generated by the four points. Practical examples of skeletonisation results as shape descriptors are shown in (c)-(e). The skeletons of the grey shapes are marked by red lines.

Geographic Base [2001] uses skeletonisation to maintain a database of watershed-based geographical boundaries. A MAT approximation that relies on Voronoi diagrams (see Section 2.3) is used to compute skeletons of lakes and rivers (see Figure 1.3).

Skeletonisation is also applied in medical imaging. Sorantin et al. [2002] used three-dimensional MATs on images of the laryngo-tracheal tract obtained by spiral computed tomography. The skeleton was used to assess site, length and degree of tracheal-stenoses automatically.

The applications of skeletons are not necessarily direct incorporations of the MAT in algorithms, but can also help to understand related concepts. Bornemann and März [2007] used skeletons in the field of image inpainting (the task of reconstructing missing or masked parts of images) to explain the behaviour of the fast inpainting algorithm of Telea [2004]. Namely, errors produced by Telea's algorithm occur because image information is not transported across the skeleton of the reconstructed areas.

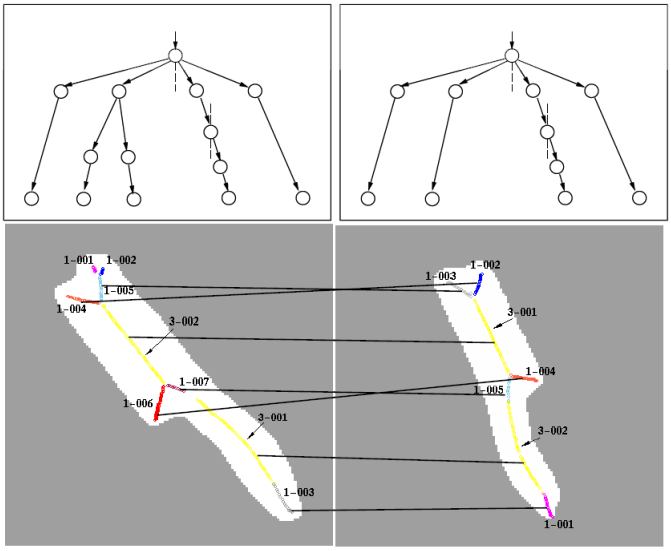


Figure 1.2: **Object recognition with shock graphs** (courtesy of Siddiqi et al. [1999]). The skeleton is divided into several segments that consist of medial points of equal shock type. In the lower images, the first number of the annotations denote the shock types. The upper pictures show trees built from the shock labels in the images below. In order to perform object recognition, tree nodes are matched as depicted by the lines connecting the labels in the lower images.

1.3 Related Work

Due to the many different applications of the medial axis, there is a wide variety of different approaches for its computation. Existing skeletonisation algorithms can be categorised by several means.

First, the different methods can be classified by their underlying theoretical background, which yields three basic classes of algorithms. For each class, due to the high number of different approaches, only two representative methods are given instead of an exhaustive list.

Thinning algorithms sequentially remove points from the object, mimicking the "burning" of patches in the grass-fire analogy. Many thinning algorithms guarantee that the resulting skeleton preserves major topological features of the original shape (i.e. the skeleton is homotopic to the shape), but do not locate skeleton points accurately. In order to achieve better results, thinning approaches are often combined with other methods.
Zhu et al. [1994] introduced a boundary-based thinning algorithm that divides the object into axis-parallel lines and reduces them to their midpoint, exploiting the object's local width. Homotopy is preserved by restoring connectivity after a disconnect occurs.
A 3-D thinning algorithm was proposed by Palagyi and Nemeth [2009]. Multiple points are removed from the object in each step using homotopy preserving rules, while skeleton endpoints are preserved. The identifica-

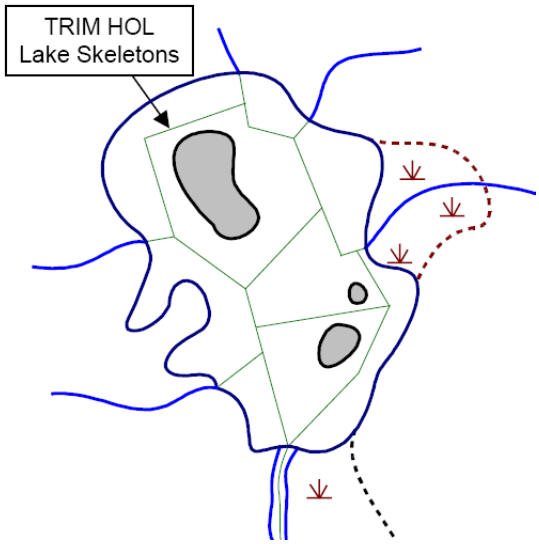


Figure 1.3: **Skeletonisation in the TRIM database** (courtesy of Crown Registry and Geographic Base [2001]). The green line represents an approximation to the medial axis of the lake which is equidistant to the shorelines (dark blue) of the lake. However, TRIM does not use exact skeletons, but medial axis that are adapted to the specific needs of the database. This modified MAT can be interpreted as the borders of the Voronoi-diagram (see Section 2.3) that is created by the shoreline segments that result from dividing at estuaries.

tion of skeleton endpoints is independent from the thinning method.

Voronoi-based algorithms exploit the fact that the MAT can be defined by the borders of a Voronoi diagram of the shape boundary (see Section 2.3). Ogniewicz and Kübler [1995] used Voronoi diagrams of the boundary points together with a topology preserving metric to compute hierarchic skeletons. The resulting MATs form a pyramid that reaches from simple skeletons with wide, but few branches to thin, complex skeletons of 2-D shapes.
Another method for 2-D shapes was proposed by Kimmel et al. [1995]. First, the object boundary is segmented at its points of maximum curvature. Afterwards, the corresponding Voronoi diagram is computed. The borders of the diagram form the skeleton and coincide with the zero level sets of the boundary segment's distance map differences.

Distance map based algorithms use the definition of medial axis as the ridges of the shape boundary's distance map (see Section 2.3).
Exploiting the fact that the locations of the medial axis points coincide with the centres of maximal inscribed discs, Rémy and Thiel [2005] proposed an algorithm to precompute lookup tables that allow to identify skeleton points by comparing the distance map values of an object-width-dependent neighbourhood to the values of the lookup table.
Malandain and Fernández-Vidal [1998] introduced two local heuristic measures to characterise the singularities of the Euclidian distance function.

Skeletons for objects of arbitrary dimensions are retrieved by thresholding both measures and applying a topological reconstruction step that restores homotopy to the original shape.

Additionally, methods differ significantly in the type of object representation that is considered for skeletonisation. Again, there are three different classes: most algorithms [e.g. Ogniewicz and Kübler, 1995; Rémy and Thiel, 2005; Zhu et al., 1994] use binary images for object representation, others rely on polygonal approximations of the object boundary [e.g. Montanari, 1968] or deal with smoothed object boundaries [e.g. P.J. Giblin, 1985].

Note that membership to the classes above is by no means mutually exclusive. For example, the method of Kimmel et al. [1995], which is mentioned above, uses both Voronoi diagrams and distance maps and also makes use of all three categories of object representation.

1.4 Overview

The main focus of this work lies on a skeletonisation method based on Hamiltonian mechanics as introduced by Siddiqi et al. [2002]. In this approach, skeleton points are located as sinks of the distance map's gradient vector field. The average outward flux of the gradient vector field is used to identify the sinks by thresholding combined with a homotopy preserving thinning process.

Several improvements to the original algorithm by Siddiqi et al. [2002] are proposed, namely updated methods for distance map computation and two modified versions of the algorithm. The modifications are designed to remove the need for manual adjustment of parameters and improve the robustness of the results under boundary perturbations. In the first new method, the flux-ordered thinning algorithm with adaptive thresholding, a secondary MAT detection method is used to precompute an approximation of the skeleton. This preliminary result is used to adjust the flux threshold to the specific properties of the shape. The second modified version of the original algorithm replaces MAT detection with flux thresholding by a maximal disc detection method [Rémy and Thiel, 2005] and only uses flux values to determine the order for the thinning process.

In order to analyse the resulting MATs, quality criteria for discrete skeletons are introduced. Based on those criteria, measures for exactness of reconstruction, skeleton minimality and skeleton complexity are defined. Those measures can be used to determine suitability of the resulting MATs for specific tasks with different priorities for skeleton properties.

Based on the quality measures, a comparison of five skeletonisation algorithms is conducted. This comparison is used to check the Hamiltonian approach against its modified versions and methods that require less complex computations. The five methods that are compared are the flux-ordered thinning algorithm by Siddiqi et al. [2002], its two modifications, a maximal disc detection algorithm by Rémy and Thiel [2005] and a new thinning algorithm.

The new maximal disc thinning algorithm that is proposed in this work resembles the approach of Pudney [1998] in that it removes points from the object in the order of their distance to the boundary using homotopy preserving rules. Skeleton endpoints are marked separately and are not removed. In contrast to

Pudney's thinning algorithm, the new method uses exact Euclidian distances instead of the Chamfer distance to identify skeleton endpoints based on the maximal disc criteria by Rémy and Thiel [2005]

1.5 Outline

Chapter 2 gives a general introduction on the grass-fire model, mathematical definitions of skeletons and their properties. It also contains basic definitions and notational conventions that are used in this work and establishes the theoretical background for Hamilton-Jacobi skeletons and the resulting flux-ordered thinning method for skeletonisation.

Improvements of the flux-ordered thinning method, an alternative thinning algorithm based on maximal disc detection and quality criteria for skeletons are presented in Chapter 3.

Details about the implementation of the methods established in Chapters 2 and 3 can be found in Chapter 4. The information in this chapter includes explanations and pseudocode descriptions of the algorithms used in this work, as well as remarks on boundary treatment and correctness tests.

A testing environment for the different skeletonisation algorithms of this work is described in Chapter 5. Additionally, the results of several test runs on different data sets from the testing environment are presented.

Chapter 6 concludes the thesis with summarising remarks and an outlook on possible future work.

Additional pseudocode and instructions for program usage can be found in the appendices.

Chapter 2

Theoretical Background

In this chapter, both an overview of the general underlying concepts of the medial axis, such as different definitions and properties, as well as the theoretical background of the Hamiltonian approach by Siddiqi et al. [2002] are given.

2.1 Definitions

This section contains basic definitions and notations that are used throughout the following sections. Most of the definitions establish two versions of the same concept: a continuous and a discrete one.

The distinction between both versions is important, since there are significant differences between continuous and discrete settings in skeletonisation processes. While the foundations of a skeletonisation method's theoretical background are usually discussed in the continuous setting, discretisation is, in most cases, non-trivial, and introduces new sources of error that must be treated separately.

In this work, the grid size h is introduced as an additional index that distinguishes mathematical objects in the discrete setting from their continuous counter parts.

Image u(x) : The binary image containing the object is represented by the function

$$u : \Omega \to \{0, 1\}$$

$\Omega \subset \mathbb{R}^2$ is called the *image domain*. Object points have value one, background points have value zero.

The discrete version of the image with grid size h and dimension $m \times n$ is denoted with

$$u_h : \Omega_h = \{1, \dots, m\} \times \{1, \dots, n\} \to \{0, 1\}, \quad \Omega_h \in \mathbb{N}^2$$

Object Domain O : Subset of the image domain that contains all object points.

$$O = \{x \in \Omega \,|\, u(x) = 1\}$$

Consequently, $\Omega \setminus O$ is the set of background points. The discrete object domain O_h is defined analogously to the continuous case.

ϵ-neighbourhood $B_\epsilon(x)$: The open ball with radius ϵ around the point x.

$$B_\epsilon(x) = \{y \in \Omega : |y - x| < \epsilon\}$$

$|\cdot|$ denotes the Euclidian norm. The ball is open in the topological sense which is equivalent to the exclusion of the ball's boundary from the set. The discrete ball $B_{\epsilon,h}$ is defined analogously to the continuous version.

Open and closed sets : Several of the following definitions rely on the topological notion of open and closed sets. For any set $A \subset \Omega$, any given point $x \in A$ is called a boundary point, if every open disc with centre x contains at least one point that is not contained in A.

Consequently, the boundary ∂A of is defined as

$$\partial A = \{x \in A | \forall \epsilon > 0 : B_\epsilon(x) \cap \Omega \setminus A \neq \emptyset\}$$

All points of A that are not boundary points are called interior points.

A is referred to as an open set, if it does not contain any boundary points (i.e. $A = A \setminus \partial A$). The complements of open sets are called closed. In other words, B is closed if and only if $\Omega \setminus B$ is an open set.

Thin sets : A set $A \in \Omega$ is referred to as thin, if and only if it only contains boundary points. This condition can be paraphrased as: A is thin if and only if $A = \partial A$.

8-Neighbourhood $\mathcal{N}_8(p)$: The 8-neighbourhood of a discrete point $p \in \Omega_h$ is only defined in the discrete setting and contains all direct neighbours $n_1(p), \ldots, n_8(p)$ of p as shown in Figure 2.1.

4-Neighbourhoods $\mathcal{N}_4(p)$ **and** $\mathcal{D}_4(p)$: The 8-neighbourhood of a point $p \in \Omega_h$ can be partitioned into two 4-neighbourhoods, namely the four diagonal neighbours $\mathcal{D}_4(p)$ and the four vertical/horizontal neighbours $\mathcal{N}_4(p)$ of p.

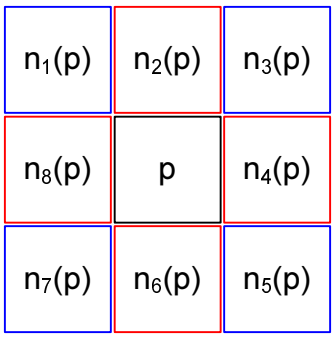


Figure 2.1: 8-Neighbourhood of a point in the discrete setting.

Object Boundary ∂O : Boundary of the object domain. In the continuous setting, the object boundary is defined as the boundary of the set O in the algebraic sense for the metric space with the Euclidean metric, i.e.

$$\{x \in O \,|\, \forall \epsilon > 0 : \; B_\epsilon(x) \cap \Omega \setminus O \neq \emptyset\}$$

The discrete object boundary can be defined in an easier way by utilising the 8-neighbourhood of the discrete points:

$$\partial O_h = \{x \in O_h \,|\, \exists y \in \mathcal{N}_8(x) : y \in \Omega_h \setminus O_h\}$$

Distance Map $D(x)$: Maps each point of the image domain Ω to its minimal distance to the object boundary, i.e.

$$D : \Omega \to \mathbb{R}_0^+, \quad D(x) = \min_{y \in \partial O} |y - x|$$

For the Hamiltonian approach, a signed distance map is used. In signed distance maps, background points have negative distance values, while object points have positive distance map values.

$$D : \Omega \to \mathbb{R}, \quad D(x) = \begin{cases} \min_{y \in \partial O} |y - x|, & \text{if } x \in O \\ -\min_{y \in \partial O} |y - x|, & \text{else} \end{cases}$$

Voronoi Region $V_O(x)$: A Voronoi region $V_O(x)$ is the set of image points that are closer to the object point x than to any other object point y.

$$V_O(x) = \bigcap_{y \in O \setminus \{x\}} \{p \in \Omega | \ |p - x| < |p - y|\}$$

Skeleton Σ : The set of centre points of maximal inscribed discs (see Section 2.3 for more details).

$$\Sigma = \{x \in O | \forall y \in O : B_{D(x)}(x) \not\subset B_{D(y)}(y)\}$$

The skeleton is also referred to as the medial axis transform (MAT) .

$D\big|_\Sigma$ is called the medial axis function (MAF). It maps each skeleton point to the radius of the corresponding inscribed disc and allows object reconstruction (see Section 2.4).

2.2 The Wave Propagation Model

Formally, the grass-fire analogy from Section 1.1 can be described by a wave propagation model. The expansion of the object's boundary ∂O, regarded as a curve, propagating with a constant velocity $F : \Omega \to \mathbb{R}$ in its inward normal direction $n : \Omega \to \mathbb{R}^2$, is observed in this model. The locations of the shocks (cancellation points), that appear during this process, form the skeleton.

The evolution of the boundary curve can be described by the change over time in the coordinate vectors of each curve point. Let $c(x, y, t)$ denote the coordinate vector of a curve point $(x, y) \in \partial O$. Then the propagation process is described by the partial differential equation

$$\frac{\partial c}{\partial t} = F \cdot n$$

Since only the evolution of the boundary in direction of the inward normal is observed, the outer skeleton is disregarded in this approach.

The special case of a circle acts as a good example for the functionality of this model: the circular wavefront continuously shrinks to a single point, the centre of the circle. No other shocks except for the centre point of the original circle occur during the process. The skeleton combined with the traveling time yields a very efficient, exact description of the circle: it is equivalent to centre and radius. Actually, as explained in Section 2.3, inscribed circles and their corresponding discs can be used to define the skeleton for arbitrary objects in an alternative way.

Note that, in this model, every point on the evolving wavefront is generated from exactly one point of the object boundary, as long as no cancellation occurs. Therefore, the wavefronts in this model carry boundary information in form of the origin points on the boundary. Since the fronts propagate with a known velocity, it is possible to compute the minimal traveling distance from a point p of the front to the shape's boundary at any given time. This traveling distance coincides with the distance of p to its origin point.

However, every time wavefronts cancel out, a part of the contained boundary information is lost from the evolving front, since the information from the original sources of the collision point are not further propagated. Therefore, keeping this information in the cancellation points is an intuitive way to reduce redundant features of the shape's boundary (local symmetries) to the skeleton without directly examining any geometric properties of the boundary.

Including the time of shock occurrences in the description yields the medial axis function (MAF) . The MAF can be used to reconstruct the original object from its skeleton, as described in Section 2.4. Equivalently, the minimal distance of collision points to the boundary can be used as MAF values.

2.3 Alternative Definitions

In addition to the grass-fire analogy there are several equivalent definitions for the MAT, each of them allowing to approach the problem from another point of view.

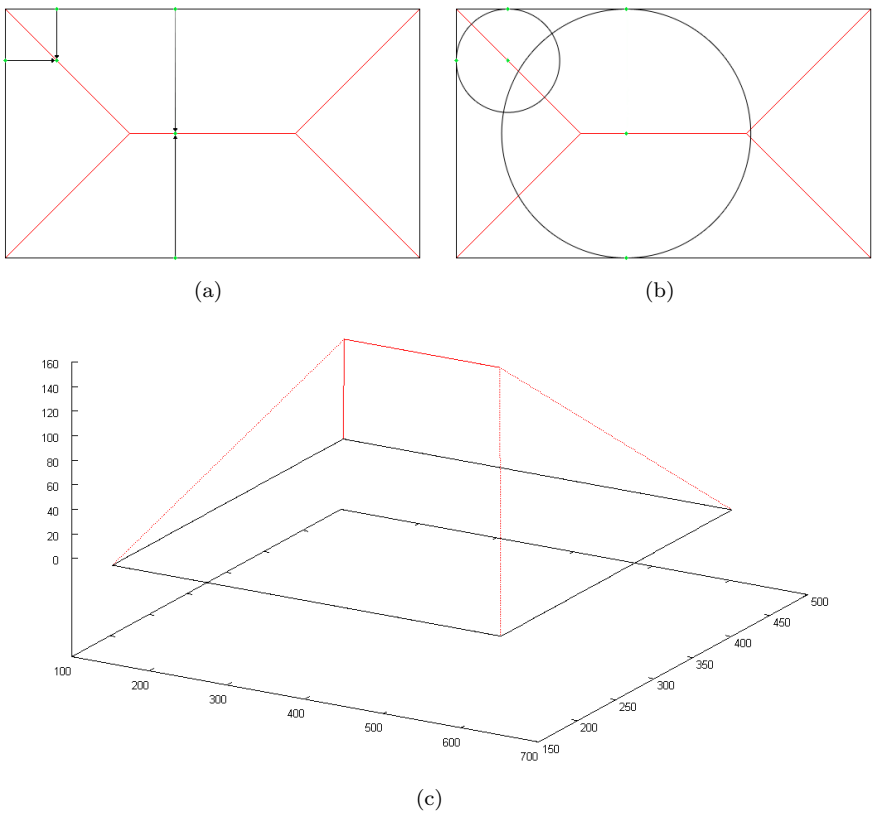
Equidistant Viewpoint

In the grass-fire model, skeleton points mark collision points of evolving wavefronts that originate from the object boundary. As mentioned in Section 2.2, each point on a wavefront corresponds to exactly one point of the object boundary, as long as no collisions occur.

Consequently, if several fronts collide in a skeleton point s, each front carries the information of a boundary point b_i ($i = 1, \ldots, k$, where k is the number of colliding fronts) that is a source point of s.

Since all of the wave fronts propagate with the same constant velocity, the distance $|s - b_i|$ is equal for all $i \in \{1, \ldots, k\}$. All non-skeleton points are passed by the evolving wavefronts without collision and therefore only exactly one point on the boundary has minimal distance to them.

Thus, an object point p is a skeleton point if and only if multiple boundary points $b_1, \ldots, b_k$ exist, which are equidistant to p. The distance of p and one of the equidistant boundary points is equal to the distance map value of p.



(a) (b)

(c)

Figure 2.2: **Alternative definitions of the MAT.** The medial axis (red) of a rectangle (black) is considered for different definitions. In (a), boundary points are connected by arrows to their corresponding skeleton point. Each skeleton point has equal distance to at least two boundary points which coincide with the source points of the wave propagation in the grass-fire model. Image (b) displays the maximal inscribed discs for two of the skeleton points and in (c), the medial axis is depicted as the ridges of a 3-D distance map representation.

Inscribed Discs

Based on the equidistant viewpoint from Section 2.3, another definition of the medial axis based on inscribed discs can be established. It allows easy reconstruction of the original object from the medial axis (see Section 2.4).

Since a point s of the object is a medial axis point, if and only if there are several boundary points with minimal distance d to s, those boundary points are all located on a circle with radius d around s.

Consequently, a point is part of the medial axis, if and only if a circle around an object point with the radius of its minimal distance to the boundary touches several boundary points. An equivalent condition for aforementioned circles is, that the disc bounded by the circle is maximal. This means that a point $x \in O$ is a medial axis point if and only if the inscribed disc around x is not contained in any other inscribed disc with centre $y \in O$, $y \neq x$. The radius of the inscribed

disc around a point $x \in O$ is its distance $D(x)$ to the boundary. Therefore, the skeleton can be defined as

$$\Sigma = \{x \in O | \forall y \in O : B_{D(x)}(x) \not\subset B_{D(y)}(y)\}$$

Voronoi Diagrams

The Voronoi diagram of the object is the collection of the Voronoi regions of all object points. A Voronoi region $V_{\partial O}(x)$, as defined in Section 2.1, contains only those points that are closer to $x \in \partial O$ than to all other object points $x \neq y \in \partial O$.

Therefore, all points that are equidistant to two boundary points are not contained in any of the Voronoi regions, but all other points are. By definition of the equidistant viewpoint, only skeleton points have equal minimal distance to two or more boundary points. If $\hat{\Sigma}$ denotes the set of inner and outer skeleton points, the image domain Ω can be partitioned in the following way:

$$\Omega = \bigcup_{x \in \partial O} V_{\partial O}(x) \cup \hat{\Sigma}$$

In particular, the skeleton points form the boundary of the Voronoi regions and Σ can thus be defined as

$$\Sigma = \left(\bigcup_{x \in \partial O} \partial V_{\partial O}(x) \right) \cap O$$

Distance Maps

The evolving wave fronts from the grass-fire analogy coincide with the level sets of the distance map. A level set (or isoline) of a function is a set of points on which the function is constant. For a distance map D, a level set with constant value c is defined as

$$\{p \in \Omega \mid D(p) = c\}, \quad c \in \mathbb{R}$$

$D(p)$ however is, for each point $p \in \Omega$, the minimal distance of p to the object's boundary. Thus, each level set with constant distance map value c corresponds to the points of the wave front after traveling the distance c from the boundary.

Interpreting the distance values that are assigned to each point of the image domain by the distance map as a height value, a 3-D representation of the distance map can be obtained. From this point of view, the medial axis corresponds to the ridges of the distance map (see Figure 2.2(c)).

Higher Dimensions

While in this work only the 2-D case is considered, all of the definitions above can be applied to arbitrary dimensions by using the n-dimensional Euclidian distance. This directly allows to apply the equidistant viewpoint.

Voronoi diagrams, distance maps and inscribed spheres in n dimensions can be defined accordingly and thus the other definitions can also be extended to arbitrary dimensions.

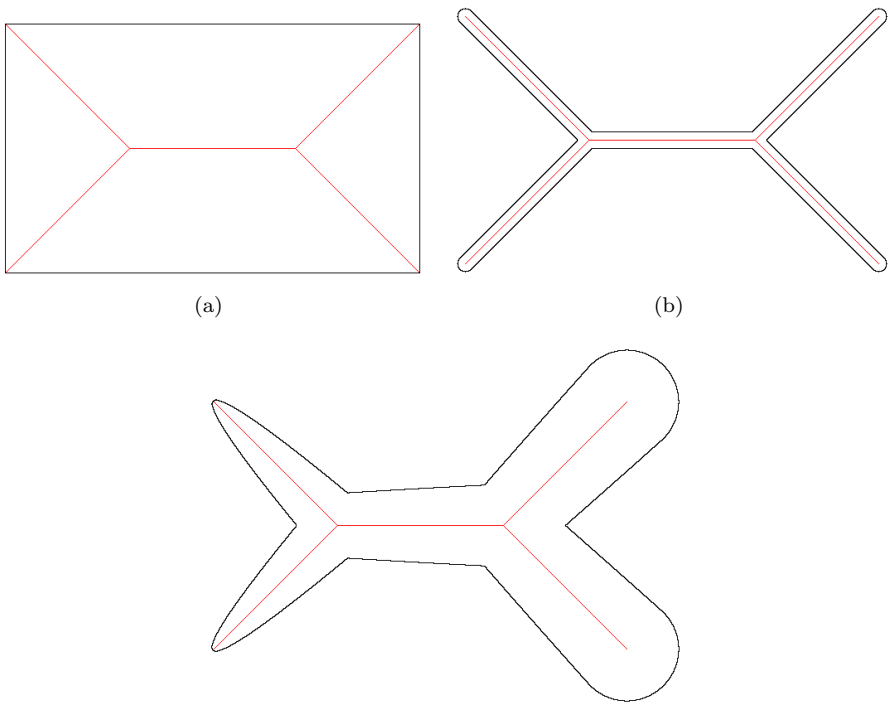


Figure 2.3: **Uniqueness of the MAF.** (a) shows the skeleton of a rectangle. The skeleton points (red) together with their distance to the boundary form a unique descriptor of the corresponding shape. The location of the medial axis alone does not suffice to define a one-on-one correspondence between skeleton and shape. For each skeleton, depending on the values of the MAF, arbitrary many shapes exist. (b) and (c) depict shapes that have the same medial axis as the rectangle (a), but different MAF-values.

2.4 Properties of the MAF

The medial axis function has several properties that qualify it as a robust alternative to the boundary as a shape descriptor. These properties are also important for designing quality criteria for skeletonisation algorithms.

Equivalent Descriptor of Shape

One central property of the MAF is its equivalence to the shape description by its boundary. This property can be observed in the wave propagation model from Section 2.2: if a wave spreads from each boundary point uniformly into all directions, as long as there are at least two points in the image, the wave fronts will eventually collide. Therefore, each point on the boundary has at least one corresponding point on the skeleton.

Conversely, each medial axis point p corresponds to at least two points on the object boundary ∂O that are also located on the boundary of the inscribed disc

$B_{D(p)}(p)$. Drawing the maximal inscribed discs for each point in the skeleton therefore yields the original shape again if O is an open set.

$$O = \bigcup_{p \in O} B_{D(p)}(p)$$

Note that the skeleton alone does not suffice for the reconstruction of an object from its MAT. The radii of the inscribed maximal discs for each skeleton point define the local width of the object, which means that one skeleton can define arbitrary many shapes (see Figure 2.3). In other words, the MAT does not correspond to a unique shape, but between the MAF and shape boundaries exists a one-on-one correspondence.

Homotopy

The medial axis is homotopic to the corresponding shape. This means that the shape and its skeleton have the same number of connected components, holes and cavities [Kong and Rosenfeld, 1989] or, in other words, the skeletonisation process is topology preserving. Thus, major topological features of the shape are preserved when reducing it to its skeleton. In particular, the skeleton of a single connected shape is also connected.

Invariances

Skeletons are invariant under Euclidian transformations, i.e. under translation and rotation. Invariant means in this context that the skeleton of a transformed shape is identical to the MAT of the original shape after the same transformation.

In other words, skeletonisation and the aforementioned transformations can be applied to an object in arbitrary order. If O is an object, $\mathcal{S}$ the skeletonisation operator ($\mathcal{S}(O) = \Sigma$) and $\mathcal{T}$ a transformation operator, the invariance is described by the equation:

$$\mathcal{S}(\mathcal{T}(O)) = \mathcal{T}(\mathcal{S}(O))$$

Thin Set

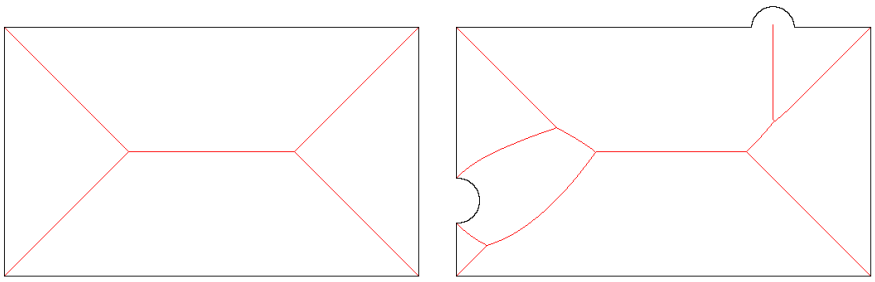
The set of skeleton points is thin. In a discrete setting, this means that the medial axis is composed of lines and arcs with a width of exactly one pixel.

For the continuous case, thin sets can be defined via the topological notion of set boundaries. For an arbitrary set $A \subset \mathbb{R}^2$, the boundary ∂A is defined as the set of all points that have at least one adjacent point in an arbitrary small neighbourhood that is not included in A.

$$\partial A = \{x \in A \,|\, \forall \epsilon > 0 : \; B_\epsilon(x) \cap \mathbb{R}^2 < \emptyset\}$$

A set is called thin if it only contains boundary points, i.e.

$$\partial A = A$$



(a) (b)

Figure 2.4: **Effects of boundary perturbations on the MAT.** Image (a) shows the skeleton of a rectangle. Modifying the boundary yields significant changes in the skeleton. Example (b) displays the effect of boundary perturbations on the skeleton with large modifications to the boundary. Note that those effects occur in the continuous setting for arbitrary small perturbations.

Convexity Criterion

The contours of convex objects only produce an inner skeleton. Points of the outside MAF correspond to boundary points that are not on the convex hull of the object's contour.

Sensitivity to Boundary Perturbations

The MAT as an object descriptor also possesses properties that can be considered as drawbacks for practical applications. It is particularly sensitive to all kinds of boundary perturbations. A slight change in the boundary might cause significant changes, like additional branches, in the skeleton (see Figure 2.4).

Therefore, noise in input images or merely the discretisation of the boundary can lead to significant errors in the computed skeleton if algorithms do not address this problem (e.g. via presmoothing of the object boundary).

2.5 The Eikonal Equation

The wave propagation model from Section 2.2 uses coordinate vectors c of the boundary curve C and models their change over time with the equation

$$\frac{\partial c}{\partial t} = F \cdot n$$

where $F(x, y)$ is the speed and $n(x, y)$ the direction of the inward normal at the point $(x, y) \in \Omega$.

Instead of using a time component and two-dimensional curves, the wave propagation can also be modelled by introducing a third dimension. The evolved boundary curves can be superimposed in three-dimensional space by assigning the corresponding time t to each curve as its height value.

Note that each point of the 2-D plane is only passed once by the evolving wave front and the set of all evolved curve points forms a connected 3-D surface

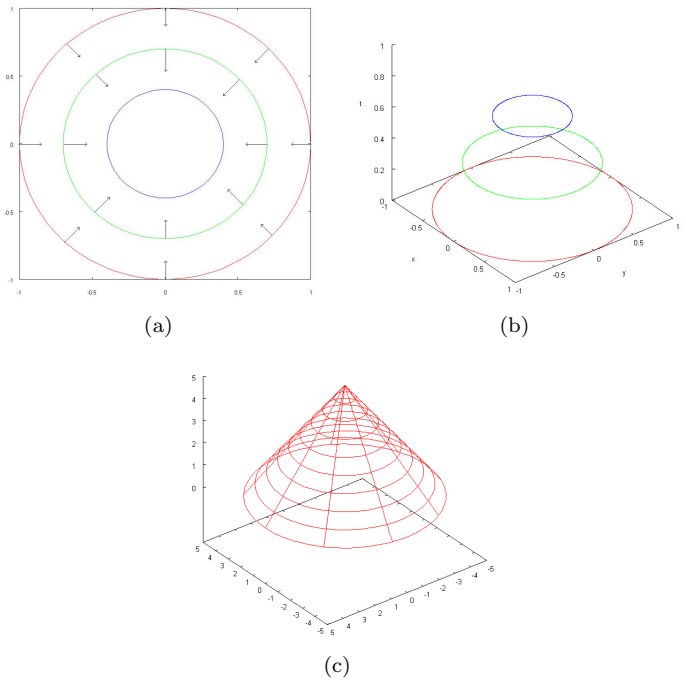


Figure 2.5: **Wave Propagation and the Eikonal Equation.** (a) shows the two dimensional wave propagation of a circular boundary as defined in Section 2.2. Superimposing the evolved curves in 3-D space yields (b). The entirety of all evolved curves form a surface, in the case of (b) a cone.

(see Figure 2.5). Since this surface contains all evolved curves, it is called the solution surface for the curve evolution process. It defines a function $T(x, y) = t$ that maps each point $(x, y) \in \Omega$ to the time at which the propagating wave front passes (x, y). The evolved curve at time t is thus given by the intersection of the solution surface with a plane parallel to the x-y-plane at height t:

$$\{(x, y) \in \Omega \mid T(x, y) = t\}$$

T can be used to express the speed of the propagating wave front. The gradient ∇T points in the direction of the steepest slope of the solution surface. Thus, its projection to the x-y-plane points in the direction of the inward normal, which coincides with the direction of the propagation. Additionally, the gradient magnitude is a measure for the rate of change of the time coordinate in this direction.

The speed $F(x, y)$ of the evolving front, however, is defined by the time that is needed for the front to travel a distance in the x-y-plane. Therefore, the speed of the evolving front is inverse proportional to the rate of change in the time coordinate.

$$F(x, y) = \frac{1}{|\nabla T(x, y)|}$$

Rearranging the equation above yields the eikonal equation

$$|\nabla T| F = 1$$

There are several methods for solving the eikonal equation, such as the fast marching method [Sethian, 1996] or the viscosity solution approach by Rouy and Tourin [1992]. The problem with those methods is, however, that the computation of the solution surface via the eikonal equation does not explicitly yield the location of skeleton points (shocks). Therefore, solving the eikonal equation alone is not enough to compute the skeleton. Additional shock detection must be applied.

This problem was the motivation for Siddiqi et al. [2002] to introduce an alternative method for solving the eikonal equation that allows a direct computation of the medial axis. The resulting Hamiltonian framework is presented in the following section.

2.6 Hamilton-Jacobi Skeletons

Siddiqi et al. [2002] introduced a framework for solving the eikonal equation that is based on classical mechanics, the science field that deals with the motion of bodies in space over time. In this section, only a brief introduction on the underlying physical theories is given, based on both the paper by Siddiqi et al. [2002] and the book about classical mechanics by Sussman and Wisdom [2001].

The Hamiltonian formalism is one of several formulations of classical mechanics, alongside the Lagrangian and Newtonian formalisms. It describes the movement of particles in a closed system with a phase space $(p(t), q(t))$ where q are the particle's coordinates and p the corresponding momenta at time t. The energy of the system is represented by the Hamiltonian $\mathcal{H}$ that consists of the sum of kinetic and potential energy. $\mathcal{H}$ fulfils Hamilton's canonical equations:

$$\dot{p} = \frac{\mathrm{d}}{\mathrm{d}t} p(t) = -\frac{\partial}{\partial q} \mathcal{H}(p, q) \tag{2.1}$$

$$\dot{q} = \frac{\mathrm{d}}{\mathrm{d}t} q(t) = \frac{\partial}{\partial p} \mathcal{H}(p, q) \tag{2.2}$$

Equivalently, the system above can be described with the Lagrangian formalism. Here, the Hamiltonian $\mathcal{H}(p, q)$ is replaced by the Lagrangian $\mathcal{L}(q, \dot{q})$. Thus, the state of the system is not described by coordinates q and momenta p, but by the coordinates and their respective velocities $\dot{q}$. The momenta p are not used to describe the state of the Lagrangian system, but can still be derived from it as shown in equation 2.3, just as the velocities $\dot{q}$ are a derived quantity in the Hamiltonian formalism (see equation 2.2).

$$p = \frac{\partial \mathcal{L}}{\partial \dot{q}} \tag{2.3}$$

With the Lagrangian formalism it is easier to establish a direct connection to the wave propagation model: the coordinates q describe the positions of points on the wave front and the velocities $\dot{q}$ are the corresponding vectors $F \cdot n$. Note that it is possible to express the Hamiltonian in terms of the Lagrangian and vice versa by using the Legendre transformation

$$\frac{\partial \mathcal{H}}{\partial q} = -\frac{\partial \mathcal{L}}{\partial q} \tag{2.4}$$

In order to apply the Hamiltonian formalism to the wave propagation model from Section 2.2, marker particles are placed on the object boundary. At the starting time t_0, a marker particle has the initial coordinates $q_0 = q(t_0)$. The particle then moves during the propagation process, assuming new coordinates $q(t)$ on the evolved wave front at time $t > t_0$. The path λ that the particle takes from the boundary to the new coordinates $q(t)$ minimises the action functional S with

$$S_{q_0,t_0}(q,t) = \int_\lambda \partial\mathcal{L}(q,\dot{q})dt \tag{2.5}$$

The action functional S describes the energy that is associated with the path λ in terms of its Lagrangian $\mathcal{L}$. In order to find a minimiser of the action functional, the associated Euler-Lagrange equation can be used:

$$\frac{\mathrm{d}}{\mathrm{dt}}\frac{\partial\mathcal{L}}{\partial\dot{q}} - \frac{\partial\mathcal{L}}{\partial q} = 0 \tag{2.6}$$

Using the Legendre transformation (equation 2.4) in combination with equation 2.6 and equation 2.3 yields

$$\frac{\partial\mathcal{H}}{\partial q} \overset{(2.4)}{=} -\frac{\partial\mathcal{L}}{\partial q} \overset{(2.6)}{=} -\frac{\mathrm{d}}{\mathrm{dt}}\frac{\partial\mathcal{L}}{\partial\dot{q}} \overset{(2.3)}{=} -\dot{q}$$

This shows that the Euler-Lagrange equation, together with equation 2.3 from the Lagrangian formalism, corresponds to Hamilton's canonical equations 2.1 and 2.2 from the Hamiltonian formalism. It is therefore possible to define the action functional for the propagation process described by the eikonal equation with an action functional in the Lagrangian setting (as in equation 2.5) and then use the Legendre transformation to switch to the Hamiltonian formalism. Then, instead of plugging $\mathcal{L}$ into the Euler-Lagrange equations, Hamilton's canonical equations are considered.

Since the equations from classical mechanics are only used for the very specific case of the grass-fire model, they can be considerably simplified using the information about the wave propagation process. The wave fronts move with constant speed $F(x,y) \equiv 1$ in the direction of the inward normal n. In this setting, the paths of least action become straight lines and the value of the action functional is the Euclidian length, i.e. $S \equiv D$. Furthermore, the Lagrangian for the special case of the grass-fire model is given by

$$\mathcal{L} = \underbrace{\frac{1}{F(x,y)}}_{=1}|\dot{q}| = |\dot{q}| \tag{2.7}$$

Additionally, two well-known equations from physics are used without proof. Huygen's principle states that p and $\dot{q}$ have conjugate directions (equation 2.8) and the Hamilton-Jacobi equation (equation 2.9) further specifies the properties of $p = \frac{\partial S}{\partial q}$ (see Sussman and Wisdom [2001] for more information).

$$p \cdot \dot{q} = 1 \tag{2.8}$$

$$\frac{\partial S}{\partial t} = -\mathcal{H}(q, \frac{\partial S}{\partial q}) \tag{2.9}$$

Using all of the established equations, the Hamiltonian can be described by the gradient field of the distance map:

$$\mathcal{H}(q,p) \overset{(2.4)}{=} p \cdot \dot{q} - \mathcal{L}(q,\dot{q}) \overset{(2.7)}{=} p \cdot \dot{q} - |\dot{q}| \overset{(2.8),(2.9),S\equiv D}{=} 1 - |\nabla D| \tag{2.10}$$

Finally, plugging the Hamiltonian into Hamilton's canonical equations yields

$$\dot{p} \overset{(2.1)}{=} -\frac{\partial}{\partial q}\mathcal{H}(p,q) \overset{(2.10)}{=} -\frac{\partial}{\partial q}(1 - |\nabla D|) = (0,0) \tag{2.11}$$

$$\dot{q} \overset{(2.2)}{=} \frac{\partial}{\partial p}\mathcal{H}(p,q) \overset{(2.10)}{=} \frac{\partial}{\partial p}(1 - |\nabla D|) = -\nabla D \tag{2.12}$$

Note that in the equations above $q = (x, y)$, $p = (S_x, S_y) = (D_x, D_y) = \nabla D$ and the derivatives in respect to q and p are regarded component-wise. Each of the two equations yields an important result.

According to equation 2.12, the path of the marker particles is determined by the gradient vector field of the distance map. This gradient vector field does not change with time, because $\frac{d}{dt}\nabla D = \dot{p} = (0,0)$ follows from equation 2.11. For practical purposes this means that the gradient of the distance map has to be computed only once. In particular, the equations 2.11 and 2.12 can be used to compute the evolved curves for each time t, but this is not necessary for achieving the goal of skeletonisation.

The medial axis points are exactly the sinks of the gradient vector field ∇D. This intuitively fits the definition of the MAT as the location of the ridges of the distance map (Section 2.3): since the gradients of the signed distance map point in the direction of the steepest slope, they point to the distance map's ridges. Thus, the ridges coincide with the sinks of ∇D. Following the same argument, the points of the outer skeleton share their locations with the vector field's sources.

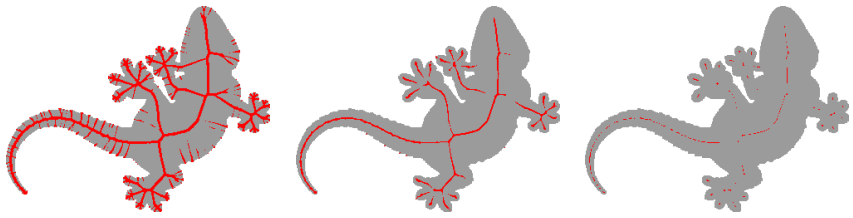
Sinks and sources of a vector field can be identified using the divergence $\mathrm{div}\nabla D$ of the field: for sources the divergence has large positive values, for sinks it has large negative values and for all other points it is close to zero.

In order to compute $\mathrm{div}\nabla D$, the average outward flux is used. In physics, the outward flux of a region R describes how much of a fluid flows out of that region and is determined by the number of sources and sinks in that region, which implicates a connection to $\mathrm{div}\nabla D$. Mathematically, the outward flux is related to divergence by the divergence theorem:

$$\int_R \mathrm{div}(\dot{q})da \equiv \int_{\partial R} < \dot{q}, n > ds$$

Here, the left side of the equation contains an area integral of the divergence on the region R while the right side contains the integral over the outward flux on the boundary of R. Thus, the divergence can be computed for each point of Ω via the average outward flux using the divergence theorem (see equation 2.13) where R is a region containing the point, $A(R)$ its area and n is the outward normal of each point on ∂R:

$$\mathrm{div}\nabla D = \lim_{A(R)\to 0} \frac{\int_{\delta R} < \nabla D, n > \mathrm{d}s}{A(R)} \tag{2.13}$$



(a) threshold -0.1 (b) threshold -0.3 (c) threshold -0.5

Figure 2.6: **Results of flux thresholding.** The pictures above show results of thresholding on a binary picture of a lizard. In (a) the skeleton is not thin and boundary artefacts are visible. (b) contains only a few boundary artefacts, but the skeleton is disconnected and not thin. In the last picture, the skeleton is mostly thin, but large parts are missing.

The skeleton points Σ can be identified by computing the limit of the outward flux as defined above. In the discrete setting, it suffices to compute the outward flux for a small neighbourhood of each object point and to mark all object points that have large negative average flux values, according to Siddiqi et al. [2002].

In practice, the quality of the result is highly dependent on the threshold that is used to identify the MAT points. If the threshold is too confining, the skeleton might not be connected, if it is too large, unwanted additional branches appear. Also, the resulting discrete skeleton is, in general, not thin (see Figure 2.6 for examples).

2.7 Homotopic Thinning

Homotopic thinning can be used as a remedy for the shortcomings of a simple flux-based thresholding algorithm. The basic idea of this method is to reduce an object sequentially to a thin set, removing points only if they do not change the object's topology.

2.7.1 Simple Points

In the 2-D setting, the removal of points is homotopy preserving if it does not disconnect the object and does not create holes in the object [Kong and Rosenfeld, 1989]. An object point that fulfils this conditions is called simple. Test conditions for simple points can be described by the graph that is implied by the configuration of the object neighbours in the 8-neighbourhood $\mathcal{N}_8(p)$ around the point $p \in O_h$ that is tested. The vertices V of the graph are the object neighbours of p.

$$V = \{x \in O_h \mid x \in \mathcal{N}_8(p)\}$$

The edges E between the vertices V are defined for all object points in the 8-neighbourhood $\mathcal{N}_8(p)$ that are either horizontal or vertical neighbours of each other. Diagonal edges are only allowed in the graph if they do not introduce cycles of length three. On a neighbourhood graph as described above, the two conditions for simple points can be formulated as follows.

Figure 2.7: **Neighbourhood graphs.** The images above show examples for neighbourhood graphs. The centre point is blue if it is simple, red if it is not. Vertices and edges of the neighbourhood graph are yellow. The picture in the middle shows a point that may not be removed because its removal would create a hole in the object. Pruning the centre point in the right image would disconnect the object.

Connector Points: If the removal of a point $p \in \Omega_h$ disconnects the object, there must be at least two components of its neighbourhood graph that are not connected by any edges, making p the only connection between them. Thus, for p to be simple, the neighbourhood graph must be connected.

Interior Points: If the removal of a point $p \in \Omega_h$ creates a hole in the object, p must be an interior point, i.e. the edges of the neighbourhood graph of p must form a cycle that encloses p. Per definition, there are no degenerate cycles in the graph. Thus, for p to be simple the graph must not contain any cycles at all.

Bringing the two conditions together yields that p is simple, if its neighbourhood graph is connected and has no cycles. In graph theory, such a graph is referred to as a tree. Thus, p is simple, if its neighbourhood graph is a tree. This condition can be easily checked, since a graph is a tree if and only if the difference between the number of vertices V and the edges E is one:

$$|V| - |E| = 1$$

2.7.2 Thinning Order and Endpoints

Removing only simple points from the shape guarantees homotopy to the original object, but no other properties. In order to preserve topology, the pruning of points from the object can be done in arbitrary order, as long as only simple points are removed. However, the results may vary significantly, as shown in Figure 2.8. An object with no holes in it will even eventually vanish, if simple points are removed successively without any termination criterion.

In order to apply homotopic thinning to skeletonisation algorithms, all thinning-based methods in this work share a common basic approach. The underlying concept is to mark skeleton endpoints that may not be removed, even if they are simple.

Since homotopic thinning does not disconnect the shape, removing all simple points but the skeleton endpoints yields a thin connection line between those endpoints. The location of the line still depends on the order of the thinning process, which means that it might not be centred in the object. This must be ensured by choosing an appropriate thinning order.

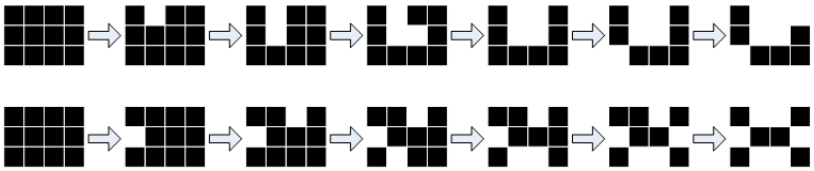


Figure 2.8: **Thinning order in homotopic thinning.** The series of images above displays the effect of different thinning orders. Both rows show the removal of six simple points from a rectangle. In the upper row, simple points are removed in a random order. The thinning process of the lower row can be reached by excluding the skeleton endpoints (the four corner points) from removal and removing simple points in the order of their distance map value.

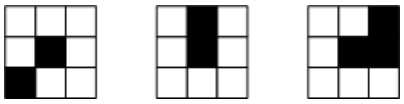


Figure 2.9: **Endpoints in homotopic thinning.** From left to right, the first two images show two possible configurations for endpoints with one object neighbour (either horizontal or diagonal). The third image shows an example for an endpoint with two object neighbours.

For the approach described above, a criterion to identify endpoints is needed. Endpoints in a discrete setting can be easily identified via their 8-neighbourhood. An object point is an endpoint if and only if it is a simple point that has either one single object neighbour or a maximum of two object neighbours, which are 4-adjacent (i.e. two object points are horizontal or vertical neighbours of each other). Examples for the possible configurations are given in Figure 2.9.

2.8 Flux Ordered Thinning

The skeletonisation algorithm given by Siddiqi et al. [2002] is based on the identification of skeleton points by their outward flux (see Section 2.6) combined with the homotopic thinning from Section 2.7.

First, the distance map D of the image and its gradient field ∇D are computed subsequently. In a third step, based on ∇D, a flux map $F : \Omega \to \mathbb{R}$ is created that maps each point p of the image domain to its average outward flux on a small region R with $R \in p$:

$$F(p) = \frac{\int_{\partial R} < \dot{q}, n(q) >}{\text{length}(\partial R)}, \qquad \text{where } n(q) \text{ is the outward normal of } \partial R \text{ in } q$$

In practice, Siddiqi et al. [2002] choose the 8-neighbourhood of p as the region R for the discrete algorithm. The fourth and last stage of the algorithm is the homotopic thinning process, which removes one point of the object in each thinning step, obeying two rules:

1. Remove weak simple points first. Weak simple points are those points that are simple and have positive or negative flux values of small magnitude.

The chosen order for the thinning process is therefore the inverse flux order $<_{\text{iflux}}$ defined by $\forall x, y \in \Omega : x <_{\text{iflux}} y \Leftrightarrow F(x) > F(y)$. With this order, the points that are closest to the skeleton are removed last (except for discretisation or approximation errors).

2. If a simple point is an endpoint, remove it only if it is bigger than a predefined negative flux threshold $\tau < 0$. This condition is used to ensure that skeleton endpoints are not removed and is identical to the simple thresholding approach from Section 2.6.

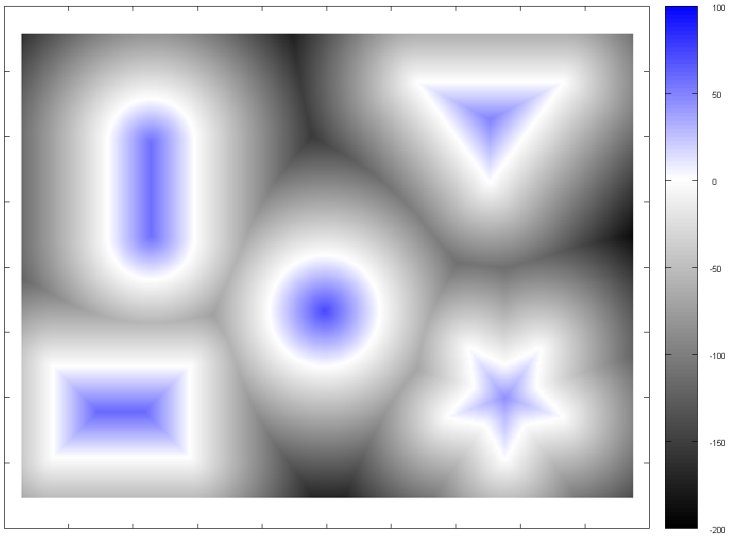
Figure 2.8 shows intermediate results of the four stages of the algorithm. In Section 4.4, details of a discrete algorithm that implements the thinning process above with a heap structure are given. Due to the choice of the thinning order, this method is referred to as flux ordered thinning (FOT). In summary, FOT with a flux threshold $\tau < 0$ can be described by the following steps:

1. **Flux map Computation:**
   (a) Compute distance map D.
   (b) Compute gradient map ∇D.
   (c) Compute flux map F.
2. **Homotopic Thinning:**
   Remove simple points from the object in descending order $<_{\text{iflux}}$. If a simple point p is an endpoint with $F(p) < \tau$, do not remove it.

Due to the two rules defined above, the algorithm reduces the object to a thin skeleton that is homotopic to the original shape. Also, the branches are located in close proximity to the continuous skeleton, because of the flux order. Nevertheless, the output of the algorithm is significantly influenced by the choice of the flux threshold.

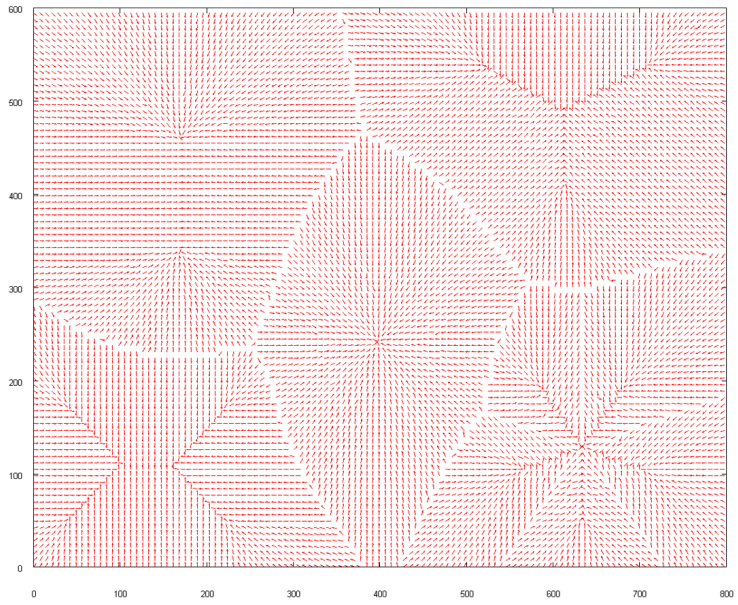
While the most obvious shortcomings of the simple thresholding scheme are eliminated by applying homotopic thinning to the object, the endpoints are still determined by thresholding. If the threshold is too high, additional unwanted endpoints appear and consequently, due to the homotopic thinning process, spurious branches are added to the skeleton. If the threshold is too low, branches (or parts of them) will be missing from the skeleton. Thresholding errors are displayed in Figure 2.6.

(a) Original Image

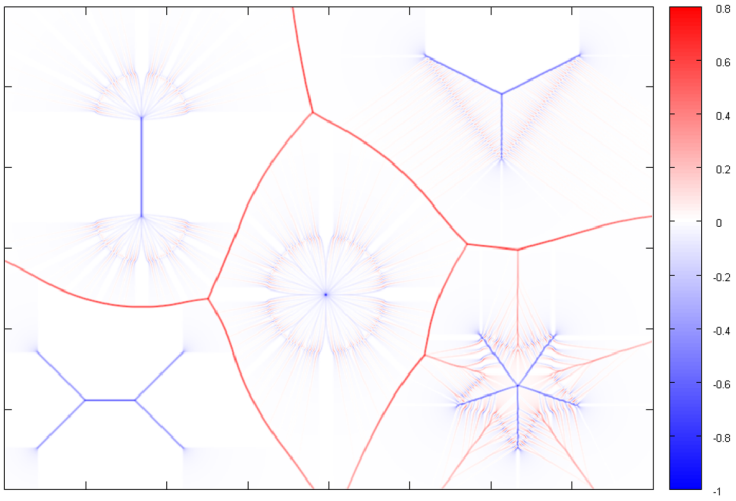


(b) Step 1: Signed Distance Map

Figure 2.10: **Flux-ordered thinning Steps (I).** Image (a) shows a binary image containing five geometric objects. The signed distance map in (b) is the result of the flux-ordered thinning algorithm's first step. Distance map values close to zero are white, negative values are black and positive values are blue.



(a) Step 2: Gradients of the Signed Distance Map



(b) Step 3: Average Outward Flux Map

Figure 2.11: **Flux-ordered thinning Steps (II).** The arrows in image (a) denote the gradients of the signed distance map in Figure 2.10(b). Based on the the gradient field the average outward flux is computed in the third step of the flux-ordered thinning process (Figure (b)).

Chapter 3

Methods

In this chapter, methods are proposed that either improve the computation of Hamilton-Jacobi skeletons or can be used to compare the results of flux-based algorithms to other skeletonisation algorithms.

3.1 Improvements of Flux Ordered Thinning

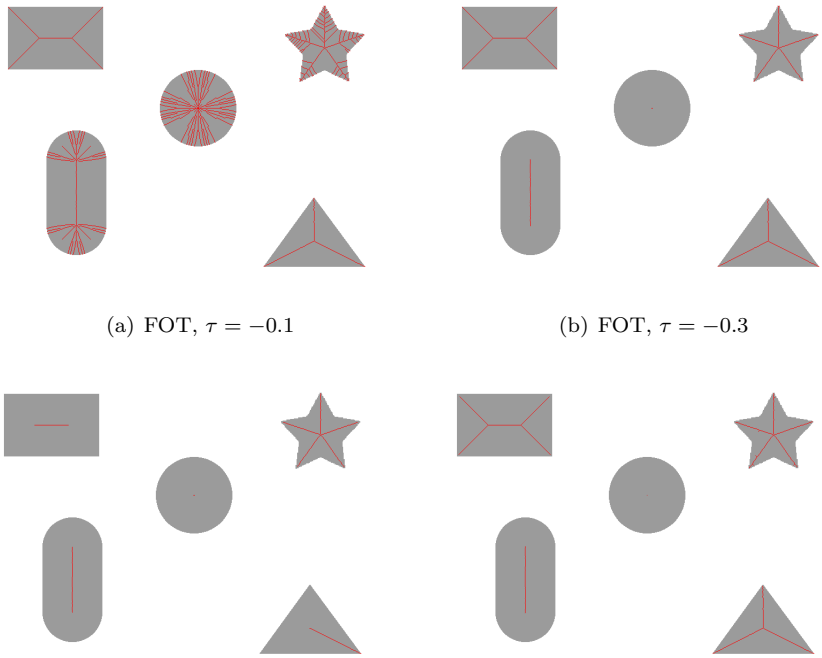
The algorithm of Siddiqi et al. [2002] does not offer many opportunities for significant speed increases except for the method of distance map computation. In Section 3.1.1, faster and more exact alternatives for the distance map algorithm used by Siddiqi et al. [2002] are proposed.

Furthermore, the standard flux ordered thinning algorithm suffers from problems related to the thresholding of the flux map. First and foremost, the threshold τ must be chosen manually and it is not obvious how to find an optimal value for τ. While connectedness and homotopy to the original shape are preserved due to the homotopic thinning, the endpoints of the skeleton are still determined by thresholding. Thus, the chosen threshold τ determines the number and length of skeleton branches and an optimal threshold can vary for different shapes.

A threshold that yields perfect results for a rectangle might give unwanted additional branches for a circle instead of just the centre point. In turn, using a more confining threshold, which gives only the centre point of the circle as its MAT, will yield an incomplete skeleton for other shapes (for examples, see Figure 3.1). In Section 3.1.2 a method for automatic threshold adaption is presented. Naturally, due to the aforementioned thresholding problem, the pruning of endpoints with a global threshold causes problems for images with several shapes in one picture or many boundary perturbations, as can be seen in Figure 3.1. A modified flux ordered thinning algorithm that is designed to counter this effect is presented in Section 3.1.3.

3.1.1 Improved Distance Map Computation

For all algorithms that rely on Euclidian distance maps, the choice of the corresponding algorithm is vital. Since distance transforms are used in many different research fields, there is a plethora of algorithms based on distinct approaches,



(a) FOT, $\tau = -0.1$ (b) FOT, $\tau = -0.3$

(c) FOT, $\tau = -0.5$ (d) FMDT

Figure 3.1: **Thresholding in flux-ordered thinning.** The images above show results of flux-ordered thinning applied to a binary image with several geometrical shapes. In (a), the skeleton of the circle, the star and the rounded rectangle contain many spurious branches. (b) features the best overall results of the three pictures. However, the skeleton of the circle still consists of more than one point (formed by multiple pixels in the unmodified image file) and there are slight skeleton perturbations at the vertices of the star. In (c), the perturbations in the star are gone, but the circle skeleton is still containing more than one point and, more importantly, large parts of the rectangle's and triangle's skeletons are missing. With flux-ordered maximal disc thinning, those problems can be avoided, as (d) displays. However, FMDT adds small spurious branches to the skeleton of the star.

each with varying efficiency and exactness. Siddiqi et al. [2002] cite the so-called D-Euclidian algorithm, a raster scanning algorithm that can be applied to arbitrary dimensions. However, the work of Borgefors [1984], which presented the D-Euclidian algorithm as a refined version of Danielsson's algorithm [Danielsson, 1980], dates back to 1984. Naturally, many new and improved algorithms were introduced since then.

While the D-Euclidian algorithm is easy to implement and can be extended directly to arbitrary dimensions, it also has several drawbacks. Namely, the distance transforms computed with the D-Euclidian method provide a good approximation, but are not exact in all cases [Borgefors, 1984; Fabbri et al., 2008]. Additionally, its speed is not competitive anymore.

In a comparative survey, Fabbri et al. [2008] examined a wide range of distance map algorithms, many of which are both exact and significantly faster than the D-Euclidian method. For this work, only a 2-D distance map is needed and thus, according to Fabbri et al. [2008], Meijster's algorithm [Meijster et al., 2002] or Maurer's method [Maurer et al., 2003] are good choices. Since Meijster's algorithm is as efficient as Maurer's method, but allows a more compact implementation, it is used in this work. Details of the functionality and implementation of Meijster's algorithm can be found in Section 4.1.

For 3-D extensions it is feasible to use the method of Saito and Toriwaki [1994] instead, which can be directly applied to higher dimensions than 2-D and is still superior to the D-Euclidian algorithm in exactness and speed. It has, however, a higher worst case runtime than Meijster's or Maurer's algorithms.

3.1.2 Adaptive Thresholding

In the standard flux-ordered thinning approach of Siddiqi et al. [2002], endpoints of the skeleton are only removed if their flux value is bigger than the threshold, since medial axis points have negative flux values and non-medial points have flux values close to zero. If the magnitude of the threshold is too low, spurious branches are preserved in the thinning process. If the magnitude is too high, parts of the skeleton are lost.

There are several possible strategies for an automatic adaption of the threshold to the image content. In their experiments, Siddiqi et al. [2002] used either fixed thresholds for all pictures or applied quantiles (e.g. choose the threshold such that 75% of object points have a higher flux value). Quantiles however only involve the area of the object in the choice of the threshold, not its shape. For a disc, only one single point, the centre, is a valid skeleton point. In contrast, an object that has the same area and is already a thin set, all object points are also skeleton points. Therefore, adequate quantiles for objects of the same area may vary substantially with shape.

Additionally, boundary artefacts in the discrete setting must be accounted for. As can be observed in the flux map in Figure 2.8, boundaries that are located between the gridlines produce small spurious areas with high flux magnitude near the object's borders (e.g. at the arc of the circle or the diagonal lines of the star). Siddiqi et al. [2002] state that they used boundary interpolation for their experiments without further detailing its effect on the exactness of flux computation. In this work, additional computations for boundary interpolation are avoided. Instead, the boundary is directly extracted from binary pictures and boundary artefacts are treated in the endpoint selection methods instead.

In order to choose the threshold depending on the boundary's shape, the Hamiltonian approach can be combined with other methods for skeleton point detection. Thinning processes in general and flux ordered thinning in particular can be combined with arbitrary other skeletonisation methods, but algorithms that require little additional runtime are preferred. Such additional computations are referred to as secondary MAT detection (SMD) . The SMD does not need to be homotopy preserving.

One possible strategy to adapt the threshold is to start with a negative threshold of low magnitude (i.e. close to zero) and then successively lower the threshold until unwanted branches are removed. In order to identify incorrect

branches, SMD is performed on endpoints that would not be deleted because their flux value is below the negative flux threshold.

If such an endpoint p is identified as a skeleton point by SMD, it is not removed. If p is not a MAT point according to SMD, p is removed from the object and the global flux threshold is set to a new value that is slightly lower than the flux value of p. Therefore, in the following thinning steps, all points with the same flux value as p are removed without further checks. While this approach only requires SMD on endpoints, its output depends on the coherence between the flux order and SMD. The flux order determines which points are endpoints and thus which points influence the overall threshold.

In order to avoid such coherence effects, which are difficult to predict for all possible cases, another approach is pursued in this work. Instead, a global flux threshold is determined before the thinning process starts, similar to the quantile approach of Siddiqi et al. [2002]. First, a SMD is performed on the object to create a preliminary skeleton $\hat{\Sigma}$ that has not to be homotopy preserving or thin and thus can be computed significantly faster than with complex homotopy preserving methods such as FOT.

A threshold for the thinning process is then computed based on the flux values of the skeleton points in $\hat{\Sigma}$. Instead of just considering the area of the object, like in the quantile approach, this threshold adaption incorporates the shape of the object. In this work, the threshold τ is computed as a modified mean of the flux values in $\hat{\Sigma}$. A parameter $\lambda \in [0, 1]$ is introduced that influences how many points of the preliminary skeleton are considered as artefacts resulting from inaccuracies of the SMD and are thus pruned by the subsequent FOT process:

$$\tau = (1 - \lambda)\frac{\sum_{x \in \hat{\Sigma}_h} F(x)}{|\hat{\Sigma}_h|}$$

Due to the introduction of the parameter λ, the algorithm is not entirely independent of input parameters, the dependency on τ is merely replaced by the dependency on λ. However, in contrast to the threshold τ, λ is not primarily used to account for properties of the input shape, but for differences in the boundary artefacts that occur due to the choice of the SMD. This reduces the need for parameter adjustments significantly. In summary, automatic flux adaptation can be performed in three steps:

1. Compute preliminary skeleton $\hat{\Sigma}$ with SMD.
2. Compute $\tau = (1 - \lambda)\frac{\sum_{x \in \hat{\Sigma}_h} F(x)}{|\hat{\Sigma}_h|}$.
3. Compute final skeleton Σ with FOT, using threshold τ.

The resulting algorithm is further referred to as flux-ordered adaptive (FOA) thinning. This method is identical to FOT except for the threshold adaption, which is performed as an additional initial step before the thinning process.

This approach requires a SMD method to decide if an endpoint is a skeleton point. In the following Section 3.1.2, a fast SMD method that is based on maximal disc detection with lookup tables is presented. Note that the choice of the mean as a basis for the adaption of τ is an intuitive one, values for λ were determined experimentally. Other obvious choices to compute a threshold based on $\hat{\Sigma}$ are the median of the flux values or quantiles.

Maximal Disc Algorithm

A simple and fast means for identifying medial axis points in the discrete setting can be constructed by using the maximal disc definition for the skeleton (see Section 2.3):

$$\Sigma = \{x \in O \mid \forall y \in O : B_{D(x)}(x) \not\subset B_{D(y)}(y)\}$$

The radii of the inscribed discs of each object point coincide with their distance map values. Note that a disc $B_{r_1}(x)$ is contained in a disc $B_{r_2}(y)$ if and only if $|x-y|+r_1 \leq r_2$. A naive algorithm based on this definition consists of a check for this criterion for all points in the object domain. If the biggest radius of all inscribed discs of the object is known, the search can be confined to the neighbourhood $B_{r_{max}-r_1}(x)$ because of the equivalence:

$$|x-y|+r_1 \leq r_2 \overset{r_2 \leq r_{max}}{\Leftrightarrow} |x-y| \leq r_{max} - r_1$$

Expanding further on those lines of thought, a lookup table can be precomputed that reduces the problem of finding maximal inscribed discs to a simple comparison of lookup table values with distance map values in a certain neighbourhood. The size and shape of the neighbourhood depends on the maximal local width of the object. Lookup tables for Euclidean discs were provided by Rémy and Thiel [2005]. In Section 4.5 the structure of the lookup tables and an implementation that uses them to detect maximal discs is described.

3.1.3 Flux-Ordered Maximal Disc Thinning

A slight variation of the FOA algorithm from Section 3.1.2 can also be used to eliminate the need for thresholding by removing the flux thresholding entirely from the algorithm, only keeping the flux map as a means of defining a thinning order.

Instead of deciding which endpoints are preserved in the homotopic thinning process via the flux threshold τ, like in the FOT method, only the SMD determines which of the endpoints are considered skeleton points. The steps of the resulting algorithm can be described as follows:

1. **Flux map Computation:**
   (a) Compute distance map D.
   (b) Compute gradient map ∇D.
   (c) Compute flux map F.
2. **Homotopic Thinning:**
   Remove simple points from the object in descending order $<_{\text{iflux}}$. If a simple point is an endpoint that is part of the skeleton according to SMD, do not delete it.

In this work, the maximal disc check with lookup tables by Rémy and Thiel [2005] is used as a SMD method and the resulting algorithm is referred to as flux-ordered maximal disc thinning (FMDT).

FMDT has an exactness advantage over FOT or FOA if it is applied on images containing multiple objects. FOT and FOA both use a global flux threshold

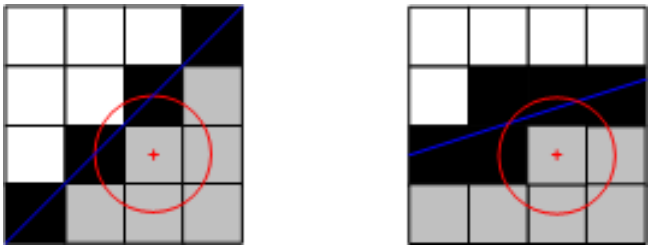


Figure 3.2: **Discretisation errors.** The two images above show discrete versions of straight boundary lines (blue). The inside of the object is marked in grey, while the background is white. In both examples, a point p with minimal distance one to the discrete boundary is marked in red. In the continuous setting, the circle of radius one around p reaches over the boundary line and is thus not an inscribed circle. Therefore, p is not a MAT point in the continuous setting. In the discrete setting, p has minimal distance one to two boundary points and is therefore part of the skeleton due to the equidistant viewpoint.

for the whole image which thus cannot be adapted to different shapes. The SMD in the FMDT method, however, takes into account the local properties of every separate shape in the image. For the same reasons, FMDT is also expected to be slower than FOT or FOA. In FMDT, due to the missing global threshold, more explicit homotopic thinning and SMD checks must be performed.

3.2 Comparison of Skeletonisation Results

One goal of this thesis is to compare skeleton quality of the FOT algorithm to its modified versions FOA and FMDT, as well as to other, less complex algorithms.

Existing publications on skeletonisation usually do not include comparisons to other algorithms and often only present images of the results for visual verification of skeleton quality [e.g. Siddiqi et al., 2002]. One of the reasons for the lack of comparisons with different skeletonisation algorithms or a more detailed analysis might be the inherent difficulty of assessing the quality of discrete skeletons.

All algorithms presented in this work compute skeletons with pixel accuracy and do not use interpolation of the object boundary. Therefore, the resulting skeletons are not desired to be exact in the mathematical sense. Whole branches can be located in between grid lines and thus need to be represented by adjacent pixels. Depending on the algorithm used, this means that the branch is not centred or not thin.

Also, even without noise in a picture, unwanted branches can occur due to discretisation inaccuracies (see Figure 3.2), e.g. for boundary lines that cannot be accurately described with pixel accuracy. In this case, not all points that formally fulfil the mathematical definition of skeleton points are desirable to occur in the skeleton and are regarded as boundary artefacts.

Additionally, depending on the application that the skeletons are intended for, there are priorities beyond the accurate location of skeleton points. For shape recognition for example, the overall structure of the skeleton, i.e. the number and spatial configuration of its branches, as well as the homotopy to

the original shape, are more important than the exact location of every single skeleton point.

For other applications, such as compression, homotopy can be ignored, the only goal is to minimise the number of points that are needed to reconstruct the shape as exactly as possible. The next section establishes quality criteria for discrete skeletons that are relevant for practical applications.

3.2.1 Quality Criteria

In Section 2.4, important mathematical properties of the medial axis in the continuous setting are described. The following criteria are designed to asses how well skeletonisation results fulfil the desired properties in the discrete setting.

Exactness of Reconstruction: A frequently used test criterion for skeletonisation algorithms is the difference between the original object and the shape that is obtained by applying a reconstruction algorithm to the MAF. While the equivalence of the MAF description to the original shape is a defining property, in the discrete setting, under practical conditions, it rivals with other desirable properties such as robustness under rotation and boundary perturbations.
Also, the exactness of reconstruction only yields little to no information about the approximation quality of the skeleton. For instance, a skeleton that is identical to the object itself yields a perfect reconstruction but is, except for rare special cases of thin objects, not the correct MAT.

Skeleton Minimality: In order to assess the approximation quality of the skeleton there are several possibilities. If test images with a ground truth (i.e. exact pixel or subpixel locations of the skeleton points) are given, a direct comparison is possible. However, for large databases with complex shapes, this is usually not the case. Thus, skeleton minimality, i.e. the fraction of object points that are contained in the skeleton, can be examined in combination with the exactness of reconstruction to assess the overall approximation quality.

Skeleton Complexity: In addition to the raw number of skeleton points, i.e. skeleton minimality, also the structure of the skeleton must be considered. The number, connectivity and length of skeleton branches defines the overall complexity of the MAT and is especially important for shape recognition by graph matching.

Thin Set: Determining if a discrete skeleton is a thin set can be done by checking the number and spatial configuration of the skeleton points' 8-neighbours. This is a binary criterion that can be observed directly, regardless of the object's shape and location.

Homotopy: Homotopy to the original shape can be checked for arbitrary shapes. For each connected component in the shape exactly one connected skeleton must exist that contains the same number of holes as the object component.

Translation Invariance: The only change for a discrete object that undergoes translation is its position relative to the image boundary. This should

not change the computed skeleton at all, thus a perfect match between the computed MAF of the original object and its translated version are expected. Translation invariances can be easily checked for arbitrary shapes by applying the inverse translation to the skeleton and comparing it directly to the original skeleton.

Rotation Invariance: For rotations other than in ninety degree steps, discretisation problems arise. The shape does not stay the same, since the boundary curve is located between the grid points and a representation with pixel accuracy is neither exact nor unique and depends on the method that is used to acquire the rotated version of the shape (for example rotation using an image editing software). A perfect match between the computed skeletons of the original and the rotated shapes is desirable, but cannot be expected for all possible shapes and angles. The robustness of the algorithm under rotation is also related to its robustness under boundary noise.

Noise Invariance: Small perturbations on the boundary, such as single object points adjacent to the boundary that are incorrectly attributed to the image background (or vice versa) should not change the skeleton significantly.

For in-depth comparisons of skeletonisation algorithms all of the criteria above should be evaluated separately, judged by the importance of each criterion for the application at hand.

The invariances can be described in terms of the other properties defined above, i.e. the skeleton should differ as little as possible under transformation, concerning exactness of reconstruction, minimality and complexity. Additionally, homotopy and thinness should be preserved.

In order to quantify the three quality criteria above, the set E of erroneous image points is defined, which contains all points of the reconstructed image that differ from the original. Let $\hat{O}_h$ be the object domain of the reconstruction.

$$E = \{x \in \Omega_h | x \in \hat{O}_h \wedge x \notin O_h \vee x \notin \hat{O}_h \wedge x \in O_h\}$$

Note that E contains false negatives (i.e. object points that are missing from the reconstruction) as well as false positives (i.e. background points that are incorrectly denoted as object points during reconstruction). $|E|$ can be used to quantify the exactness of reconstruction.

Analogously, skeleton minimality can be expressed by $|\Sigma|$. The complexity of the skeleton is defined by the number and configuration of its branches. Instead of counting branches and comparing their connectivity explicitly, a straightforward approach to quantify complexity is to count the number of skeleton endpoints and branching points.

Endpoints are defined in Section 2.7.2. Branching points are those locations of the skeleton, where several branches connect. Since the MAT is a thin set, each of those spots coincides with a single point. In the discrete setting, branching points can be located by exploiting properties of their neighbourhood graphs.

A skeleton point is a branching point if and only if its neighbourhood graph contains at least three disconnected components. Each of those components

corresponds to a branch. The set of all endpoints and branching points is denoted P and accordingly, skeleton complexity can be expressed by $|P|$.

Based on $|E|$, $|\Sigma|$ and $|P|$, three quality measures can be defined, which map the skeletonisation results to the interval $[0;1] \subset \mathbb{R}$:

$$e(u,\Sigma) = \frac{\min\{|E|,|O_h|\}}{|O_h|} \quad \text{(exactness of reconstruction)} \tag{3.1}$$

$$m(u,\Sigma) = \frac{\min\{|\Sigma|,|O_h|\}}{|O_h|} \quad \text{(skeleton minimality)} \tag{3.2}$$

$$c(u,\Sigma) = \frac{\min\{|P|,|\Sigma|\}}{|\Sigma|} \quad \text{(skeleton complexity)} \tag{3.3}$$

The measures defined above help to relate the quantities of the respective quality criteria to the properties of the original object. Exactness of reconstruction, as well as minimality, are given as percentages of the total object points, whereas skeleton complexity is measured as the percentage of endpoints and branching points in relation to the total number of skeleton points. Values close to zero are desirable, values close to one denote undesirable results.

However, especially for invariance tests, where the object changes slightly due to discretisation errors or noise, the measures e, m and c will vary, even if the skeleton does not change under transformation. Therefore, the raw numbers $|E|$, $|\Sigma|$ and $|P|$ should be considered in combination with the corresponding measures.

3.2.2 Graph Matching

For invariance tests or images where a ground truth exists (i.e. the exact location of the skeleton is known), another comparison method can be applied. If there is a reference skeleton to which the results can be compared (e.g. the exact result or the skeleton before transformation), the skeletons can be converted to graphs that can be analysed with graph matching.

Graph matching is a core functionality of many skeleton-based shape recognition algorithms and naturally, there are sophisticated algorithms that serve this purpose, such as the shock graph method by Siddiqi et al. [1999].

While the computation of ground truths for a large image base and the application of complex graph matching algorithms for skeleton comparisons is beyond the scope of this work, results of a graph matching approach are presented in Chapter 5.

Due to time constraints, no graph matching method that is tailored to the particular needs of skeleton comparison could be implemented. In order to back up the proposal of graph matching for comparative purposes, *graphdiff*, an approximate graph matcher by Shasha and Wang [2000], is used to demonstrate that even with methods that are not specialised to the problem at hand, skeleton quality can be assessed.

Graphdiff computes a one-to-one mapping of nodes of a query graph Q and a database graph D, using a scoring function that takes node types and edge weights into account. Each possible mapping has a total score and the mapping with the highest score is considered as the best match.

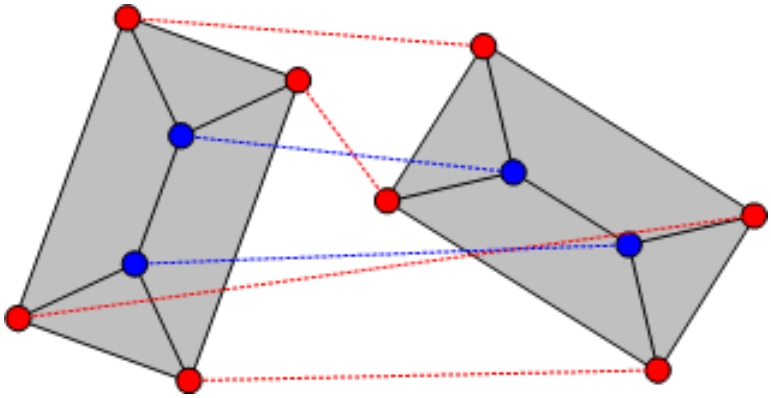


Figure 3.3: **Graph matching.** The two images above show two rectangles and their respective skeleton graphs that are matched by *graphdiff*. Red discs represent graph nodes of the type endpoint, blue discs are branching point nodes. The dotted lines show the one-to-one node mappings that are computed by *graphdiff*.

If i, j are nodes in Q that are mapped to i' and j' in D, respectively, the score for the matches is determined by the matching edges of the nodes. The edge (i, j, w) with weight w matches (i', j', w') if i and i', as well as j and j' are of the same node type. Then, the score is computed as $\min(w/w', w'/w)$.

For the purpose of skeleton matching, in this work, endpoints and branching points are considered as nodes of the graph, defining two different node types. Edges between those nodes are defined, if the corresponding points are connected by skeleton branches. The length of connection branches defines the weight of the corresponding graph edge. An example for such skeleton graphs and an application of *graphdiff* is given in Figure 3.3.

3.2.3 Skeleton Quality as a Minimisation Problem

In this section, the basic foundations for another view on skeleton quality are presented, namely a minimisation approach that is closely related to the quality measures that are defined in the previous sections. Due to the scope of this thesis, the minimisation approach was not thoroughly explored and is only added as an incentive for possibly future work.

As described in the last section, exactness of reconstruction rivals skeleton minimality and complexity. The goal of skeletonisation algorithms is to provide a skeleton that is as minimal and structurally simple as possible, while yielding a reconstruction that is as similar as possible to the original shape.

In other words, the size of the skeleton, the number of special points (branching and endpoints) and the error of reconstruction must be minimised simultaneously. Since different applications prioritise the aforementioned quality criteria differently, they must be individually balanced for each application.

A straightforward approach is to design a cost functional that maps a skeleton to a real value that is minimal for desired skeletonisation results. In order to define such a cost functional, some additional definitions are introduced.

s is an indicator function for skeleton membership and coincides with the output of the algorithms:

$$s : \Omega_h \to \{0, 1\}, s(x) = 1 \Leftrightarrow x \in \Sigma_h$$

The image that is reconstructed from the skeleton is denoted as $R(s, x)$:

$$R(s, x) = 1 \Leftrightarrow \{\exists y \in \Sigma_h : |x - y| < D_h(y)\}$$

The indicator function for end- and branching points is referred to as $P(s, x)$.

With the new definitions, the cost functional $C_u(s)$ can be defined, which maps s to a real value of weighted terms corresponding to the three quality criteria described above. α, β and $\gamma \in \mathbb{R}$ are parameters that can be used to adapt the weighting of the criteria to the demands of a concrete application. Minimisers of C_u are preferred skeletonisation results.

$$C_u(s) = \alpha \sum_{x \in \Omega_h} (u(x) - R(s, x))^2 + \beta \sum_{x \in \Omega_h} s(x)^2 + \gamma \sum_{x \in \Omega_h} P(s, x)^2 \tag{3.4}$$

Note that the cost functional C_u does not account for additional skeleton properties like homotopy to the original shape or thinness and is solely designed to compare skeletons that were computed under additional constraints. The properties of minimisers of C_u vary significantly with changes in the weights α, β and γ. Consider for example an exact, discrete Euclidean disc with radius $r \in \mathbb{N}, r > 1$. For $\alpha = \beta = \gamma = 1$, the minimiser of C_u is the skeleton s' that consists only of the centre point.

$$C_u(s') = \alpha \underbrace{\sum_{x \in \Omega_h} (u(x) - R(s', x))^2}_{=0} + \beta \underbrace{\sum_{x \in \Omega_h} s'(x)^2}_{=1} + \gamma \underbrace{\sum_{x \in \Omega_h} P(s', x)^2}_{=1} = 2$$

Adding additional points to the skeleton while retaining the centre point increases the minimality and complexity parts while the exactness term stays constant. Removing the centre point causes reconstruction errors. The amount of errors can be reduced by introducing more points to the skeleton s'' that does not contain the centre point. However, as soon as s'' consists of at least two distinct points, it holds that $C_u(s'') > 2 = C_u(s')$. Thus, s' is the minimiser of C_u. While the example above describes the desired behaviour of the cost function, in that it has an unique minimiser that coincides with the exact skeleton, it can be easily shown that this behaviour cannot be expected in general.

A thin shape that consists of just two discrete points ($O_h = \{x, y\}$) acts as a counter example. The exact skeleton of the shape is identical to the whole shape: $\Sigma = O_h$. In this case, there are only four possible outputs for a skeletonisation algorithm: either the correct skeleton $\Sigma = \{x, y\}$, a one-point skeleton $\Sigma_x = \{x\}$ or $\Sigma_y = \{y\}$ or an empty skeleton Σ_0. The cost function C_u with $\alpha = 2$, $\beta = \gamma = 1$ yields a cost of 16 for Σ, a cost of 3 for Σ_x and Σ_y, respectively, and a cost of 16 for Σ_0. Thus, the minimiser of C_u does, in this case, yield a skeleton that is different from the exact one. Additionally, the second example demonstrates that minimisers of C_u are not necessarily unique.

There are many possibilities to extend on the basic approach presented above. Besides adding additional constraints to the cost function, nonquadratic error penalisation or the incorporation of the quality measures e, m and c as defined in Section 3.2.1 are potential improvements that could help to eliminate the drawbacks demonstrated by the examples above.

3.2.4 Alternative Algorithms

In order to assess how much the flux component of the FOT method compares to algorithms that are not based on classical mechanics, implementations of non-flux methods are needed. In this work, two alternative algorithms that do not use the outward flux to locate skeleton points are compared to the FOT algorithm and its modifications FOA and FMDT.

First, there is the basic maximal disc (MD) algorithm by Rémy and Thiel [2005]. This algorithm is straightforward to implement and also fairly efficient, since it just consists of a single scan over all object points. For each point in the object, the maximal disc criterion described in Section 3.1.2 is verified and the object point is marked as a skeleton or a background point accordingly. A pseudocode version of this method is given in Section 4.5. However, in general, the MD algorithm does not preserve homotopy and does not yield thin skeletons.

The second algorithm is newly proposed in this work and extends the MD method with homotopic thinning. Object points are pruned layer by layer from the boundary inwards, only retaining points identified as skeleton points via the MD method of Rémy and Thiel [2005] and obeying the homotopy preserving erosion rules.

To define an order for the thinning process, the Euclidian distance to the boundary is used, i.e. all possible points from the Euclidian distance map's level set with the lowest distance are removed before the algorithm moves on to the next level set. This ensures that the connection lines between the skeleton points are centred in the object. The corresponding order is referred to as Euclidean distance order $<_{\text{euclid}}$ and can be defined as:

$$x <_{\text{euclid}} y \Leftrightarrow D(x) < D(y)$$

where D denotes an unsigned Euclidean distance map of the image boundary. The resulting algorithm is referred to as maximal disc thinning (MDT) and is similar to the FMDT approach. The two algorithms differ only in the chosen order for the thinning process.

Chapter 4

Implementation

The implementations of all algorithms and tools used for this work were done with ANSI C. Except for ANSI C's lack of a standard heap structure, the descriptions in this section abstract from the programming language by using pseudocode.

Distance map computation and the basic homotopic thinning rules, as well as the heap implementation, are the same for all algorithms that make use of it. Therefore, the first sections of this chapter deal with common algorithms and data structures.

After the description of common elements, the skeletonisation algorithms are presented. The chapter closes with remarks about boundary conditions and correctness tests.

4.1 Meijster's Algorithm

Meijster's algorithm [Meijster et al., 2002] is an independent scanning algorithm for distance map computation. Independent scanning means that Meijster's algorithm can be parallelised, since it scans rows (or columns) without relying on the information of other rows/columns [see Fabbri et al., 2008].

The distance transform is computed in two phases where phase two relies on the results of phase one. Thus, parallelisation can be applied only for the row/column scans in each phase respectively.

In the following sections it is assumed that Meijster's algorithm computes 1-D distance maps for each row in phase one and uses this information for a minimisation scan per column in phase two. Alternatively, in phase one column scans can be used which implies row scans for phase two.

Phase 1

The first phase consists of computing a 1-D distance map for each separate row R_j ($j \in \{1, \ldots, n\}$). In other words, for each point $x = (i, j)$ in any row R_j, the minimal squared distance to the object points in this row is computed, which means that the intermediate result G can be defined as

$$G(i,j) = \min_{(y,j) \in R_j} \{(i-y)^2 | (i,y) \in O_h\}$$

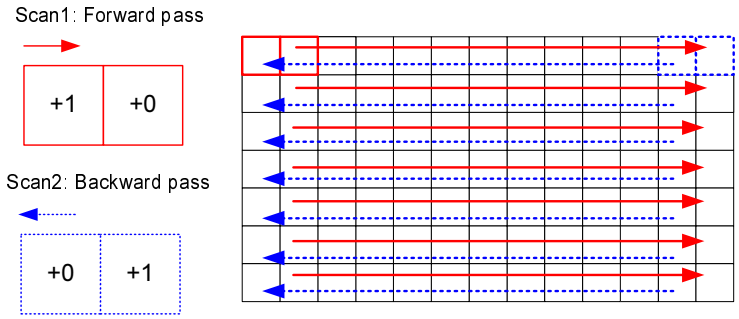


Figure 4.1: **Computation of vertical 1D distance maps in phase 1 of Meijster's algorithm.** On each line, a forward pass (red arrows) and a backward pass (blue arrows) are performed subsequently. The masks represent the minima of neighbouring pixels that are computed. For the forward pass, the left pixel under the mask is assigned the minimum of its own distance value and the value of its right neighbour, increased by one. The backward pass works analogously, but with reversed roles of the mask points.

This first step can be efficiently computed with two scans per row R_j ($j \in \{1, \ldots, n\}$), one forward and one backward scan, as displayed in Figure 4.1.

In the forward scan, each point $p = (i, j) \in R_j$ is assigned its distance to the nearest object point to its left and ∞ if it does not have a left object neighbour. This yields the intermediate result $G_L(i, j)$:

$$G_L(i,j) = \min_{(y,j)\in R_j} \{(i-y)^2 | (i,y) \in O_h \wedge y < i\} \cup \{\infty\}$$

G_L is computed by initialising the first pixel of the row with zero if it belongs to the object and infinity if it is a background point. Then sequentially, from left to right, the other points of the row are assigned a distance value based on two conditions: if the visited point is an object point its distance is set to zero, i.e. $G_L(i, j) = 0$. Otherwise, it is assigned the distance of its left neighbour $n_8(p)$ (see Figure 2.1) increased by one, i.e. $G_L(i, j) = n_8((i, j)) + 1$.

At the end of this pass, all object points in the row have the correct value zero, and all background points that have at least one row neighbour on their left that is an object point were assigned the distance to the closest object point on their left. The remaining points that do not have any object points to their left have distance value infinity.

The second pass, from right to left, is a corrective stage that ensures that all background points have the correct value $G(i, j)$. Computing the distance for each row point to its closest right object neighbour $G_R(i, j)$ allows to derive $G(i, j) = \min\{G_L(i, j), G_R(i, j)\}$ with

$$G_R(i,j) = \min_{(y,j)\in R_j} \{(i-y)^2 | (i,y) \in O_h \wedge y > i\} \cup \{\infty\}$$

In the backward pass, all background points are assigned $G_R(i, j) + 1$, and their own distance value from the forward pass. Thus, object points that have a right object neighbour that is closer to them as a previously discovered left object neighbour, or do not have a left object neighbour, are assigned the new

value $G_R(i,j)$. All other points already had the correct distance after the forward pass and do not need to be changed.

At the end of the second pass, $G(i,j)$ is an exact 1-D distance map for all rows R_j that contain at least one object point. In rows without any object points, all values of the intermediate distance map are ∞.

Algorithm 1 Phase 1 of Meijster's algorithm. Each row is treated separately in a forward and a backward pass. In the forward pass (left to right) each point in the row is assigned the minimal distance to its left object neighbours. The backward pass (right to left) corrects the distances if there are closer object neighbours on the right.

```
// for practical purposes replace ∞ by the maximal possible distance m + n
for j ∈ {1, ..., n} do
  // forward pass (row j: left to right)
  if (1, j) ∈ O_h then
    G(1, j) = 0
  else
    G(1, j) = ∞
  end if
  for i = 2, ..., m do
    if (i, j) ∈ O_h then
      G(i, j) = 0
    else
      G(i, j) = g(i − 1, j) + 1
    end if
  end for
  // backward pass (row j: right to left)
  for i = m − 1, ..., 1 do
    if G(i + 1, j) < G(i, j) then
      G(i, j) = G(i + 1, j) + 1
    end if
  end for
end for
```

Phase 2

In the first phase, only 1-D distance maps for the rows are computed, i.e. only horizontal distances are considered. In order to find the minimal overall distance, vertical distance information of two points $(x,y) \in \Omega_h$ and $(i,j) \in \Omega_h$ must also be considered. The squared Euclidean distance between (x,y) and (i,j) is given by $(x-i)^2 + (y-j)^2$.

Regarding this distance as a minimisation problem in (i,j), the 1-D row distance maps can be used to eliminate i from the equation, since the function G provides horizontal distance minima for each row. Thus, the term $x-i$ can be replaced by $G(x,j)$, i.e. instead of minimising the x-coordinate i, the horizontal distance can be minimised in terms of G and the row number j. The resulting minimisation problem is given by the equation

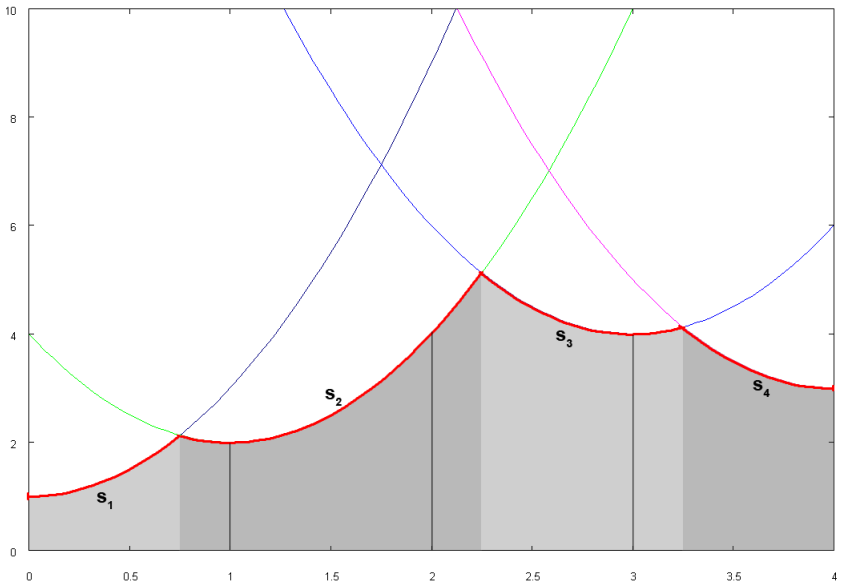


Figure 4.2: **Lower envelope of parabolas.** The image above displays several parabolas as they are encountered in phase two of Meijster's algorithm. The thick red line is the lower envelope of the four parabolas in the image. It consists of the parabola segments $s_1, \ldots, s_4$ with minimal y-value. The corresponding domains $t_1, \ldots, t_4$ are the intervals on the y-axis marked by the alternating shades of grey of the area below the parabola segments. In Meijster's algorithm, the lower envelope only needs to be evaluated at integer points, that are marked with straight vertical black lines in the picture above.

$$D(x,y) = \min_{j \in \{1,\ldots,m\}} \{G(x,j)^2 + (y-j)^2\}, \quad (x,y) \in \Omega_h$$

For a fixed row x and fixed value $j \in \{1, \ldots, m\}$, the graph of the function F_j, with $F_j(y) = F_{x,j}(y) = G(x,j)^2 + (y-j)^2$, $y \in [1,m]$, is a parabola with vertex at $(G(x,j), j)$.

The minimisation problem can be interpreted as finding the right parabola F_j for each point $(x,y) \in R_x$ that provides a minimal value for $F_j(y)$. Thus, a function $F_{\min}$ that consists of all minimal curve segments of the intersecting parabolas, evaluated at integer points, gives the solution to the minimisation problem above, i.e. $D(x,y) = F_{\min}(y)$. Formally, $F_{\min}$ can be defined as $F_{\min}(y) = \min_{j \in \{1,\ldots,m\}}\{F_j(y)\}$ and is called the *lower envelope* of the parabolas F_j.

Figure 4.2 shows the lower envelope of several parabolas. The curve segments are denoted as $s_1, \ldots, s_\mu$, where μ is the number of segments and the corresponding domains on the y-axis are denoted by $t_1 \ldots, t_\mu$.

Which parabola is minimal for a point can be determined by solving the

following equation for $\alpha < \beta$:

$$\begin{aligned} & F_\alpha(y) \leq F_\beta(y) \\ \overset{\alpha<\beta}{\Leftrightarrow} & G(x,\alpha)^2 + (y-\alpha)^2 \leq G(x,\beta)^2 + (y-\beta)^2 \\ \overset{y\in\mathbf{N}}{\Leftrightarrow} & y \leq (\beta^2 - \alpha^2 + G(x,\beta)^2 - G(x,\alpha)^2)\mathrm{div}(2(\beta-\alpha)) \end{aligned}$$

Using this equation together with the fact that the parabolas only need to be evaluated at integer values, allows to compute the segments s_i and the regions t_i efficiently in one forward pass over each column (top to bottom). In a backward pass, only $F_{\min}$ needs to be evaluated to obtain the final distance map value $D(x,y)$. For more details on the aforementioned computations see Algorithm 2 and the original work of Meijster et al. [2002].

Algorithm 2 Phase 2 of Meijster's algorithm. Each column is treated separately in a forward and a backward pass. In the forward pass (top to bottom) the lower envelope $F_{\min}$ of the parabolas $F_j(y) = G(x,j)^2 + (y-j)^2$ is computed. The backward pass allows to derive $D(x,y)$ by evaluating $F_{\min}$.

```
for j from 0 to m − 1 do
  q := 0, s(0) := 0, t(0) := 0
  for u from 1 to n − 1 do {scan 3}
    while q ≥ 0 ∧ (t(q) − s(q))² + g(s(q), j)² > (t(q) − u)² + g(u, j)² do
      q := q − 1
    end while
    if q < 0 then
      q := 0, s(0) := u
    else
      w := 1 + (u² − s(q)² + g(u, j)² − g(s(q), j)²) div (2(u − i))
      if w < m then
        q := q + 1, s(q) := u, t(q) := w
      end if
    end if
  end for
  for u from m − 1 downto 0 do {scan 4}
    dt(u, j) := (u − s(q))² + g(s(q), j)²
    if u = t(g) then
      q := q − 1
    end if
  end for
end for
```

4.2 Homotopic Thinning

The homotopic thinning rules that are described in Section 2.7 are used in several of the implemented algorithms.

Since the rules are only based on the 8-neighbourhoods of each pixel, the implementation is straightforward. In order to identify simple points, the function $isSimple(p)$ computes the vertices V and edges E of the neighbourhood graph

of p. The output can the be determined with the rule $isSimple(p) =$ TRUE $\Leftrightarrow$ $|V| - |E| = 1$. Algorithm 3 displays the function in pseudocode.

Algorithm 3 Function *isSimple(p)*. Compute the number of edges and vertices of the neighbourhood graph of p. If and only if $|V| - |E| = 1$ return TRUE.

Require: $v = 0$ // vertices
Require: $e = 0$ // edges
  if $\neg p \in O_h$ **then**
    return FALSE
  end if
  // horizontal edges (with $n_9(p) = n_1(p)$, $n_0(p) = n_8(p)$)
  for $i \in \{2, 4, 6, 8\}$ **do**
    if $n_i(p) \in O_h$ **then**
      v++
      if $n_{i-1}(p) \in O_h$ **then**
        e++
      end if
      if $n_{i+1}(p) \in O_h$ **then**
        e++
      end if
    end if
  end for
  // horizontal edges (with $n_9(p) = n_1(p)$, $n_0(p) = n_8(p)$)
  for $i \in \{1, 3, 5, 7\}$ **do**
    if $n_i(p) \in O_h$ **then**
      v++
    else
      if $n_{i-1}(p) \in O_h$ **then**
        e++
      end if
      if $n_{i+1}(p) \in O_h$ **then**
        e++
      end if
    end if
  end for
  return $(v - e = 1)$

Endpoints can also be identified by counting the edges. The function *isEndpoint(p)* returns TRUE if and only if $|V| = 1$ or $|V| = 2$ and the two object neighbours of p are 4-adjacent (i.e. they are either horizontal or vertical neighbours (see Algorithm 4)).

4.3 Heap

ANSI C does not supply data structures like heaps oder priority queues. Therefore, a type-independent heap implementation with automatic memory allocation was implemented for use in the homotopic thinning algorithm.

A MIN(MAX)-heap is a binary tree that fulfils the heap property: each parent node is less (greater) or equal than its child nodes. Also, the tree is

Algorithm 4 Function *isEndpoint(p)*. Compute the number of vertices of the neighbourhood graph of p. If and only if $|V| = 1$ or $|V| = 2$ and the two object neighbours of p are 4-adjacent return TRUE.

Require: $v = 0$ // vertices
  if $\neg p \in O_h$ **then**
    return FALSE
  end if
  for $x \in \mathcal{N}_8(p)$ **do**
    v++
  end for
  if $v = 1$ **then**
    return TRUE
  end if
  // $n_9(p) = n_1(p)$, $n_0(p) = n_8(p)$
  if $v = 2 \wedge \forall i, j \in \{1, \ldots, 8\} : \{n_i(p), n_j(p)\} \subset O_h \Rightarrow |i - j| = 1$ **then**
    return TRUE
  end if
  return FALSE

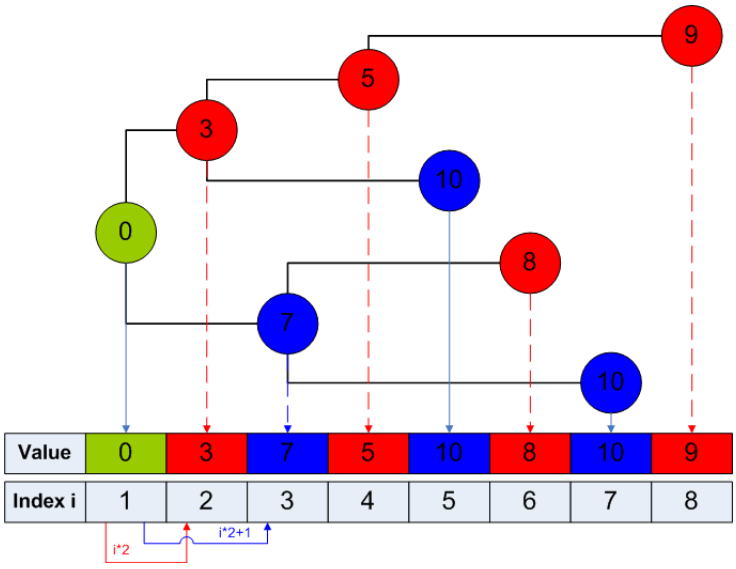


Figure 4.3: **Heap implementation.** Example of a min-heap in the list format: The root node (green) has index 1 and for every node i its right neighbour (red) has index $2i$ and its left neighbor (blue) has index $2i + 1$.

completely filled on all levels but possibly the lowest. An array is used to store a list based binary heap, similar to the concept described by Cormen et al. [2001].

The root node has array index 1. For all nodes with index i the right child is stored in the array field with index $2i$ and the left child in the field $2i + 1$, respectively (also refer to Figure 4.3).

On this list representation of a binary tree structure, the usual operations are defined: *insert*, which adds a new value to the heap and restores the heap property, if necessary, and *pop* (also referred to as *delete-min/max*), which removes the root-node while preserving the heap property.

Additionally, the heap must be of variable size and thus the insert operation is modified to include a memory reallocation for the case that the heap size exceeds the preallocated memory. A pseudocode-version of the operations can be found in Appendix B.

4.4 Flux-Ordered Thinning

The implementations of all basic components that are needed for flux ordered thinning besides the core algorithm are presented in the sections above: a heap structure and rules for homotopic thinning are given, as well as a method for computing the distance map.

In order to compute the outward flux, the gradients ∇D of the distance map must be known. The approximation of ∇D can be done with several means, e.g. finite differences. For this implementation, the Sobel operator [Sobel and Feldman, 1973] was chosen to approximate ∇D.

After computing ∇D, the average outward flux $F(p)$ must be derived for a small region around each point $p \in \Omega_h$. Using the smallest region possible, the 8-neighbourhood $\mathcal{N}_8(p) = \{n_1(p), \ldots, n_8(p)\}$, the outward flux can be written as

$$\sum_{i=1}^{n} \frac{< \nabla D(n_i(p)), N(n_i(p)) >}{8}$$

$N(n_i(p))$ is the outward normal of the the neighbourhood at $n_i(p)$. For the 4-adjacent (horizontal and diagonal) neighbours $\mathcal{N}_4(p)$ the normals are parallel to the grid axis, and the normals for the diagonal neighbours $\mathcal{D}_4(p)$ coincide with the grid axis vectors after a 45 degree rotation.

With the flux values known for all points, the homotopic thinning can be applied to the picture. First, all boundary points from ∂O_h are added to a max-heap that orders the points by their flux value.

Subsequently, each point on the heap is checked for removal. Points are removed if they are simple and no endpoints. Endpoints are only removed if their flux value is less than the flux threshold that is given as a parameter. On removal of a point p, all 8-neighbours $\mathcal{N}_8(p)$ are added to the heap if they became simple points when p was removed.

Algorithm 5 Flux-ordered thinning algorithm. In this pseudocode representation, a precomputed distance map is required (see Section 4.1). Note that boundary treatment is omitted from the pseudocode. The functions *isSimple* and *isEndpoint* are the core methods of homotopic thinning and are defined in Section 4.2

```
Require: dm // precomputed distance map
Require: hp // MIN-heap which orders points by decreasing flux
  // gradient approximation with sobel operator
  for p = (i, j) ∈ Ω_h do
    v.x = (dm[i+1][j-1]+2·dm[i+1][j]+dm[i+1][j+1]
           -dm[i-1][j-1]-2·dm[i-1][j]-dm[i-1][j+1])/8
    v.y = (dm[i-1][j+1]+2·dm[i][j+1]+dm[i+1][j+1]
           -dm[i-1][j-1]-2·dm[i][j-1]-dm[i+1][j-1])/8
    grad[p] = v;
  end for
  // flux computation
  for p = (i, j) ∈ Ω_h do
    // N: list of precomputed outward normals
    flux[p] = ∑_{q∈N_8(p)} ⟨N[q], grad[q]⟩
  end for
  // initialize heap with object boundary
  for p = (i, j) ∈ ∂O_h do
    if isSimple(p) then
      heapInsert(hp, p, flux[p]) // insert p into heap with its flux value
    end if
  end for
  // Perform homotopic thinning
  while hp.size > 0 do
    p = heapPop(hp) // get root of the heap
    if isSimple(p) ∧ (¬isEndpoint(p) ∨ flux[p] > minflux) then
      remove(p) // delete p from object
      for q ∈ N_8(p) do
        // add all simple neighbours of p to the heap
        if isSimple(q) then
          heapInsert(hp, q, flux[q])
        end if
      end for
    end if
  end while
```

4.5 Maximal Disc Algorithm

The maximal disc algorithm uses the precomputed lookup tables provided by Rémy and Thiel [2005]. The lookup tables are given as text files that contain two tables in a predetermined format.

First, there is a vector lookup table (VLU) that contains offset vectors for all neighbours that need to be checked for maximal disc detection, given the maximal local width of the object. The maximum of local width coincides with the maximum of the distance map values in O_h and can thus be easily determined. Each line of the file contains an offset vector and a corresponding maximal radius, i.e. the VLU does not need to be read entirely, only those vectors that correspond to a radius that is smaller or equal to the maximal local width of the object at hand need to be considered.

The second table in a lookup file contains the main lookup table (LUT) that maps each possible radius of an inscribed disc, to the maximal radii of the neighbours defined in the VLU. In order to check if a disc is maximal, for each point $x = c + v$ that can be reached from the centre c of the disc by adding an offset vector v from the VLU, the corresponding value LUT(x) has to be smaller than the disc's radius $D(c)$. Otherwise, the disc around c is not maximal and therefore, c is no skeleton point.

This check is performed on all object points. After visiting each object point once and removing points that fail the maximal disc check, the remaining object points form the skeleton Σ.

Algorithm 6 Maximal disc algorithm. The maximal disc algorithm uses a precomputed lookup table and the distance map to determine, if an inscribed disc is maximal.

```
Require: dm // distance map
Require: vlu // lookup table for displacement vectors
Require: lut // lookup table for distances
  for p = (i, j) ∈ O_h do
    k = 0
    while INOBJ(p) ∧ k < n do
      v = vlu[k]
      for i ∈ {1, ..., 8} do
        // n_{v,i} is v rotated by i · 45°
        if dm[p + n_{v,i}] ≥ lut[dm[p]][k+1] then
          remove(p) // delete p from object
        end if
      end for
      k = k+1
    end while
  end for
```

4.6 Maximal Disc Thinning

The maximal disc thinning algorithm extends the flux ordered thinning algorithm by an endpoint determination phase that uses maximal disc checks as

described in Section 4.5. Flux ordered maximal disc thinning and distance ordered maximal disc thinning are identical except for the chosen order. Therefore, only one pseudocode version is given for both methods (see Algorithm 7).

As the maximal disc algorithm suffers from boundary artefacts for boundaries that cannot be accurately represented in the discrete setting, a straightforward pruning method is used in this work. Spurious skeleton points produced by the maximal disc algorithm are, in general, isolated, i.e. there are no other skeleton points close to them. Therefore, all skeleton points that do not have at least two other skeleton points in their 5×5 square-shaped neighbourhood are pruned before the homotopic thinning process is initiated.

Algorithm 7 Maximal Disc Thinning. The maximal disc thinning algorithm combines a homotopy preserving thinning process with the maximal disc algorithm as a secondary MAT detection. For this pseudocode representation, the results from the maximal disc stage are considered as given, as well as additional computations that are needed for the chosen order.

Require: order // precomputed array of ordering values
Require: hp // MIN-heap which orders points by the chosen order
Require: ep // lookup array for endpoints, computed with maximal disc algorithm
// Prune spurious points, using 5×5 neighborhood $\mathcal{N}_{24}$
for $p \in O_h$ **do**
  if $|\{x \in \mathcal{N}_{24}(p) | ep[x] = \text{TRUE}\}| < 2$ **then**
    ep[p] = FALSE
  end if
end for
// initialize heap with object boundary
for $p = (i, j) \in \partial O_h$ **do**
  if isSimple(p) **then**
    heapInsert(hp, p, order[p]) // insert p into heap with its ordering value
  end if
end for
// Perform homotopic thinning
while hp.size > 0 **do**
  p = heapPop(hp) // get root of the heap
  if isSimple(p) ∧ (¬isEndpoint(p) ∨ ep[p]=FALSE) **then**
    remove(p) // delete p from object
    for $q \in \mathcal{N}_8(p)$ **do**
      // add all simple neighbors of p to the heap
      if isSimple(q) **then**
        heapInsert(hp, q, order[q])
      end if
    end for
  end if
end while

4.7 Adaptive Flux-ordered Thinning

As described in Section 3.1.2, the flux-ordered thinning algorithm with adaptive thresholding (FOA) uses the maximal disc algorithm to compute a preliminary skeleton $\hat{\Sigma}$. Based on $\hat{\Sigma}$ the adapted threshold τ is computed as:

$$\tau = (1 - \lambda)\frac{\sum_{x \in \hat{\Sigma}_h} F(x)}{|\hat{\Sigma}_h|}$$

Using τ, a standard flux-ordered skeletonisation process is applied to the image after the threshold computation. A condensed pseudocode version of FOA is given in Algorithm 8.

Algorithm 8 Flux-ordered thinning with adaptive thresholding uses the maximal disc algorithm to compute preliminary locations of skeleton points. Based on the location of this intermediate MAT, a threshold for flux ordered thinning is computed.

Require: flux // precomputed array of flux values
Require: hp // MIN-heap which orders points by the chosen order
Require: ep // lookup array for endpoints, computed with maximal disc algorithm
  // compute threshold
  sum = 0; total = 0
  for $p \in O_h$ **do**
    if ep[p] = TRUE **then**
      sum = sum + flux[p]; total = total + 1
    end if
  end for
  minflux = $(1 - \lambda) \cdot sum/total$
  // initialize heap with object boundary
  for $p \in \partial O_h$ **do**
    if isSimple(p) **then**
      heapInsert(hp, p, order[p]) // insert p into heap with its ordering value
    end if
  end for
  // Perform homotopic thinning
  while hp.size > 0 **do**
    p = heapPop(hp) // get root of the heap
    if isSimple(p) $\wedge$ ($\neg$isEndpoint(p) $\vee$ flux[p] > minflux) **then**
      remove(p) // delete p from object
      for $q \in \mathcal{N}_8(p)$ **do**
        // add all simple neighbors of p to the heap
        if isSimple(q) **then**
          heapInsert(hp, q, order[q])
        end if
      end for
    end if
  end while

4.8 Boundary Treatment

All pseudo code versions of the algorithms presented in this chapter omit the treatment of the image boundaries for the sake of readability. Since, in this work, it is assumed that objects are fully contained in the input images, the boundary can be arbitrarily extended with background points without losing exactness of the object representation. This property of the input images can be exploited for boundary treatment.

Critical computations that may depend on values outside of the image boundary are the distance map, gradient and flux derivations. Each of those computations relies on the results of the previous one. Consequently, boundary conditions are considered in reverse order of the actual computations.

Flux computation is performed on 8-neighbourhoods, therefore, the corresponding 3×3 mask reaches outside of the image at boundary points. Extending the size of the gradient map by one in each direction suffices to deal with this problem.

The boundary treatment for gradient computation depends on the method that is used, but most of the well-known methods rely on the 8-neighbourhoods. Note that the size of the gradient map must be extended by two in each direction because of the boundary requirements for flux computation. Thus, the size of the distance map must be extended by two in each direction.

Finally, boundary treatment for distance map computation is dependent on the method, but for Meijster's algorithm, again, 8-neighbourhoods are used. Incorporating the extensions that are necessary for the aforementioned computations, the image size must be extended by three in each direction, using additional background points for padding, in order to remove the need for additional boundary checks.

4.9 Correctness of the Implementation

The flux-ordered thinning algorithm has several nontrivial stages that should be tested separately for correctness of implementation. Otherwise, errors become difficult to trace. In this section, some brief commentary on the correctness tests that were applied for this implementation are given.

4.9.1 Correctness of Distance Map Computation

The correctness of any algorithm that computes the Euclidian distance transform can be verified by comparing its results to precomputed exact distance maps.

Using test cases similar to those of Fabbri et al. [2008] (see Figure 4.4), first an exact distance map is computed with a brute force algorithm. Those results are then compared to the results of Meijster's algorithm.

For a correct implementation, a perfect match in all cases is expected.

4.9.2 Correctness of Homotopic Thinning

Homotopic thinning relies on both, a correct heap implementation and correctness of the two functions *isSimple* and *isEndpoint*.

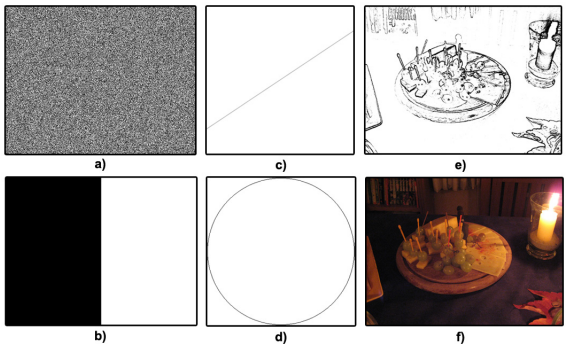


Figure 4.4: **Test images for distance map correctness.** a) Images containing varying amounts of uniform noise (9 pictures, 10%-90%) b) Images with a varying amount of solid fill (9 pictures, 10%-90%). c) Line through image centre with varying rotation (10 pictures, 0°-90°). d) Inscribed circle. e) Outlines of a photograph. f) Original photograph.

Both of those functions can be tested exhaustively by applying them to all possible object membership configurations of the 8-neighborhood of a point.

There are 2^8 different configurations for an 8-neighborhood and the subset of them that yields TRUE for *isSimple* or *isEndpoint* can be determined in a straightforward manner by exploiting symmetries. Thus, a comparison with a precomputed truth table suffices.

Chapter 5

Results and Discussion

In this section, various tests that were conducted with the algorithms described in Chapter 4, are presented. Namely, performance and skeleton quality of flux-ordered thinning (FOT), flux-ordered adaptive thinning (FOA), flux-ordered maximal disc thinning (FMDT), maximal disc detection (MD) and maximal disc thinning (MDT), were compared.

The tests use several different methods and were designed to achieve the following goals:

- Assess the effect of using Meijster's algorithm [Meijster et al., 2002] instead of the D-Euclidean algorithm [Borgefors, 1984] in flux-ordered thinning.
- Compare overall performance of FOT, FOA, FMDT, MD and MDT under practical conditions.
- Assess, in how far skeletonisation results of the different algorithms are invariant under rotation, translation and boundary noise.
- Compare overall quality of the new methods FOA, FMDT and MDT to the original flux-ordered thinning method.
- Check in how far the results obtained with the tools for quality assessment (quality criteria and graph matching, see Chapter 3) correspond to quality assessment by manual visual examination.
- Assess how well *graphdiff* [Shasha and Wang, 2000] in combination with the thinning algorithms (FOT, FOA, FMDT, MDT) can be used for shape recognition (without adaption of *graphdiff* to this task).

The following section introduces the testing environment, including a description of different representative test cases, as well as soft- and hardware specifications.

5.1 Testing Environment

All algorithms were implemented in ANSI C and compiled with gcc 3.4.2. The tests were conducted on Windows XP SP3, using an dual core processor with 2.71 GHz and 3.5 GB RAM.

The test images that are described in the following sections are binary images in the portable grey map (pgm) ascii format with 255 grey tones. Object points have value zero (black), while all other points are regarded as background points.

In addition to the ANSI C versions of the skeletonisation algorithms, several command line tools implementing computational methods for the quality criteria from Section 3.2.1 were used. *Graphdiff*, the program that the graph matching tests are based on, is provided by Shasha and Wang [2000] as a K-file. K is a programming language that is designed for high performance database operations. For a short description of the command line tools, see Appendix A.

5.1.1 Runtime Tests

Runtime optimisation is not a main focus of this work, but nevertheless, the performance of the algorithms was documented for all test cases. Runtime was measured directly by the skeletonisation program, using the C function *clock()*.

Additional command line tools written in ANSI C were used to compute the average runtime of the algorithms for each test set. While runtime was recorded for all tests described in Sections 5.1.2 and 5.1.3, only two test sets are specifically designed for runtime analysis.

Two image collections were created to test runtime behaviour for images of varying size with constant content. Shapes for the runtime experiments were chosen to be natural objects of variable size and complexity.The test set *size_apple* contains rescaled versions of the quadratic binary image of an apple. For the images of this set, the picture height takes the values 50, 100, 200, 400, 800, 1200 and 1800. A second test set, *size_human*, contains rescaled versions of a human silhouette (heights: $100, 300, 500, 800, 1000, 1200, 1400$). For all images, the runtime of each algorithm is recorded.

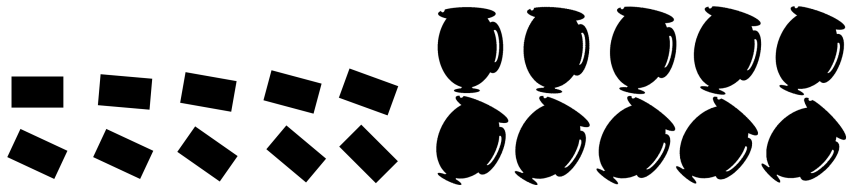
5.1.2 Invariance Tests

Four sets of images were used to determine invariance of the skeletonisation results of different algorithms under Euclidean transformations and under noise, respectively.

All of the four test sets that are described in this section are based on two original images, one of a rectangle with axis-parallel outlines and one of a jar. The two base images are chosen as representatives of different shape classes. The rectangle is a simple geometric object with no holes or curved boundary lines and is, in the original, unmodified image, described exactly by its discrete boundary. In contrast, the jar is a natural and more complex shape that has a hole and cannot be described with full accuracy by a discrete binary picture, because its boundary lines are curved.

The rotation invariance sets *rot_rect* and *rot_jar* contain rotated versions of the aforementioned base images. Rotations were performed in an external image editing software in five degree steps. The image sets contain rotations by 5, 10, 15, 20, 25, 30, 35, 40 and 45 degrees (see Figure 5.1).

Following the same basic approach, the image collections *transl_rect* and *transl_jar* feature the same original objects, a rectangle and a jar, that are translated by nine different two-dimensional translation vectors. Those two image sets are representatives for translation invariance tests.



(a) (b)

Figure 5.1: **Test images for rotation invariance.** The two test sets *rot_rect* and *rot_jar* contain rotated versions of the two respective base images, one of a rectangle with axis-parallel outlines and one of a jar.

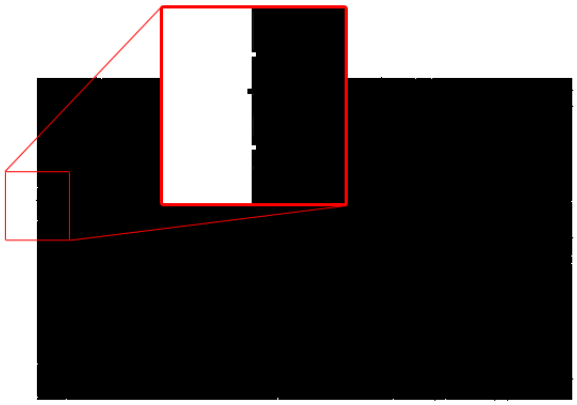


Figure 5.2: **Test image for noise invariance.** Boundary noise is composed by single additional points adjacent to the boundary (black) and points that are removed from the rectangle's boundary. In order to make the small perturbations visible, the region marked with a red rectangle is enlarged.

In order to test invariance of the skeletonisation results under small-scale boundary noise, random noise was added to the boundary of the rectangle and the jar. Boundary noise refers, in this context, to object points that are deleted from the boundary of the original shape (false negatives) and additional object points adjacent to the original's boundary (false positives). The amount of boundary noise reaches from five to thirty pixels (in steps of five). The corresponding picture sets are named *noise_rect* and *noise_jar*.

For all invariance tests, two testing strategies are applied. For the first testing method, the quality measures for exactness of reconstruction, skeleton minimality and skeleton complexity (see Section 3.2.1) are computed and compared.

The second strategy is based on graph matching and uses the approximate graph matcher *graphdiff* [Shasha and Wang, 2000] to compare the skeletons of rotated shapes to the skeleton of the original shape, as described in Section 3.2.2.

5.1.3 Shape Database Tests

In order to test the skeletonisation algorithms under conditions that are closer to practical applications than the invariance test described in Section 5.1.2, the skeletonisation methods are applied to a shape database of 1400 images. The test images of this database were taken from the MPEG-7 core experiment database for shape descriptors (CE-Shape-1) and converted to the input format of the skeletonisation algorithms. CE-Shape-1 is widely used and specifically designed for testing shape descriptors [Latecki et al., 2000].

For all images in the database and for all algorithms, the corresponding medial axis transform (MAT) is computed. The skeleton analysis based on the quality measures from Section 3.2.1. FOT is the only one of the six algorithms that requires a parameter, the flux threshold τ. For the shape database test, several thresholds for FOT were used and τ was chosen such that, for most pictures, spurious branches are pruned.

Since for CE-Shape-1 no exact reference skeletons for comparison exists, graph matching cannot be applied directly, as for the invariance tests. Instead, a small-scale shape recognition experiment is conducted in order to complement the comparison of quality criteria.

For the shape recognition experiment, due to high runtime, only a small subset of the shape database that is displayed in Figure 5.3 is used. In addition to the database *small_db* (see Figure 5.3), five query images are chosen from the full image set *shape_db*, that correspond to similar shapes in *small_db*. Skeletonisation with FOT, FOA, FMDT and MDT is performed on all of the aforementioned images and the resulting MATs are converted to skeleton graphs. For each algorithm separately, matching scores for all pairs containing one query image and one image from *small_db* are computed using *graphdiff*. The image pair with the highest score is considered the query result and represents the database entry that is most similar to the query shape.

5.2 Test Results

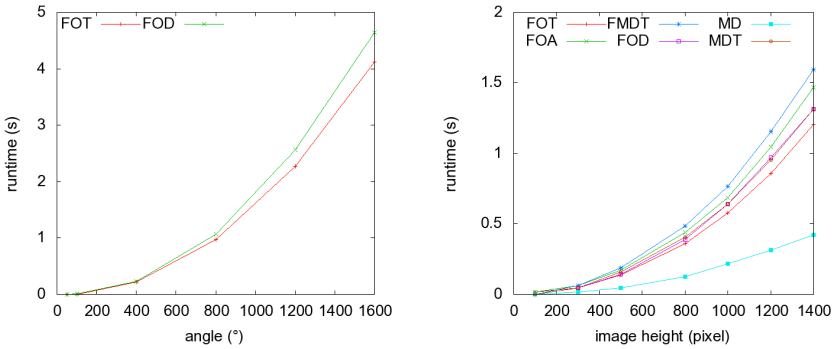
The results of the tests described in Section 5.1 are presented in the following sections. In order to make interpretation of the data easier, the large amount of information that was collected during the test runs is given in detail for specific cases only and in terms of averages for the rest of the image database. Additionally, each type of data (runtime information, quality scores and graph matching results) is treated separately.

All of the following sections deal with one distinct aspect of skeletonisation analysis. Runtime data is presented in detail for invariance tests and runtime tests on images of variable size. The runtime of the algorithms for the large image database is given in terms of averages. The two approaches for determining skeleton quality, namely scores for the quality criteria from Section 3.2.1 and the graph matching method (see Section 3.2.2) do not depend on each other. Therefore, the corresponding data is presented in separate sections.

An additional graph matching data set shows the results of a shape recognition experiment that was conducted in order to display the practical use of *graphdiff* in combination with several different skeletonisation algorithms.

Another section is dedicated to preservation of homotopy and thinness, since

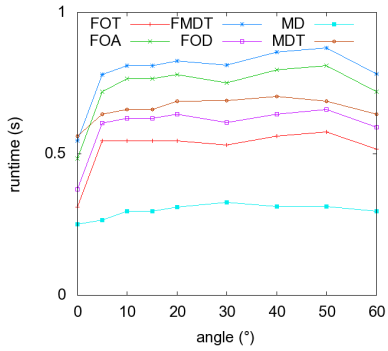
Figure 5.3: **Examples from the Shape Database.** The twenty images above are an excerpt from the shape database *shape_db*, which includes 1400 different shapes. The examples conform with the contents of *small_db*, which is used for the shape recognition experiment.



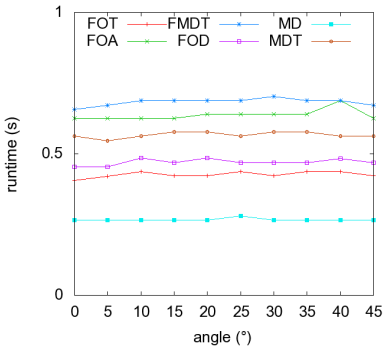
(a) *size_apple* (b) *size_human*

Figure 5.4: **Runtime for varying picture size.**

those are binary criteria that can be checked independently from runtime and the other quality criteria. Interpretations and a discussion of implications that result from the different test runs can be found in Section 5.3.

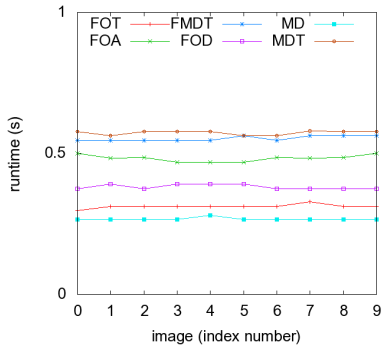


(a) *rot_rect*

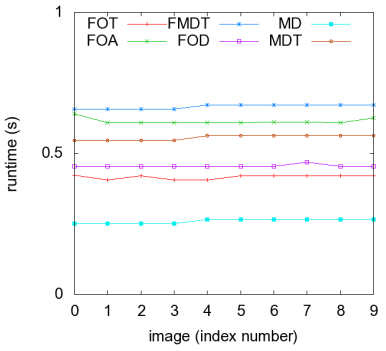


(b) *rot_jar*

Figure 5.5: **Runtime for rotation test sets.**



(a) *transl_rect*



(b) *transl_jar*

Figure 5.6: **Runtime for translation test sets.**

5.2.1 Runtime

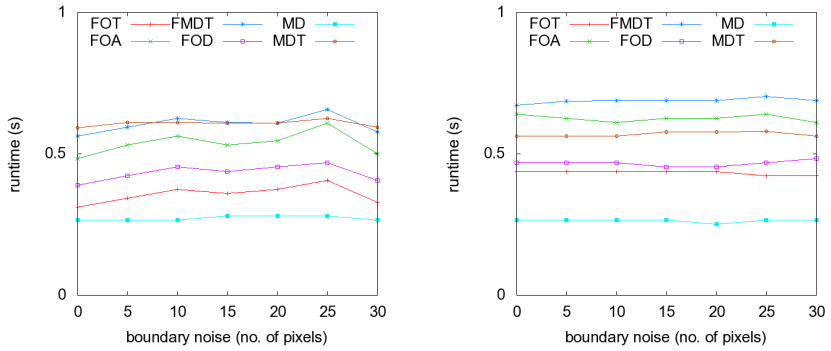
Comparison of Distance Map Algorithms

A direct comparison of the original flux-ordered thinning with the D-Euclidian algorithm [Borgefors, 1984] and the modified flux-ordered thinning algorithm that uses Meijster's algorithm [Meijster et al., 2002] reveals that FOT is consistently faster. The runtime advantages of FOT over FOD are however small for both the invariance test sets (see figures 5.5-5.7) and the images of the shape database (see Table 5.1). More distinct differences become visible for larger image sizes (see Figure 5.4).

Invariance Tests

For the invariance tests, MD is by far the fastest method, followed in ascending order by FOT, FOD, MDT, FOA and FMDT. This runtime order can be observed in figures 5.5-5.7, as well as in Table 5.1 that displays average runtime for the invariance tests.

The only exceptions from this rule are the images that contain a rectangle



(a) *noise_rect* (b) *noise_jar*

Figure 5.7: **Runtime for noise test sets.**

Table 5.1: Runtime averages

test set	FOT	FOA	FOD	FMDT	MD	MDT
rect_rot	0.520556	0.732556	0.597444	0.790111	0.297000	0.658111
rect_transl	0.312100	0.483000	0.381400	0.553300	0.267300	0.573500
jar_transl	0.416600	0.614000	0.454600	0.665600	0.259600	0.556100
jar_noise	0.433143	0.625143	0.466429	0.687714	0.263571	0.569429
shape_db	0.078000	0.140000	0.093000	0.141000	0.062000	0.125000

with axis-parallel outlines, where MDT is the slowest algorithm, falling behind FOA and FMDT speed-wise. Such rectangles with exact discrete representation are contained in the original picture for the invariance test sets *rot_rect* (angle: 0°) and all of the images in *transl_rect*. For the noise invariance tests, MDT and FMDT vary in their position in the runtime order, depending on the location and amount of noise (see Figure 5.7).

Shape Database

The test runs under practical conditions using the shape database *shape_db* (see Table 5.1) confirm the tendencies that are described in Section 5.2.1. On average, MD is the fastest method, followed in ascending order by FOT, FOD, MDT, FOA and FMDT.

For the image sizes of the shape database (varying from roughly 250×250 to 640×640), FOT does not have significant runtime advantages over FOD and is not significantly slower than the MD algorithm. FOA, FMDT and MDT are, on average, roughly two times slower as FOT and MD, respectively.

Table 5.2: Quality averages for *rot_jar*

algorithm	error score	minimality score	complexity score
FOT	319.700012	1249.800049	10.200000
FOA	294.299988	1269.699951	12.400000
FMDT	270.500000	1299.699951	44.700001
MD	213.399994	2606.399902	297.200012
MDT	345.899994	1235.199951	11.000000

Table 5.3: Quality averages for *rot_rect*

algorithm	error score	minimality score	complexity score
FOT	20.777779	929.444458	6.000000
FOA	18.888889	933.777771	6.000000
FMDT	26.000000	948.444458	26.666666
MD	8.555555	2229.888916	286.777771
MDT	32.777779	918.444458	6.000000

5.2.2 Quality Criteria

In this section, the quality criteria of Section 3.2.1, namely exactness of reconstruction, skeleton minimality and skeleton complexity are assessed by using quality scores.

Exactness of reconstruction is measured using the number of erroneous pixels in the image that is reconstructed from the skeletonisation results. Erroneous pixels are image points that deviate from the original image. The corresponding quality score is referred to as the error score (ERR). The minimality score (MIN) is used to assess skeleton minimality and is derived as the total number of skeleton points in the skeletonisation results. Finally, the complexity score (CMP) is computed by counting all branching and endpoints (see Sections 2.7 and 3.2.1) in the MATs that were computed during the test runs.

In order to allow direct comparisons, the scores are not normalised to the quality measures e, m and c described in Section 3.2.1. Quality scores are expressed as a percentage only for the shape database test.

Invariance Tests

Table 5.2 and 5.3 display the averages for both rotation invariance tests, *rot_rect* and *rot_jar*. Detailed information on the quality scores for each test image of *rot_jar* is given in Table 5.7.

For the test set *rot_jar*, the MD method produces the smallest amount of errors, followed in ascending order by FMDT, FOA, FOT and MDT. The exactness order is similar for *rot_rect*, except for FMDT, which ranks second to

Table 5.4: Quality averages for *noise_jar*

algorithm	error score	minimality score	complexity score
FOT	97.571426	1330.571411	11.142858
FOA	92.142860	1328.285767	13.142858
FMDT	59.000000	1319.000000	34.000000
MD	34.714287	2326.857178	285.857147
MDT	129.714279	1265.428589	9.571428

Table 5.5: Quality averages for *noise_rect*

algorithm	error score	minimality score	complexity score
FOT	14.571428	805.000000	6.000000
FOA	16.000000	805.000000	6.000000
FMDT	21.571428	790.142883	6.000000
MD	16.428572	926.142883	27.714285
MDT	34.142857	786.428589	6.000000

last in the exactness order, followed by MDT.

In both test sets, the average minimality score of all algorithms is similar, the only exception being the MD method, which produces roughly two times as much skeleton points as the other algorithms. The skeleton complexity score of FOT, FOA and MDT is similar for all test images. FMDT has a notably higher complexity score than the aforementioned algorithms (roughly four times larger). An even more significant difference in complexity occurs for MD, which produces skeletons with scores that are roughly fifty times higher than those of the other algorithms.

A picture by picture comparison of the scores for *rot_jar* and *rot_rect* reveals that for all algorithms, there is a noteworthy increase in error from the original image (angle: 0°) to the rotated shapes. However, for the rotated shapes, the error stays stable with comparatively small changes from picture to picture.

In Table 5.7, the skeleton complexity score varies only slightly for MDT (10-12), FOA (10-14) and FOT (8-14). In contrast, the CMP score of FMDT ranges from 32 to 62 and every skeleton has a distinct number of end and branching points, no two scores are equal. Skeleton complexity for MD is consistently higher than for all other algorithms and fluctuates more.

For the translation tests, no separate data is given, since all scores are identical to the original shape for all translated objects and for all algorithms. The score values for the translated images coincide with the scores for the unrotated objects from the rotation test sets. The noise invariance tests (see tables 5.4, 5.5 and 5.8) reveal tendencies similar to the rotation invariance tests.

Table 5.6: Quality averages for *shape_db*

algorithm	error score	minimality score	complexity score
FOT	346.423218	995.720276	31.683489
FOA	322.351837	1023.875244	31.985580
FMDT	145.362656	1070.573120	48.845711
MD	90.453499	1878.093750	229.994949
MDT	299.688538	979.944458	31.224945

Database Tests

For the tests on the test set *shape_db*, only averages for the quality scores exactness of reconstruction, skeleton minimality and skeleton complexity are presented in this work, due to the large amount of data (more than 16.000 quality scores). The averaged quality scores can be found in Table 5.6.

The maximal disc algorithm produces, on average, the most exact reconstructions, since its average reconstruction error is the lowest (see Table 5.6). Skeletons computed by the MD method are also significantly larger and more complex, as the skeleton minimality and skeleton complexity scores suggest.

FMDT yields skeletons with a better reconstruction quality than all other homotopy preserving algorithms. Again, similar to the MD method, the advantage in exactness comes at the cost of higher minimality and complexity scores. The skeletonisation results of FOT, FOA and MDT have very similar complexity. Additionally, the minimality score varies only slightly with FOA having the largest skeletons. However, FOA skeletons are also more exact than those produced by FOT.

In Table 5.6 and Figure 5.8 it can be observed that the MDT algorithm, the only homotopy preserving algorithm in the selection that does not rely on a flux order, has the lowest (i.e. the best) values for all three quality scores. Figure 5.8 displays the quality scores in terms of percentages, relative to the total number of object points or, for skeleton complexity, relative to the total number of skeleton points. This method of representation shows that all algorithms produce less than 1% erroneous pixels, relative to the total number of object points.

5.2.3 Graph Matching

The results in this section were obtained by performing graph matching as described in Section 3.2.2, using the approximate graph matcher *graphdiff* [Shasha and Wang, 2000]. *Graphdiff* computes the best match for a query graph and a database of graphs that results in the output of a node mapping and a score value. The score represents the quality of the match as the quantity of matched nodes, normalised in relation to the total number of nodes.

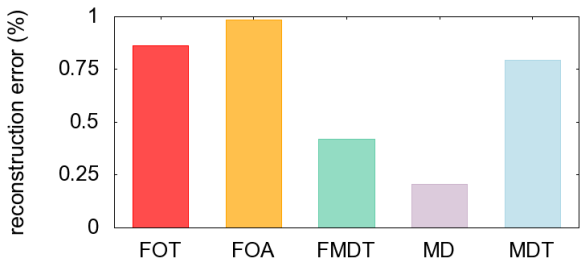
Since the exact node mappings are irrelevant for skeleton quality, in this section only the scores are presented. The highest possible score is one, which corresponds to a perfect match, while the lowest possible score is zero (no match found).

Table 5.7: Quality Criteria for Invariance test *rot_jar*

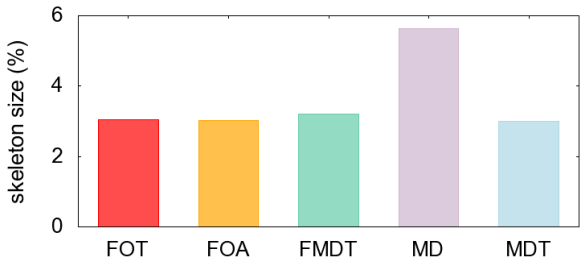
Angle (°)	FOT			FOA			FMDT			MD			MDT		
	ERR	MIN	CMP	ERR	MIN	CMP	ERR	MIN	CMP	ERR	MIN	CMP	ERR	MIN	CMP
0	60	1285	10	57	1294	12	30	1316	32	10	2286	284	86	1266	10
5	345	1285	10	339	1291	10	272	1354	54	185	2572	264	352	1273	12
10	355	1270	8	345	1282	10	295	1337	52	233	2545	330	393	1276	10
15	371	1270	10	337	1304	14	329	1320	41	292	2617	311	411	1257	12
20	341	1258	10	314	1284	14	303	1309	48	269	2632	278	371	1254	10
25	340	1246	10	320	1255	12	294	1285	39	238	2687	320	322	1243	12
30	348	1237	8	294	1274	12	278	1308	62	236	2698	307	348	1228	10
35	293	1253	14	288	1259	12	279	1272	49	224	2678	312	346	1206	12
40	353	1207	12	329	1216	14	295	1254	34	215	2646	269	398	1172	12
45	391	1187	10	320	1238	14	330	1242	36	232	2703	297	432	1177	10

Table 5.8: Quality Criteria for Invariance test *noise_jar*

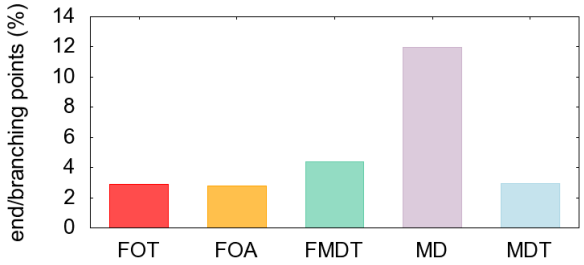
	FOT			FOA			FMDT			MD			MDT		
noise	ERR	MIN	CMP	ERR	MIN	CMP	ERR	MIN	CMP	ERR	MIN	CMP	ERR	MIN	CMP
0	60	1285	10	57	1294	12	30	1316	32	10	2286	284	86	1266	10
5	65	1285	10	57	1294	12	34	1318	34	13	2317	284	102	1261	8
10	72	1285	10	64	1294	12	42	1314	30	19	2296	286	96	1266	8
15	107	1285	10	102	1294	12	73	1321	36	44	2323	289	154	1267	10
20	136	1385	12	104	1316	14	78	1318	32	51	2359	290	163	1263	11
25	131	1285	10	145	1294	12	73	1330	42	55	2326	286	158	1268	10
30	112	1504	16	116	1512	18	83	1316	32	51	2381	282	149	1267	10



(a) percental exactness of reconstruction



(b) percental skeleton minimiality



(c) percental skeleton complexity

Figure 5.8: **Percental database results.** Exactness of reconstruction and skeleton minimality are given in terms of percentages of the total object points. Skeleton complexity is given as a percentage of the total skeleton points.

Invariance Tests

Matching scores for the application of *graphdiff* on the test sets *rot_rect* and *rot_jar* can be found in Table 5.9 and Table 5.10. FOT, FOA and MDT feature similar graph matching results, with high matching scores that vary between 0.64 and 0.95 for *rot_rect*. While those algorithms are not entirely rotation invariant, the high scores suggest that the configuration of the branches stays the same, only branch length varies. In contrast, FMDT has consistently lower matching scores and for one case, no match can be found by *graphdiff*, at all. For *rot_jar*, the tendencies are similar, but overall the scores are lower for all algorithms.

For the translation tests, both for the rectangle and the jar test sets, *graphdiff*

Table 5.9: Graph Matching: Invariance test *rot_rect*

rotation angle (°)	FOT	FOA	FMDT	MDT
5	0.9453704	0.7430553	0.3776216	0.7300299
10	0.8947018	0.8916416	0.06075862	0.8883656
15	0.6539656	0.6566482	0.00000000	0.6409290
20	0.8238728	0.8130085	0.54800000	0.8136446
25	0.7966479	0.7865378	0.79441380	0.7826887
30	0.7680811	0.7629597	0.34068970	0.7618113
35	0.7430201	0.7402330	0.48583010	0.7324194
45	0.7477580	0.7470745	0.74984300	0.7280779

Table 5.10: Graph Matching: Invariance test *rot_jar*

rotation angle (°)	FOT	FOA	FMDT	MDT
5	0.8696369	0.7364974	0.08883411	0.4898076
10	0.5578834	0.6992813	0.1154723	0.6859648
15	0.8602323	0.6960153	0.09060973	0.5542712
20	0.6298299	0.7153714	0.1313318	0.5814962
25	0.9208212	0.7721554	0.1762637	0.5362848
30	0.5934053	0.6839552	0.139588	0.6249221
35	0.2755947	0.4672229	0.06131341	0.5482262
40	0.5928457	0.5477218	0.1057576	0.5772137
45	0.4027719	0.4217458	0.1634385	0.4868252

reports a matching score of 1.0 for all tests. This means, that perfect matches are found for all combinations of test images and algorithms and thus, all algorithms are fully invariant under translation.

The noise invariance tests show similar results as the rotation invariance tests. FMDT has poor scores, while the results for FOT, FOA and MDT are significantly higher. However, a bigger gap in the matching scores can be observed between FOT/FOA and MDT. FOT and FOA both feature several perfect matches, while MDT produces consistently lower scores.

Database Queries

The shape recognition experiment that was conducted with the set of twenty database images from *small_db* and five query images yielded mixed results, depending on which algorithm was used to compute the skeletons. The query

Table 5.11: Graph Matching: Invariance test *noise_jar*

noise pixels	FOT	FOA	FMDT	MDT
5	1.0000000	1.0000000	0.8284314	0.4949118
10	1.0000000	0.9149658	0.6065511	0.8498562
15	1.0000000	1.0000000	0.9108073	0.9992832
20	0.7029765	0.7266314	0.2187128	0.6371714
25	1.0000000	1.0000000	0.2104022	0.9778802
30	0.4024742	0.6244489	0.4311077	0.5545693

Table 5.12: Graph Matching: Database query results

	FOT		FOA		FMDT		MDT	
query	match	score	match	score	match	score	match	score
bone-13 	cup-5 	0.534	bone-14 	0.686	beetle-9 	0.244	car-16 	0.384
cup-4 	cup-5 	0.506	cup-5 	0.469	phone-17 	0.327	phone-2 	0.380
fly-10 	fly-8 	0.134	bat-15 	0.134	turtle-5 	0.156	bat-4 	0.167
bat-14 	bat-4 	0.180	bat-4 	0.124	jar-8 	0.140	bat-15 	0.202
apple-1 	apple-3 	0.567	apple-3 	0.608	apple-2 	0.340	apple-2 	0.548

images and the resulting best match from the database, along with its *graphdiff* score are displayed in Table 5.12.

Queries based on FOT and FOA results yield a correct corresponding shape from the database in four of five cases. Shape recognition with MDT succeeds in two of five cases, while FMDT produces only one correct result.

The matching scores for MDT and FMDT are notably lower than the scores

for FOT and FOA. Another tendency that can be observed is, that for more complex skeletons (like the fly's skeleton) the matching score is lower than for simple skeletons (like the apple's skeleton).

5.2.4 Homotopy and Thinness

Homotopy and thinness were checked manually for all pictures in the various test sets. As expected, all algorithms that use the homotopy preserving thinning rules (FOT, FOA, FOD, FMDT, MDT) produce skeletons that are both thin and homotopic to the original shapes. All skeletons computed by the aforementioned algorithms that contain disconnected components correspond to original images that also contain multiple disconnected shapes.

The maximal disc algorithm, in contrast, ensures neither homotopy to the original shape, nor thinness, which is confirmed by the test runs. Virtually all skeletons computed with the MD method are not thin and feature several disconnected components.

5.3 Discussion

In the following subsections, the results of runtime and skeleton analysis of Section 5.2 are interpreted.

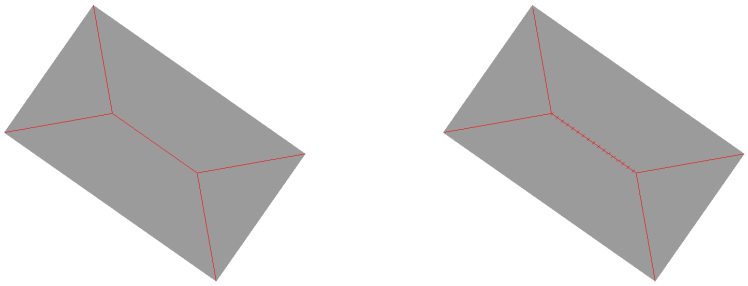
5.3.1 Runtime Discussion

The results on the image database of the MPEG7 Core Experiment revealed some unexpected tendencies: due to the relatively small image size, Meijster's algorithm does not display significant performance advantages in comparison to the D-Euclidean algorithm. For the same reasons, FOT is only marginally slower than the MD method, though MD does not use homotopic thinning rules, but only performs one single check on a shape-dependent neighbourhood of each image point.

Overall, the runtime of the newly proposed methods FOA, FMDT and MDT is significantly higher than the one of FOT. This however follows directly from the secondary MAT detection that is applied in all of the three methods. FOA, FMDT and MDT perform maximal disc detection on the image as a preliminary computation step. Thus, the runtime of MD is added to the runtime of the thinning method that is also applied by FOT. The missing flux computation of the MDT algorithm explains its advantages over FMDT, but is otherwise negligible.

5.3.2 Quality Score Discussion

The quality scores based on the criteria from Section 3.2.1 turn out to provide a much deeper insight into skeleton quality than just the number of reconstruction errors or visual examination. Comparing the minimality and complexity scores of FOT, FOA or MDT with the results of FMDT shows clearly that FMDT produces qualitatively inferior skeletons, though it features very competitive exactness values. The high complexity and skeleton size suggests that there are spurious branches. This fact corresponds with visual examination of the results



(a) FOT (b) FMDT

Figure 5.9: **Spurious branches in FMDT results.** The skeleton on the right features many spurious branches extending from the branch in the middle. Those unwanted branches only appear for FMDT, the expected result is displayed on the left (skeletonisation result of FOT).

(see Figure 5.9). Concerning MD, the CMP and MIN scores are significantly higher than all other algorithms in the field, which results from the lack of homotopy preservation of the method.

Interpreting the scores for the invariance tests, all algorithms are clearly fully invariant under translation, while none of them are entirely invariant under rotation or boundary noise. The scores of FOA, FOT and MDT are mostly stable for the rotation and noise sets which suggests minor differences in the skeletons, but an overall similar structure. However, for FMDT and MDT, the highly variable CMP scores imply that skeleton structure changes significantly for those algorithms. Again, this observations can be confirmed by manual visual examination of the results.

While quality scores prove to be a valid tool for skeleton comparison, this method requires a manual comparison of three individual scores, leaving room for different interpretations of the data. For practical applications, priorities of the different scores might shift, as the following examples demonstrate: for compression, MIN is the most important score, followed by ERR, while CMP is insignificant. Since the shape should be represented by minimal number of skeleton points, while exactness of reconstruction is as high as possible, skeleton complexity is irrelevant. For shape recognition with graph matching, CMP is by far the most important score, since CMP assesses the configuration of branches in the skeletons and therefore also gives insight on the structure of the resulting skeleton graphs.

5.3.3 Graph Matching Discussion

Graph matching turns out to be an effective method for invariance tests. *Graphdiff* provides a single matching score for each comparison and thus represents the similarity of two skeletons much more efficiently and clearly than the quality scores. The results are similar for both methods: full translation invariance is suggested by perfect matches for the translation invariance tests. Rotation invariance and noise invariance are reasonably good for FOT, FOA and MDT. FMDT and MD achieve very low graph matching scores, which suggests poor

invariance under rotation and boundary noise. Those tendencies correspond to the visually determined observations.

However, the implementation of graph matching in this thesis also features a significant weak point. Only the configuration and length of branches is incorporated in the skeleton graphs that are compared with *graphdiff*. The MAF values are not considered in this model. This means, that even a perfect match of two skeleton graphs does not guarantee that the corresponding shapes are identical (see Figure 2.3), since the MAT alone is not a unique shape descriptor.

While the shape recognition experiment that was conducted on *small_db* is not representative for the full capacities of *graphdiff* for skeleton-based shape recognition, due to the small size of the database and the limited number of queries that were analysed, the results reinforce the validity of the quality criteria for skeletons. The experiment shows that skeletons with spurious branches, like the ones of the FMDT method, are significantly less well-suited for shape recognition than skeletons with a less complex structure (FOT, FOA).

5.3.4 Comparison of Quality Assessment Methods

Both, quality scores (Section 3.2.1) and graph matching (Section 3.2.2), prove to be valid tools for skeleton analysis. The implications for skeleton quality that result from the application of both methods correspond to the manual, visual analysis of skeleton quality and can thus add credibility to prevalent methods of skeleton analysis that are based solely on visual confirmation and reconstruction quality.

The two methods offer individual advantages and drawbacks. Graph matching produces one single matching score for the analysed skeleton, which is a very clear way of describing skeleton quality. This method however requires a reference skeleton graph. For invariance tests, such a reference skeleton is provided by the MAT of the base image, but for database tests, for each image, the exact skeleton must be known in order to build a skeleton graph from it. Additionally, the conversion of skeletons to skeleton graphs is non-trivial and, in the implementation that was used in this thesis, has significant weak points (missing incorporation of MAF values).

In contrast to graph matching, quality scores feature three different scores that must be interpreted holistically. This is both an advantage over graph matching and a drawback. On one hand, each score gives information on a single aspect of skeleton quality (exactness, minimality and complexity) that cannot be extracted from the graph matching score, but on the other hand, combining the scores into one single quality score is non-trivial, as demonstrated in Section 3.2.3. A clear advantage of the quality scores over graph matching is however, that no reference skeleton is needed. The scores can be computed from an arbitrary MAF and the original picture without the need of any additional information.

Chapter 6

Conclusions

6.1 Summarising Remarks

In this work, the flux-ordered thinning (FOT) method based on Hamiltonian mechanics, originally proposed by Siddiqi et al. [2002], was used as a basis for new skeletonisation algorithms. FOT identifies skeleton points by applying a threshold τ to the outward flux values of the points in the image domain. Points with a negative flux value of high magnitude coincide with the sinks of the distance map's gradient vector field and thus with the skeleton points (see Chapter 2). In a homotopic thinning process, points are removed from the object, in inverse order of their flux value, until a thin discrete set remains that has the same major topological features as the original shape. Endpoints of this thin set of lines and arcs are removed based on the aforementioned threshold criterion for MAT points.

Two general methods that remove the need for parameter adjustment in the FOT method by removing the need for the flux threshold τ were proposed: flux-ordered adaptive thinning (FOA), that uses a secondary MAT detection (SMD) method to adapt the flux threshold to the input shape and flux-ordered maximal disc thinning (FMDT). FMDT uses a SMD to replace the threshold criterion entirely. For a concrete implementation of FOA and FMDT, a maximal disc detection (MD) algorithm by Rémy and Thiel [2005] was used as the SMD. An additional thinning algorithm was proposed, which does not rely on the outward flux: maximal disc thinning (MDT) uses the distance to the shape boundary to define the thinning order and relies on the MD method to identify skeleton endpoints. For all of the new algorithms, an improved method of distance map computation was applied, Meijster's algorithm [Meijster et al., 2002].

In order to compare the new methods to the original FOT method, two distinct means of quality analysis for discrete skeletons were proposed. Several quality criteria were introduced, and three primary criteria were used to define quality measures: exactness of reconstruction (the deviation of the reconstructed shape from the original), skeleton minimality (the number of skeleton points) and skeleton complexity (number of branching and endpoints of the skeleton). As a second method for quality assessment, graph matching was proposed. Skeletons are transformed into graphs by using endpoints and branching points as nodes, connecting them with edges if they are connected by branches

in the skeleton. The branch lengths are used as edge weights. An approximate graph matcher, *graphdiff* [Shasha and Wang, 2000] was used for an implementation of graph matching.

A series of tests was conducted, both to assess performance and output quality of the newly proposed algorithms and to compare them to the original FOT method, and in order to test the practical use of the proposed quality assessment methods. The tests relied on image sets that were specifically designed to check invariance of the algorithms under rotation, translation and boundary noise. Additionally, tests were conducted on a widely used test database for shape descriptors, the MPEG7 core experiment CE-Shape-1 database [Latecki et al., 2000].

The tests confirmed that Meijster's algorithm offers performance improvements over the distance map computation in the original FOT algorithm. Those advantages are however smaller than expected and show primarily for images of large size. The new methods FOA and MDT yielded results comparable or superior to FOT without the need for parameter adjustments. Both algorithms are however slower than FOT, due to the additional runtime of the SMD. FMDT needs further work in order to be competitive to the other algorithms in the field, because its skeletons feature many spurious branches.

Quality scores and graph matching both yielded solid skeleton analysis results that correspond to manual visual observations and can be used as a basis for more elaborate methods for comparisons of skeletonisation methods. Several proposals for improvements of the quality measures and the skeletonisation algorithms described above are given in the following section.

6.2 Outlook

The modified versions of the flux-ordered thinning algorithm, namely FOA and FMDT offer ample opportunities for improvements. While pruning of spurious branches works well on FOA, FMDT needs more elaborate treatment of boundary artefacts. Since both FOA, FMDT and, additionally, MDT are general methods that are not limited to a specific choice of the secondary MAT detection algorithm, the application of SMDs other than the maximal disc method [Rémy and Thiel, 2005] can be used to improve performance and output quality of the aforementioned algorithms.

Considering pruning methods for spurious branches, the minimisation approach that was only briefly discussed in Section 3.2.3 may harbour the potential for the creation of new pruning algorithms. By further refining the cost function and finding methods of approximating the cost change for removing branches, correction stages could be designed. In order to base full skeletonisation algorithms on this approach, the cost function would have to undergo major changes that make minimisers unique or significantly reduce the amount of possible candidates.

The quality criteria introduced in Section 3.2.1 yield valid results for the analysis of skeleton quality, but do not incorporate all of the skeleton's important properties, such as the MAF values. Adding further quality measures to the existing ones and finding a way to combine them into a robust general measure for skeleton quality, similar to the graph matching score, could be the focus of future work.

Finally, graph matching as a comparison tool offers many possibilities for further research. The application of the *graphdiff* method by Shasha and Wang [2000] to skeleton analysis and shape recognition can be further improved. Both more complex edge weights and node types could be used to enhance the exactness of skeleton comparison, for example by incorporating the medial axis function's value of endpoints and branching points into the node type, as well as the number of branches. Naturally, not only the input graphs can be modified to improve results, but also the graph matching method. In particular, using graph matching methods that are specifically designed for determining similarity of skeleton graphs [Siddiqi et al., 1999; Sebastian et al., 2001] is promising.

In the future, an optimised graph matching method, combined with an image database like CE-Shape-1, extended by skeleton graphs of exact skeletonisation results, could act as a reliable method for comparison of skeletonisation algorithms.

Bibliography

Blum, H. A Transformation for Extracting New Descriptors of Shape. In Wathen-Dunn, W, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

Borgefors, G. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics and Image Processing*, 27(3):321–345, 1984.

Bornemann, F and März, T. Fast image inpainting based on coherence transport. *Journal of Mathematical Imaging and Vision*, 28(3):259–278, 2007.

Cormen, T. H, Leiserson, C. E, Rivest, R. L, and Stein, C. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 2nd edition, 2001.

Crown Registry and Geographic Base. User Manual For The British Columbia TRIM HoL (Height-of-Land) Database, 2001. URL http://ilmbwww.gov.bc.ca/crgb/pba/trim/#manual. Last checked: February 23, 2010.

Danielsson, P. E. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248, 1980.

Fabbri, R, Costa, L. D. F, Torelli, J. C, and Bruno, O. M. 2D Euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys*, 40 (1):1–44, 2008.

Kimmel, R, Shaked, D, Kiryati, N, and Bruckstein, A. Skeletonization via distance maps and level sets. *Computer Vision and Image Understanding*, 62 (3):382–391, 1995.

Kong, T. Y and Rosenfeld, A. Digital topology: introduction and survey. *Computer Vision, Graphics and Image Processing*, 48(3):357–393, 1989.

Kraml, G. Binary Heap Priority Queue: Introduction and ansi C reference implementation. http://www.sbhatnagar.com/SourceCode/pqueue.html. Last checked: February 23, 2010.

Lakshmi, C. V, Singh, S, J., R, and Patvardhan, C. A novel approach to skeletonization for multi-font OCR applications. In *Proc. 3rd International Conference on Pattern Recognition and Machine Intelligence*, volume 5909, pages 393–399, New Delhi, India, 2009.

Latecki, L, Lakämper, R, and Eckhardt, U. Shape descriptors for non-rigid shapes with a single closed contour. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, South Caronlina, 2000.

Malandain, G and Fernández-Vidal, S. Euclidean skeletons. *Image and Vision Computing*, 16(5):317–327, 1998.

Maurer, Jr., C. R, Qi, R, and Raghavan, V. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

Meijster, A, Roerdinkand, J, and Hesselink, W. A general algorithm for computing distance transforms in linear time. In Goutsias, J, Vincent, L, and Bloomberg, D. S, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18 of *Computational Imaging and Vision*. Springer, Dordrecht, 2002.

Montanari, U. A method for obtaining skeletons using a quasi-euclidean distance. *Journal of the ACM*, 15(4):600–624, 1968.

Ogniewicz, R. L and Kübler, O. Hierarchic Voronoi skeletons. *Pattern Recognition*, 28(3):343–359, 1995.

Palagyi, K and Nemeth, G. Fully parallel 3D thinning algorithms based on sufficient conditions for topology preservation. In *Proc. 15th IAPR International Conference on Discrete Geometry for Computer Imagery*, pages 481–492, Montréal, Canada, 2009.

P.J. Giblin, S. B. Local symmetry of plane curves. *American Math Monthly*, 92:689–707, 1985.

Pudney, C. Distance-ordered homotopic thinning: a skeletonization algorithm for 3D digital images. *Computer Vision and Image Understanding*, 72(3): 404–413, 1998.

Rémy, E and Thiel, E. Exact medial axis with euclidean distance. *Image and Vision Computing*, 23(2):167–175, 2005.

Rouy, E and Tourin, A. A viscosity solutions approach to shape-from-shading. *SIAM Journal on Numerical Analysis*, 29:867–884, 1992.

Saito, T and Toriwaki, J.-I. New algorithms for euclidean distance transformations of an n-dimensional digitised picture with applications. *Pattern Recognition*, 27(11):1551–1565, 1994.

Sebastian, T. B, Klein, P. N, and Kimia, B. B. Recognition of shapes by editing shock graphs. In *Proc. Eighth IEEE International Conference on Computer Vision*, pages 755–762, Vancouver, Canada, 2001.

Sethian, J. A. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591, 1996.

Shasha, D and Wang, J. Graphdiff: Approximate Graph Matcher and Clusterer. http://cs.nyu.edu/shasha/papers/agm.html, 2000. Last checked: February 23, 2010.

Siddiqi, K, Shokoufandeh, A, Dickinson, S. J, and Zucker, S. W. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999.

Siddiqi, K, Bouix, S, Tannenbaum, A, and Zucker, S. W. Hamilton-Jacobi skeletons. *International Journal of Computer Vision*, 48(3):215–231, 2002.

Sobel, I and Feldman, G. A 3×3 isotropic gradient operator for image processing. *Pattern Classification and Scene Analysis*, pages 271–272, 1973.

Sorantin, E, Halmai, C, Erdohelyi, B, Palagyi, K, Nyul, L, Olle, K, Geiger, B, Lindbichler, F, Friedrich, G, and Kiesler, K. Spiral-CT-based assessment of tracheal stenoses using 3-D-skeletonization. *Medical Imaging*, 21(3):263–273, 2002.

Sussman, G. J and Wisdom, J. *Structure and interpretation of classical mechanics*. MIT Press, Cambridge, MA, USA, 2001.

Telea, A. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 9(1):23–34, 2004.

Zhu, S. C and Yuille, A. L. FORMS: A flexible object recognition and modeling system. *International Journal of Computer Vision*, 20:187–212, 1995.

Zhu, Y, Seneviratne, L. D, and Earles, S. W. E. A fast boundary based thinning algorithm. In *Proc. IAPR Workshop on Machine Vision Applications*, pages 548–551, Kawasaki, Japan, 1994.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

FMDT	flux-ordered maximal disc thinning
FOA	flux-ordered adaptive thinning
FOT	flux-ordered thinning
MAF	medial axis function
MAT	medial axis transform
MD	maximal disc detection
MDT	maximal disc thinning
PGM	portable grey map
SMD	secondary MAT detection

Appendix A

Command Line Tools

In this appendix, a short description and usage manual for the command line tools used for skeleton analysis is given. Each tool is described by its name, followed by a list of possible parameters, Optional parameters are marked with square brackets, alternative values for a certain parameter are separated by vertical bars. The purpose of the parameters is briefly described in an argument list, followed by a short text overview of the tool's core functionality.

analyser in_filename out_filename [-a]

> **in_filename:** name of input file.
> **out_filename:** name of output file.
> **-a** : optional parameter for averaging mode.
>
> Without the parameter -a, the *analyser* reads the pgm file with the input filename and the associated distance map, as well as the skeletonisation results for all algorithms, obeying the naming conventions of the output files produced by *skel*. For each input skeleton, *analyser* computes a reconstruction of the original shape and uses it to compute the quality scores described in Section 3.2.1.
> The parameter -a enables averaging mode. The input file is a text file with quality scores as created by *analyser* in normal mode. Runtime information from the input file is averaged and written to the specified output file.

graphtool in_filename out_filename graph_title

> **in_filename:** name of skeleton file.
> **out_filename:** name of output file.
> **graph_title:** title of output graph.
>
> The *graphtool* takes a skeleton file in the pgm format and computes a skeleton graph based on the end- and branching points of the skeleton. The output format is a graph file for the approximate graph matcher *graphdiff* [Shasha and Wang, 2000].

runtime in_filename out_filename [-a]

in_filename: name of input file.
out_filename: name of output file.
-a : optional parameter for averaging mode.

Without the parameter -a, the runtime tool takes a runtime file created by *skel* as input and appends the runtime results for all algorithms to the output file in LaTeX table format.
The parameter -a enables averaging mode. The input file is a table with truncated LaTeX format as created by *runtime* in normal mode. Runtime information from the input file is averaged and written to the specified output file.

skel [-fot|-foa|-fod|-fmdt|-md|-mdt|-all][threshold] in_filename

algorithm: -fot, -foa etc. specify the algorithms that are used, multiple options can be combined.
threshold: threshold τ for FOT and FOD (real number).
in_filename: name of input file (binary pgm image).

Skel computes the MAT of a binary input image in the .pgm format. The output files are a single distance map file and skeletonisation results for the chosen algorithms. The skeletons and distance maps are saved as pgm images. Additionally, *skel* writes a text file with the runtime of each skeletonisation process. All output filenames are generated from in_filename.

Appendix B

Heap Functions

Memory allocation in the ANSI C implementation of the binary heap is based on the public domain reference information by Kraml.

Algorithm 9 Insert value x into a list based binary MIN-heap.

Require: size // heap size
Require: max // available memory
Require: h // heap array
Require: x // element that is inserted into the array

```
if size ≥ max then
  realloc(h,2·max) // double array size
end if
size = size+1
if size = 1 then
  s[1] = x
else
  i = size
  while i > 1 ∧ x < s[i/2] do
    s[i] = s[i/2] // swap parent end child node
    i = i/2 // set insertion index to former parent node
  end while
  s[i]=x // insert x
end if
```

Algorithm 10 Return the value of the root node and remove the root node while preserving the heap property.

```
Require: size // heap size
Require: h // heap array
  root = s[1] // save root
  s[1] = s[size] // move last leaf to root
  size = size-1
  i = 1
  while i ≤ size/2 do
    // restore heap property
    j = 2i
    if size ≥ 2i + 1 ∧ s[2i + 1] < s[2i] then
      j = 2i + 1
    end if
    if s[j] < s[i] then
      // swap parent and child node
      tmp = s[i]
      s[i] = s[j]
      s[j] = tmp
      i = j
    else
      return root
    end if
  end while
  return root
```