Fakultät für Informatik
der Technischen Universität München
Lehrstuhl Univ.-Prof. Dr. B. Radig

# Unbiased Extraction of Curvilinear Structures from 2D and 3D Images

Dissertation

*Carsten Steger*

Fakultät für Informatik
der Technischen Universität München
Lehrstuhl Univ.-Prof. Dr. B. Radig

# Unbiased Extraction of Curvilinear Structures from 2D and 3D Images

*Carsten Steger*

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

| | | |
|---|---|---|
| Vorsitzender: | | Univ.-Prof. Dr. Jürgen Eickel |
| Prüfer der Dissertation: | | |
| | 1. | Univ.-Prof. Dr. Bernd Radig |
| | 2. | Univ.-Prof. Dr. Heinrich Ebner |

Die Dissertation wurde am 21.1.1998 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 20.5.1998 angenommen.

# Abstract

This dissertation proposes a new approach to extract curvilinear structures from 2D images. The major novelties are the use of explicit geometric models for line profiles and the systematic analysis of their behavior in linear scale-space. From this analysis, algorithms to extract lines and their widths with sub-pixel resolution are derived, and it is shown that the line positions and widths are necessarily biased by the Gaussian smoothing used in the extraction. Since the mapping that describes the bias is invertible, it can be removed, leading to unbiased results. Furthermore, a new approach to link individual line points into a topologically sound data structure of lines and junctions is proposed. It is noted that the extracted junction information is often incomplete. Therefore, the behavior of the line detector in the vicinity of junctions is analyzed, and a new method to extract complete junction information is derived from this analysis.

The method to extract lines from 2D images leads to a novel approach of extracting edges in 2D images with sub-pixel resolution by regarding them as lines in the gradient image. The analysis of the problem of missing junction information is extended for edges, and it is shown that the algorithm to extract complete junction information can be applied for edges as well.

The 2D line detection approach is extended to 3D images in two ways: Methods to extract 3D surfaces possessing a line profile and to extract 3D lines are proposed. The extraction of surface positions and widths is done similarly to the extraction of 2D line points and their widths. Furthermore, the bias exhibited by surface extraction results is the same as for 2D lines, and can therefore be removed in the same manner. The major difference to the 2D case is that it is more complicated to link surface points into surfaces.

The extraction of 3D lines and their widths, i.e., perimeter curves, is closely related to the extraction of extrema and their perimeters in 2D images, for which a new algorithm is proposed as a preparation for the full 3D line extraction algorithm. A scale-space analysis of a model extremum shows that the positions and perimeters of 2D extrema, and hence by extension those of 3D lines, are generally biased in a similar manner as the 2D lines. However, contrary to the 2D case, this bias can in general not be predicted analytically. Therefore, a method to remove the bias by edge focusing is proposed. The algorithms developed to extract 2D extrema and their widths are extended to extract 3D lines. The resulting line points are linked into lines by an extended version of the 2D line linking algorithm.

Finally, an extensive analysis of the quality of the extraction results is carried out. In order to test the unbiasedness of the line extraction results, noiseless and noisy synthetic test images are used. Furthermore, the localization quality of line and edge points in the presence of noise is analyzed analytically, and new formulas describing this relationship are derived. Tests on synthetic noisy images show the formulas to hold very well in practice. Experiments on real images show that sub-pixel accuracy better than one tenth of a pixel is possible in typical inspection tasks.

# Acknowledgements

First of all, I would like to thank my thesis advisor, Prof. Dr. Bernd Radig, for giving me the opportunity to work on this dissertation. Research is often akin to Brownian motion, and I would like to express my gratitude towards Prof. Radig for not restricting my often chaotic movements in any way.

Furthermore, I would like to express my thanks to Prof. Dr. Heinrich Ebner for being the second reporter on this dissertation.

I owe a great debt to my colleagues at the Forschungsgruppe Bildverstehen (FG BV) at the Department of Computer Science of the Technische Universität München: Christoph Zierl, Stefan Lanser, and most notably Wolfgang Eckstein, who was always one step ahead of me when it counted. They all took time to proofread this dissertation even though their schedules were extremely tight.

Furthermore, I am indebted to my colleagues Helmut Mayer and Albert Baumgartner at the Chair for Photogrammetry and Remote Sensing of the Technische Universität München for proofreading and for being understanding that throughout the process of writing this dissertation I had limited time to work on our joint research project.

Finally, I would like to thank the individuals or groups of people who have provided me datasets to test my algorithms on. Dr. Matthew G. Fleming of the Department of Dermatology at the Medical College of Wisconsin supplied me with the dermatoscopic image used in Chapter 3. Dr. Udo Bürgel of the University of Mainz provided me the coronary angiogram used in Chapter 3. Dr. Stephen Aylward of the Department of Radiology at the University of North Carolina (UNC), Chapel Hill, supplied two of the MR angiography datasets used in Chapter 4. The third MR angiogram was taken from the image archive of the United Medical and Dental Schools of Guy’s and St. Thomas’ Hospitals’ Image Processing Group, London (UMDS-IPG). Finally, the CT and MR datasets used in Chapter 4 were obtained from the University of North Carolina, Chapel Hill, volume rendering test dataset.

# Contents

# Chapter 1

# Introduction

*Geometric modeling of curvilinear structures and systematic analysis of their behavior in linear scale-space leads to unbiased extraction results of very high sub-pixel precision and accuracy.* This is the thesis of this dissertation.

The term “curvilinear structure” or “line” is used in many applications without giving a precise definition. What then constitutes a curvilinear structure in a two-dimensional image? Conceptually, one might think of a line as a one-dimensional manifold in the two-dimensional space $\mathbb{R}^2$. Although it is conceivable that such infinitely thin lines exist, at least mathematically, all physically occurring curvilinear structures of interest have another important characteristic: they possess a width, and hence are not really one-dimensional manifolds. However, their center line, often also called medial axis, is a one-dimensional manifold. In other words, curvilinear structures in 2D can be modeled as 1D manifolds, i.e., curves, that exhibit a characteristic 1D line profile in the direction perpendicular to the line.

Curvilinear structures are important objects for many applications. In 2D they arise quite naturally as the low-level primitives of choice to extract from images. For example, in photogrammetry and remote sensing, roads, railroads, and rivers can be regarded as lines in aerial or satellite images. The extraction of these kinds of objects is very important for the data capture or update of geographic information systems (GIS) [2, 10, 11, 20, 35, 69, 129, 157, 173, 196, 197]. Obviously, for these applications the width of the line is essential since it characterizes the type of a road, for example. Another area where curvilinear structures are important is in medical applications. For instance, in cardiographic angiography one is interested in detecting stenoses, i.e., strictures, of the coronary arteries in order to determine whether a patient is in danger of sustaining a heart attack. In the X-ray images or image sequences used for diagnosis the blood vessels appear as lines [34, 45]. Of course, to detect the risk of a heart attack the diameter of a blood vessel plays a crucial role. Furthermore, in dermatoscopic applications the characteristics of the network of pigment cells of the skin is a major decision criterion in determining whether a skin lesion is benign or malignant [174]. As above, the width of the individual lines of the pigment network is essential here.

Edges, another important class of low-level features, can also be modeled as curves that exhibit a characteristic profile in the direction perpendicular to the curve. In contrast to lines, their characteristic profile is usually not regarded as having a well defined width, although some applications do define the notion of the width of an edge. The possible applications for edges are too numerous to mention. They include, for example, road [2, 10, 150] and building

[18, 46, 70, 110] extraction from high-resolution aerial images, object recognition [12, 13, 107], and pose estimation [99, 102, 120, 193].

For 3D images, two obvious extensions of the above definition of lines are possible. The first one is to maintain the part of the definition that requires lines to be curvilinear structures, i.e., one-dimensional manifolds in $\mathbb{R}^3$. In this case, the definition of the width of a line must be modified because the profile of the line is not one-dimensional any more. Instead, the width must be modeled by a perimeter curve around a line point. The second extension of the 2D line definition is to maintain the requirement that objects must exhibit a characteristic 1D line profile perpendicular to the object. This definition leads to two-dimensional manifolds in $\mathbb{R}^3$, i.e., surfaces.

These two types of features have applications in medicine, where three-dimensional images of different modalities, e.g., computerized tomography (CT) or magnetic resonance (MR) images, are used for various diagnostic purposes. Often, 3D lines are important objects, which occur, for example, if images of blood vessels in the skull or abdomen are taken [8, 62, 64, 88, 118, 119, 126, 153, 159, 190]. Here, the main application is to detect aneurysms, i.e., enlargements of blood vessels, which may be fatal if the blood vessel eventually ruptures. In other applications it is often useful to extract surface-like structures having a well defined width, e.g., the bones in the skull, from CT or MR images for registration purposes [123].

The lack of geometric modeling of lines in existing approaches and the absence of a systematic analysis of their behavior motivates the major goal of this dissertation: to develop a low-level algorithm to extract lines and their widths from 2D and 3D images and surfaces and their widths from 3D images. Although surfaces strictly speaking do not constitute curvilinear structures, they at least exhibit a similar profile as lines in 2D images. For this reason, and since the main emphasis of this work lies on linear structures, surfaces are not included explicitly in the title of this dissertation.

Since the resolution of the images used in an application is often quite low compared to the precision requirements of the application, a major requirement of the algorithm is that it should return its extracted features with sub-pixel resolution.

In this dissertation, lines are defined by differential geometric features of the image. In order to extract any kind of feature depending on the partial derivatives of an image some sort of smoothing has to be performed because of the inherent noisiness of images. It has been shown that the optimum method to smooth an image is to convolve it with a Gaussian kernel [53, 91, 113, 181, 180, 191]. Since the Gaussian kernel has one free parameter, its standard deviation $\sigma$, this leads to a so-called scale-space description of the image. Because the convolution with the Gaussian kernel is a linear operation, this is called a linear scale-space.

Curvilinear structures are quite intricate objects because under the influence of Gaussian smoothing their position and width strongly depend on various factors. These include, most notably, the contrast to the background on both sides of the line or surface in 2D and 3D images, respectively, and the contrast to the background on the entire perimeter of the line in 3D images. One of the major new methodologies introduced in this dissertation is the systematic analytical modeling of the position and width of a line or surface. This analysis shows that Gaussian smoothing necessarily introduces biased line positions and widths. However, by carrying the analysis one step further it is shown that the analytically predicted bias can be removed, and therefore unbiased extraction results are obtainable.

As a by-product, the proposed line detection algorithm quite naturally suggests a way to implement a novel edge detection algorithm with sub-pixel resolution by regarding edges as lines in the gradient image. The edge detection algorithm is only implemented for 2D images, but could be extended to 3D images without major efforts.

With the same methodology used to model the bias of the line positions and widths, it is shown that by necessity most line and edge detectors are unable to extract complete junction information in 2D images. As above, the scale-space analysis of appropriate model junctions suggests a new sophisticated approach to extract the missing line and edge junctions, and hence to improve the extraction results.

Finally, a rigorous performance evaluation of the proposed line and edge detection algorithms is performed. The aim here is on the one hand to verify the unbiasedness of the extracted features for noiseless and noisy synthetic images, as well as for real images, and on the other hand to derive quality measures for the extracted features. This serves to show that the extracted features are returned not only with sub-pixel resolution, but with high sub-pixel precision and accuracy. Following [56], for the purpose of this dissertation precision refers to the repeatability of the extraction results, i.e., their variance, and accuracy refers to their absolute errors, including possible bias.

The structure of this dissertation is as follows: Chapter 2 discusses relevant previous approaches to line, surface, and sub-pixel edge detection. In Chapter 3, the line and edge detection algorithms for 2D images are derived. The line detection algorithm is extended to extract 3D lines and surfaces in Chapter 4. In Chapter 5, a thorough evaluation of the quality of the results returned by the 2D line and edge detection algorithms is performed. The dissertation is concluded with a summary and possible directions for further research in Chapter 6.

# Chapter 2

# Previous Work

The approaches to extract curvilinear structures from two- and three-dimensional images can be grouped into four categories, ordered roughly according to their level of sophistication: Gray-value-based approaches only use the gray values of the original images or some sort of local differences to extract line points. Differential geometric approaches regard the image as a surface in three or four-dimensional space, and try to extract lines as ridges by using various differential geometric features. In a third category, special filters enhance linear structures, so that lines can easily be obtained by thresholding. Finally, some approaches use an explicit line model to detect lines and their width. These categories are discussed in turn in four sections below. However, since most of the algorithms discussed at least implicitly make use of the scale of an operator and the objects in the image, the relevant concepts of the field of scale-space theory will be discussed first. Furthermore, since the line extraction algorithm that will be developed in Section 3.1 can easily be modified to implement a sub-pixel precise edge detection algorithm, relevant previous approaches to sub-pixel edge detection will also be discussed in this chapter. Finally, from the discussion of the existing approaches to line extraction, requirements for a line detection algorithm are derived.

Throughout the discussion, operators will be referred to as detectors and extractors. As is customary in low-level computer vision, these two terms are used synonymously. Furthermore, since an approach to feature extraction often is equivalent to its implementation, the terms approach, scheme, algorithm, and implementation are used synonymously.

## 2.1 Scale-Space Theory

Scale-space theory is a relatively new branch of computer vision, which tries to formalize the intuitive concept of “scale” for a signal and, more importantly, also for objects [113, 191]. The notion of scale is, of course, well known to anyone who has used a map or an atlas. In cartography, the scale of a map gives a good indication of which types of objects can appear in that map. For example, in a large-scale $1:5000$ topographic map, buildings are depicted as their outlines, whereas in a small-scale $1:100000$ topographic map only complete city blocks are represented. The problem of how to represent which types of objects in a map is known as generalization in cartography [71]. This “simple” example from cartography already shows the essence of the scale of an object: roughly speaking, it is the size of an object which determines whether and how it should be represented in a description of the real world of a certain scale. In cartography, as in computer vision, the problem of which objects to represent can be traced back to the finite resolution of the medium in which the representation is done. For maps, which should be legible by a human observer, objects must, for example, be separated by a minimum distance, e.g., 0.2 mm, in order for a human to be able to discern them with bare eyes. In computer vision, the finite resolution is caused by the physics of the device used to acquire an image, e.g., the size of an element of a CCD sensor or the granularity of the film. Obviously, objects having a scale smaller than the so-called “inner scale” of the imaging device, i.e., the pixel distance, cannot be represented [91]. On the other hand, objects larger than the “outer scale” of the imaging device, i.e., the imaging area, also cannot be represented.

From this discussion it can be seen that the essential characteristic of an object is whether it is visible at a certain scale. Hence, an image represented at a coarse scale should somehow be simpler than an image represented at a fine scale, i.e., information should be suppressed at coarser scales. Note that the terms coarse and fine scale have been used instead of the terms small and large scale used in cartography because the latter usage causes a great deal of confusion in computer vision. The reason for this is given below. The basic idea to simplify an image is to smooth it with successively stronger smoothing operators. More formally, the original image $f : \mathbb{R}^n \mapsto \mathbb{R}; x \mapsto f(x)$ is embedded into a one-parameter family of derived images $L : \mathbb{R}^n \times \mathbb{R}^+ \mapsto \mathbb{R}; (x, t) \mapsto L(x, t)$, where an additional parameter $t$ is introduced representing the scale of the image. $L(x, t)$ is called the scale-space description of the image $f(x)$. One basic assumption about the smoothing operator used to derive the scale-space description is that it is linear and shift-invariant because images can be assumed to show no preferred locations and signals can be superposed linearly. The assumption of linearity leads to the field of linear scale-space theory. Other possible non-linear assumptions are briefly discussed below. Linearity implies that the smoothing of the image can be described by a convolution

$$L(x,t) = g(x,t) * f(x) = \int\limits_{y \in \mathbb{R}^n} g(y,t) f(x-y)\, dy \ . \tag{2.1}$$

It is also required that the scale-space description at zero scale should be identical to the original image:

$$L(x,0) = f(x) \ . \tag{2.2}$$

Implicitly, this definition assumes that larger values of $t$ lead to smoother images. Therefore, larger values of $t$ correspond to more details being suppressed, which is opposite to the scale definition used in cartography, where small scales correspond to more details being suppressed. Therefore, the terms coarse and fine scale will be used in this presentation to avoid confusion.

The most important result of scale-space theory is that by placing additional mild restrictions, described below, on the smoothing operator $g$, the $n$-dimensional Gaussian function

$$g(x,t) = \frac{1}{(2\pi t)^{n/2}} e^{-\frac{x^T x}{2t}} \tag{2.3}$$

is singled out as the only possible smoothing kernel. Note that $t$ plays the role of the variance of the Gaussian, i.e., $t = \sigma^2$.

Over the years, several equivalent assumptions have been given which uniquely lead to the Gaussian kernel. In the original formulation for 1D signals [191] it was observed that no new local extrema or, equivalently, no new zero crossings could be introduced by the smoothing operator, hence it is sufficient to yield a scale space description. In 2D images this is not the case, since additional extrema can be created by Gaussian smoothing [113].

The first set of necessary conditions for arbitrary dimensions was given in [91]. The most important assumption introduced there is causality, i.e., no new level surfaces $L(x,t) = L_0$ must be created by the scale-space representation when $t$ is increased. Furthermore, isotropy and homogeneity were required for the representation, meaning that all spatial positions and scales should be treated in the same manner. With these requirements, one can show that the scale-space description fulfills the diffusion equation

$$\frac{\partial L}{\partial t} = \frac{1}{2}\nabla^2 L = \frac{1}{2}\left(\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}\right) \ . \tag{2.4}$$

Since from the theory of partial differential equations it is known that the Gaussian kernel is the Green's function of the diffusion equation, the Gaussian is singled out as the unique scale-space smoothing kernel. Similar assumptions are made in [9, 195] which lead to the same result.

A different set of assumptions in addition to linearity and shift invariance was used in [111]. There, for 1D signals the non-creation of extrema was required. In addition, the scale-space description was assumed to fulfill a semi-group property, i.e., successive applications of the smoothing operator should "add up": $g(x,t+s) = g(x,t) * g(x,s)$. If the kernels are required to be normalized, again the Gaussian is obtained. However, this set of assumptions is only sufficient for 1D kernels.

A final set of assumptions leading uniquely to the Gaussian was given in [53, 180, 181]. Here, linearity, shift invariance, the semi-group property, and scale invariance are assumed. The assumption of scale invariance is taken from physics, where it is known that physical laws must be independent of the choice of fundamental parameters or, equivalently, a function that relates physical observables must be independent of the choice of dimensional units.

The second important result of scale-space theory is that the Gaussian kernel possesses strong regularization properties that make the inherently ill posed problem of differentiation well posed [53, 113, 182]. Here, a problem is called well posed if a solution exists, is unique, and depends continuously on the input data. For differentiation, a small disturbance of the input image

$$f(x) \mapsto f(x) + \epsilon \sin \omega x \ , \tag{2.5}$$

where $\epsilon$ is small and $\omega$ is large, can lead to arbitrarily large disturbances of the derivative

$$f_x(x) \mapsto f_x(x) + \omega\epsilon \cos \omega x \tag{2.6}$$

if only $\omega$ is large enough. If, however, the image is smoothed with a Gaussian kernel, the scale-space description of the derivative is given by

$$L_x(x,t) \mapsto L_x(x,t) + \epsilon\omega e^{-\frac{\omega^2 t}{2}} \cos \omega x \ . \tag{2.7}$$

Obviously, now the disturbance can be made arbitrarily small if $t$ is chosen large enough. An additional important aspect of the scale-space description $L(x,t)$ is that it is infinitely often differentiable for $t > 0$.

Feature extraction in scale-space theory is almost always done based on certain kinds of singularities of the image or scale-space description, e.g., extrema of the image [109, 111, 113].

Often the features are tracked in scale-space to obtain an estimate of the saliency of the object by its lifetime in scale-space. For 1D signals this task is relatively simple since no new extrema can be created as $t$ is increased. Using results from catastrophe theory [22], one can show that generically the only possible event that can happen as $t$ is increased is the mutual annihilation of a minimum and a maximum. Such events are therefore called scale-space events. For 2D images, extrema can be created as $t$ is increased [113]. Therefore, linking of extrema across scale is more complicated in 2D. In [111], an image segmentation based on so-called "blobs" was proposed. Essentially, a blob is an extremum, e.g., a maximum, in the image and the surrounding region defined by the gray values larger than that of the closest saddle point to the extremum. One can show that only four possible scale-space events can occur for 2D blobs: annihilation, merge, split, and creation of blobs, with the latter two events occurring only rarely in practice. Since the blobs define a region in space the linking problem becomes much easier because the overlap of regions can be used to link blobs across scale. The image segmentation thus obtained is called the scale-space primal sketch in [111]. The 4D volume of a blob, defined by the 2D-image region, the scale, and the intensity, is used to measure its saliency.

One of the advantages of linear scale-space theory is that an analytical analysis of the behavior of features in scale-space is often possible, either by explicitly calculating the convolution of an explicitly represented model feature with the desired derivatives of the Gaussian kernel, or by using the diffusion equation (2.4). For example, for 1D edges defined by the zero-crossings of the second derivative one can show that the edge drift velocity for an edge point $e$ in position $x_0$ at scale $t_0$ is given by

$$\frac{\partial}{\partial t}e(t) = -\frac{1}{2}\frac{L_{xxxx}(x_0,t_0)}{L_{xxx}(x_0,t_0)} \quad . \tag{2.8}$$

This means that edges can momentarily be displaced with arbitrary large speed if $L_{xxx}(x_0,t_0)$ is close to zero. This happens in the vicinity of a maximum-minimum annihilation event. However, the total edge drift over a finite interval of scale will always be finite. As above, this result can be used in linking edges across scale.

Linear scale-space theory has spawned several other non-linear multi-scale representation schemes. The first such scheme, anisotropic diffusion, which should more correctly be called non-homogeneous diffusion, was introduced in [148]. The basic idea of the approach is to introduce a conductance term into the diffusion equation:

$$\frac{\partial L}{\partial t} = \frac{1}{2}\nabla(c\nabla L) \quad . \tag{2.9}$$

Analogously to heat diffusion in a non-homogeneous medium, the conductance term can be used to define areas across which heat cannot dissipate. In the original approach $c$ is the gradient of the image, leading to the behavior that there will be little smoothing across object boundaries.

Another approach is to smooth the image according to the geometry of its level curves and surfaces [89, 141, 166]. The evolution of a level curve $c$ can be fully described by its curvature $\kappa$ as a function of arclength $s$

$$\frac{\partial c}{\partial s} = f(\kappa(s))n(s) \quad , \tag{2.10}$$

where $n$ is the normal vector of the curve. The most often used case is $f(\kappa) = \alpha + \beta\kappa$. If $\alpha = 0$ this leads to the curve $c$ being smoothed according to its curvature, i.e., its geometry, only. It can be shown that all curves evolve into points of constant curvature, i.e., round points, in finite time, i.e., for a finite amount of smoothing [89]. If $\beta = 0$ the curve evolves only in the normal direction, leading to an erosion or dilation process, i.e., to mathematical morphology [75]. If both $\alpha$ and $\beta$ are non-zero, the resulting multi-scale representation is called the reaction-diffusion or entropy scale-space [89], where $\beta\kappa$ is the diffusion and $\alpha$ is the reaction term.

Finally, a multi-scale representation can also be obtained by using the gray scale morphological operations of erosion and dilation [84, 97, 98]. Here, the image is eroded and dilated with circular structuring elements of varying sizes, leading to the selective suppression of light or dark objects of a size smaller than the structuring element.

These different types of multi-scale representations possess many properties which are desirable in certain applications. However, the major problem with these approaches is that, unlike linear the scale-space representation, analytical results on the behavior of the derived signals are extremely difficult to obtain. Therefore, these representations are not considered further in this dissertation.

## 2.2 Gray-Value-Based Approaches

Approaches in the first category of line detectors extract lines by considering the gray values of the image only and use purely local criteria, e.g., local gray value differences. One problem with these approaches is that usually many points in the image fulfill the respective line criteria, i.e., there are many false hypotheses for line points. Therefore, elaborate and computationally expensive perceptual grouping schemes have to be used to select salient lines in the image, such as the general grouping approaches given in [48, 134], which assume a significant amount of noise in the extraction results, or the special approaches implemented for each individual extractor discussed below.

The line extraction scheme described in [50] smoothes the image and then scans for one-dimensional local maxima along the row and column directions. Local maxima are points for which the gray value is larger than the gray values of its immediate neighbors. Each maximum is given two attributes: The maximum gray value difference to the two closest minima on each side of the maximum and the medium gray value difference to the two points with the smallest gray values to the left and right of the maximum. Based on these attributes, salient line points are selected by a thresholding operation. The thresholds are estimated from the image, and are relatively lenient. Therefore, many false hypotheses are generated. Perceptually significant lines are extracted by the grouping algorithm described below. A more complex line extraction scheme is given in [47]. Here, outputs of several gray value operators are combined into a cost array. The operators are divided into the ones that yield false positives for lines with a low probability, but may lead to broken lines, e.g., the Duda road operator, and the ones that yield high responses for roads, but also return many false positives, e.g., the Sobel edge operator. From the output of the operators with a low probability of false positives, candidate line points are extracted by thresholding and the calculation of a minimum spanning tree. The weighted combination of the output of the operators with high probability of detecting roads defines a cost array, which is regarded as a graph. From this array, optimal paths between salient line points are constructed by a graph searching algorithm. Once a line between two points is found it is masked out in the cost array, and the procedure is repeated. Again, a minimum spanning tree is constructed from the found line segments to prune short segments. A very similar approach is given in [42]. The only difference is that operators of the second kind are not used.

A completely different approach is presented in [101]. Here, lines are enhanced by calculating the dot product of the gradient for each four opposite pixels of the current pixel, i.e., the line response is given by $\max(0, -\nabla(i_k) \cdot \nabla(i_{k+4}))$, where $i_0, \ldots, i_7$ denote the eight neighbors of a pixel. The gradients are computed from small Gaussian convolution masks. The rationale behind this approach is that gradients tend to point to opposite directions on each side of a line. From this it is immediately obvious that the operator will generate large responses for all features for which this is true, e.g., corners and circular symmetrical features.

The final gray-value-based operator is described in [60, 61, 86]. A probability for a pixel to be part of a line is derived from six gray values in the image. The operator returns a high probability if the gray value difference between the central two pixels and the corresponding four pixels perpendicular to the central pixels is large. Again, this will generate many false line point hypotheses. Therefore, the algorithm tries to track a line from a starting point, which is given by the user, but could conceivably also be determined automatically. To do the tracking, a kind of beam search algorithm is used. One problem of this approach is that because of the instable line extractor a very sophisticated heuristic for the beam search has to be defined. It involves the Bayesian modeling of the direction changes of the line and its gray value model. The actual heuristic is based on entropy minimization, i.e., the next pixel to expand the line is chosen according to the expected information gained by looking at it. Another problem with this approach is that the extracted lines usually will lie several pixel away from the true line position.

## 2.3 Differential Geometric Approaches

In this section differential geometric approaches to extract lines will be discussed. In these schemes the image is regarded as a function $f : \mathbb{R}^n \to \mathbb{R}$. For the most part of this section it will be implicitly assumed that $n = 2$ because most of the presented approaches use two-dimensional images, and because the function $f$ can be easily visualized in this case. With this interpretation of the image, lines are defined to be ridges and ravines in $f$. In the following, only ridges will be discussed since ravines simply are the ridges of $-f$. The notion of a ridge, however, is not well defined, and hence several ways to define ridges have been proposed. A good historical overview of this topic is given in [93, 156]. The intuitive notion typically associated with a ridge is that it separates regions where water falling onto the mountain range defined by $f$ is accumulating when running downhill. In other words, while walking on a ridge, one typically looks downward when turning left or right.

A classification of the various definitions for ridges was presented in [44]. In the following this classification will be used to discuss the various extraction schemes for ridges that have been proposed.

### 2.3.1 Ridges from Curvature Maxima of the Contour Lines

One way to define ridges is to assume they are given by the locations in the image where the contour lines (often also called isohypses or isophotes) of $f$ exhibit a local maximum in their curvature, i.e., have a vertex. This definition of ridges is presumably motivated by the way humans look for ridges in topographic maps, which is a very useful skill for mountaineers and treckers. Here, the third dimension is only implicitly given by the contour lines, but humans nevertheless are able to find ridges without much difficulty.

While this definition of ridges is useful in many cases, there are numerous important cases in which it will either lead to wrong results or does not work at all. First of all, if a ridge is perfectly horizontal, the contour lines around this ridge will all be parallel to each other and to the ridge. Therefore, this definition fails to detect this kind of ridge. This happens quite frequently, even for real data. Furthermore, vertices of the contour lines do not necessarily correspond to the location of the ridges. This can happen, for example, if the ridge has a flat profile which drops off sharply at some distance from the ridge. If the ridge profile is very flat, the correct ridge position will in fact be given by the local minimum of curvature of the contour lines! Some more examples of where this definition leads to incorrect results are given in [93, 123].

One way to extract ridges using the above definition is to apply it literally to the contour lines of an image. This is the approach of [100]. The algorithm first extracts the contour lines by using multiple thresholding at constant intervals on the image. The contour lines then are assembled into a so-called “topographic change tree” and peaks and pits are detected from the tree. They are defined by closed contour lines which are higher or lower than their enclosing contour lines. One counterintuitive consequence of this definition is that horizontal ridges and, more generally, ridges wholly contained in one height interval, are labeled as peaks and pits as well. What is more, the peaks and pits thus detected will even have a two-dimensional area greater than zero, i.e., have a blob-like shape. After this step, curvature maxima are extracted from the contour lines. Since in this approach the contour lines are only extracted with pixel accuracy, and since they are obtained by thresholding, thus being very sensitive to noise, the curvature of the contours is computed by using windows of varying width. The algorithm still has to select appropriate window widths and curvature thresholds for each contour point because of its noise sensitivity. Although the results of this procedure are likely better than using one window width for the curvature alone, there are still many extraneous vertices detected. Therefore, the final step of linking the vertices into ridges has to resort to grouping the vertices based on many heuristics. In addition to the basic problems mentioned above, this scheme suffers from another drawback: for ridges that have a very low gradient the contour lines will become widely separated, and thus hard to link.

Another way to extract the maxima of curvature on the contour lines is to give an explicit formula for that curvature and its direction, and to search for local maxima perpendicular to the direction of maximum curvature in the resulting image [44, 58, 123]. The curvature of the contour lines is given by

$$-\frac{z_x^2 z_{yy} + z_y^2 z_{xx} - 2 z_x z_y z_{xy}}{(z_x^2 + z_y^2)^{\frac{3}{2}}} \quad . \tag{2.11}$$

From this equation it is immediately obvious that this definition will fail for perfectly flat ridges, where $(z_x, z_y) = (0, 0)$. While Sard’s theorem [22] tells us that for generic functions such points will be isolated, they occur quite often in real images and lead to fragmented lines without a semantic reason. Furthermore, this operator also suffers from the problems of wrongly extracted ridge positions described above [93, 123]. While the ridge points extracted with this approach could be linked into ridges, none of the above approaches does this. The algorithm described in [44] contends itself with returning a binary image of the ridge points, while [123] stops with the image resulting from (2.11). An interesting approach is given in [58], where the ridges extracted from the image are matched to the watershed boundaries detected in the same image. This approach also tries to track the ridges and watersheds through scale-space.

The definition of ridges as vertices of the contour lines is also used implicitly in [179] to improve the quality of a digital terrain model (DTM), which was obtained by triangulating contour lines, by inserting triangles containing ridges and ravines. The approach constructs the medial axes between contour lines of adjacent height. In [179] it is observed that the sections of the resulting medial axes that correspond to a contour line of same height often correspond to sections of ridges and ravines. However, no mathematical proof of this conjecture is given, although it is intuitive to assume that vertices in the contour lines lead to this type of special medial axes. Because ridges often do not correspond to vertices of the contour lines the algorithm often returns fragmented and incomplete ridges. Nevertheless, the quality of the improved DTM appears to be much better than that of a DTM without taking ridges and ravines into account.

This definition of ridges can be extended to the $n$-dimensional case in the obvious manner [44]. As in the two-dimensional case, $d$-dimensional contour sets of the image intensity can be determined, which describe $d$-dimensional manifolds in $\mathbb{R}^n$. Each point on this manifold has $d$ principal directions and curvatures. A ridge of dimension $d$ is defined by the contour set having a local maximum of its principal curvatures in all $d$ principal directions. It is obvious that this definition will have the same problems as described above.

## 2.3.2 Ridges from Curvature Maxima in the Image

Ridges can also be defined by points exhibiting local maxima of curvature in the direction of the larger principal curvature. This definition is motivated by the definition of ridges in advanced differential geometry [92, 151], where ridges are defined for $(n-1)$-dimensional manifolds in $\mathbb{R}^n$. In order to apply this definition to images, the image is regarded as a $n$-dimensional manifold in $\mathbb{R}^{n+1}$ via the map $\mathbb{R}^n \to \mathbb{R}^{n+1} : x \to (x, z(x))$.

This definition of ridges is more useful than that in the previous section since it will also detect horizontal ridges. However, it still leads to counterintuitive or wrong results in many cases. Consider again the case of a ridge with a flat profile, turning down sharply some distance away from the ridge. It is clear that this definition will find two ridge points to the left and right of the true ridge [4, 44]. As for the ridge definition in the previous section, the true ridge will be given by the local minimum of curvature in this case. Furthermore, it will also declare the bright part of a step edge as a ridge and the dark part of an edge as a ravine, since they are positive and negative extrema of curvature, respectively. This makes it almost impossible to extract meaningful ridges from an image.

An approach using this ridge definition is given in [131]. Since the curvature depends on the second derivatives of the image, and its local maxima are needed, the algorithm starts by convolving the image with derivatives of a Gaussian kernel to obtain the first, second, and third partial derivatives of the image. The direction of the largest curvature is determined, and the directional derivative of the curvature in this direction is calculated. Both the direction and the derivative of the curvature can be calculated explicitly. However, since the resulting system of formulas in [131] is very large it is not reproduced here. After this, a sign image is computed which holds information on the zero crossings of the directional derivative. Finally, the ridges are extracted as the zero crossings, while additionally requiring that the curvature along the principal direction must be larger than a threshold. The experiments done in [131] show exactly the problems discussed above, i.e., shifted double responses to a single ridge and responses to edges. The problem is a little alleviated because thresholding is done on the curvature, and therefore most edges are not selected as ridges since an edge of equal amplitude as a line will have a much smaller curvature. The same approach is also discussed and used in [44], where qualitatively similar problems occur.

This approach can be extended in a straightforward manner to extract two-dimensional ridges, e.g., blood vessels, in volumetric images [3, 133, 153]. In this case, the first two principal curvatures must be local maxima in the corresponding principal directions. This results in the same drawbacks discussed above. Furthermore, since the third partial derivatives are needed by the algorithm, it requires an enormous amount of memory, i.e., 19 volumetric temporary images. This makes the approach infeasible even for moderately sized images.

Finally, like the approach in the previous section, this definition of ridges can be extended to the $n$-dimensional case [44]. A point in the image is declared a $d$-dimensional ridge if the first $d$ principal curvatures are local maxima in the corresponding principal directions. Again, there are the same problems as discussed above.

### 2.3.3 Ridges from Restricted Maxima in the Image

Another approach to define ridges is based on the following observation: If one walks along a ridge the terrain drops off to the left and right of the path. This means that there is a local maximum in height perpendicular to the ridge direction [44]. The direction of the ridge can be obtained as follows: Since the terrain drops off to both sides of the ridge, the second derivative perpendicular to the ridge direction has a very large negative value. Therefore, the direction perpendicular to the ridge is given by the eigenvector corresponding to the largest negative eigenvalue of the Hessian matrix of $z(x)$. This definition extends to the $n$-dimensional case in a straightforward manner. If the eigenvalues of the Hessian are sorted according to their magnitude, a $d$-dimensional ridge or ravine must have the first $d$ eigenvalues of the same sign and must have a local maximum or minimum in the direction given by the $d$ corresponding eigenvectors [44].

This ridge definition has by far the fewest shortcomings compared to the definitions in the previous two sections. It will detect the ridges in the intuitive position for all usual cases, and will only fail for very contrived counterexamples [93].

The above ridge definition implies that in order to calculate the ridge points the local second order Taylor polynomial of the image is needed. In many approaches [24, 25, 65, 72, 187, 188], the coefficients of this polynomial are determined by using the facet model [75, 76], i.e., by a least squares fit of the polynomial to the image data over a window of a certain size.

The ridge extraction scheme in [72] uses a cubic polynomial fit to the image obtained using $5 \times 5$ masks. The author derives an explicit formula for the angle that maximizes the absolute value of the second directional derivative. Plugging this direction into the cubic polynomial and setting the directional derivative to zero yields a quadratic equation with two solutions. The algorithm chooses the solution that lies closest to the current pixel's center. In order to declare the pixel a ridge point, the solution of the quadratic equation has to be closer to the pixel's center than a threshold, which is set to values between 0.85 and 1. Obviously, this will lead to many double responses, and ridges will generally be more than one pixel wide. Furthermore, no attempt is made to link the individual ridge points into ridges. Additionally, the third order polynomial tends to "overshoot" on the sides of the ridge, which may change the topography of the surface and cause ridge points to be missed. Furthermore, the algorithm declares all points on a radially symmetrical function as ridge points. This is partially solved by introducing additional criteria that are not fulfilled for such constellations, i.e., the ratio of the gradient to the second derivative must be small, and the gradients at the side of the ridge must point to sufficiently different directions than on the ridge.

A similar approach is described in [187, 188]. Third-order Chebyshev polynomials are fitted to the image over a window of size $n \times n$ for each pixel. They are used to compute the first and second partial derivatives of the image. The image is segmented into the topographic categories peak, pit, ridge, ravine, saddle, flat, and hillside points, similar to [76]. The rationale for this comes from the application of the approach, namely optical character recognition (OCR). In OCR, characters with thick strokes may lead to flat points on the strokes for small mask sizes, which nevertheless should be regarded as ridges. As above, the Hessian matrix is used to derive the ridge direction. Ridge points are not calculated with the sub-pixel scheme of [72]. Instead, the ridge directions are discretized to the pixel grid, and the check for a local maximum perpendicular to the ridge direction is made based on comparing gray values in the appropriate direction. Since this makes use of the original, noisy gray values, ridge pixels may be missed by this approach. After this, a topographical feature graph is computed, which basically consists of the connected components of ridges, peaks, saddles, and flat points, if ridges are to be extracted, and similarly ravines, pits, saddles, and flat points if ravines are desired. Peaks and saddles are regarded as junctions between ridges, and are thinned to a single point. Ridges and flat points are thinned to one pixel width and are approximated by polygons. This leads to the undesirable consequence that junctions of ridges generally are separated from the ridges by several pixels since the junction was represented by an extended area before thinning. Because of these problems, a scheme to segment the extracted ridges into arcs and straight lines is presented in [187]. It replaces the original approach based on thinning by a discrete contour following algorithm which implicitly also thins the result by removing some of the extracted ridge points. The linked pixels are then classified into arcs and line segments based on the, in this case undiscretized, ridge directions, by using a threshold on the change of the ridge direction. This will lead to junctions being represented by extended areas of pixels classified as arcs because the direction tends to change rapidly in junction areas.

The most advanced method using the facet model is presented in [24, 25]. A second order polynomial is calculated for every image point using masks of size $n \times n$. As above, the line direction is obtained from the Hessian matrix. Then a parabola is constructed perpendicular to the line direction. The derivative of this parabola is set to zero to obtain the line point. In contrast to [72], this is a linear equation, and hence only one solution results. A pixel is declared a line point if the local maximum or minimum lies within the pixel's boundaries, and if the second directional derivative is larger than a threshold. The threshold is estimated from the image data by taking three times the median value of the second directional derivatives of all isolated line pixels. The rationale behind this approach is that isolated line pixels are assumed to be caused by noise. After this step, line ends are detected by looking for line points which do not have neighboring line pixels in three pixels determined from the line direction for at least one of the two valid line directions. Next, lines are thinned to be one pixel wide. This is done taking the line direction into account, i.e., only pixels perpendicular to the line are deleted. Finally, junctions are detected as line pixels in the thinned image which have more than two adjacent line points. The advantage of this approach is that lines are obtained with sub-pixel resolution. One shortcoming is that the extracted line points are thinned, which often causes correct line pixels to be discarded because the sub-pixel positions are not taken into account in the thinning. Furthermore, the definition of the junctions leads to extended junction areas since, by symmetry, typically at least three pixels have more than three neighbors.

Finally, the algorithm given in [65] can be seen as a special case of the algorithm of [25]. It uses $3 \times 3$ facet model masks to obtain the partial derivatives of the image. Again, the line direction is computed from the Hessian. The algorithm checks for local maxima or minima perpendicular to the line by interpolating the gray values in this direction and comparing the resulting gray values. If the pixel is a local maximum, a parabola is fitted through these three points to extract the maximum position with sub-pixel resolution. The most interesting aspect of this approach is the ridge point linking algorithm because it does not require thinning. The procedure starts with the point having the largest second derivative and successively adds the most suitable neighbor from a neighborhood of a specified size to the ridge. The criterion used to check for suitability is a weighted sum involving three measures for the difference in direction of the ridge points and their distance. However, the paper fails to explain what the difference between the three angles involved in the sum is, and how to properly choose the weights of the sum. The quality of the extraction results seems to be similar to that of the results of [25].

One problem inherent to all facet model ridge detectors is that the convolution masks that are used to determine the coefficients of the Taylor polynomial are rather poor estimators for the partial derivatives. Therefore, these approaches usually lead to multiple responses to a single ridge and to inaccurately extracted ridge positions, especially when masks larger than $5 \times 5$ are used to suppress noise. Since for ridges with a flat profile the mask size has to be at least as large as the width of the ridge, this will lead to progressively inaccurate line positions as the mask size is increased. Therefore, the approach does not scale well and cannot be used to detect ridges wider than the necessarily small mask size. Because of these limitations of the facet model, several approaches have been proposed which estimate the derivatives using Gaussian filter masks. Interestingly, they all make use of a, sometimes rudimentary, line model, and consequently are discussed in Section 2.5.

### 2.3.4 Ridges as Watersheds

Because the definitions of ridges in Sections 2.3.1 and 2.3.2 often and the definition in Section 2.3.3 sometimes contradict the way water runs downhill, several different definitions have been given, which try to model this property.

The first definition was proposed in the second half of the 19th century by Maxwell, Jordan, and Cayley (see [67, 138, 154] and references therein). It is based on the observation that for generic surfaces there is a unique slope line through every non-critical point of the surface. Each slope line is the curve of steepest ascent, and can be split into a ascending and descending part for each surface point $p$. The ascending part is a solution of the ordinary differential equation $\dot{x}(t) = \nabla z(x(t))$, while the descending part is the solution of $\dot{x}(t) = -\nabla z(x(t))$, both with the initial condition $x(0) = p$. All slope lines converging at the same maximum are said to form a hill, while all slope lines converging at the same minimum comprise a valley. Obviously, there must be special slope lines separating different hills and valleys. These separatrices are the ridges and courses, respectively. The intersection of the regions given by hills and valleys defines slope districts. All slope lines in these districts converge to the same maximum and minimum, and the closure of the districts constitutes a complete segmentation of the surface or image. An important aspect of this definition is that ridges and courses are given by special slope lines emanating from saddle points, and running to a maximum or minimum, respectively. It can be shown that there are only four generic types of slope district boundaries [138], i.e., distributions of maxima, minima, and saddles on a boundary. The scale-space behavior of separatrices has been studied in [67]. There, it has been conjectured that the topology of separatrices can only change in four distinct types of scale-space events.

One problem of this definition of ridges and courses is that it is hard to implement correctly as a computer algorithm. To see this, consider that the ridge and course lines are singular solutions of the above differential equations. This means that if the algorithm does not start exactly on the special slope line it will never lie on it. Fortunately, if a slightly wrong initial point is chosen, the slope line will eventually converge to the true ridge or course line. Hence, approximate solutions to the problem are possible. One such approximation is given in [33]. The image is triangulated by inserting a point in the center between each four neighboring pixels, resulting in four triangles. Saddles, maxima, and minima are then extracted by comparing the gray value of a pixel to those of its neighbors in the triangulation. After this, so-called maximum gradient paths are extracted by following the four special slope lines from each saddle point. This is done by calculating the gradient direction, and adding the pixel in the gradient direction to the current slope line. The result of this algorithm is a discrete approximation of the slope district boundaries, i.e., ridges and courses. In a later paper [66], the approach is extended by a rather heuristic multi-scale scheme. The algorithm starts with the original triangulation and computes slope districts from it. After this, a new set of points is defined by the centers of gravity of each slope district, which has the average gray value of the entire slope district as its gray value. The new set of points is triangulated, and the procedure is repeated until there is only one data point in the image. It is not obvious how this implicit definition of scale corresponds to a meaningful scale in image coordinates.

A similar approach of following special slope lines is used in [7] to improve the quality of a triangulated DTM, similar to the approach described in [179] discussed in Section 2.3.1. The input data structure to the algorithm is a set of triangulated contour lines, as in [179]. Contrary to the definitions above, special slope lines are not constructed starting at saddle points, because it is relatively difficult to extract them from triangulated DTMs. Instead, it is noted that ridges and courses often pass through horizontal triangles of the DTM. These special triangles are classified into three classes: peaks, pits, and ridge and course line triangles. The starting points for ridges are selected within pit areas, while the starting points for courses are selected within peaks. From these points, ridges are constructed by following a special slope line, which is obtained by taking small steps in the gradient direction. The corresponding gradient vector field is constructed by interpolating the gradient directions at the vertices of the triangulation, which in turn are obtained by constructing normals to the contour lines. Since ridges and courses often do not pass through horizontal triangles of the DTM, the extracted separatrices are often fragmented and incomplete.

A different characterization of ridges is given in [156]. If one replaces the gradient vector field in the above differential equation $\dot{x}(t) = \nabla z(x(t))$, where $\nabla z = (z_x, z_y)$, by its dual

1-form, one can see that the slope line also are solutions of the ordinary differential equation $z_x \mathrm{d}y - z_y \mathrm{d}x = 0$. This differential equation usually does not have a general integral since the integrability constraint $-\frac{\partial}{\partial y} z_y = \frac{\partial}{\partial x} z_x$ does not hold. Therefore, one must look for an integrating divisor $\theta(x, y)$ such that $\frac{z_x}{\theta}\mathrm{d}y - \frac{z_y}{\theta}\mathrm{d}x = 0$ is an exact differential equation, i.e., such that $-\frac{\partial}{\partial y}\frac{z_y}{\theta} = \frac{\partial}{\partial x}\frac{z_x}{\theta}$. If $\theta$ is chosen in this way there exists a function $w(x, y)$, called the general integral, fulfilling the equation $\theta \mathrm{d}w = z_x \mathrm{d}y - z_y \mathrm{d}x$. The solutions curves of the differential equation are given by the level curves $w(x, y) = c$. The integrating divisor $\theta(x, y)$ fulfills the first order partial differential equation $\nabla z \cdot \nabla \theta = \theta \Delta z$ [93]. The solution to this differential equation is not unique. In fact, if $\theta_0$ is such a solution, all other solutions are of the form $\theta_0 f(w)$. Amongst the solution curves $\theta(x, y) = 0$ there are special slope lines for which the determinant $\frac{\partial w}{\partial x}\frac{\partial \theta}{\partial y} - \frac{\partial w}{\partial y}\frac{\partial \theta}{\partial x}$ does not vanish. These are the special slope lines, which contain the ridges and course lines. Since they are singular solutions of the above differential equation, ordinary slope lines will "crowd" around them, i.e., the density of ordinary slope lines is much higher in the vicinity of the special slope lines. A special slope line is a ridge or course line if ordinary slope lines reach them tangentially or asymptotically at infinity. Essentially, this removes special slope lines that are degenerate, e.g., perfectly horizontal, since water would not flow on such degenerate slope lines.

The biggest problem with this definition is that it is almost impossible to implement as a computer algorithm. The defining equation of the integrating divisor is a first order partial differential equation, which could be solved by the method of characteristic strips. However, no boundary conditions, i.e., no initial curve on the surface $w$ required for this method, can be given. Therefore, the only implementation which uses this particular ridge definition uses the above mentioned fact that ordinary slope lines tend to crowd around the desired special slope lines [116]. It calculates the gradient of the image with Gaussian filter masks. Then, slope lines are constructed from $n^2$ sub-pixel positions in each pixel. The gradient at these positions is obtained by bicubic interpolation. The slope lines are calculated by integrating the above ordinary differential equation using a fourth order Runge-Kutta method. A "creaseness" image is obtained by counting how many slope lines are passing through each pixel. The ridge and course lines are pixels where the number of slope lines passing through them is larger than a threshold. The major drawback of this approach is that since the slope lines never reach extrema or saddles, and since therefore the corresponding pixels will accumulate very few slope lines, the ridge and course lines are very fragmented, and therefore not very useful since a separate grouping step would be required to obtain meaningful results.

A third way to extract watersheds from the image is to use the fact that water will accumulate at the minima of the landscape. This means that each minimum in the image defines a valley or water catchment basin. Watersheds are the boundaries between different basins. One way to implement this is to simulate the flow of the water when it falls onto the surface, i.e., to construct descending slope lines from every point in the image. The algorithm in [58] starts by extracting all local minima from a slightly smoothed version of the image. Smoothing is done mainly to remove plateaus. Then, from each pixel in the image, a discretized version of the gradient direction is followed downhill until it reaches a minimum. To speed up processing, the entire path is labeled with the label of the minimum so that following slope lines can stop once they reach an already labeled pixel. The actual watersheds are finally extracted by returning the pixels in the image where label changes occur. The scale-space behavior of the watersheds is used in [58, 143, 144] to derive a heuristic to link the watersheds across different scales of the image. In [58] this is used to assign a scale to each watershed segment and to associate each watershed with a ridge curve extracted by the approach described in Section 2.3.1, while in [143, 144] it is used to track the watershed positions in the image to a more precise location at fine scales. Another way to implement the catchment basin definition is to take exactly the opposite approach, and to flood the landscape from the minima [130, 139, 186]. Watersheds are then defined by the places where water emanating from different minima meets. The algorithm in [186] works by sorting the gray values in the image by their magnitude. Then, the regions having the smallest gray value define the initial set of minima, which may have extended areas. In each following step of the algorithm the pixels with the next higher gray value are considered. The regions thus obtained are checked for whether they are new minima not connected to previous basins, whether they can be matched to a single previous basin, or whether they overlap several previous basins. In the latter case, watersheds are constructed between the different minima. One of the advantages of this approach is that it can be implemented very efficiently in a linear time algorithm.

## 2.4 Approaches Based on Special Filters

Apart from the detection of lines from gray values or as ridges in the image, several approaches have been proposed which make use of special filters to detect lines in the image. Most of these approaches combine the output of a linear filter, especially the derivatives of a Gaussian, in a non-linear manner.

The algorithm to extract edges and lines given in [83] rests on the observation that most linear filters respond to the features they were designed for as well as to other features. For example, the convolution of an image with the second derivative of a Gaussian will have a large absolute value at lines as well as near edges. Ideally, however, there should be zero response at non-features. The authors argue that, therefore, non-linear combinations of the linear filter output are needed to get good results, and develop a boolean algebra with continuous values designed for combining filter outputs. Then, they describe the conditions an ideal line and edge must fulfill using this algebra. The conditions used perpendicular to the line are more or less the same as the ones used for the ridge detectors, i.e., a line must have a local maximum perpendicular to its direction. The most interesting aspect of this work is the introduction of conditions which prevent the algorithm from following the line past the line end. The problems with this approach are manifold. First, since the algorithm does not estimate the line direction from the image, it has to be run multiple times with spatially tuned filters, resulting in high computational load. For this reason, line directions are only extracted coarsely. Furthermore, the algorithm can respond to the same line point in several directions. Finally, the filter always produces two responses for the same line or edge. The only advantage of this approach is that it produces responses to edges and lines in junction areas.

The signal theoretic approach given in [63, 64] tries to enhance lines in the image so that they can be extracted using simple thresholding. For the enhancement, second derivatives of the Gaussian are used. However, since the second derivative produces a response for edges and the sides of the line as well, these unwanted responses have to be suppressed. This is done using anisotropic diffusion. A conductance term is defined taking into account the second derivatives as well as the original image intensity. However, this does not completely remove the unwanted responses, and hence thresholding the resulting image still poses a significant problem. To get a symbolic description of the 3D-lines, a thinning algorithm [62] can optionally be applied to the result obtained by thresholding.

A similar approach for 3D images is presented in [159]. Again, the goal is to enhance linear structures such that they can be extracted by simple thresholding. The line direction and response are obtained from the Hessian matrix of an image filtered with Gaussian smoothing kernels. This is similar to the definition of ridges by restricted extrema used in Section 2.3.3. However, contrary to the approaches based on this definition, no check for the extremality of the first derivative is made. Instead, enhancement is done by applying a transformation to the obtained principal second derivatives which makes linear structures more prominent, provides continuity in the line direction in case of signal loss, and suppresses unwanted structures such as blobs or surfaces having a line profile. Since the line enhancement function favors lines with circular cross sections over lines with elliptical cross sections, the latter lines may be severely distorted. Furthermore, another kind of distortion may be introduced by the continuity term, which will extract lines with a small diameter with a too large width because narrow lines are interpreted as signal loss. This problem is partially overcome by integrating the responses of the filter at multiple scales, where the individual responses are combined taking into account a noise estimate, which is obtained from a region in the image marked by the user.

The approach given in [94, 95] is based on the model that a line has two corresponding edges, one on each side of the line. The edges can be extracted by applying two edge detection filters, which are first derivatives of the Gaussian with different sign. If the two respective responses are combined in a non-linear way, a sharp maximum of the filter at the position of the line is obtained. The algorithm uses the minimum of the two responses, and discards negative results. In order for this to work, the $\sigma$ of the Gaussians has to be commensurate with the line width, i.e., $\sigma$ must be half the line width. The direction in which the two edge operators have to be applied is determined from the eigenvalues of the Hessian matrix of the image, as for the ridge detectors in Section 2.3.3. The single-scale operator thus detects lines of a certain, fixed width. The procedure can be iterated for various $\sigma$. If the maximum of the filter over all $\sigma$ is taken, an estimate of the width and contrast of the line are returned by the algorithm in addition to the line position. This approach can also be extended to 3D images, where surfaces and lines can be extracted. For surfaces, the extension is obvious: The edge operators are applied in the direction of the largest eigenvalue of the Hessian, as in 2D. For 3D lines, the edge operators are applied in the directions of the two largest eigenvalues, and the minimum of the two responses is returned. The problem with this approach is that it is computationally very expensive since multiple convolutions with different $\sigma$ have to be performed. Furthermore, the line positions are only returned with pixel accuracy. For 3D lines that do not have an exactly circular diameter, the algorithm chooses some kind of average of the two principal diameters for the width of the line. Finally, the width estimation can only be as precise as the quantization for $\sigma$. Hence there is a tradeoff between execution time and accuracy.

A completely different approach is taken in [135, 136]. Lines are defined as the “fuzzy” medial axes of the image. The algorithm first extracts a measure of “boundariness” from the image by applying a Gaussian gradient operator, i.e., an edge detector. After this, a medial response image is computed by integrating the boundariness along a circle commensurate with the scale $\sigma$ used for the edge detector. This is similar to a Hough transformation, and yields an accumulator image for one particular scale, i.e., width of lines. The procedure is repeated for selected scales to get a three dimensional multi-scale response image. Finally, lines are extracted from this 3D image by using the restricted extremum definition of Section 2.3.3. Since the image and scale dimensions use different units, the calculation of the first and second derivatives has to be appropriately modified. The ridge detection is implemented as a zero crossing detector, and therefore usually yields two pixel wide ridges. One problem with this approach is that the medial response yields large values not only for lines, but also for objects for which the circles used to integrate the response touch the object in more than one place, e.g., at edges meeting in a corner. Furthermore, since multiple, finely sampled scales have to be used to make the algorithm produce meaningful results, the approach is computationally very expensive.

The above approach is extended in [137, 149] to different boundariness integrating functions in arbitrary dimensions. The resulting medial axes are called "cores" in these papers. A semi-automatic approach to extract 3D lines and their widths based on these ideas is presented in [8]. In this paper, lines are defined by restricted maxima in the image, similar to the approaches of Section 2.3.3. To extract a single ridge, the user must click near the line location to indicate a starting point. An initial point on the ridge is then found by using a conjugate gradient algorithm that searches for points where the sum of the squares of the largest two eigenvalues of the Hessian matrix is sufficiently close to zero. All other points on the ridge are subsequently extracted by taking a small step in the ridge direction and using the conjugate gradient algorithm to get back onto the true ridge. An estimate of the line width is then obtained by finding an optimal $\sigma$ of the integrating function of the boundariness of the line, in this case a Laplacian of Gaussian operator. The line width is returned as this optimal $\sigma$. One advantage of this approach is that because of the core definition used to extract the line width, the estimated width is relatively insensitive to small disturbances. However, as discussed in [8], this insensitivity causes the method to work well only for lines with roughly circular cross sections.

Finally, another entirely different approach is the basis of [55], where regions, line-like, and point-like features are extracted simultaneously from the image. Line-like features are either edges or lines, while point-like features are either junctions or circular symmetrical structures. The whole approach is based on using the average squared gradient, i.e., the matrix $\Gamma z = g_\sigma * \nabla z \nabla z^T$, where $g_\sigma$ is a Gaussian smoothing kernel. Feature extraction is done in four steps. First, the noise variance, for which a linear dependency on the gray values is assumed, is estimated from the image. Knowledge about the noise characteristics is important for the later classification of the image points into the three feature types, since the noise variance determines various parameters in the significance tests used to do the classification. Second, an information preserving restoration is performed. This is based on locally estimating the true gradient of the image by optimizing an energy function which takes the local statistics of the gradient into account. After this, the image is classified into the three feature types. Regions are made up by points for which the trace of $\Gamma z$ is small, i.e., where the gradient magnitude is small. Line and point like features are discerned by the ratio of the two eigenvalues of $\Gamma z$. For point like features this ratio will be close to one. Finally, a localization step is performed. The location of points and edges is determined by minimizing a function which describes the sum of the distances of a point or edge from all the lines given by the gradient directions and points in a window of a certain size. Currently, the localization step is only performed for point-like features. What is interesting in this approach is that lines can be extracted using the average squared gradient. This works because the gradient of a line will yield two impulse responses at its edges. Although, technically speaking, the square of an impulse function does not exist, one can assume a very slightly smoothed line to be able to obtain the squared gradient. If these two impulses are convolved with a Gaussian kernel, the two maxima will move towards each other, until there is only one maximum for an appropriately chosen $\sigma$. Of course, edges result in a single maximum for all $\sigma$. Hence, the line-like features have to be classified into lines and edges. This is done based on a complex function involving averaged first and second derivatives. One problem with this approach is that lines with an asymmetrical profile may be classified as edges. Another problem is that asymmetrical lines will lead to the extracted line positions being biased. A general problem of the approach in general is that the classification is not complete, i.e., point and line-like features are usually surrounded by points which do not belong to any of the three classes, and are classified as “background.”

## 2.5 Approaches Using a Line Model

This section discusses algorithms that make use of an explicit line model. At heart, they are basically ridge detection schemes. The line model is mainly used to obtain an estimate of the line width.

The approach presented in [114, 115] assumes the lines to have a cross section in the shape of a Gaussian function. The half width of the line is assumed to be given by the standard deviation $\sigma_0$ of the Gaussian. The algorithm uses multiple scales, i.e., multiple values of $t = \sigma^2$ to extract the lines. Line points at a single scale are detected by the scheme described in Section 2.3.3, i.e., the first derivative in the direction of maximum curvature must be zero. The novelty of this approach is that it defines a so-called $\gamma$-normalized scale-space response, which basically is $t^\gamma$ times a certain measure $R$ of “ridgeness,” e.g., the maximum of the magnitude of the two principal second derivatives or the squared difference of their squared magnitudes. The parameter $\gamma$ is chosen such that the response reflects the width of the line, i.e., it takes on its maximum value for the scale $t$ equal to the with of the line. Therefore, the additional condition the algorithm uses is that $\frac{\partial}{\partial t}R(x, y, t) = 0$. Because the scale-space description of the signal fulfills the diffusion equation $\frac{\partial}{\partial t}z(x, y, t) = \frac{1}{2}\Delta z(x, y, t)$, the partial derivative along the scale direction can be expressed by the partial derivatives in the spatial domain. This procedure leads to a line being a one dimensional manifold in the two-plus-one dimensional scale-space. In order to select salient lines, the $\gamma$-normalized scale-space response is integrated along this curve to define a saliency measure. This definition has one big deficiency. For objects that consist of multiple lines as sub-objects, e.g., a highway with two separate carriage ways, there will be multiple salient lines for the same object. In the example, the algorithm would detect one line for each carriage way at a fine scale, while additionally returning a single line representing the entire highway at a coarse scale. For more complex objects it may even happen that the extracted lines of different scales intersect each other. Furthermore, since partial derivatives of up to order six are required at multiple scales, the algorithm is extremely computationally expensive. Finally, for asymmetrical lines, i.e., lines with different contrast on each side of the line, the extracted results will in general be rather far from the true line positions because they are extracted at a scale at which smoothing has already introduced a significant bias.

A similar approach to extract 2D and 3D lines is given in [118, 119]. The authors discuss four types of line profiles: Gaussian, bar-shaped, triangular, and parabolic. As in the above approach, a $\gamma$-normalized scale-space response is defined for these profiles based on the second derivatives of the image intensity. For two-dimensional lines, the response is given by $\sigma^\gamma|\lambda_p|/|\lambda_l|$, while for three-dimensional $\sigma^\gamma(|\lambda_{p_1}| + |\lambda_{p_2}|)/|\lambda_l|$ is used. Here, $\lambda_l$ denotes the second derivative in the line direction, while the $\lambda_p$ denote the second derivatives perpendicular to the line direction. The authors show that for each of the profiles, the optimal $\sigma$ selected by their response function is close to $0.4$ times the width of the line at half the maximum intensity. However, since they do not require the intensity to have a maximum perpendicular to the line direction, the algorithm responds to edges as well as lines. Therefore, an "edgeness" criterion is introduced to remove these unwanted responses. Line points are then extracted by conventional non-maximum-suppression [29]. Whether this is done on the edgeness image or the response image is not clear from the articles. The optimal $\sigma$ needed to extract the lines is either estimated from an iteration through scale-space, similar to the above approach, or assumed to be given by the user. This latter approach in effect means that lines are assumed to either have constant width or to have constant contrast. Under these assumptions, the width and contrast of the line can be estimated from the $\gamma$-normalized scale-space response. Since this assumption will rarely be true for real images, the extracted width often does not correspond to the true width of the lines in the image. Furthermore, the 3D response function essentially assumes lines to be cylindrical, i.e., have a circular cross section. In order to overcome this problem, a snake-based approach is used to extract the boundaries of the line perpendicular to the line direction. However, no details about the algorithm are given.

A completely different approach to the problem of extracting lines along with their width is presented in [26, 27]. This scheme continues the work begun in [24, 25], described in Section 2.3.3. As mentioned there, the approach is a restricted-extrema-type ridge detector. It uses a parabola as the model of the line profile. Since the algorithm cannot be analyzed in scale-space because of the facet model, this essentially means that the assumed parabolic model is meaningless. The true model is that a line is bounded by an edge on each side, which can be extracted by an edge detector. In the algorithm, edges are defined by requiring the second derivative in the direction of the gradient to be zero within a pixel's boundaries, and the third directional derivative to be negative. Therefore, a cubic facet model is used to extract line and edge points simultaneously, and to link them into lines and edges individually. After this, two edges are matched to each extracted line as a whole object. No details on how this is done are given in [26, 27]. The paper leaves open many questions, e.g., what happens at junctions, where edges typically continue around the junction and must be matched to another line as well. Furthermore, because the facet model convolution masks are rather poor estimators for the partial derivatives of the image, the line and edge positions may be severely biased, especially for asymmetrical lines. Since this behavior cannot be modeled analytically, the heuristic to correct the line positions by putting them in the middle of the two edges proposed in [26, 27] may improve the extraction results, but in many cases this correction will not be meaningful since the edge positions will be biased as well, and very likely not by the same amount for each edge.

## 2.6 Sub-Pixel Edge Detection

Sub-pixel edge detection operators fall into three broad categories. Algorithms in the first category achieve sub-pixel resolution by assuming an edge model, typically a step edge. The gray value moments of the assumed edge model over a window of a certain extent are calculated, and the parameters of the step edge are locally determined by matching the gray value moments in the image to the moments of the ideal edge.

The approach described in [176] determines the edge location in the 1D case by matching the first three moments of a non-decreasing sequence of gray values to the step edge model. The resulting estimates of the edge location and gray values on each side of the edge can be calculated analytically. An analysis of the operator in the presence of noise is carried out, which shows that additive noise biases the edge locations to the center of the window. In the 2D case, the edge model has four parameters: the two intensities on both sides of the edge, the edge orientation, and the distance of the edge from a suitably chosen origin. The edge location and intensities are found by matching the first three gray value moments in the image to the model edge over a circle containing 69 pixels. Finally, the edge orientation is estimated from a suitably defined weighted mean of the gray values over this disc.

A very similar approach is given in [121]. Although the authors claim to use spatial moments, it is not clear how edge positions could be calculated from spatial moments alone, and the formulas given in [121] clearly are gray value moments. Again, the edge location and contrast can be derived analytically, and the formulas are quite similar to the ones in [176]. The only difference to [176] seems to be that the authors notice that the sub-pixel edge location is biased if a square optical transfer function is assumed for the imaging device. This bias can be modeled, and hence removed. Furthermore, the authors claim better behavior of their approach compared to other edge detection schemes in the presence of noise, most notably compared to the zero crossings of a Laplacian of Gaussian filter. However, their experiments indicate that this is only true for very high signal to noise ratios, and that their operator deteriorates much quicker for low signal to noise ratios.

The second category of operators obtains a sub-pixel estimate of the edge position by a least-squares fit of an assumed edge model to the image data over a window of a certain size.

The algorithm in [140] first determines a rough estimate of the orientation of the edge by a least-squares fit of a plane to the image data in a window of size $5 \times 5$. Then, the direction estimate is improved by fitting a 1D cubic curve, which is extruded to a 2D surface, to the data. The decision whether an edge pixel is present is made based on the $F$-statistic. After this, an extruded 1D $\tanh$ curve is fitted in the direction perpendicular to the edge. With this, the edge contrast as well as the sub-pixel edge position are determined. Additionally, an extruded 1D quadratic curve is fitted to the data to check if the edge may be a line. Finally, the edge pixels are thresholded using their contrast. One disadvantage of this approach is that the least-squares fits are non-linear, and thus computationally expensive to solve since they cannot be rewritten as convolutions.

The approach given in [87] uses a rotated straight step edge as the model to fit to the image. This is the same model that was also used above in [176]. In the continuous case, the straight edge is approximated locally by five Fourier coefficients, and from these, explicit formulas for the four parameters of the edge are derived. In the discrete case, the five Fourier coefficients are obtained by convolving the image with five suitably chosen $3 \times 3$ masks. Since the purpose of the operator is to enhance the resolution of images during a scaling operation, no study about the precision of the edge positions is made.

A different approach is given in [90]. Edges are extracted using a quadrature filter pair, which essentially are second and third derivative operators. In 1D, image points are classified into step, roof, and ramp edges based of the output of the quadrature filters. For example, a positive step edge is given by the maximum of the anti-symmetrical filter, i.e., the third derivative, and a zero crossing in the symmetrical filter, i.e., the second derivative. Based on this classification, the respective edge model is fitted to the image data to calculate the sub-pixel edge location. In 2D, the edge direction is calculated by maximizing the energy of the quadrature filters using a least-squares regression algorithm, and the 1D edge model is fitted in the direction perpendicular to the edge. Experiments on real images show that an accuracy of one tenth of a pixel can be achieved despite the fact that the equivalent of a third derivative is used.

In the final category of algorithms, sub-pixel resolution is achieved by interpolation of the image data or their derivatives. The algorithm described in [82] first determines the zero crossings of a Laplacian of Gaussian filtered image at the original image resolution. The zero crossings are obtained by a predicate-based approach on the 8-neighborhood of a pixel, which uses eleven distinct cases to obtain 24 possible zero crossing locations. The predicates are chosen carefully such that edge connectivity is assured. A local polynomial of the form $k_0 + k_1 r + k_2 c + k_3 r^2 + k_4 rc + k_5 c^2 + k_6 r^2 c + k_7 c^2 r + k_8 r^2 c^2$ is fitted locally at each zero crossing pixel by using facet model convolution masks, and the image is re-sampled at a $n$ times higher resolution around the zero crossing points. Then, zero crossings are again calculated using the above algorithm in the higher resolution image. Thus, sub-pixel resolution is at most $n$ times the normal resolution of the image. Furthermore, an analysis is carried out on how much an image can be down-sampled before the extracted edges in the lower resolution image deteriorate noticeably. If the original image is convolved with a Laplacian of Gaussian with scale $\sigma$, the authors derive that the image can be sub-sampled by averaging by a factor of $k < 0.75\sigma$. Thus, substantial computational savings are possible even for $\sigma$ as small as $2.7$.

A rather different scheme is proposed in [145]. Here, the image is re-sampled to a hexagonal grid, and six first differences are computed for each pixel, corresponding to the six directions of the hexagonal grid. After this, the difference of the positive and negative parts of the six first differences is computed, and a rough estimate of the edge direction is obtained. Then, the algorithm makes the assumption that the maximum of the output of this filter in the direction perpendicular to the edge can be approximated by a Gaussian of width $\sigma = 1.7$ and extrapolates the position of the maximum of this Gaussian as the edge position. The algorithm is rather heuristic in nature and no formal justification of the approach is given.

In contrast to the above two approaches, the edge detection scheme given in [177] achieves sub-pixel accuracy by comparing the output of an edge detector, e.g., a Deriche, Canny, or Laplacian of Gaussian operator, obtained from the image with an analytical prediction based on the characteristics of the respective operator. Implicitly this requires some sort of smoothing to be done in the imaging device. However, the required smoothing function is not elaborated. From the pixel precise edge location and the response of the operator, e.g., the first and second derivatives in case of the Canny or Deriche operators, a correction is calculated, which is added to the edge location to gain a sub-pixel precise estimate of the edge location.

## 2.7 Requirements for a Line Detection Algorithm

After having discussed the existing line, ridge, and edge detection algorithms, a list of requirements that a good line extraction algorithm should fulfill can be derived.

- *Lines should be detected with a very high rate of true positives, while returning as few false positives as possible.* This will on the one hand make the linking of the individual line points into lines much easier, and on the other hand provide better input for later, high-level stages of reasoning about the objects in the scene. Note that this implicitly also requires that the algorithm should return no double responses. Therefore, simple gray-value-based schemes are immediately ruled out as candidates for line detection algorithms, while facet model ridge detectors are only moderately desirable because of the large number of double responses.

- *Salient lines should be easily selectable without complex grouping algorithms.* This is strongly connected with the previous requirement, and rules out watershed-based approaches since it is well known that they severely over-segment an image, responding even to the smallest watershed in the image, making the problem of selecting salient lines very complex.

- *The criterion to detect lines should possess as few inherent problems as possible.* For example, the extraction of ridges by curvature maxima of the contour lines does not work for perfectly horizontal ridges and for ridges with a flat profile, as discussed in Section 2.3.1, while the definition of ridges by curvature maxima of the image also does not work for ridges with a flat profile and responds to edges as well. While one could argue that such cases are not generic [154], most of the interesting lines in images are man-made, e.g., roads in aerial images, and therefore almost per definition non-generic.

- *Lines of arbitrary widths should be detectable.* The algorithm should not be restricted to small line widths.

- *In order to give meaning to the term "width," the line detector should use an explicit model of a line.* This suggests using some sort of scale-space approach to extract the line width. As discussed in Section 2.1, analytical results of the behavior of the line model in scale space can almost exclusively be obtained with linear scale-space descriptions only. Therefore, other scale-space definitions, e.g., anisotropic diffusion, can be ruled out as candidates.

- *It must be possible to model the bias of the extracted features.* It is well known that features will be displaced, i.e., biased, due to scale-space smoothing. Therefore the major requirement for the algorithm is that it must be able to model this bias and to remove it.

- *The line position and width should be returned with sub-pixel resolution and accuracy.* This is often highly desirable in cases where the inner scale of the image cannot be increased, e.g., if the camera is unable to zoom to the object in question, but precise measurements of the object are nevertheless paramount. Furthermore, sub-pixel accuracy enables feature extraction on a scaled-down copy of an image in order to save valuable computational resources.

- *The algorithm should link individual line points into lines.* This facilitates later reasoning processes such as perceptual grouping. The linking algorithm should produce a topologically sound representation of the lines and their junctions, i.e., no extended junction areas.

- *Finally, the line detector should be reasonably fast.* This means, for instance, that line widths should be extracted without resorting to a scale-space iteration, if possible. Instead, the line width should be extracted from one scale of the image.

# Chapter 3

# Line and Edge Extraction in 2D Images

As was discussed in Chapter 1, conceptually, one may think of a line as a one-dimensional manifold in the two-dimensional space $\mathbb{R}^2$ with another very important characteristic: a well defined width. Therefore, curvilinear structures in 2D can be modeled as curves $s(t)$ that exhibit a characteristic 1D line profile in the direction perpendicular to the line, i.e., perpendicular to $s'(t)$. By far the most important type is a line with equal polarity, i.e., the line is either brighter or darker than the background on each side of the line. The extraction of this type of line is discussed in Section 3.1, which describes the line detection algorithm in much more detail than the author's previous publications on this subject [167, 168, 169, 171, 172]. Another type of line, which does not occur too frequently for interesting objects in real images, has different polarity, i.e., the background is darker on one side of the line, while it is lighter on the other side. This type of lines is discussed in Section 3.2. Finally, in Section 3.3 it is shown that the algorithm for line extraction developed in Section 3.1 quite naturally leads to a new method of sub-pixel precise edge extraction.

## 3.1 Lines with Equal Polarity

### 3.1.1 Models for Line Profiles

As described above, lines exhibit a characteristic profile across the line at each line point. Hence, many of the problems concerning the extraction of lines are essentially one-dimensional in nature since the analysis can be carried out for the one-dimensional line profile. Therefore, this section first discusses different line profiles in 1D.

When considering one-dimensional line profiles, probably the first model that comes to mind is the symmetrical bar-shaped profile given by

$$f_b(x) = \begin{cases} h, & |x| \le w \\ 0, & |x| > w \end{cases} . \tag{3.1}$$

This type of profile occurs for many interesting classes of features under ideal imaging conditions, e.g., the strokes of a character [187, 188]. An example profile of this type is shown in

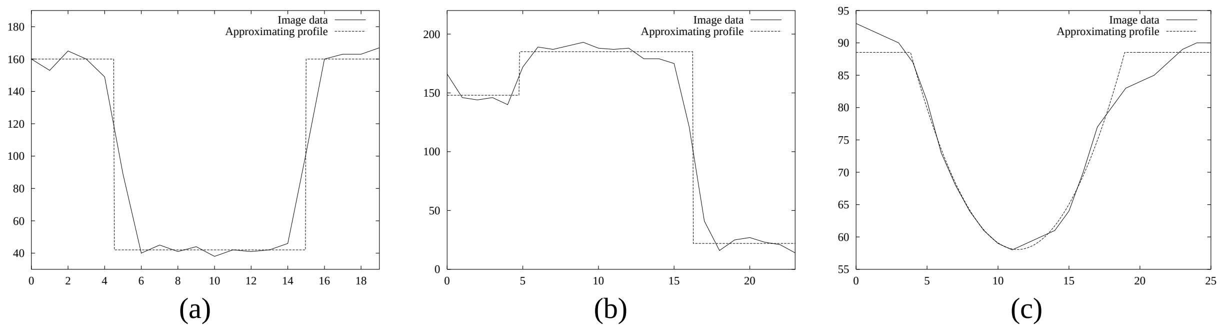


Figure 3.1: Examples of the three types of line profiles considered matched to profiles obtained from typical images. (a) Profile from an image containing written text exhibiting profile $f_b$. (b) Profile from an aerial image containing a road having profile $f_a$. (c) Profile from an X-ray angiogram containing a blood vessel exhibiting profile $f_p$.

Figure 3.1(a). However, often the symmetry of the profile does not hold in reality, e.g., for roads in aerial or satellite images [2, 10, 11, 69, 129, 157], where often different types of objects or vegetation leads to significantly different gray values on each side of the line. Nevertheless, the flat profile across the line does hold for these types of objects. Therefore, the asymmetrical bar-shaped profile will be considered as the most relevant type of line profile in this work. It is given by

$$f_a(x) = \begin{cases} 0, & x < -w \\ 1, & |x| \le w \\ a, & x > w \end{cases} , \tag{3.2}$$

where $a \in [0, 1]$. General lines of height $h$ can be obtained by considering a scaled asymmetrical profile, i.e., $hf_a(x)$. See Figure 3.1(b) for an example from an aerial image. Although the asymmetrical bar-shaped profile is the most commonly occurring profile, sometimes lines do not exhibit the sharp drop in intensity at the sides of the line. A more gradual drop in intensity is modeled by the symmetrical parabolic profile given by

$$f_p(x) = \begin{cases} h(1 - (x/w)^2), & |x| \le w \\ 0, & |x| > w \end{cases} . \tag{3.3}$$

This profile is an excellent approximation of the type of profile occurring, e.g., in finger prints [85, 124], or for the elliptical profile that often occurs in X-ray imaging [30, 34]. Figure 3.1(c) displays an example profile from a coronary angiogram. The advantage of the parabolic profile is that it is much easier to analyze analytically than the elliptical profile. However, the main use of the parabolic profile in this work is to motivate the derivation of the line extraction algorithm and to derive properties a line extraction algorithm should possess.

### 3.1.2 Detection of Line Points on Profiles

In order to detect line points with a profile given by (3.3) in an image $z(x)$ without noise, it is sufficient to determine the points where $z'(x)$ vanishes. However, it is usually convenient to select only salient lines. A useful criterion for salient lines is the magnitude of the second derivative $z''(x)$ in the point where $z'(x) = 0$. Bright lines on a dark background will have $z''(x) \ll 0$ while dark lines on a bright background will have $z''(x) \gg 0$. Please note that for the ideal line profile $f_p''(x) = -2h/w^2$ for all $|x| \leq w$.

Real images will contain a significant amount of noise, and thus the scheme described above is not sufficient. In this case, as was discussed in Section 2.1, the first and second derivatives of $z(x)$ should be estimated by convolving the image with the derivatives of the Gaussian smoothing kernel since, under certain, very general, assumptions, it is the only kernel that makes the inherently ill posed problem of estimating the derivatives of a noisy function well posed. The Gaussian kernels are given by:

$$\begin{aligned} g_\sigma(x) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} && (3.4) \\ g_\sigma'(x) &= \frac{-x}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2}{2\sigma^2}} && (3.5) \\ g_\sigma''(x) &= \frac{x^2-\sigma^2}{\sqrt{2\pi}\sigma^5} e^{-\frac{x^2}{2\sigma^2}} \ . && (3.6) \end{aligned}$$

The responses, i.e., the estimated derivatives, of the parabolic line profile will then be:

$$\begin{aligned} r_p(x,\sigma,w,h) &= g_\sigma(x) * f_p(x) \\ &= \frac{h}{w^2}\Big((w^2 - x^2 - \sigma^2)(\phi_\sigma(x+w) - \phi_\sigma(x-w)) - \\ &\quad 2\sigma^2 x(g_\sigma(x+w) - g_\sigma(x-w)) - \\ &\quad \sigma^4(g_\sigma'(x+w) - g_\sigma'(x-w))\Big) && (3.7) \\ r_p'(x,\sigma,w,h) &= g_\sigma'(x) * f_p(x) \\ &= \frac{h}{w^2}\Big(-2x(\phi_\sigma(x+w) - \phi_\sigma(x-w)) + \\ &\quad (w^2 - x^2 - 3\sigma^2)(g_\sigma(x+w) - g_\sigma(x-w)) - \\ &\quad 2\sigma^2 x(g_\sigma'(x+w) - g_\sigma'(x-w)) - \\ &\quad \sigma^4(g_\sigma''(x+w) - g_\sigma''(x-w))\Big) && (3.8) \\ r_p''(x,\sigma,w,h) &= g_\sigma''(x) * f_p(x) \\ &= \frac{h}{w^2}\Big(-2(\phi_\sigma(x+w) - \phi_\sigma(x-w)) - \\ &\quad 4x(g_\sigma(x+w) - g_\sigma(x-w)) + \\ &\quad (w^2 - x^2 - 5\sigma^2)(g_\sigma'(x+w) - g_\sigma'(x-w)) - \\ &\quad 2\sigma^2 x(g_\sigma''(x+w) - g_\sigma''(x-w)) - \\ &\quad \sigma^4(g_\sigma'''(x+w) - g_\sigma'''(x-w))\Big) && (3.9) \end{aligned}$$

where

$$\phi_\sigma(x) = \int_{-\infty}^{x} e^{-\frac{t^2}{2\sigma^2}} dt \ . \qquad (3.10)$$

Equations (3.7)–(3.9) give a complete scale-space description of how the parabolic line profile $f_p$ will look like when it is convolved with the derivatives of a Gaussian kernel. Figure 3.2 shows the responses for an ideal line with $w = 1$ and $h = 1$, i.e., a bright line

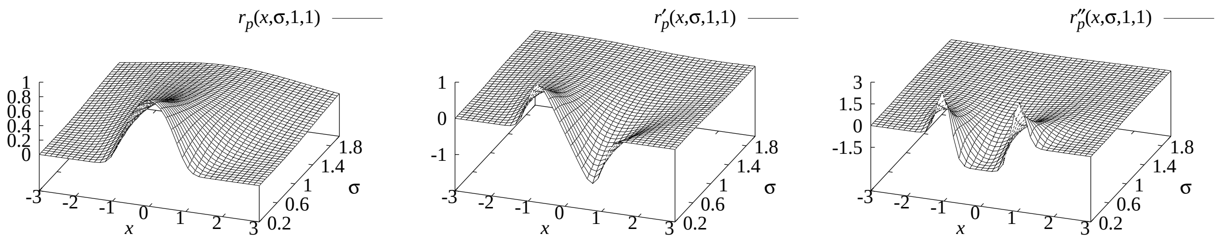


Figure 3.2: Scale-space behavior of the parabolic line $f_p$ when convolved with the derivatives of Gaussian kernels for $x \in [-3, 3]$ and $\sigma \in [0.2, 2]$.

on a dark background, for $x \in [-3, 3]$ and $\sigma \in [0.2, 2]$. As can be seen from this figure, $r'_p(x, \sigma, w, h) = 0 \Leftrightarrow x = 0$ for all $\sigma$. Furthermore, $r''_p(x, \sigma, w, h)$ takes on its maximum negative value at $x = 0$ for all $\sigma$. This is depicted in Figure 3.4(a), where the scale-space behavior of the second derivative $r''_p(x, \sigma, w, h)$ of parabolic line at $x = 0$ is shown. Hence it is possible to determine the precise location of the line for all $\sigma$. In addition it can be seen that the ideal line will be flattened out as $\sigma$ increases as a result of smoothing. This means that if large values for $\sigma$ are used, the threshold to select salient lines will have to be set to an accordingly smaller value.

From this analysis it can be seen that the desirable properties a line extraction algorithm should exhibit are that it can detect the precise line position for an arbitrary amount of smoothing, and that the magnitude of the second derivative is always maximum at the line position. Thus, salient lines can be selected based on their second derivative for all $\sigma$.

After the analysis of the parabolic profile has given us desirable properties for a line extraction algorithm, the more common case of a bar-shaped profile is considered next. For this type of profile without noise no simple criterion that depends only on $z'(x)$ and $z''(x)$ can be given since $z'(x)$ and $z''(x)$ vanish in the interval $[-w, w]$. However, if the bar profile is convolved with the derivatives of the Gaussian kernel, a smooth function is obtained in each case. The responses are:

$$\begin{aligned} r_b(x, \sigma, w, h) &= g_\sigma(x) * f_b(x) \\ &= h\Big(\phi_\sigma(x + w) - \phi_\sigma(x - w)\Big) \qquad (3.11) \\ r'_b(x, \sigma, w, h) &= g'_\sigma(x) * f_b(x) \\ &= h\Big(g_\sigma(x + w) - g_\sigma(x - w)\Big) \qquad (3.12) \\ r''_b(x, \sigma, w, h) &= g''_\sigma(x) * f_b(x) \\ &= h\Big(g'_\sigma(x + w) - g'_\sigma(x - w)\Big) \ . \qquad (3.13) \end{aligned}$$

Figure 3.3 shows the scale-space behavior of a bar profile with $w = 1$ and $h = 1$ when it is convolved with the derivatives of a Gaussian. It can be seen that the bar profile gradually becomes "round" at its corners. The first derivative will vanish only at $x = 0$ for all $\sigma > 0$ because of the infinite support of $g_\sigma(x)$. Hence, in theory lines can be detected precisely for all $\sigma$. However, the second derivative $r''_b(x, \sigma, w, h)$ does not take on its maximum negative value at $x = 0$ for small $\sigma$. In fact, for $\sigma \leq 0.2w$ it is very close to zero, as can be seen from Figure 3.4(b). Furthermore, there will be two distinct minima in the interval $[-w, w]$. It is,

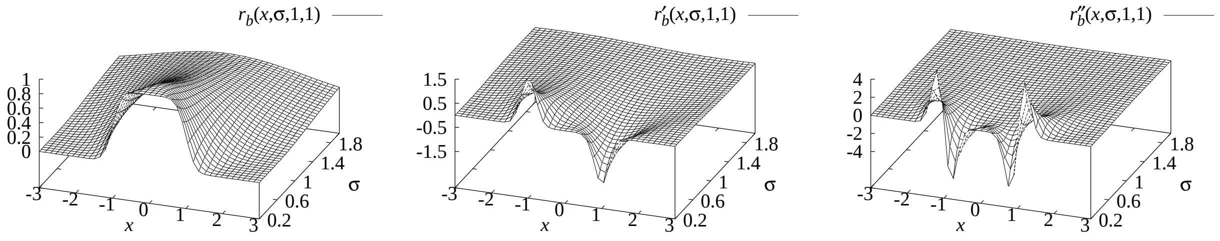


Figure 3.3: Scale-space behavior of the bar-shaped line $f_b$ when convolved with the derivatives of Gaussian kernels for $x \in [-3,3]$ and $\sigma \in [0.2,2]$.

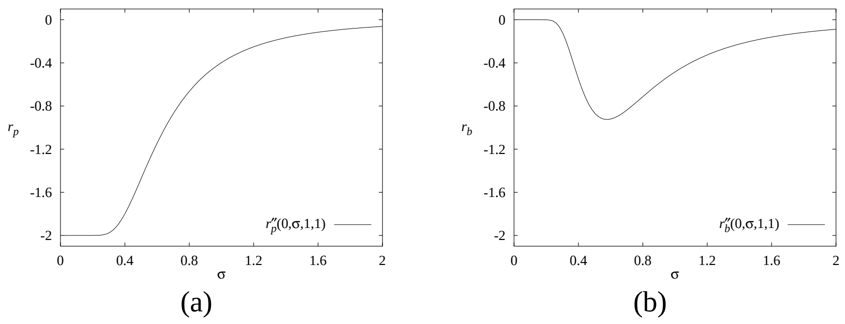


Figure 3.4: Scale-space behavior of the parabolic line $f_p$ (a) and the bar-shaped line $f_b$ (b) at $x = 0$ for $\sigma \in [0,2]$. Clearly, $f_b$ reaches the maximum magnitude of response at $\sigma = w/\sqrt{3}$, while $f_p$ has its maximum response at $\sigma = 0$. Note that the responses are negative because a bright line was used.

however, desirable for $r''_b(x,\sigma,w,h)$ to exhibit a clearly defined minimum at $x = 0$ since salient lines are selected by this value. This value of $\sigma$ is given by the solution of

$$\frac{\partial}{\partial\sigma} r''_b(0,\sigma,w,h) = 0 \ . \tag{3.14}$$

It is not very hard to show that

$$\sigma \geq \frac{w}{\sqrt{3}} \tag{3.15}$$

has to hold for this. Furthermore, it is obvious that $r''_b(x,\sigma,w,h)$ will have its maximum negative response in scale-space for $\sigma = w/\sqrt{3}$. This means that the same scheme as described above can be used to detect bar-shaped lines as well. However, the restriction (3.15) should be observed to ensure that salient lines can be selected based on the magnitude of their second derivative.

The discussion so far was concerned with extracting the line points. Of course, the second most important feature of a line is its width $w$. From the line models given in Section 3.1.1 it is obvious that a line is bounded by an edge on each side. The same methods as above can be used to derive the scale-space behavior of the line width, i.e., the edge positions. They are given by the maxima of $|r'_b(x,\sigma,w,h)|$ or the zero crossings of $r''_b(x,\sigma,w,h)$, where additionally $r'''_b(x,\sigma,w,h)r'_b(x,\sigma,w,h) < 0$ is required. In the one-dimensional case these definitions are equivalent. It is well known that in the two-dimensional case they lead to the same results if and only if the mean curvature of the image $z(x, y)$ vanishes [16]. As will be discussed in Section 3.1.9, the implementation uses the maxima of the gradient definition. This will give the correct edge locations unless the curvature of the line is very high compared to $\sigma$ [41], in which case the width will be extracted too small. Since the scale-space analysis to derive the edge positions involves equations which cannot be solved analytically, the calculations must be done using a root finding algorithm [152]. This analysis results in a first contact with biased extraction results.

## 3.1.3 Examples of Biased Extraction Results

Figure 3.5 shows the location of the line and its corresponding edges for $w \in [0, 4]$ and $\sigma = 1$. Note that the ideal edge positions are given by $x = \pm w$. From (3.13) it is apparent that the edges of a line can never move closer than $\sigma$ to the real line, and thus the width of the line will be estimated significantly too large for narrow lines. This effect was also studied qualitatively in the context of edge detection in [162] and [31], where a solution was proposed based on edge focusing by successively applying filters with a smaller value of $\sigma$, while tracking the edges through the scale-space description thus constructed. However, for the bar-shaped model presented here it is possible to invert the map that describes the edge position, and therefore the edges can be localized very precisely once they are extracted from an image without repeatedly applying the filter, provided the bar-shaped line model holds for the lines in the image.

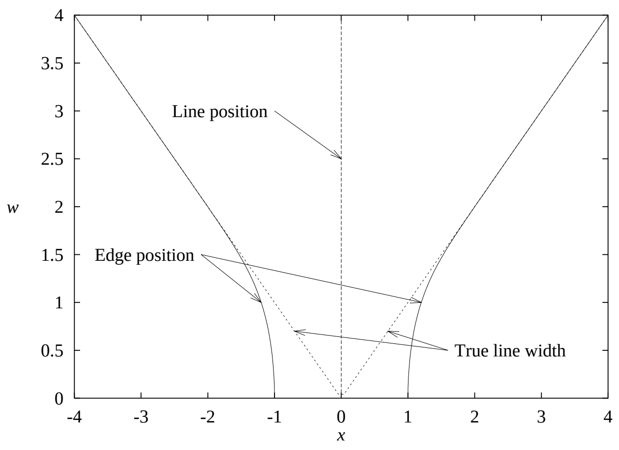


Figure 3.5: Location of a bar-shaped line with width $w \in [0, 4]$ and its edges for $\sigma = 1$.

For completeness sake, the same analysis has been done on the parabolic line profile. The results of extracting the line and edge locations on this type of profile for $w \in [0, 4]$ and $\sigma = 1$ are displayed in Figure 3.6. Analogously to the bar-shaped line, the extracted width cannot move closer to the line than $\sigma$. Again, the width of small lines will be extracted significantly too large. What is interesting about the parabolic profile is that the extracted widths will be extracted significantly too small for a range of widths. In fact, the extracted line width may be as low as $0.7055w$. Asymptotically, for large $w$ the extracted line width will lie close to the true line width again. Similarly to the bar-shaped line, the map that describes the relation between the extracted and the true widths can be inverted, and hence the true line width can be estimated with high accuracy, again provided the parabolic model holds in the image.

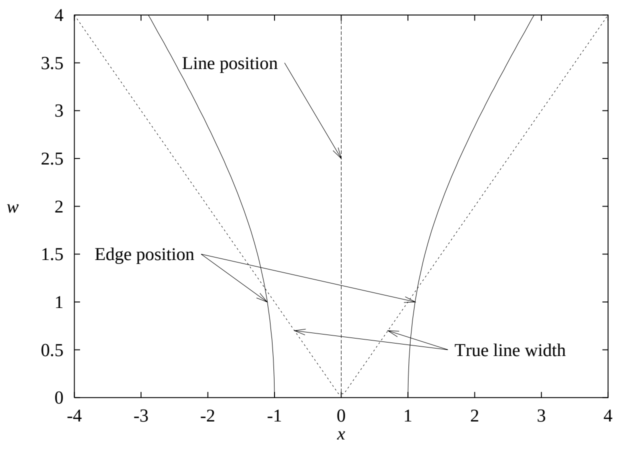


Figure 3.6: Location of a parabolic line with width $w \in [0, 4]$ and its edges for $\sigma = 1$.

The discussion so far has assumed that lines have the same contrast on both sides, which is rarely the case for real images. Let us now turn to the asymmetrical bar-shaped lines given by (3.2). Their responses are

$$\begin{aligned} r_a(x, \sigma, w, a) &= \phi_\sigma(x + w) + (a - 1)\phi_\sigma(x - w) && (3.16) \\ r'_a(x, \sigma, w, a) &= g_\sigma(x + w) + (a - 1)g_\sigma(x - w) && (3.17) \\ r''_a(x, \sigma, w, a) &= g'_\sigma(x + w) + (a - 1)g'_\sigma(x - w) \ . && (3.18) \end{aligned}$$

The location where $r'_a(x, \sigma, w, a) = 0$, i.e., the position of the line, is given by

$$l = -\frac{\sigma^2}{2w} \ln(1 - a) \ . \qquad (3.19)$$

This means that the line will be estimated in a wrong position whenever the contrast is significantly different on both sides of the line. The estimated position of the line will be within the actual boundaries of the line as long as

$$a \leq 1 - e^{-\frac{2w^2}{\sigma^2}} \ . \qquad (3.20)$$

The location of the corresponding edges can again only be computed numerically. Figure 3.7 gives an example of the line and edge positions for $w = 1$, $\sigma = 1$, and $a \in [0, 1]$. It can be seen that the position of the line and the edges is greatly influenced by line asymmetry. As $a$ gets larger the line and edge positions are pushed to the weak side, i.e., the side that possesses the smaller edge gradient.

Note that (3.19) gives an explicit formula for the bias of the line extractor. Suppose that $w$ and $a$ were known for each line point. Then it would be possible to remove the bias from the line detection algorithm by shifting the line back into its proper position. Section 3.1.10 will describe the solution to this problem.

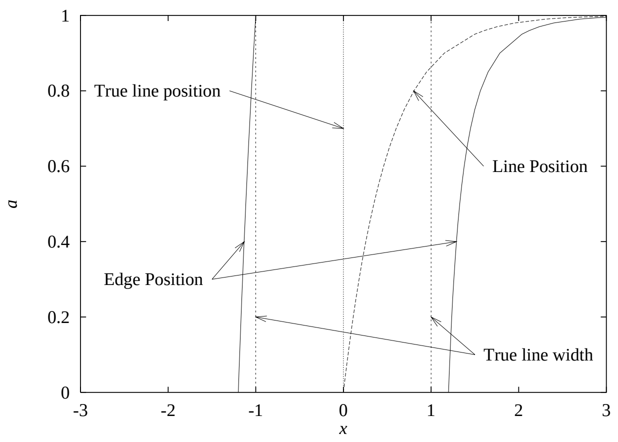


Figure 3.7: Location of an asymmetrical line and its corresponding edges with width $w = 1$, $\sigma = 1$, and $a \in [0, 1]$.

From the above analysis it is apparent that failure to model the surroundings of a line, i.e., the asymmetry of its edges, can result in large errors of the estimated line position and width. Algorithms that fail to take this into account may not return very meaningful results.

It should be noted that in cases where a different asymmetrical line profile might be more appropriate, e.g., an appropriately modified parabolic profile [30, 34], the methodology presented here can be used to model the behavior of the profile in scale-space, and thus yield more precise extraction results.

### 3.1.4 Detection of Line Points on Profiles in Discrete Space

The analysis so far has been carried out for analytical functions $z(x)$. For discrete signals only two modifications have to be made. The first is the choice of how to implement the convolution in discrete space. Integrated Gaussian kernels [111, 112, 113] were chosen as convolutions masks, mainly because the scale-space analysis of Section 3.1.2 directly carries over to the discrete case. An additional advantage is that they give automatic normalization of the masks and a direct criterion on how many coefficients are needed for a given approximation error. The integrated Gaussian is obtained if one regards the discrete image $z_n$ as a piecewise constant function $z(x) = z_n$ for $x \in (n - \frac{1}{2}, n + \frac{1}{2}]$. In this case, the convolution masks are given by:

$$
\begin{aligned}
g_{n,\sigma} &= \phi_\sigma(n + \tfrac{1}{2}) - \phi_\sigma(n - \tfrac{1}{2}) && (3.21)\\
g'_{n,\sigma} &= g_\sigma(n + \tfrac{1}{2}) - g_\sigma(n - \tfrac{1}{2}) && (3.22)\\
g''_{n,\sigma} &= g'_\sigma(n + \tfrac{1}{2}) - g'_\sigma(n - \tfrac{1}{2}) \ . && (3.23)
\end{aligned}
$$

For the implementation the approximation error $\epsilon$ is set to $10^{-4}$ because for images that contain gray values in the range $[0, 255]$ this precision is sufficient. The number of coefficients used is $2\lceil x_0 \rceil + 1$, where $x_0$ is given by $g_\sigma(x_0) < \epsilon/2$ for $g_{n,\sigma}$, and analogously for the other kernels. Of course, other schemes, like discrete analog of the Gaussian [111, 112, 113] or a recursive computation [40, 194] are suitable for the implementation as well. However, for small $\sigma$ the scale-space analysis will have to be modified because these filters have different coefficients compared to the integrated Gaussian.

The second problem that must be solved is the determination of the line location in the discrete case. In principle, one could use a zero crossing detector for this task. However, this would yield the position of the line only with pixel accuracy. In order to overcome this, the second order Taylor polynomial of $z_n$ is examined. Let $r$, $r'$, and $r''$ be the locally estimated derivatives at point $n$ of the image that are obtained by convolving the image with $g_n$, $g'_n$, and $g''_n$. Then the Taylor polynomial is given by

$$p(x) = r + r'x + \frac{1}{2}r''x^2 \quad . \tag{3.24}$$

The position of the line, i.e., the point where $p'(x) = 0$ is

$$x = -\frac{r'}{r''} \quad . \tag{3.25}$$

The point $n$ is declared a line point if this position falls within the pixel's boundaries, i.e., if $x \in [-\frac{1}{2}, \frac{1}{2}]$ and the second derivative $r''$ is larger than a user-specified threshold. Note that in order to extract lines, the response $r$ is unnecessary and therefore does not need to be computed. The discussion of how to extract the edges corresponding to a line point is deferred to Section 3.1.9.

### 3.1.5 Detection of Lines in 2D

As mentioned in the introduction to this chapter, curvilinear structures in 2D can be modeled as curves $s(t)$ that exhibit a characteristic 1D line profile in the direction perpendicular to the line, i.e., perpendicular to $s'(t)$. Let this direction be $n(t)$. Combined with the argument in the last sections, this means that the first directional derivative in the direction $n(t)$ should vanish and the second directional derivative should be of large absolute value. No assumption can and should be made about the derivatives in the direction of $s'(t)$ since changes of the contrast $h$ along the line may result in second directional derivatives along the line direction which are relatively large. If a threshold were imposed on this derivative undesirable broken lines would result.

The only remaining problem is to compute the direction of the line locally for each image point. In order to do this, the partial derivatives $r_x$, $r_y$, $r_{xx}$, $r_{xy}$, and $r_{yy}$ of the image have to be estimated, and this can be done by convolving the image with the following kernels

$$\begin{aligned} g_{x,\sigma}(x,y) &= g'_\sigma(x)g_\sigma(y) && (3.26)\\ g_{y,\sigma}(x,y) &= g_\sigma(x)g'_\sigma(y) && (3.27)\\ g_{xx,\sigma}(x,y) &= g''_\sigma(x)g_\sigma(y) && (3.28)\\ g_{xy,\sigma}(x,y) &= g'_\sigma(x)g'_\sigma(y) && (3.29)\\ g_{yy,\sigma}(x,y) &= g_\sigma(x)g''_\sigma(y) \quad . && (3.30) \end{aligned}$$

The direction in which the second directional derivative of $z(x, y)$ takes on its maximum absolute value is used as the direction $n(t)$. This direction can be determined by calculating the eigenvalues and eigenvectors of the Hessian matrix

$$H(x,y) = \begin{pmatrix} r_{xx} & r_{xy} \\ r_{xy} & r_{yy} \end{pmatrix} \quad . \tag{3.31}$$

The calculation can be done in a numerically stable and efficient way by using one Jacobi rotation to annihilate the $r_{xy}$ term [152]. Let the eigenvector corresponding to the eigenvalue of maximum absolute value, i.e., the direction perpendicular to the line, be given by $(n_x, n_y)$ with $\|(n_x, n_y)\|_2 = 1$. As in the 1D case, a quadratic polynomial is used to determine whether the first directional derivative along $(n_x, n_y)$ vanishes within the current pixel. This point can be obtained by inserting $(tn_x, tn_y)$ into the Taylor polynomial

$$z(x, y) = r + \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} r_x \\ r_y \end{pmatrix} + \frac{1}{2} \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} r_{xx} & r_{xy} \\ r_{xy} & r_{yy} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} , \tag{3.32}$$

and setting the derivative along $t$ to zero. Hence,

$$\begin{aligned} z(tn_x, tn_y) &= r + \begin{pmatrix} tn_x & tn_y \end{pmatrix} \begin{pmatrix} r_x \\ r_y \end{pmatrix} + \frac{1}{2} \begin{pmatrix} tn_x & tn_y \end{pmatrix} \begin{pmatrix} r_{xx} & r_{xy} \\ r_{xy} & r_{yy} \end{pmatrix} \begin{pmatrix} tn_x \\ tn_y \end{pmatrix} \\ &= r + tn_x r_x + tn_y r_y + \frac{1}{2} t^2 n_x^2 r_{xx} + t^2 n_x n_y r_{xy} + \frac{1}{2} t^2 n_y^2 r_{yy} . \end{aligned} \tag{3.33}$$

The derivative of this expression is

$$\frac{\partial}{\partial t} z(tn_x, tn_y) = n_x r_x + n_y r_y + tn_x^2 r_{xx} + 2tn_x n_y r_{xy} + tn_y^2 r_{yy} . \tag{3.34}$$

Hence, the maximum or minimum point is given by

$$(p_x, p_y) = (tn_x, tn_y) , \tag{3.35}$$

where

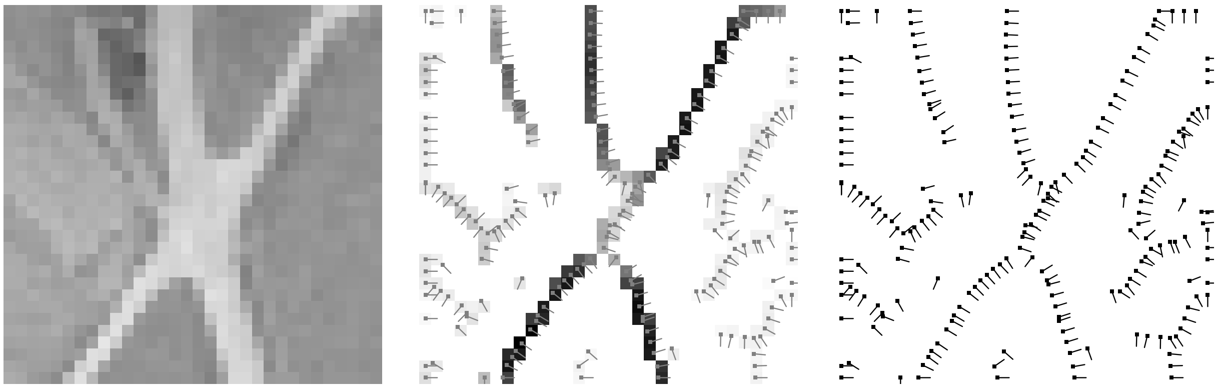
$$t = -\frac{n_x r_x + n_y r_y}{n_x^2 r_{xx} + 2n_x n_y r_{xy} + n_y^2 r_{yy}} . \tag{3.36}$$

Again, $(p_x, p_y) \in [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ is required in order for a point to be declared a line point. As in the 1D case, the second directional derivative along $(n_x, n_y)$, i.e., the maximum eigenvalue, can be used to select salient lines.

As can be seen, this definition of lines is essentially the same as the restricted extrema ridge definition of Section 2.3.3, which is re-derived here using an explicit line model. The main difference is that no claim is made that the line definition here models the way water runs downhill, and that the surroundings of the line are modeled as well.

Figures 3.8(b) and (c) give an example of the results obtainable with the presented approach. Here, bright line points were extracted from the input image shown in Figure 3.8(a) with $\sigma = 1.1$. This image is part of an aerial image with a ground resolution of 2 m. The sub-pixel location $(p_x, p_y)$ of the line points and the direction $(n_x, n_y)$ perpendicular to the line are symbolized by vectors. The strength of the line, i.e., the absolute value of the second directional derivative along $(n_x, n_y)$ is symbolized by gray values. Line points with high saliency have dark gray values.

From Figure 3.8(b) it might appear that the proposed approach returns multiple responses to each line, if an 8-neighborhood were used. However, when the sub-pixel location of each line point, shown in Figure 3.8(c), is taken into account it can be seen that there is always a single response to a given line since all line point locations line up perfectly. Therefore, linking will be considerably easier than in approaches that yield multiple responses, e.g., [25, 83, 188], and no thinning is needed.



(a) Input image (b) Line points and their response (c) Line points and their direction

Figure 3.8: Bright line points detected in an aerial image (a) of ground resolution 2 m. In (b) the line points and directions of (c) are superimposed onto the magnitude of the response.

### 3.1.6 Linking Line Points into Lines

After individual line pixels have been extracted, they need to be linked into lines. It is necessary to do this at this point because the later stages of determining the line width and removing the bias will require a data structure that uses the notion of a left and right side of an entire line. Therefore, the normals to the line have to be oriented in the same manner as the line is traversed. As is evident from Figure 3.8, the procedure so far cannot do this since line points are regarded in isolation, and thus preference between two valid directions $n(t)$ is not made.

In order to facilitate later mid-level vision processes, e.g., perceptual grouping, the data structure that results from the linking process should contain explicit information about the lines as well as the junctions between them. This data structure should be topologically sound in the sense that junctions are represented by points and not by extended areas as in [25] or [187]. Furthermore, since the presented approach yields only single responses to each line point, no thinning operation needs to be performed prior to linking. This assures that the maximum information about the line points will be present in the data structure.

Since there is no suitable criterion to classify the line points into junctions and normal line points in advance without having to resort to extended junction areas another approach has been adopted. From the algorithm in Section 3.1.5 the following data are obtained for each pixel: the orientation of the line $(n_x, n_y) = (\cos\alpha, \sin\alpha)$, a measure of strength of the line (the second directional derivative in the direction of $\alpha$), and the sub-pixel location of the line $(p_x, p_y)$.

Starting from the pixel with maximum second derivative, lines will be constructed by adding the appropriate neighbor to the current line. Because the maximum point typically does not lie at the endpoints of the line, this is done for both directions $n^\perp$ and $-n^\perp$. Since it can be assumed that the line point detection algorithm will yield a fairly accurate estimate for the local direction of the line, only three neighboring pixels that are compatible with this direction are examined. For example, if the current pixel is $(c_x, c_y)$ and the current orientation of the line is in the interval $[-22.5°, 22.5°]$, only the points $(c_x + 1, c_y - 1)$, $(c_x + 1, c_y)$, and $(c_x + 1, c_y + 1)$ are examined. Two of the eight possible cases are given in Figure 3.9, where pixels checked for possible line

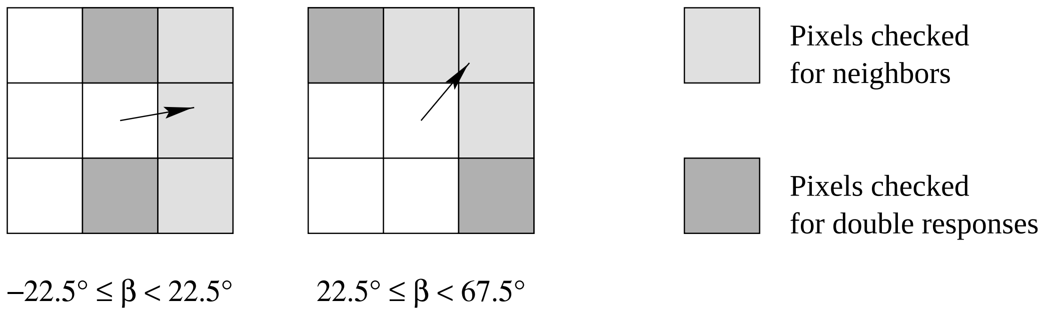

Pixels checked for neighbors
Pixels checked for double responses
−22.5° ≤ β < 22.5°
22.5° ≤ β < 67.5°

Figure 3.9: Two of the eight possible neighborhoods examined during the linking process.

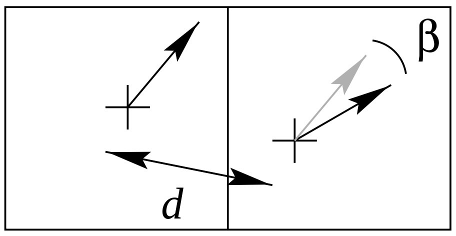
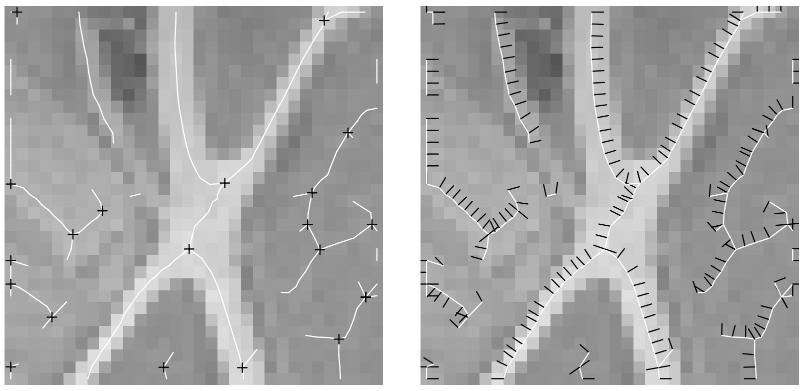


Figure 3.10: The criteria used to determine the best neighbor to add to the current line: the distance $d$ between the two line points and the angle difference $\beta$ of their normals.

continuation are marked as light gray squares. This figure also shows pixels which are checked for double responses. The reason for this will be explained below. The choice regarding the appropriate neighbor to add to the line is based on the distance between the respective sub-pixel line locations and the angle difference of the two points. Let $d = \|p_2 - p_1\|_2$ be the distance between the two points and $\beta = |\alpha_2 - \alpha_1|$, such that $\beta \in [0, \pi/2]$, be the angle difference between those points. The neighbor that is added to the line is the one that minimizes $d + c\beta$. In the current implementation, $c = 1$ is used. Figure 3.10 displays these criteria graphically. This algorithm will select each line point in the correct order. At junction points, it will select one branch to follow without detecting the junction, which will be detected later on. The algorithm of adding line points is continued until no more line points are found in the current neighborhood or until the best matching candidate is a point that has already been added to another line. If this happens, the point is marked as a junction, and the line that contains the point is split into two lines at the junction point. If the best matching candidate is the starting point of the current line, a closed curve has been found, and no junction is generated.

New lines will be created as long as the starting point has a second directional derivative that lies above a certain, user-selectable upper threshold. Points are added to the current line as long as their second directional derivative is greater than another user-selectable lower threshold. This is similar to a hysteresis threshold operation [29].

The problem of orienting the normals $n(t)$ of the line is solved by the following procedure. First, at the starting point of the line the normal is oriented such that it is turned $-90°$ to the direction the line is traversed, i.e., it will point to the right of the starting point. Then at each line point there are two possible normals whose angles differ by $180°$. The angle that minimizes the



(a) Linked lines and junctions (b) Lines and oriented normals

Figure 3.11: Linked lines detected using the new approach (a) and oriented normals (b). Lines are drawn in white while junctions are displayed as black crosses and normals as black lines.

difference between the angle of the normal of the previous point and the current point is chosen as the correct orientation. This procedure ensures that the normal always points to the right of the line as it is traversed from start to end.

With a slight modification the algorithm is able to deal with multiple responses if it is assumed that not more than three parallel responses are generated. For the facet model, for example, no such case has been encountered for mask sizes of up to $13 \times 13$. For Gaussian masks double responses may occur if $\sigma$ is chosen smaller than the restriction given by (3.15). Under this assumption, the algorithm can proceed as above. Additionally, if there are multiple responses to the line in the direction perpendicular to the line, e.g., the pixels $(c_x, c_y - 1)$ and $(c_x, c_y + 1)$ in the example above, they are marked as processed if they have roughly the same orientation as $(c_x, c_y)$. These points are marked as dark gray squares in Figure 3.9. The termination criterion for lines has to be modified to stop at processed line points instead of line points that are contained in another line.

Figure 3.11(a) shows the result of linking the line points in Figure 3.8 into lines. The results are overlaid onto the original image. In this case, the upper threshold was set to zero, i.e., all lines, no matter how faint, were selected. The lines obtained with the proposed approach are very smooth and the sub-pixel location of the line is quite precise. Figure 3.11(b) displays the way the normals to the line were oriented for this example. As can be seen, at junction points the normals of different lines of the junction are not oriented with respect to one another. This explains the fact that sometimes at junction points two normal vectors are drawn.

### 3.1.7 Parameter Selection

The selection of thresholds is very important to make an operator generally usable. Ideally, semantically meaningful parameters should be used to select salient objects. For the proposed line detector, these are the line width $w$ and its contrast $h$. However, as was described above, salient lines are defined by their second directional derivative along $n(t)$. To convert thresholds on $w$ and $h$ into thresholds the operator can use, first a $\sigma$ should be chosen according to (3.15).

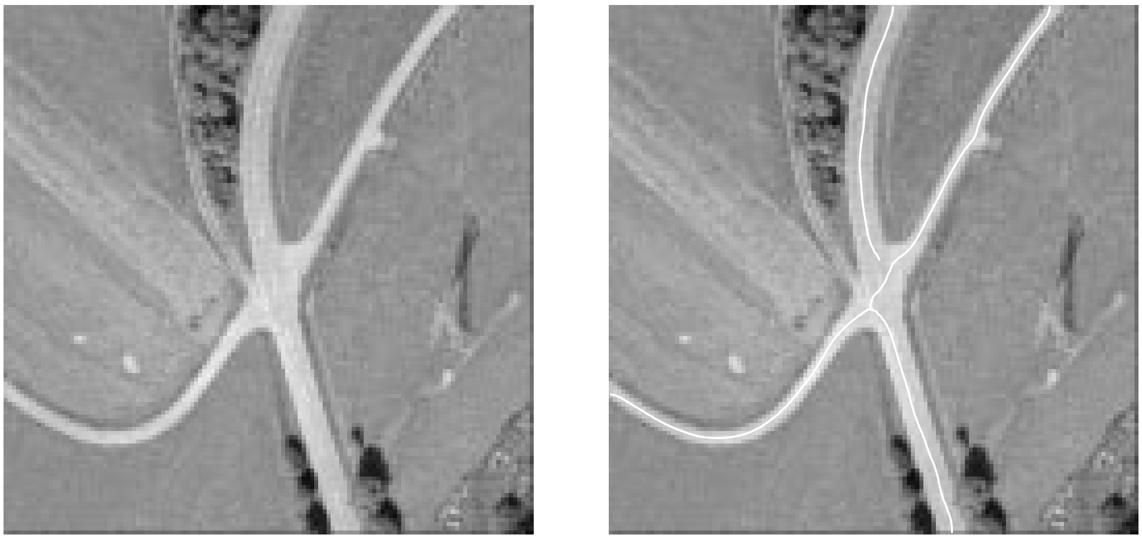

(a) Aerial image (b) Detected lines

Figure 3.12: Lines detected (b) in an aerial image (a) of ground resolution 1 m.

Then, $\sigma$, $w$, and $h$ can be plugged into (3.13) to yield an upper threshold for the operator.

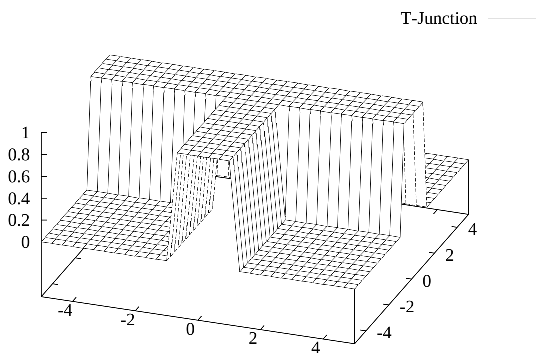
Figure 3.12 exemplifies this procedure and shows that the presented line detector can be scaled arbitrarily. In Figure 3.12(a) a larger part of the aerial image in Figure 3.11 is displayed, but this time with a ground resolution of 1 m, i.e., twice the resolution. If $7$ pixel wide lines are to be detected, i.e., if $w = 3.5$, according to (3.15), a $\sigma \geq 2.0207$ should be selected. In fact, $\sigma = 2.2$ was used for this image. If lines with a contrast of $h \geq 70$ are to be selected, from (3.13) it follows that these lines will have a second derivative of $\approx -5.17893$. Therefore, the upper threshold for the absolute value of the second derivative was set to $5$, while the lower threshold was $0.8$ to follow the roads as far as possible into the road intersection. Figure 3.12(b) displays the lines that were detected with these parameters. As can be seen, all of the roads were detected.

## 3.1.8 Extraction of Missed Junctions

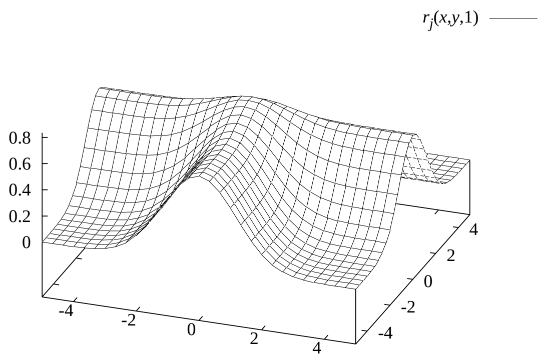
The example in Figure 3.12(b) shows that the proposed algorithm sometimes fails to extract junction points. In this section, the reasons for these failures will be explored by modeling junctions and examining their scale-space behavior. Furthermore, a solution to this problem will be proposed.

Consider a T-junction where lines of different width and different contrast meet. A model for this type of junction is

$$f_j(x,y) = \begin{cases} h_1, & x \geq 0 \wedge |y| \leq w_1 \\ h_2, & x < 0 \wedge |y| \leq w_2 \\ h_3, & |x| \leq w_3 \wedge y < -w_1 \\ 0, & \text{otherwise} \enspace . \end{cases} \tag{3.37}$$



(a) T-junction



(b) Response for $\sigma = 1$

Figure 3.13: A T-junction with lines of equal width and contrast (a) and the response $r_j(x, y, 1)$ (b).

The scale-space description of this junction is given by

$$\begin{aligned} r_j(x,y,\sigma) &= h_1(\phi_\sigma(y+w_1) - \phi_\sigma(y-w_1))\phi_\sigma(x) + \\ &\quad h_2(\phi_\sigma(y+w_2) - \phi_\sigma(y-w_2))(1-\phi_\sigma(x)) + \\ &\quad h_3(\phi_\sigma(x+w_3) - \phi_\sigma(x-w_3))(1-\phi_\sigma(y+w_1)) \end{aligned} \tag{3.38}$$

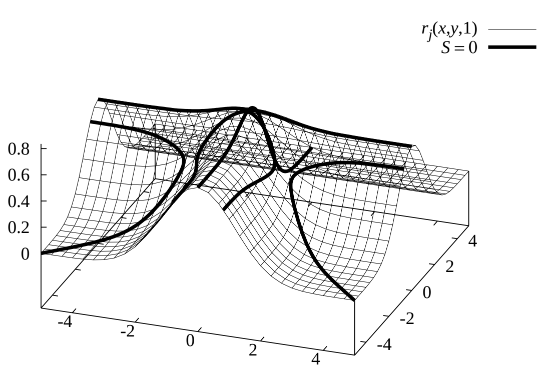
and its partial derivatives of first and second order. Figure 3.13(a) shows a T-junction with lines of equal widths $w_1 = w_2 = w_3 = 1$ and equal contrasts $h_1 = h_2 = h_3 = 1$, and the corresponding response $r_j(x, y, 1)$ for $(x, y) \in [-5, 5] \times [-5, 5]$.

In order to analyze the scale-space behavior of the line positions the zero crossings of the first directional derivative in the direction of the maximum second directional derivative must be obtained. Since the algorithm uses the restricted extremum definition for ridges, De Saint-Venants condition $S = 0$ can be used to do this [93]. More specifically, this condition is given by

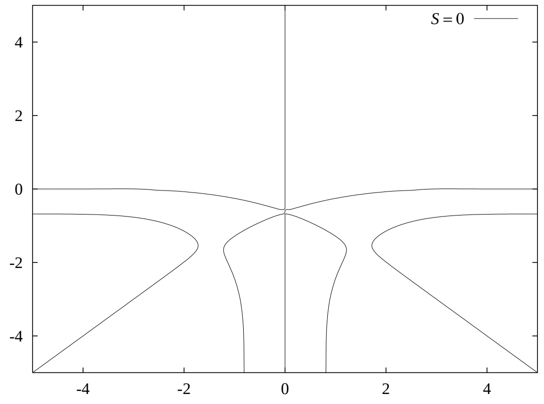
$$S = \frac{\nabla r_j^T H \nabla r_j^\perp}{\nabla r_j^T \nabla r_j} = \frac{r_x r_y (r_{xx} - r_{yy}) - (r_x^2 - r_y^2) r_{xy}}{r_x^2 + r_y^2} = 0 \ . \tag{3.39}$$

This condition will extract a superset of the lines the proposed algorithm extracts because it will also extract the points where there is a restricted extremum in the direction of minimum second directional derivative, and because it does not discern between restricted maxima and minima. Nevertheless, because the extracted lines form a superset of the lines returned by the proposed algorithm, the reasons for missed junctions can be studied.

Figure 3.14 shows the lines extracted by the condition $S = 0$ for the T-junction of Figure 3.13. One thing immediately strikes the eye: the “hump” in the middle of the Gaussian smoothed junction. As can be seen from Figure 3.14, the three extracted lines seem to intersect in the middle of the junction, at the maximum of the hump. However, closer inspection reveals that the line of the $y$-axis actually is a restricted maximum in the direction of the *minimum* second directional derivative in the vicinity of the junction. Therefore, failures to extract junctions, even for lines with identical contrast, e.g., as in Figure 3.12 can be explained. Fortunately, for lines of equal contrast the area where the lines entering the junction are restricted extrema in the direction of the minimum second directional derivative is usually rather small, typically smaller

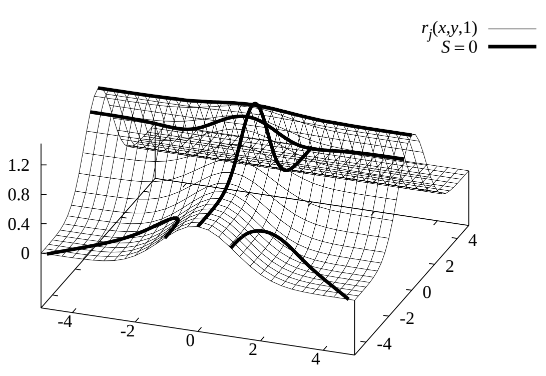


(a) Extracted lines

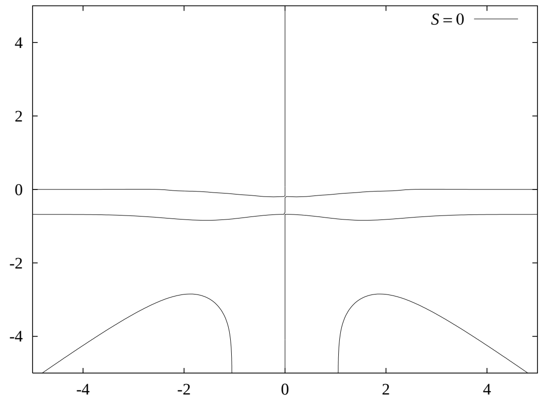


(b) Lines in the $(x, y)$-plane

Figure 3.14: Extracted lines by the De Saint-Venant condition for a T-junction with lines of equal width and contrast: (a) mapped onto the response $r_j(x, y, 1)$, and (b) in the $(x, y)$-plane.



(a) Extracted lines



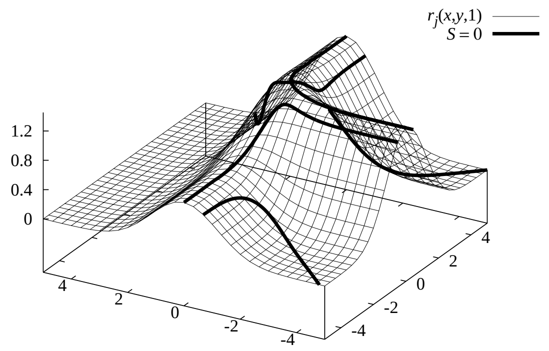
(b) Lines in the $(x, y)$-plane

Figure 3.15: Extracted lines by the De Saint-Venant condition for a T-junction with lines of equal width and different contrasts of $h_1 = h_2 = 2$ and $h_3 = 1$: (a) mapped onto the response $r_j(x, y, 1)$, and (b) in the $(x, y)$-plane.

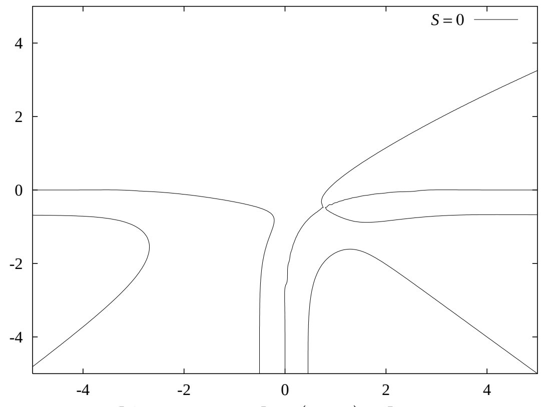
than one pixel for narrow lines. Consequently, these junctions will be extracted by the linking algorithm because the lines will meet in 8-neighborhood.

A more difficult case is given, when the line on the $y$-axis has a different contrast than the lines on th $x$-axis, e.g., $h_1 = h_2 = 2$ and $h_3 = 1$. This configuration is depicted in Figure 3.15. As can be seen from this figure, the point where the extracted line changes from being a restricted maximum in the direction of the maximum second directional derivative to being a restricted maximum in the direction of the minimum second directional derivative now lies outside of the lines on the $x$-axis. However, the three lines still intersect in the expected junction point. From this it might seem that one should allow lines to be restricted extrema in the direction of the minimum second directional derivative as well, in order to extract all junctions. The next example shows that this is not the case.

Figure 3.16 shows a junction with $h_1 = h_3 = 2$ and $h_2 = 1$, i.e., the left line on the $y$-axis is

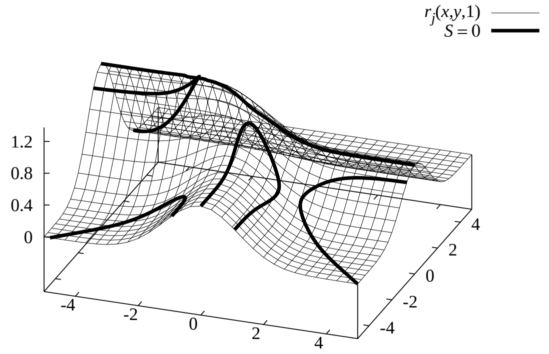


(a) Extracted lines

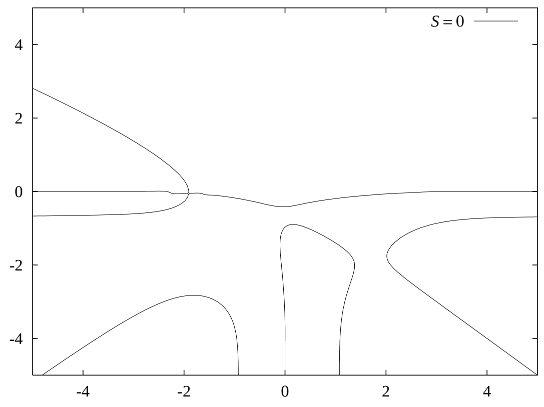


(b) Lines in the $(x, y)$-plane

Figure 3.16: Extracted lines by the De Saint-Venant condition for a T-junction with lines of equal width and different contrasts of $h_1 = h_3 = 2$ and $h_2 = 1$: (a) mapped onto the response $r_j(x, y, 1)$, and (b) in the $(x, y)$-plane.



(a) Extracted lines



(b) Lines in the $(x, y)$-plane

Figure 3.17: Extracted lines by the De Saint-Venant condition for a T-junction with lines of equal width and different contrasts of $h_1 = h_3 = 1$ and $h_2 = 2$: (a) mapped onto the response $r_j(x, y, 1)$, and (b) in the $(x, y)$-plane.

now darker than the other two lines. As can be seen, the line on the left part of the $y$-axis again changes to a restricted maximum in the direction of the minimum second directional derivative. However, as this line approaches the other two lines it rapidly turns away from the junction, and ends up running parallel to the line on the $y$-axis. Therefore, allowing lines to be restricted extrema in the direction of the minimum second directional derivative is definitely a wrong approach. What is striking, though, is that the direction of the line on the $y$-axis at the point where it changes its type points exactly at the junction. This is also true for the configuration of $h_1 = h_3 = 1$ and $h_2 = 2$ displayed in Figure 3.17, and for numerous other configurations not shown here.

Therefore, a good algorithm to extract the missed junctions is to search for other lines along the last extracted line direction. As can be seen from the examples, the lines not continuing to the junction will in general be the lines with the smaller contrast. Hence, while following the line, the gray value in the image should increase monotonically for bright lines and decrease monotonically for dark lines, respectively. Since it is desirable to avoid using the image intensity directly, this condition can be reformulated to the condition that the dot product of the gradient of the image and the line direction must always be positive for bright lines and always negative for dark lines. Of course, the length of the line along which the search is performed should not be arbitrarily long. A good restriction is to use the same length $2.5\sigma$ as for the determination of the line width, discussed in Section 3.1.9. In order to perform the search the algorithm needs to determine all pixels intersected by the line starting at the sub-pixel position of the last line point. A slight modification of the Bresenham line drawing algorithm [155] suffices to return these pixels.

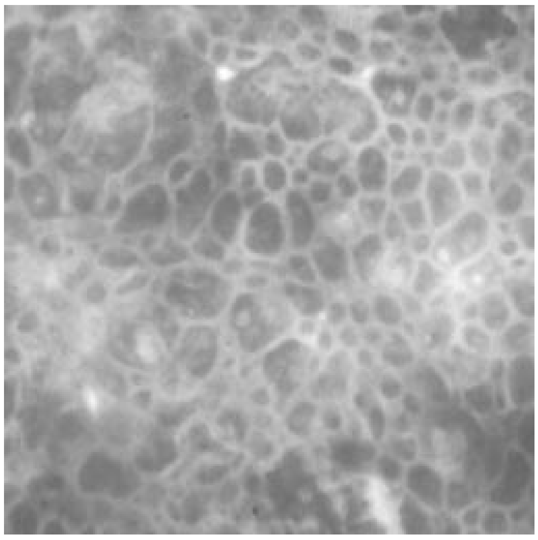
The resulting junction completion algorithm can be summarized as follows: First, an array containing all pixels of the search line is constructed. Then, these pixels are checked in order, whether the dot product of the gradient and the line direction has the correct sign. If not, or if all pixels have been checked without finding another line the algorithm terminates without adding a junction. On the other hand, as soon as a pixel containing a point on another line is found, a junction is added.

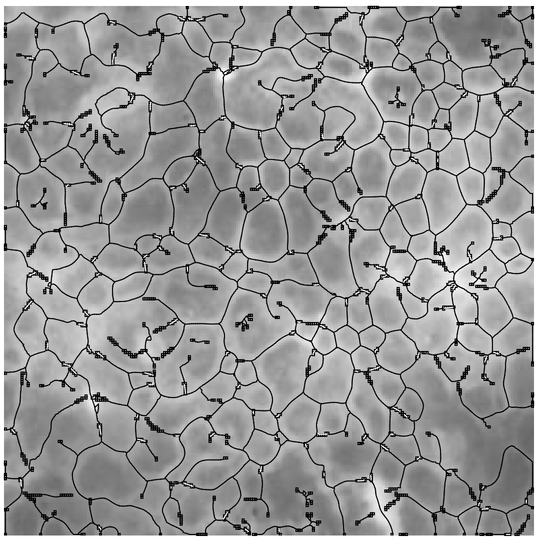
Figure 3.18 shows the extracted lines in a dermatoscopic image, which contains numerous junctions of lines with different intensities. Here, it is especially important to extract all junctions in the pigment network. The extracted lines are shown in black, while the pixels examined during the check for junctions are shown as black squares. The line segments leading to added junctions are shown in white. As can be seen, the algorithm does a very good job of adding the correct junctions. Almost all of the cells enclosed by the pigment network are completely surrounded by lines.

### 3.1.9 Determination of Line Width

The width of a line is an important feature in its own right. As discussed in Chapter 1, many applications are interested in obtaining the width of an object as precisely as possible. Furthermore, the width can, for instance, be used in perceptual grouping processes to avoid the grouping of lines that have incompatible widths. However, the main reason that width is important in the proposed approach is that it is needed to obtain an estimate of the true line width such that the bias which is introduced by asymmetrical lines can be removed.

From the discussion in Section 3.1.2 it follows that a line is bounded by an edge on each side. Hence, to detect the width of the line, for each line point the closest points in the image, to the left and to the right of the line point, where the absolute value of the gradient takes on its maximum value need to be determined. Of course, these points should be searched for exclusively along a line in the direction $n(t)$ of the current line point. Again, the modified Bresenham algorithm is used to determine all pixels that the search line intersects. The analysis in Section 3.1.2 shows that it is only reasonable to search for edges in a restricted neighborhood of the line. For ideal symmetrical lines, the line to search would have a length of $\sqrt{3}\sigma$. In order to ensure that almost all of the edge points are detected, the implementation uses a slightly larger search line length of $2.5\sigma$. This covers most of the asymmetrical lines as well, for which the width can grow beyond $\sqrt{3}\sigma$. As will be shown in Section 3.1.10, only very asymmetrical, narrow lines are not covered.


(a) Dermatoscopic image


(b) Dermatoscopic image

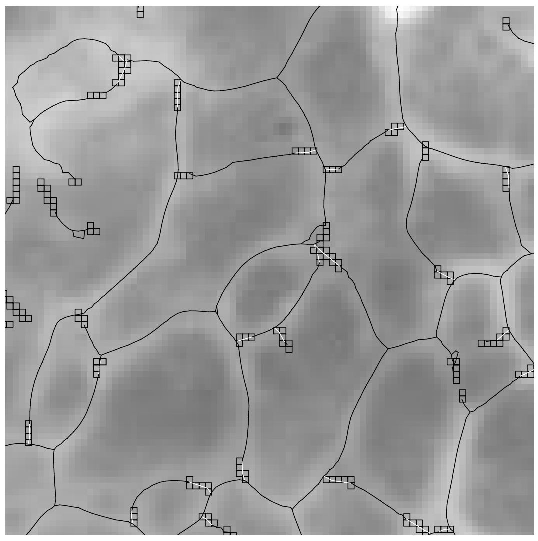

(c) Detail of (b)

Figure 3.18: Pixels examined during the process of adding junctions for the dermatoscopic image (a). The extracted lines are shown as black lines, while the pixels examined are shown as black squares. Line segments accepted as leading to junctions are shown as white lines.

In an image of the absolute value of the gradient of the image, the desired edges appear as bright lines. Figure 3.19 exemplifies this for the aerial image of Figure 3.12(a). In order to extract the lines from the gradient image

$$e(x,y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2} = \sqrt{f_x^2 + f_y^2} \tag{3.40}$$

where

$$f(x,y) = g_\sigma(x,y) * z(x,y) \tag{3.41}$$

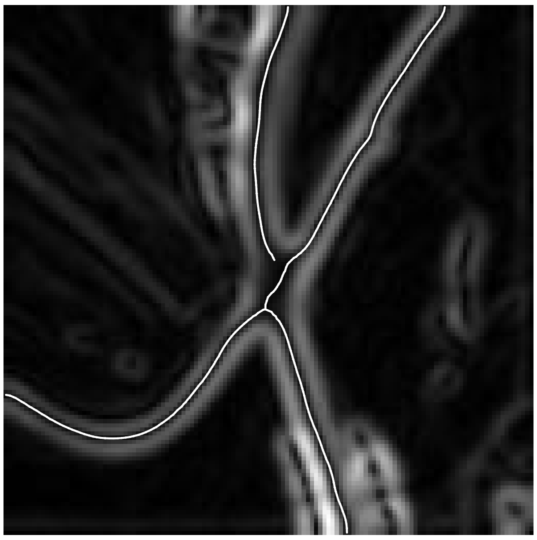


Figure 3.19: Lines and their corresponding edges in an image of the absolute value of the gradient.

the following coefficients of a local Taylor polynomial need to be computed:

$$
\begin{aligned}
e_x &= \frac{f_x f_{xx} + f_y f_{xy}}{e} && (3.42)\\
e_y &= \frac{f_x f_{xy} + f_y f_{yy}}{e} && (3.43)\\
e_{xx} &= \frac{f_x f_{xxx} + f_y f_{xxy} + f_{xx}^2 + f_{xy}^2 - e_x^2}{e} && (3.44)\\
e_{xy} &= \frac{f_x f_{xxy} + f_y f_{xyy} + f_{xx} f_{xy} + f_{xy} f_{yy} - e_x e_y}{e} && (3.45)\\
e_{yy} &= \frac{f_x f_{xyy} + f_y f_{yyy} + f_{xy}^2 + f_{yy}^2 - e_y^2}{e} \;. && (3.46)
\end{aligned}
$$

This has three main disadvantages. First of all, the computational load increases by almost a factor of two since four additional partial derivatives with slightly larger mask sizes have to be computed. Furthermore, the third partial derivatives of the image would need to be used. This is clearly undesirable since they are very susceptible to noise. Finally, the expressions above are undefined whenever $e(x, y) = 0$. However, since the only interesting characteristic of the Taylor polynomial is the zero crossing of its first derivative in one of the principal directions, the coefficients can be multiplied by $e(x, y)$ to avoid this problem.

It might appear that an approach to solve these problems would be to use the algorithm to detect line points described in Section 3.1.5 on the gradient image in order to detect the edges of the line with sub-pixel accuracy. However, this would mean that some additional smoothing would be applied to the gradient image because the image would be again convolved with Gaussian derivative masks. This is undesirable since it would destroy the correlation between the location of the line points and the location of the corresponding edge points. Therefore, the edge points in the gradient image are extracted with a facet model line detector which uses the

$$k_1 = \frac{1}{6}\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \qquad k_2 = \frac{1}{6}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

$$k_3 = \frac{1}{6}\begin{pmatrix} 1 & 1 & 1 \\ -2 & -2 & -2 \\ 1 & 1 & 1 \end{pmatrix} \qquad k_4 = \frac{1}{4}\begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} \qquad k_5 = \frac{1}{6}\begin{pmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \end{pmatrix}$$

Figure 3.20: Facet model convolution masks for the polynomial $k_1r + k_2c + k_3r^2 + k_4rc + k_5c^2$.

same principles as described in Section 3.1.5, but uses different convolution masks to determine the partial derivatives of the image [25, 65]. The facet model kernels compute the coefficients $k_1, \ldots, k_5$ of a local second order polynomial $k_1r+k_2c+k_3r^2+k_4rc+k_5c^2$ for the current pixel, and are given in Figure 3.20. Note that this definition is slightly different from the local Taylor polynomial used in (3.32) and (3.33). Here, the Hessian matrix from which the line direction is computed is given by

$$H(x,y) = \begin{pmatrix} 2k_3 & k_4 \\ k_4 & 2k_5 \end{pmatrix} . \tag{3.47}$$

Therefore, the value of $t$ used in (3.35) to detect the line position in the direction of the largest eigenvector $n$ is given by

$$t = -\frac{n_xr_x + n_yr_y}{2(n_x^2r_{xx} + n_xn_yr_{xy} + n_y^2r_{yy})} . \tag{3.48}$$

The smallest possible mask size ($3 \times 3$) is used since this will result in the most accurate localization of the edge points while yielding as little of the problems mentioned in Section 2.3.3 as possible. It has the additional benefit that the computational costs are very low. Experiments on a large number of images have shown that if the coefficients of the Taylor polynomial are computed in this manner, they can, in some cases, be significantly different than the correct values. However, the positions of the edge points, especially those of the edges corresponding to salient lines, will only be affected very slightly. Figure 3.21 illustrates this on the image of Figure 3.8(a). Edge points extracted with the correct formulas are displayed as black crosses, while those extracted with the $3 \times 3$ facet model are displayed as white crosses. It is apparent that because third derivatives are used in the correct formulas there are many more spurious responses. Furthermore, five edge points along the salient line in the upper middle part of the image are missed because of this. Finally, it can be seen that the edge positions corresponding to salient lines differ only minimally, and therefore the approach presented here seems to be justified.

The edge points found by this procedure will usually not lie on the search line itself, but some distance away on a line from the pixel's center in the direction perpendicular to the gradient. Figure 3.22 shows a typical configuration. Therefore, a procedure is needed to get a meaningful edge position on the search line. The most accurate results will usually be given if the edge point is projected onto the search line along the edge direction, i.e., perpendicular to

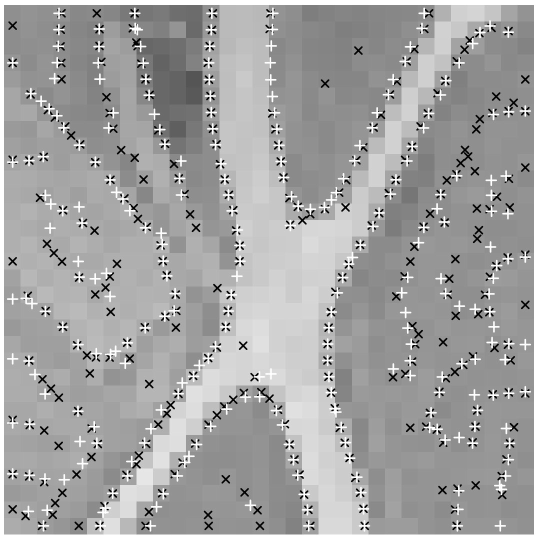


Figure 3.21: Comparison between the locations of edge points extracted using the correct formula (black crosses) and the $3 \times 3$ facet model (white crosses).

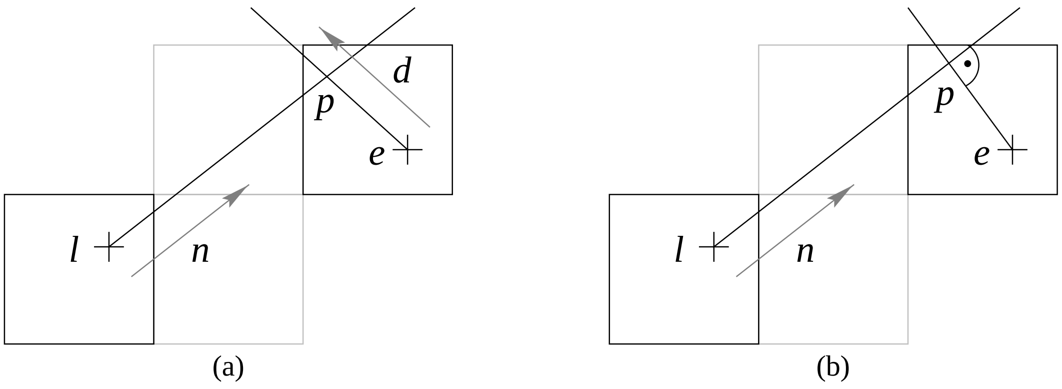


Figure 3.22: Two methods of projecting the extracted edge position onto the search line. (a) Along the edge direction. (b) Perpendicular to the search line.

the gradient. This is depicted in Figure 3.22(a). Here, the line point is given by $l$, the direction perpendicular to the line, i.e., the search direction, by $n$, the edge point by $e$, and the edge direction by $d$. Thus, one has to solve the linear equation system

$$l + \lambda n = e + \mu d \ , \tag{3.49}$$

and the edge point $p$ on the search line will be given by $p = l + \lambda n$. Unfortunately, sometimes the edge direction forms a very acute angle with the search direction, and therefore the point $p$ may lie rather far away from the edge point on the search line. Therefore, if the intersection is further than two pixel units away from the edge point, the algorithm projects the edge point onto the search line perpendicularly, as depicted in Figure 3.22(b). Here, the condition for the point $p$ is that $l + \lambda n - e \perp n$, i.e.,

$$(l + \lambda n - e) \cdot n = 0 \ . \tag{3.50}$$

Again, $p$ will then be given by $p = l + \lambda n$. With this approach, very precise edge locations can be obtained in all cases.

One final important issue is what the algorithm should do when it is unable to locate an edge point for a given line point. This might happen, for example, if there is a very weak and wide gradient next to the line, which does not exhibit a well defined maximum. Another case where this typically happens are the junction areas of lines, where the line width usually grows beyond the range of $2.5\sigma$. Since the algorithm does not have any other means of locating the edge points, the only viable solution to this problem is to interpolate or extrapolate the line width from neighboring line points. It is at this point that the notion of a right and a left side of the line, i.e., the orientation of the normals of the line, becomes crucial since it is best to interpolate missing widths from widths on the same side of the line.

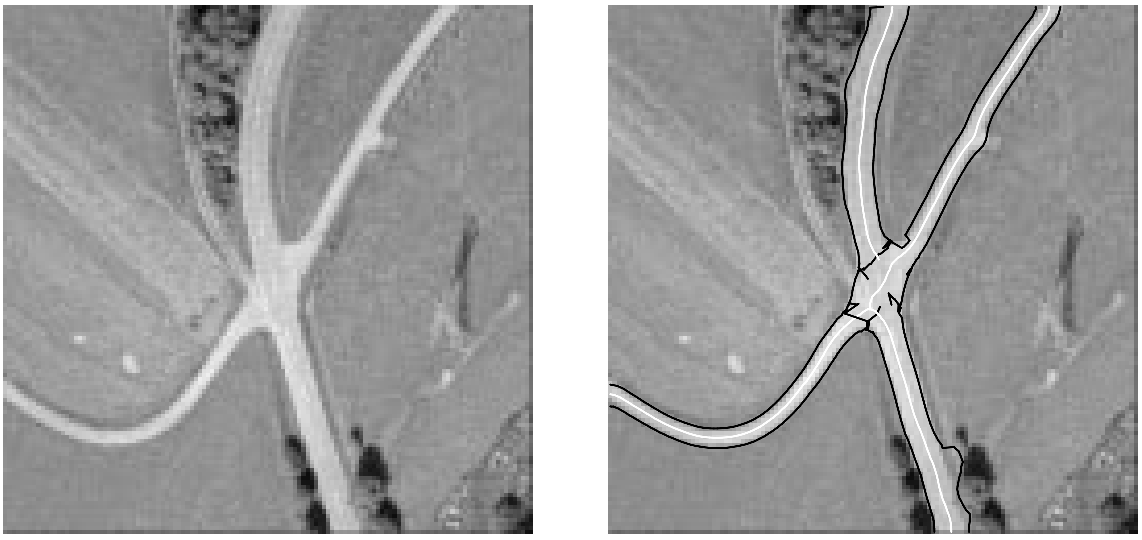
The algorithm is as follows: First of all, the width of the line is extracted for each line point, if possible. After this, if there is a gap in the extracted widths on one side of the line, i.e., if the width of the line is undefined at some line point, but there are some points before and after the current line point that have a defined width, the width for the current line point is obtained by linear interpolation. This can be formalized as follows. Let $i$ be the index of the last point and $j$ be the index of the next point with a defined line width, respectively. Let $a$ be the length of the line from $i$ to the current point $k$ and $b$ be the total line length from $i$ to $j$. Then the width of the current point $k$ is given by

$$w_k = \frac{b-a}{b} w_i + \frac{a}{b} w_j \quad . \tag{3.51}$$

This scheme can easily be extended to the case where either $i$ or $j$ are undefined, i.e., the line width is undefined at either end of the line. The algorithm sets $w_i = w_j$ in this case, which means that if the line width is undefined at the end of a line, it will be extrapolated to the last defined line width.

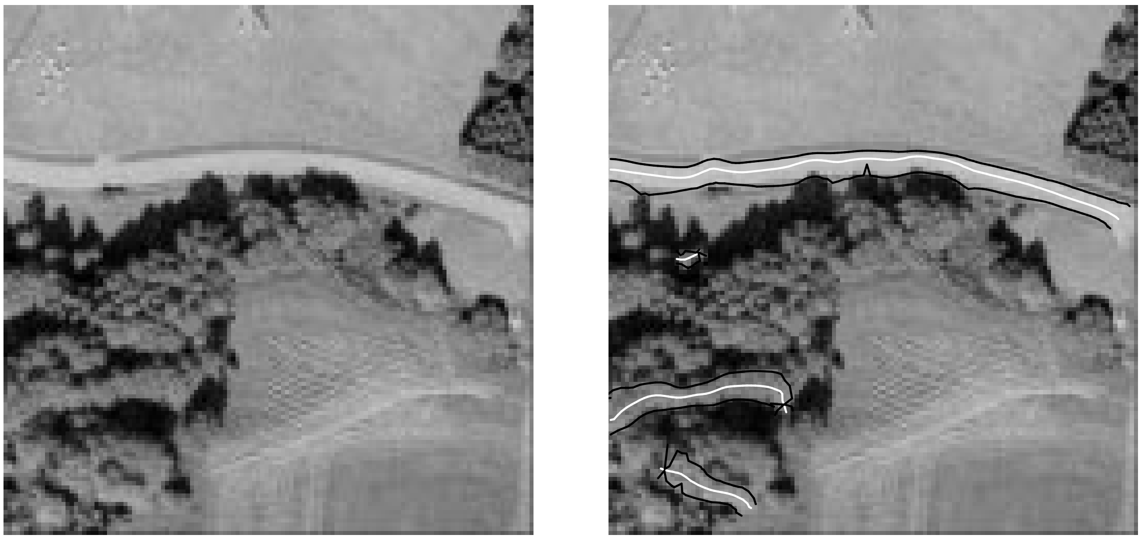
Figure 3.23(b) displays the results of the line width extraction algorithm for the example image of Figure 3.12. This image is fairly good-natured in the sense that the lines it contains are rather symmetrical. From Figure 3.23(a) it can be seen that the algorithm is able to locate the edges of the wider line with very high precision. The only place where the edges do not correspond to the semantic edges of the road object are in the bottom part of the image, where nearby vegetation causes a strong gradient and causes the algorithm to estimate the line width too large. Please note that the width of the narrower line is extracted slightly too large, which is not surprising taking the discussion in Section 3.1.3 into account. Revisiting Figure 3.5 again, it is clear that an effect like this is to be expected. How to remove this effect is the topic of Section 3.1.10. A final thing to note is that the algorithm extrapolates the line width in the junction area in the middle of the image, as discussed above. This explains the seemingly unjustified edge points in this area.

Figure 3.24(b) exhibits the results of the proposed approach on another aerial image of the same ground resolution, given in Figure 3.24(a). Please note that the line in the upper part of the image contains a very asymmetrical part in the center part of the line due to shadows of nearby objects. Therefore, as is predictable from the discussion in Section 3.1.3, especially from Figure 3.7, the line position is shifted towards the edge of the line that possesses the weaker gradient, i.e., the upper edge in this case. Please note also that the line and edge positions are very accurate in the rest of the image.



(a) Aerial image (b) Detected lines and their width

Figure 3.23: Lines and their width detected (b) in an aerial image (a). Lines are displayed in white while the corresponding edges are displayed in black.



(a) Aerial image (b) Detected lines and their width

Figure 3.24: Lines and their width detected (b) in an aerial image (a).

### 3.1.10 Removing the Bias from Asymmetrical Lines

Recall from the discussion at the end of Section 3.1.3 that if the algorithm knew the true values of $w$ and $a$ it could remove the bias in the estimation of the line position and width. Equations (3.16)–(3.18) give an explicit scale-space description of the asymmetrical line profile $f_a$.

The position $l$ of the line can be determined analytically by the zero-crossings of $r'_a(x,\sigma,w,a)$ and is given in (3.19). The total width of the line, as measured from the left to right edge, is given by the zero-crossings of $r''_a(x,\sigma,w,a)$. Unfortunately, these positions can only be computed by a root finding algorithm since the equations cannot be solved analytically. Call these positions $e_l$ and $e_r$. Then the width to the left and right of the line is given by $v_l = l - e_l$ and $v_r = e_r - l$. The total width of the line is $v = v_l + v_r$. The quantities $l$, $e_l$, and $e_r$ have the following useful property:

**Proposition 1** *The values of $l$, $e_l$, and $e_r$ form a scale-invariant system. This means that if both $\sigma$ and $w$ are scaled by the same constant factor $c$ the line and edge locations will be given by $cl$, $ce_l$, and $ce_r$.*

*Proof:* Let $l_1$ be the line location for $\sigma_1$ and $w_1$ for an arbitrary, but fixed $a$. Let $\sigma_2 = c\sigma_1$ and $w_2 = cw_1$. Then $l_{1,2} = -(\sigma_{1,2}^2/2w_{1,2})\ln(1-a)$. Hence, $l_2 = -(\sigma_2^2/2w_2)\ln(1-a) = -(c^2\sigma_1^2/(2cw_1))\ln(1-a) = cl_1$.

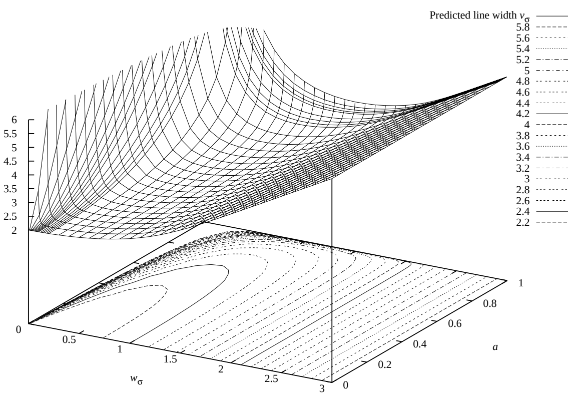
Now let $e_1$ be one of the two solution of $r''_a(x,\sigma,w_1,a) = 0$, either $e_l$ or $e_r$, and likewise for $e_2$, with $\sigma_{1,2}$ and $w_{1,2}$ as above. The expression $r''_a(x,\sigma,w_1,a) = 0$ can be transformed to $(a-1)(e_2-w_2)/(e_2+w_2) = \exp(-2e_2w_2/\sigma_2^2)$. If $\sigma_2 = c\sigma_1$ and $w_2 = cw_1$ is plugged in, it can be seen that this expression can only be fulfilled for $e_2 = ce_1$ since only then will the factors $c$ cancel everywhere. Of course, this property also holds for the derived quantities $v_l$, $v_r$, and $v$. $\Box$

The meaning of Proposition 1 is that $w$ and $\sigma$ are not independent of each other. In fact, we only need to consider all $w$ for one particular $\sigma$, e.g., $\sigma = 1$. Therefore, for the following analysis only values that are normalized with regard to the scale $\sigma$ need to be discussed, i.e., $w_\sigma = w/\sigma$, $v_\sigma = v/\sigma$, and so on. A useful consequence is that the behavior of $f_a$ can be analyzed for $\sigma = 1$. All other values can be obtained by a simple multiplication by the actual scale $\sigma$.

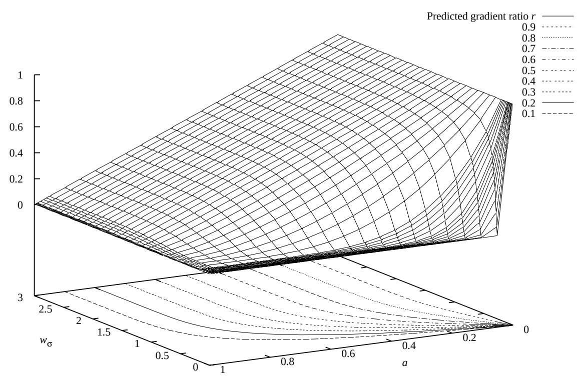
With all this being established, the predicted total line width $v_\sigma$ can be calculated for all $w_\sigma$ and $a \in [0,1]$. Figure 3.25 displays the predicted $v_\sigma$ for $w_\sigma \in [0,3]$. It be seen that $v_\sigma$ can grow without bounds for $w_\sigma \downarrow 0$ or $a \uparrow 1$. Furthermore, it can be proved that $v_\sigma \in [2,\infty]$. Therefore, in Figure 3.25 the contour lines for $v_\sigma \in [2,6]$ are also displayed.

Section 3.1.9 gave a procedure to extract the quantity $v_\sigma$ from the image. This is half of the information required to get to the true values of $w$ and $a$. However, an additional quantity is needed to estimate $a$. Since the true height $h$ of the line profile $hf_a$ is unknown this quantity needs to be independent of $h$. One such quantity is the ratio of the gradient magnitude at $e_r$ and $e_l$, i.e., the weak and strong side. This quantity is given by $r = |r'_a(e_r,\sigma,w,a)|/|r'_a(e_l,\sigma,w,a)|$. It is obvious that the influence of $h$ cancels out. Furthermore, it is easy to convince oneself that $r$ also remains constant under simultaneous scalings of $\sigma$ and $w$. The quantity $r$ has the advantage that it is easy to extract from the image. Figure 3.25 displays the predicted $r$ for $w_\sigma \in [0,3]$. It is obvious that $r \in [0,1]$. Therefore, the contour lines for $r$ in this range are displayed in Figure 3.25 as well. It can be seen that for large $w_\sigma$, $r$ is very close to $1-a$. For small $w_\sigma$ it drops to near-zero for all $a$.

The discussion above can be summarized as follows: The true values of $w_\sigma$ and $a$ are mapped to the quantities $v_\sigma$ and $r$, which are observable from the image. More formally, there is a function $f : (w_\sigma, a) \in [0,\infty] \times [0,1] \mapsto (v_\sigma, r) \in [2,\infty] \times [0,1]$. From the discussion in



(a) Predicted line width



(b) Predicted gradient ratio

Figure 3.25: Predicted behavior of the asymmetrical line $f_a$ for $w_\sigma \in [0, 3]$ and $a \in [0, 1]$. (a) Predicted line width $v_\sigma$. (b) Predicted gradient ratio $r$.

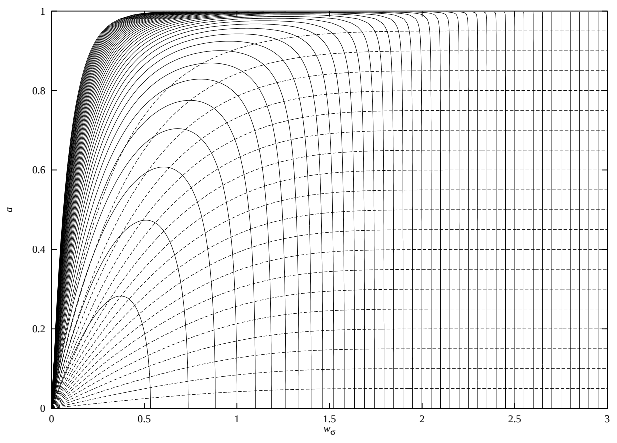
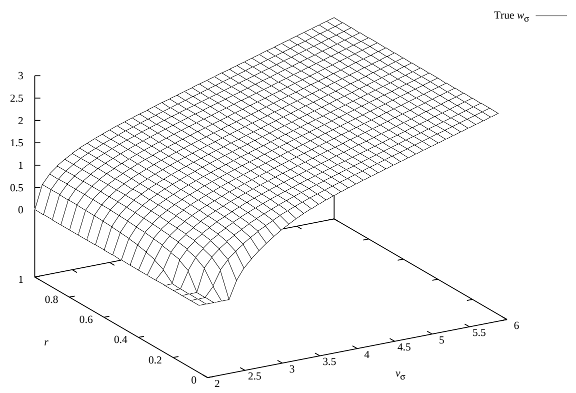


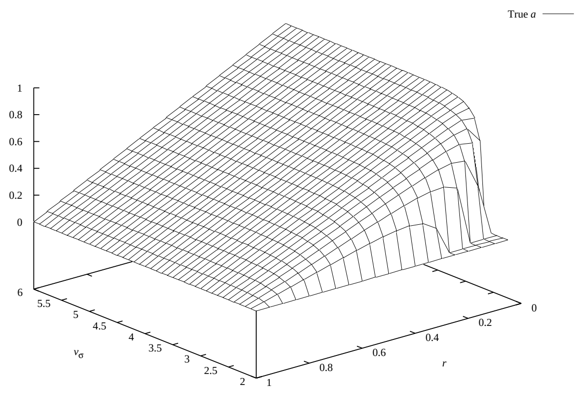
Figure 3.26: Contour lines of $v_\sigma \in [2, 6]$ (solid) and $r \in [0, 1]$ (dashed).

Section 3.1.9 it follows that it is only useful to consider $v_\sigma \in [0, 5]$. However, for very small $\sigma$ it is possible that an edge point will be found within a pixel in which the center of the pixel is less than $2.5\sigma$ from the line point, but the edge point is farther away than this. Therefore, $v_\sigma \in [0, 6]$ is a good restriction for $v_\sigma$. Since the algorithm needs to determine the true values $(w_\sigma, a)$ from the observed $(v_\sigma, r)$, the inverse $f^{-1}$ of the map $f$ has to be determined. Figure 3.26 illustrates that $f$ is invertible. It displays the contour lines of $v_\sigma \in [2, 6]$ and $r \in [0, 1]$. The contour lines of $v_\sigma$ are U-shaped with the tightest U corresponding to $v_\sigma = 2.1$. The contour line corresponding to $v_\sigma = 2$ is actually only the point $(0, 0)$. The contour lines for $r$ run across with the lowermost visible contour line corresponding to $r = 0.95$. The contour line for $r = 1$ lies completely on the $w_\sigma$-axis. It can be seen that, for any pair of contour lines from $v_\sigma$ and $r$, there will only be one intersection point. Hence, $f$ is invertible.

To calculate $f^{-1}$, a multi-dimensional root finding algorithm has to be used [152]. To obtain maximum precision for $w_\sigma$ and $a$, this root finding algorithm would have to be called at each line point. This is undesirable for two reasons. Firstly, it is a computationally expensive operation.



(a) True $w_\sigma$



(b) True $a$

Figure 3.27: True values of the line width $w_\sigma$ (a) and the asymmetry $a$ (b).

More importantly, however, due to the nature of the function $f$, very good starting values are required for the algorithm to converge, especially for small $v_\sigma$. Therefore, the inverse $f^{-1}$ is pre-computed for selected values of $v_\sigma$ and $r$ and the true values are obtained by interpolation. The step size of $v_\sigma$ was chosen as $0.1$, while $r$ was sampled at $0.05$ intervals. Hence, the intersection points of the contour lines in Figure 3.26 are the entries in the table of $f^{-1}$. Figure 3.27 shows the true values of $w_\sigma$ and $a$ for any given $v_\sigma$ and $r$. It can be seen that despite the fact that $f$ is very ill-behaved for small $w_\sigma$, $f^{-1}$ is quite well behaved. This behavior leads to the conclusion that bilinear interpolation can be used to obtain good values for $w_\sigma$ and $a$.

One final important detail is how the algorithm should handle line points where $v_\sigma < 2$, i.e., where $f^{-1}$ is undefined. This can happen, for example, because the facet model sometimes gives a multiple response for an edge point, or because there are two lines very close to each other. In this case the edge points cannot move as far outward as the model predicts. If this happens, the line point will have an undefined width. These cases can be handled by the interpolation procedure given in Section 3.1.9 that fills such gaps.

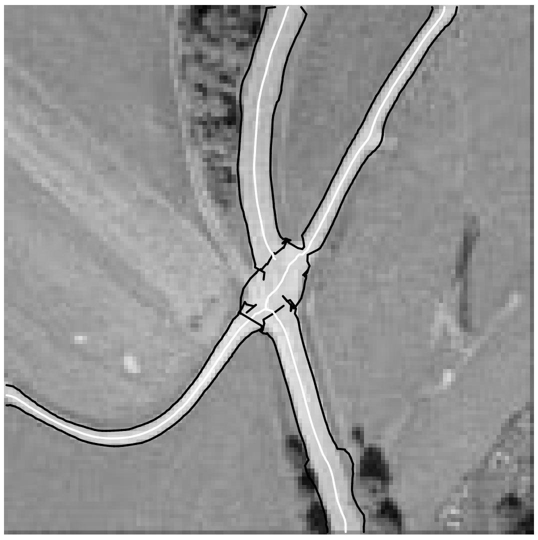
With all this information calculated it is now a simple matter to calculate the true contrast $h$ of the line. It is given by the ratio of predicted response $r''_a$ according to (3.18) and the observed response $r''$ in the image:

$$h = \frac{r''}{r''_a(l, \sigma, w, a)} \quad , \qquad (3.52)$$

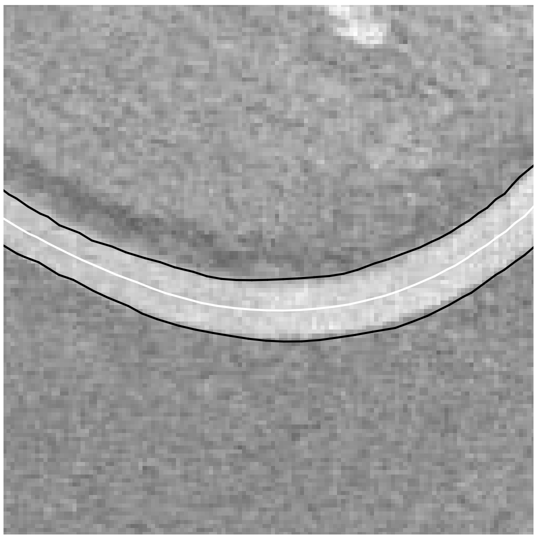
where $l$ is the line position according to (3.19). In order to achieve maximum accuracy, $r''$ has to be determined with sub-pixel precision from the image by interpolation.

### 3.1.11 Examples

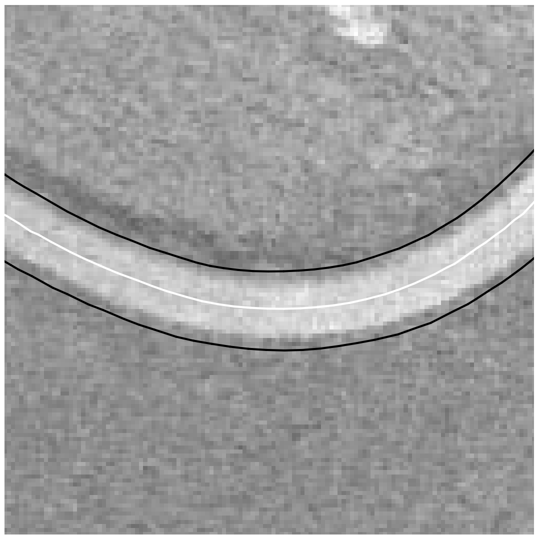
Figure 3.28 shows how the bias removal algorithm is able to adjust the line widths in the aerial image of Figure 3.23. Note from Figure 3.28(a) that because the lines in this image are fairly symmetrical, the line positions have been adjusted only very slightly. Furthermore, it can be seen that the line widths correspond much better to the true line widths. Figure 3.28(b) shows a four times enlarged part of the results superimposed onto the image in its original ground resolution of 0.25 m, i.e., four times the resolution in which the line extraction was carried out.



(a) Lines detected with bias removal



(b) Detail of (a)

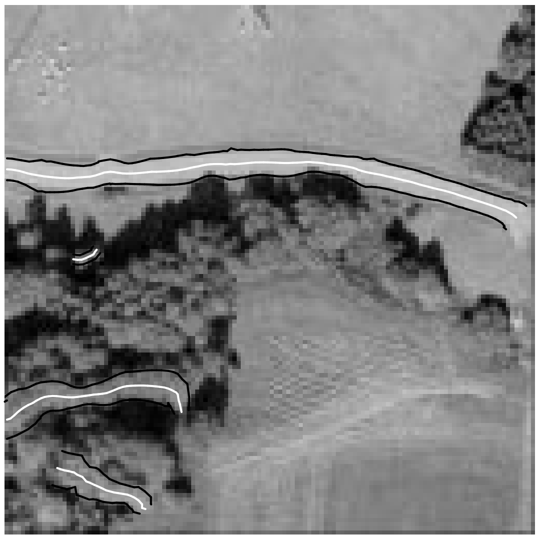


(c) Detail of (a) without bias removal

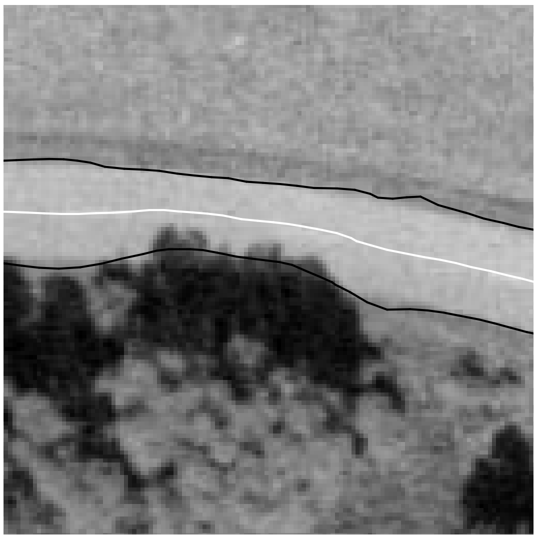
Figure 3.28: Lines and their width detected (a) in an aerial image of resolution 1 m with the bias removed. A four times enlarged detail (b) superimposed onto the original image of resolution 0.25 m. (c) Comparison to the line extraction without bias removal.

For most of the line the edges are well within one pixel of the edge in the larger resolution. Figure 3.28(c) shows the same detail without the removal of the bias. In this case, the extracted edges are about 2–4 pixels from their true locations. The bottom part of Figure 3.28(a) shows that sometimes the bias removal can make the location of one edge worse in favor of improving the location of the other edge. However, the position of the line is affected only slightly.

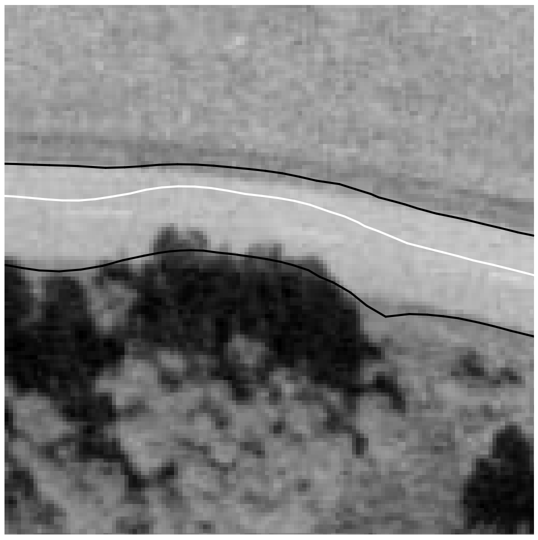
Figure 3.29 shows the results of removing the bias from the test image in Figure 3.24. Please note that in the areas of the image where the line is highly asymmetrical the line and



(a) Lines detected with bias removal



(b) Detail of (a)



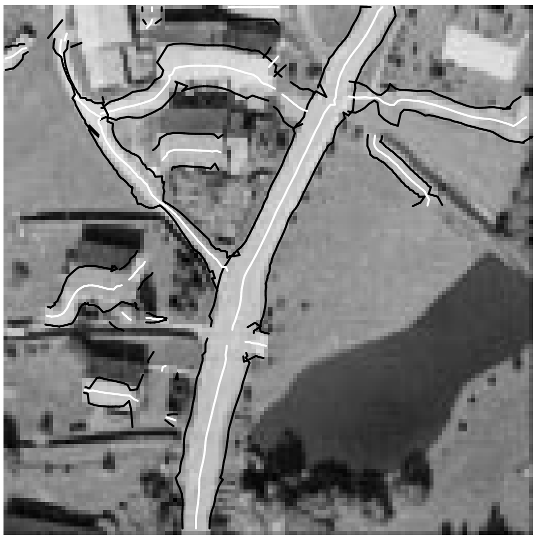
(c) Detail of (a) without bias removal

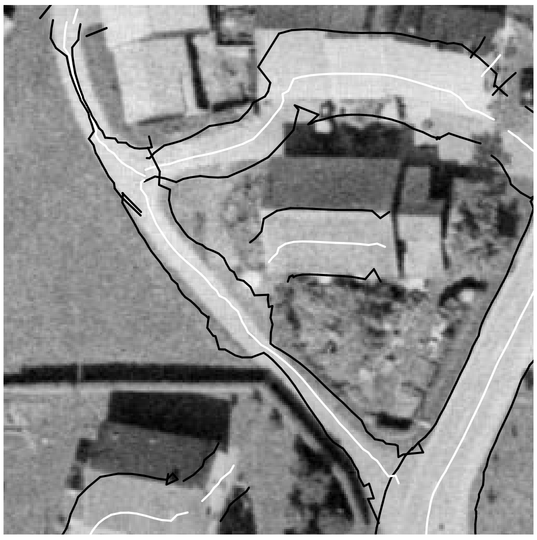
Figure 3.29: Lines and their width detected (a) in an aerial image of resolution 1 m with the bias removed. A four times enlarged detail (b) superimposed onto the original image of resolution 0.25 m. (c) Comparison to the line extraction without bias removal.

edge locations are much improved. In fact, for a very large part of the road the line position is within one pixel of the road markings in the center of the road in the high resolution image. Again, a four times enlarged detail is shown in Figure 3.29(b). When this is compared to the detail in Figure 3.29(c) the significant improvement in the line and edge locations becomes apparent.

The final example in the domain of aerial images is a much more difficult image since it contains much structure. Figure 3.30(a) shows an aerial image, again of ground resolution 1 m.

(a) Input image


(b) Lines detected with bias removal


(c) Detail of (b)

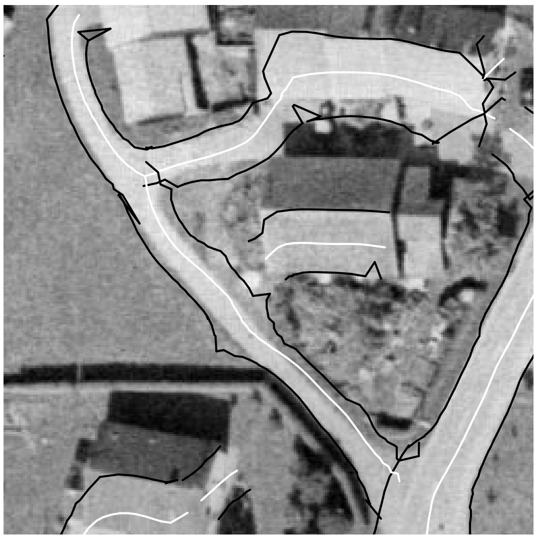

(d) Detail of (b) without bias removal

Figure 3.30: Lines and their width detected (b) in an aerial image of resolution 1 m (a) with bias removal. A two times enlarged detail (c) superimposed onto the original image of resolution 0.25 m. (d) Comparison to the line extraction without bias removal.

This image is very hard to process correctly because it contains a large area where the line model does not hold. There is a very narrow line on the left side of the image that has a very strong asymmetry in its lower part in addition to another edge being very close. Furthermore, in its upper part the house roof acts as a nearby line. In such cases, the edge of a line can only move outward much less than predicted by the model. This complex interaction of multiple lines or lines and multiple edges is very hard to model. An insight into the complexity of the scale-space interactions going on in such cases can be gained from [128], where line models of multiple parallel lines are examined. Figure 3.30(b) shows the result of the line extraction algorithm with bias removal. Since in the upper part the edges of the line cannot move as far outward as the model predicts, the width of the line is estimated as almost zero. The same holds for the lower part of the line. The reason that the bias removal corrects the line width to near zero is that small errors in the width extraction lead to a large correction for very narrow lines, i.e., if $v_\sigma$ is close to 2, as can be seen from Figure 3.25(a). Furthermore, in the area of the strong shadow caused by the hedge, the line width is extracted too large because the edge point is drawn to this strong edge. This effect can best be seen in the low resolution image, where it is clear that the true road edge has been eliminated by the smoothing and sub-sampling process. Please note, however, that the algorithm is still able to move the line position to within the true line in its asymmetrical part. This is displayed in Figures 3.30(c) and (d). The extraction results are enlarged by a factor of two and superimposed onto the original image of ground resolution 0.25 m. Please note also that despite the fact that the width is estimated incorrectly, the line positions nevertheless correspond very closely to the true line positions in the whole image.

The next example is taken from the domain of medical imaging. Figure 3.31(a) shows a magnetic resonance (MR) image of a human head. The results of extracting bright lines with bias removal are displayed in Figure 3.31(b), while a three times enlarged detail from the left center of the image is given in Figure 3.31(c). The extracted line positions and widths are very good throughout the image. Whether or not they correspond to “interesting” anatomical features is application dependent. Note, however, that the skin and fatty tissue of the head, the gyri of the brain, and several other features are extracted with high precision. Compare this to Figure 3.31(d), where the line extraction was done without bias removal. Note that the line positions are much worse for the gyri of the brain since they are highly asymmetrical lines in this image.

The final example is again from the domain of medical imaging, but this time the input is an X-ray image. Figure 3.32 shows the results of applying the proposed approach to a coronary angiogram. Since the image in Figure 3.32(a) has very low contrast, Figure 3.32(b) shows the same image with higher contrast. Figure 3.32(c) displays the results of extracting dark lines from Figure 3.32(a), the low contrast image, superimposed onto the high contrast image. A three times enlarged detail is displayed in Figure 3.32(d). In particular, it can be seen that the algorithm is very successful in delineating the vascular stenosis in the central part of the image. Note also that the algorithm was able to extract a large part of the coronary artery tree. The reason that some arteries were not found is that very restrictive thresholds were set for this example. Therefore, it seems that the presented approach could be used in a system like the one described in [34] to extract complete coronary trees. However, since the presented algorithm does not generate many false hypotheses, and since the extracted lines are already connected into lines and junctions, no complicated perceptual grouping would be necessary, and the rule base would only need to eliminate false arteries, and could therefore be much smaller.

## 3.2 Lines with Different Polarity

This section discusses lines that have different polarity, i.e., for which the background is darker on one side of the line, while it is brighter of the other side. As mentioned in the introduction to this chapter, this type of lines does not occur frequently for interesting objects in real images.

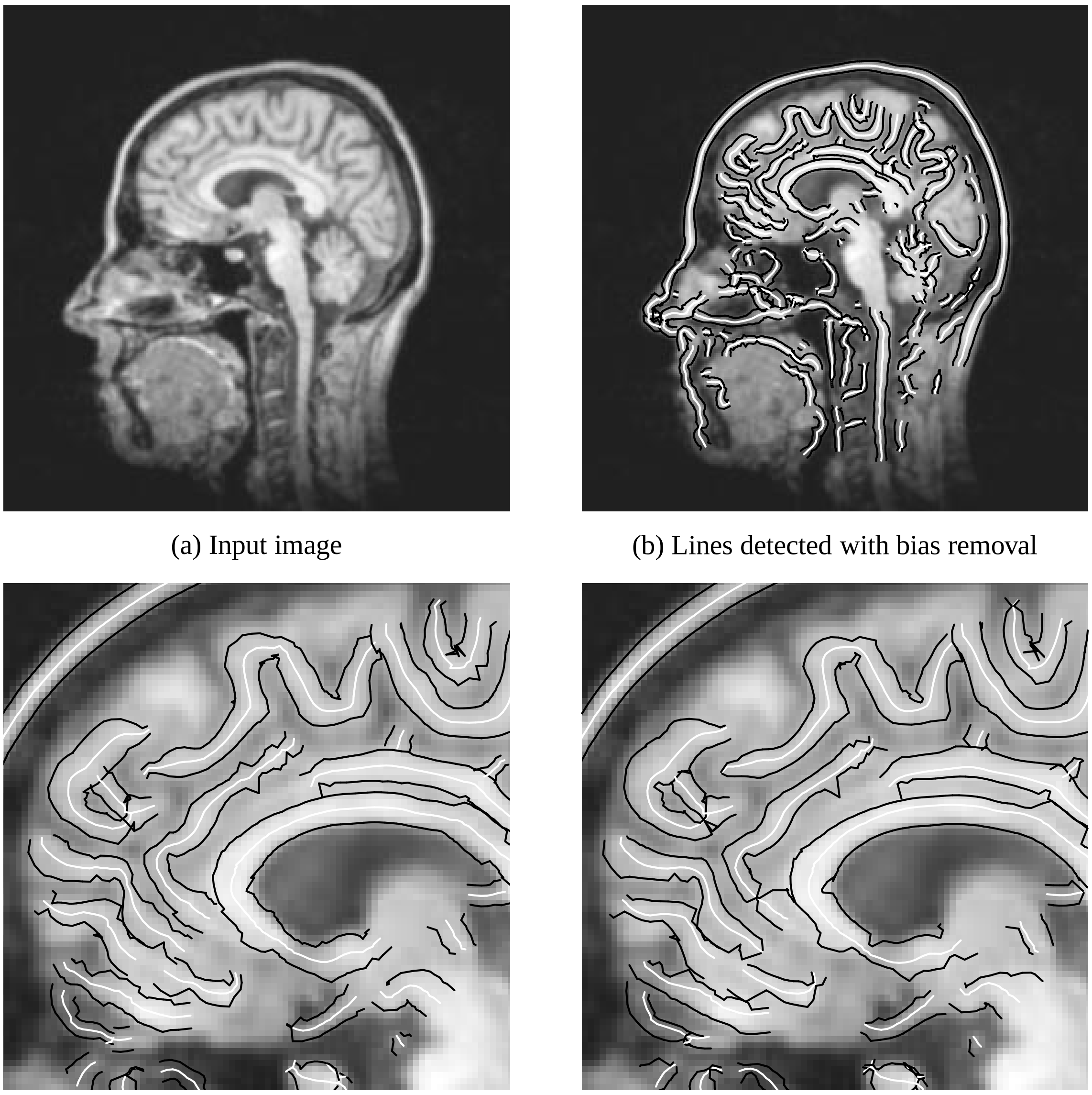
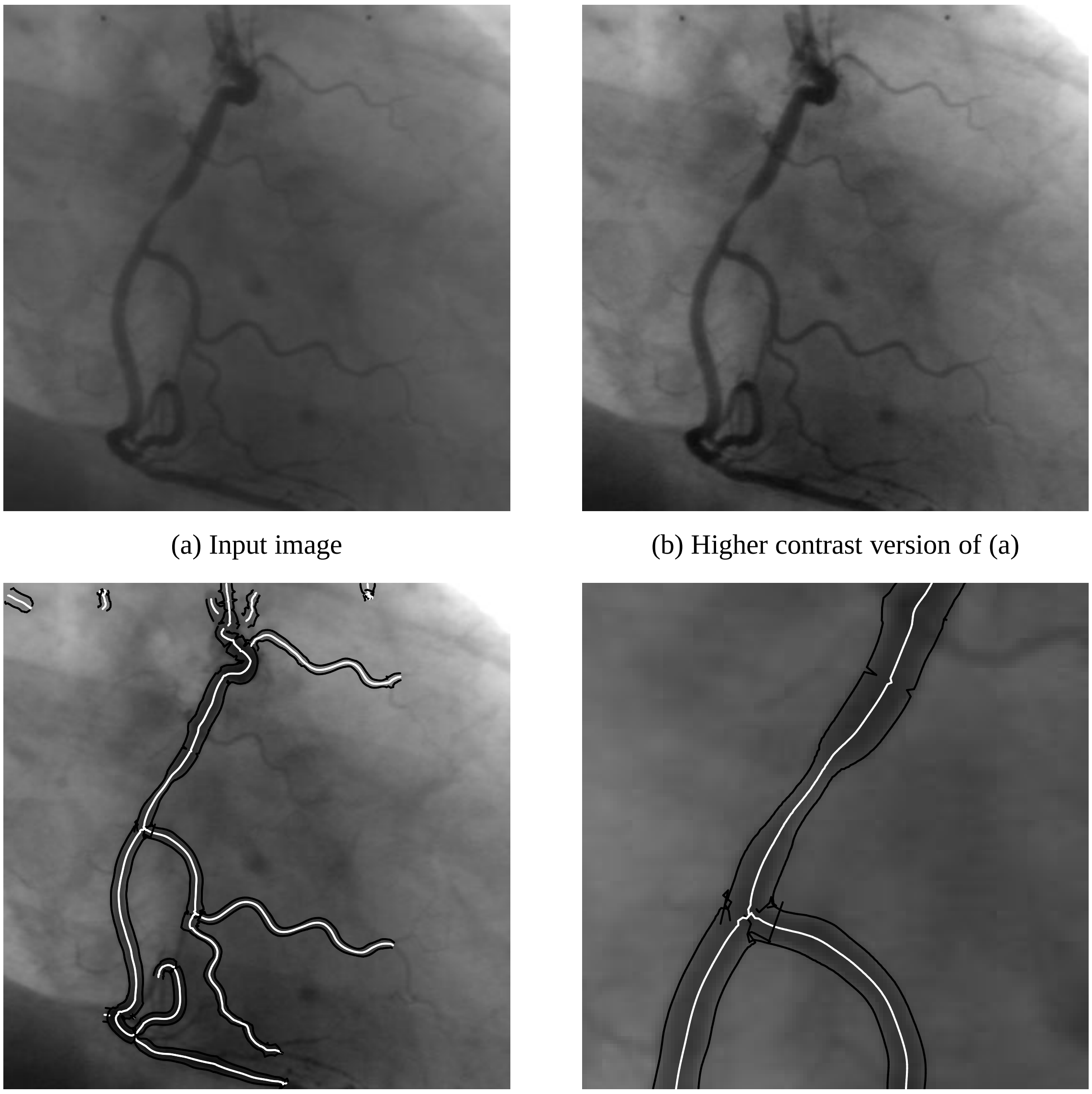

(a) Input image
(b) Lines detected with bias removal
(c) Detail of (b)
(d) Detail of (b) without bias removal

Figure 3.31: Lines and their width detected (b) in a MR image (a) with the bias removed. A three times enlarged detail (c) superimposed onto the input image. (d) Comparison to the line extraction without bias removal.

In fact, very few were found by the author in a large collection of images. Nevertheless, they are interesting from a theoretical point of view. For this reason, the discussion in this section will mainly use synthetic test images.

### 3.2.1 Line Model

As is the case for lines with equal polarity, lines with different polarity can be modeled as curves $s(t)$ with a characteristic profile perpendicular to the line direction. For this type of



(a) Input image

(b) Higher contrast version of (a)

(c) Lines and their widths detected in (a)

(d) Detail of (c)

Figure 3.32: Lines detected in the coronary angiogram (a). Since this image has very low contrast, the results (c) extracted from (a) are superimposed onto a version of the image with better contrast (b). A three times enlarged detail of (c) is displayed in (d).

lines, a useful model profile is given by an asymmetrical “staircase” profile, i.e., a combination of two step edges of different contrast. The generic model profile thus can be parameterized as follows:

$$f_s(x) = \begin{cases} 0, & x < -w \\ a, & |x| \le w \\ 1, & x > w \ . \end{cases} \tag{3.53}$$

As was the case for the asymmetrical bar-shaped line, lines of arbitrary contrast can be obtained by scaling this profile.

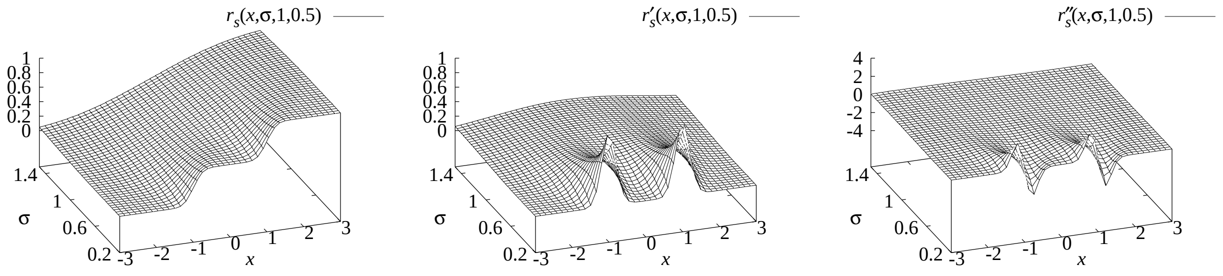


Figure 3.33: Scale-space behavior of the staircase line $f_s$ with $w = 1$ and $a = 0.5$ when convolved with the derivatives of Gaussian kernels for $x \in [-3, 3]$ and $\sigma \in [0.2, 1.5]$.

### 3.2.2 Detection of Line Points

In order to extract line points with the profile given by (3.53), the profile must be convolved with the derivatives of a Gaussian in order to get a smooth function. The respective responses are given by

$$r_s(x, \sigma, w, a) = a\phi_\sigma(x + w) + (1 - a)\phi_\sigma(x - w) \tag{3.54}$$

and its derivatives. Figure 3.33 shows the scale-space behavior of a staircase line with $w = 1$ and $a = 0.5$. As can be seen, in the one-dimensional case the position of the lines is given by the inflection point of the smoothed line profile, where additionally the inflection must be "flat," i.e., the line position is given by $r''_s(x, \sigma, w, h) = 0$ with $r'''_s(x, \sigma, w, h)r'_s(x, \sigma, w, h) > 0$. This is very similar to the definition of edges in the 1D case, given in Section 3.1.2, where the inflection point is required to be non-flat. Considering Figure 3.33, it is clear that the flat inflection points will turn non-flat if $\sigma$ is chosen large enough compared to the line width $w$, i.e., the line will turn into an edge. Because for $a = 0.5$ the line or edge position is at $x = 0$ for all $\sigma$, this can be analyzed analytically by requiring $r'''_s(0, \sigma, w, 0.5) = 0$. With this, by a straightforward calculation it can be shown that lines can only be extracted for $\sigma < w$. For other values of $a$, it can be expected that the line can be extracted for even smaller values of $\sigma$ only. Since for applications $\sigma$ is constant, while $a$ and $w$ vary in the image, it is interesting to determine the range of $a$ and $w$ for which lines can be extracted with a given $\sigma$. This range is delimited by the two-dimensional roots of the equation

$$\begin{pmatrix} r''_s(x, \sigma, w, a) \\ r'''_s(x, \sigma, w, a) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} . \tag{3.55}$$

After a very complicated calculation it can be shown that lines can only be extracted for

$$1 - a_s(w, \sigma) < a < a_s(w, \sigma) \ , \tag{3.56}$$

where

$$a_s(w, \sigma) = \frac{1}{1 - \dfrac{w^2 - \sigma^2 + w\sqrt{w^2 - \sigma^2}}{w^2 - \sigma^2 - w\sqrt{w^2 - \sigma^2}} e^{-2\frac{w\sqrt{w^2-\sigma^2}}{\sigma^2}}} . \tag{3.57}$$

The location $x_s(w, \sigma)$, at which the root of (3.55) occurs is given by

$$x_s(w, \sigma) = \pm\sqrt{w^2 - \sigma^2} . \tag{3.58}$$

The interested reader may verify the correctness of (3.57) and (3.58) by plugging them into (3.55). Figure 3.34 shows the range of $w$ and $a$ for which lines can be extracted with $\sigma = 1$. It can be seen that the range of asymmetries grows for larger line widths. Furthermore, theoretically it is possible to extract lines of arbitrary width with $\sigma = 1$ in the continuous case.

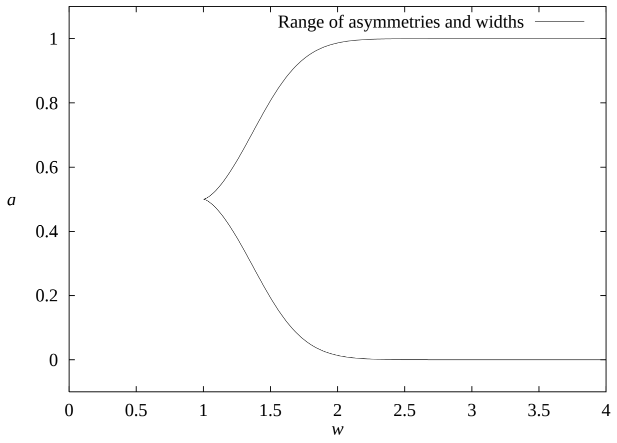


Figure 3.34: Range of widths $w$ and asymmetries $a$ for which lines can be extracted with $\sigma = 1$. The extractable range is contained between the two curves.

Finally, it should be noted that salient staircase lines must be selected based on the magnitude of $r_s'''$ since the first two derivatives will be very small and zero at line points, respectively. As was the case for the bar-shaped line in Section 3.1.2, the magnitude of $r_s'''$ does not exhibit a unique maximum if $\sigma$ is small compared to $w$. Since salient lines are selected based on this value it is necessary to derive a restriction on $\sigma$ similar to (3.15). This analysis can only be carried out for symmetrical staircase lines with $a = 0.5$ since only then the line position can be determined analytically. With these parameters, the point at which a unique maximum of $r_s'''(x, \sigma, w, a)$ is obtained is given by

$$\frac{\partial}{\partial\sigma} r_s'''(0, \sigma, w, 0.5) = 0 \ . \tag{3.59}$$

It is not very hard to show that

$$\sigma \geq \sqrt{\frac{3-\sqrt{6}}{3}} w \tag{3.60}$$

has to hold for this. Combined with the above restriction $\sigma < w$, the value of $\sigma$ should be chosen in the range

$$\sqrt{\frac{3-\sqrt{6}}{3}} w \leq \sigma < w \tag{3.61}$$

if lines of width $w$ are to be extracted.

### 3.2.3 Examples of Biased Extraction Results

If staircase lines are extracted with the above definition and their edges with the definition of Section 3.1.2, the results will again generally be biased. The simplest example of this behavior occurs for the symmetrical case $a = 0.5$. Figure 3.35 displays the line and edge positions for a line with $w \in [0, 4]$ extracted with $\sigma = 1$. As can be seen, only lines with $w > 1$ can be extracted. Furthermore, for $w \approx 1$ the line width is extracted significantly too small, while for large $w$ the extracted line width is close to the true line width. Since the staircase is symmetrical the extracted line position equals the true line position for all $w$. Interestingly, the shape of the map that describes the relation between the true and extracted widths qualitatively is very similar to the map of Figure 3.5 that describes this relation for the bar-shaped line, if that map is inverted.

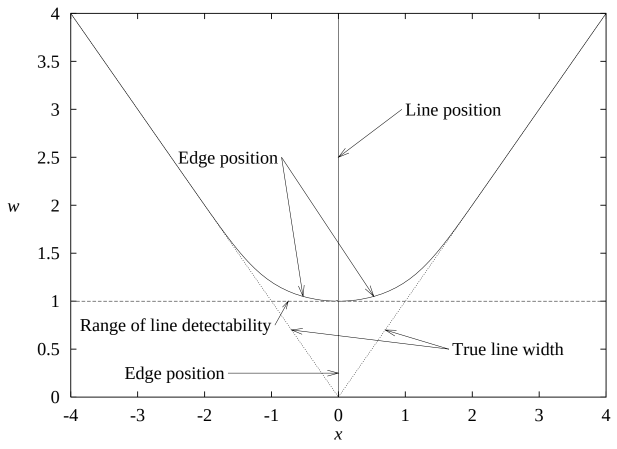


Figure 3.35: Location of a staircase line with $a = 0.5$, $w \in [0, 4]$ and its edges for $\sigma = 1$. Note that only lines with $w > 1$ can be extracted.

A more complex behavior occurs for non-symmetrical lines. Figure 3.36 shows the line and edge positions for a staircase line with $a = 0.75$ for $w \in [0, 4]$ extracted with $\sigma = 1$. As can be seen, lines can only be extracted for $w > 1.424$ for this choice of $a$. Furthermore, the line position moves to the weak side of the line, i.e., the side with the smaller gradient, for small $w$. Again, for large $w$ the extracted line widths correspond closely to the true line widths, while the extracted line position asymptotically approaches the true line position. However, this convergence is very slow.

Finally, it is instructive to examine the behavior of the line positions and widths if $\sigma$ and $w$ are kept constant, while the asymmetry $a$ is varied. An example of this behavior for $\sigma = 1$ and $w = 1.5$ is displayed in Figure 3.37. As is to be expected, the behavior of the line and edge position is symmetrical with respect to $a = 0.5$. The line position is biased by a large amount for small and large $a$. The edge position is also biased, albeit by a smaller amount, for these choices of $a$.

From these examples it can be seen that the extracted line and edge positions of staircase lines generally exhibit a bias, similar to the asymmetrical bar-shaped lines. The major difference is that for staircase lines the extracted line position always remains within the true line. Furthermore, the extracted line width is always too small, while it is always too large for lines of equal polarity. Therefore, for lines with different polarity a bias removal algorithm is also essential to obtain meaningful results.

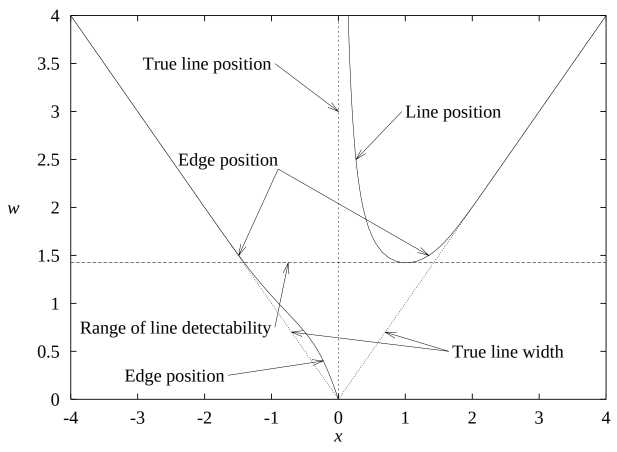


Figure 3.36: Location of a staircase line with $a = 0.75$, $w \in [0, 4]$ and its edges for $\sigma = 1$.

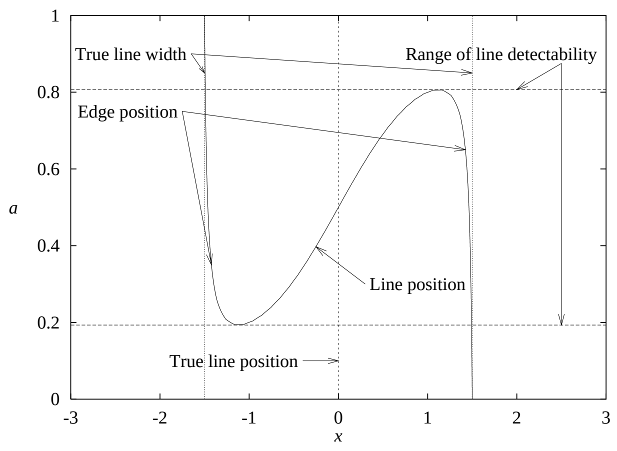


Figure 3.37: Location of a staircase line with $w = 1.5$, $a \in [0, 1]$ and its edges for $\sigma = 1$.

### 3.2.4 Detection of Lines in 2D

The discussion so far has been concerned with the extraction of line points in 1D. It is not immediately obvious how to derive a 2D extraction algorithm for lines with different polarity. However, if the gradient image is considered, it can be seen that such a line manifests itself as a dark line there. An example of this is displayed in Figure 3.38(a), where two rings of lines of different and equal polarity are displayed. In the gradient image in Figure 3.38(b) the line of different polarity can be seen as a dark line. Therefore, one way to extract lines with different polarity is to use the facet model line detector used to extract the line width in Section 3.1.9 to extract dark lines from the gradient image. However, as Figure 3.38(b) shows, lines with equal polarity also result in dark lines in the gradient image. Since only lines with different polarity should be detected, this simple approach is not sufficient.

Fortunately, there is a simple criterion by which lines of different and equal polarity can be discerned. From (3.54) it is obvious that the gradients point in the same direction on both sides of a line of different polarity, while they point in opposite directions for lines with equal polarity.

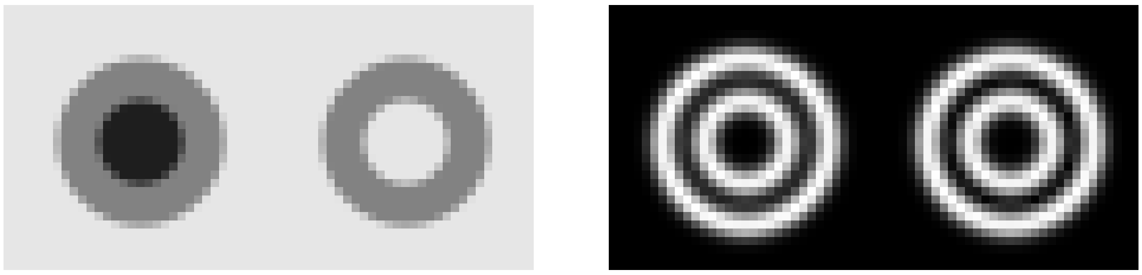

(a) Input image (b) Gradient image

Figure 3.38: An image containing two rings of lines with different and equal polarities, respectively (a). The gradient image (b) shows that both types of lines result in dark lines in the gradient image.

Of course, this only holds if the gradient direction is evaluated at a reasonable distance from the line point. If the gradients are taken too close to the line, they might be close to zero, rendering the direction estimation unreliable. The same holds true if the gradients are evaluated too far away from the line. In practice, a good distance at which to evaluate the gradients has turned out to be $\sigma$. Thus, to discern lines of different and equal polarity, the gradients at a distance of $\sigma$ in both directions perpendicular to the dark line in the gradient image are evaluated, and their dot product is calculated. Only if the dot product is positive, a line of different polarity has been found. However, it is useful to be more restrictive in the selection of lines. For example, junctions of three edges are also extracted as dark lines from the gradient image in the vicinity of the junction. Therefore, in the implementation a dark line point is only accepted as a line point of different polarity if the directions of the gradients are sufficiently close. Here, sufficiently close is defined by the gradient directions forming an angle of less than $30°$. With these criteria, lines points of different polarity can be extracted fairly reliably.

To link the individual line points into lines the algorithm of Section 3.1.6 is used. The two hysteresis thresholds are applied to the eigenvalues of the dark lines in the gradient image, which are equivalent to the third directional derivative in the direction perpendicular to the original line. Figure 3.39 displays the results of extracting lines of different polarity with $\sigma = 1.5$ from the image in Figure 3.38(a). As can be seen, with the approach discussed above the line of equal polarity is not extracted. Furthermore, since the depicted line is symmetrical, the line location is very good.

In contrast, Figure 3.40(b) shows the result of extracting lines with $\sigma = 1.7$ from the input image displayed in Figure 3.40(a). Since this image contains a line with varying asymmetry $a$, the line positions are severely biased. As was to be expected from the discussion in Section 3.2.3, the line is shifted towards the side with the smaller gradient. This result shows that bias removal is essential to obtain meaningful line positions. However, before bias removal can be done the line widths needs to be determined.

### 3.2.5 Determination of Line Width

The procedure to extract the width of a staircase line is completely analogous to that for asymmetrical bar-shaped lines discussed in Section 3.1.9. For each line point, the closest edge point

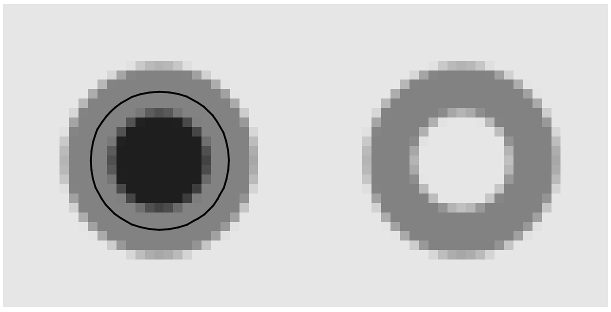


Figure 3.39: Extracted line of different polarity.

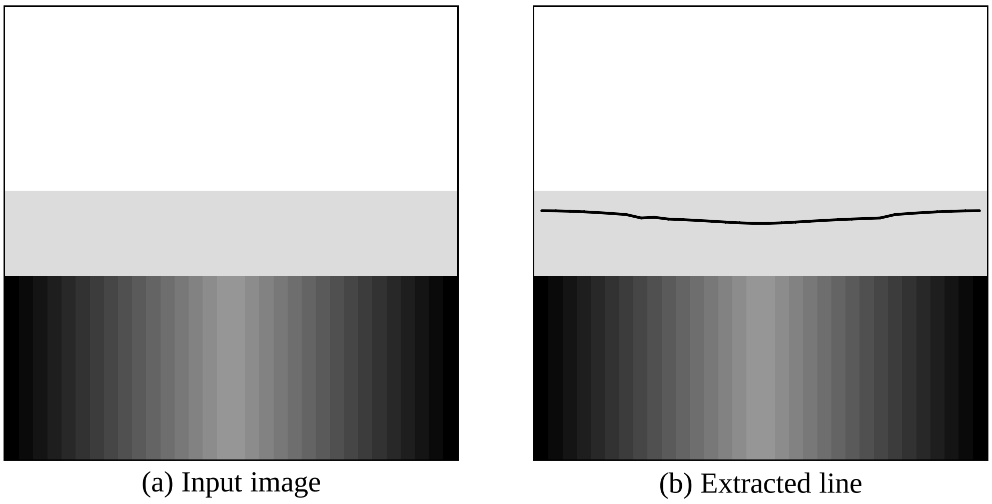


Figure 3.40: Image containing a line with varying asymmetry (a). Result (b) of extracting and linking lines with different polarity from (a).

on each side of the line is extracted as bright line points from the gradient image. The major difference for staircase lines is that the length of the search line must be chosen differently. From (3.60) it follows that ideally this length should be $\sqrt{3+\sqrt{6}}\sigma$. As was done for lines of equal polarity, it is advisable to enlarge this length slightly to $3.5\sigma$. Since for staircase lines the edges can only move inward this ensures that almost all edge points will be found for asymmetrical lines as well. Because the edge points detected by this procedure usually do not lie exactly on the search line they are projected onto the search line by the method discussed in Section 3.1.9 to obtain the line width. Furthermore, if some line widths could not be extracted, they are interpolated with the same method discussed there.

Figure 3.41(a) displays a line of varying width having an asymmetry of $a = 0.33$. The extracted line and its width is shown in Figure 3.41(b). As can be seen, the line position is shifted to the side with the smaller gradient for small line widths, while it is close to the true line position for large line widths. The extracted line widths correspond to the true line widths except for very small line widths, where they are extracted too small.

Figure 3.42 exhibits the results of extracting the line width from the test image shown in

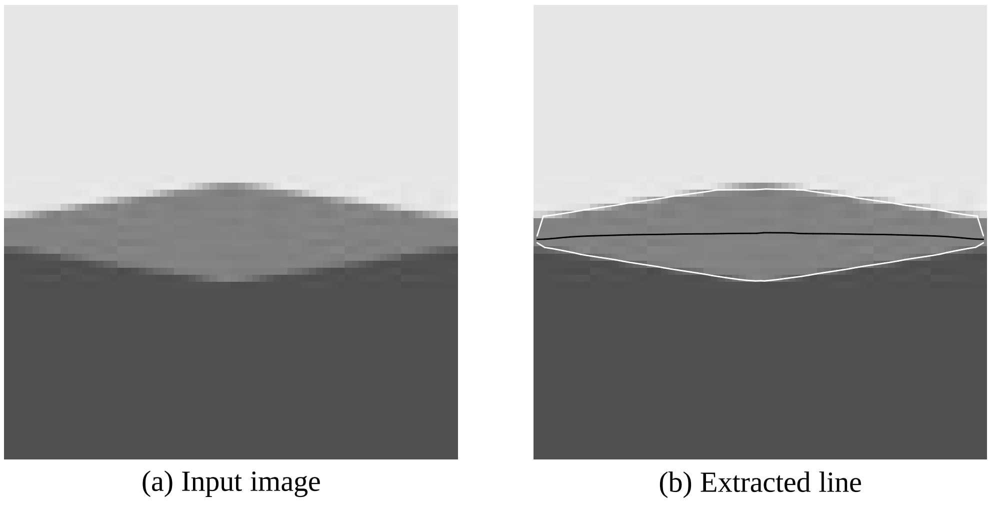

(a) Input image (b) Extracted line

Figure 3.41: Image containing a line with varying width (a). Result (b) of extracting lines with different polarity and their width from (a).

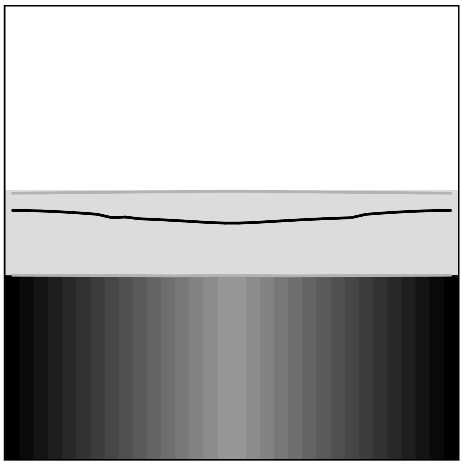


Figure 3.42: Result of extracting lines and their width from the test image containing a line with varying asymmetry.

Figure 3.40(a). As noted there, the line position is strongly biased. In contrast, the line width corresponds to the true line width for most of the line. Only for very high asymmetries the width is extracted too small.

Finally, an image containing a ring with asymmetry $a = 0.66$ is depicted in Figure 3.43(a). This image is intended to test the correctness of the line positions and widths for different line orientations. The result of extracting lines and their width is displayed in Figure 3.43(b). Again, the line positions are shifted to the side with the smaller gradient. Obviously, the quality of the line positions is not affected by the line orientation. The extracted widths correspond closely to the true line widths, except for two small "notches" in the lower right part of the ring, where a double response of the edge detector has caused the line width to be extracted too small.

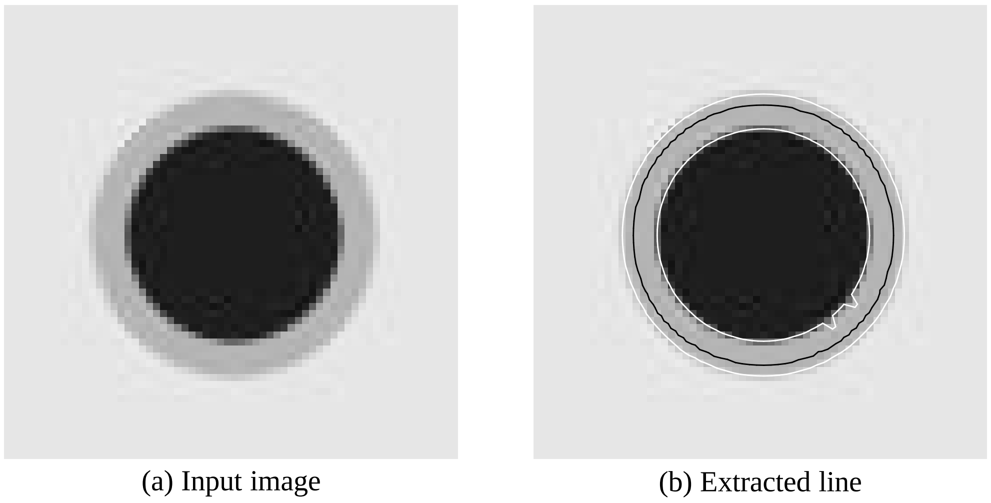


Figure 3.43: Image containing an annular line with $a = 0.66$ (a). Result (b) of extracting lines with different polarity and their width from (a).

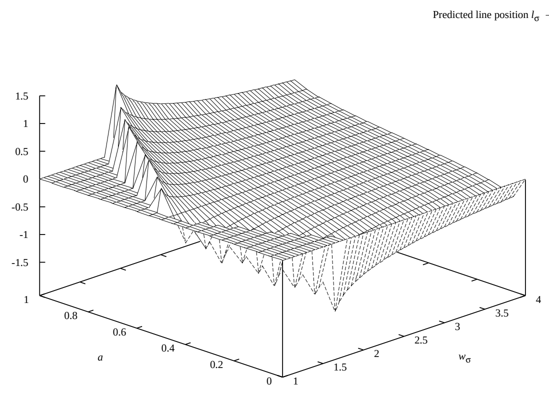
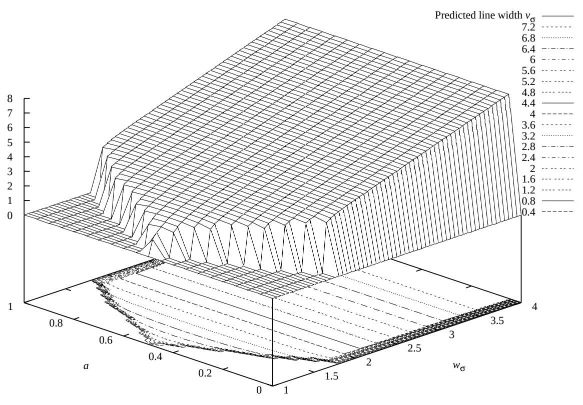


Figure 3.44: Predicted line position $l_\sigma$ of the staircase line $f_s$ for $w_\sigma \in [1, 4]$ and $a \in [0, 1]$.

## 3.2.6 Removing the Bias from Asymmetrical Lines

In order to model the bias of lines with different polarity the technique developed in Section 3.1.10 can be used with small modifications. Recall that from the scale-space description (3.54) of the staircase line it is possible to calculate the line position $l$ and the edge positions $e_l$ and $e_r$ by a one-dimensional root finding algorithm. Thus, the widths $v_l = l - e_l$ and $v_r = e_r - l$ to the left and right side of the line can be determined. With this, the total width $v = v_l + v_r$ of the line can be calculated. A simple calculation suffices to convince oneself that Proposition 1 of Section 3.1.10 also holds for lines with different polarity. Thus, only quantities normalized with respect to the scale parameter $\sigma$ need to be considered, e.g., $w_\sigma = w/\sigma$, and the analysis can be carried out for $\sigma = 1$.

With these preliminaries, the predicted line position $l_\sigma$ can be calculated for all $w_\sigma$ and $a \in [0, 1]$. The result is shown in Figure 3.44. It can be seen that the bias of the line position is largest for small line widths and large asymmetries. Furthermore, the bias is symmetrical


(a) Predicted line width

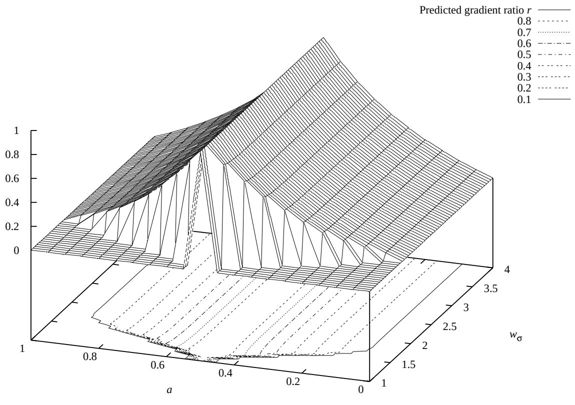
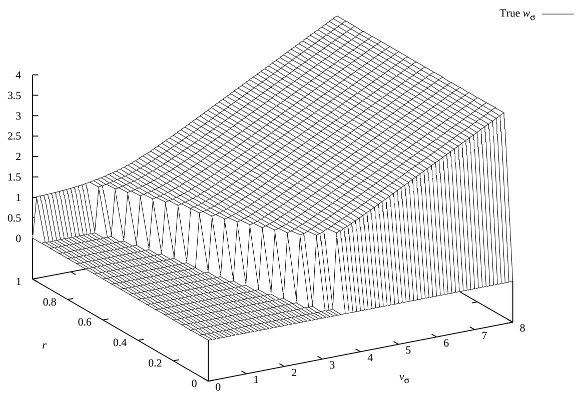

(b) Predicted gradient ratio

Figure 3.45: Predicted behavior of the staircase line $f_s$ for $w_\sigma \in [1,4]$ and $a \in [0,1]$. (a) Predicted line width $v_\sigma$. (b) Predicted gradient ratio $r$.


(a) True $w_\sigma$

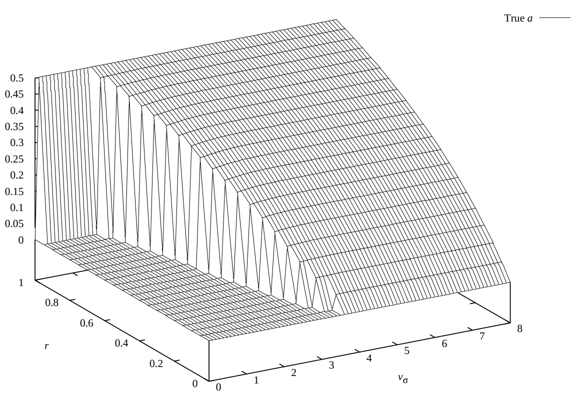

(b) True $a$

Figure 3.46: True values of the line width $w_\sigma$ (a) and the asymmetry $a$ (b) of the staircase line $f_s$.

with respect to $a = 0.5$. Note that the positions are only defined within the range of $a$ and $w$ defined by (3.56). Furthermore, the predicted line width $w_\sigma$ and gradient ratio $r$ can be calculated. They are displayed in Figure 3.45 along with their contour lines. As can be seen, the predicted line width behaves much better than its counterpart for lines with equal polarity shown in Figure 3.25. The gradient ratio is also very well behaved. Note that again, everything is symmetrical with respect to $a = 0.5$. Therefore, only asymmetries $a \leq 0.5$ need to be considered. For larger asymmetries, the width and gradient ratio remain unchanged, while the sign of the predicted line position changes.

As is obvious from Figure 3.45, for $a \leq 0.5$ the bias function is invertible since the contour lines of $v_\sigma$ and $r$ intersect almost perpendicularly everywhere. Hence, the inverted bias function can be calculated by a multi-dimensional root finding algorithm to obtain the true values of $w_\sigma$ and $a$. They are shown in Figure 3.46. Furthermore, the absolute value of the correction $c_\sigma$ to be applied to the line position in order to obtain the true line position is displayed in Figure 3.47.

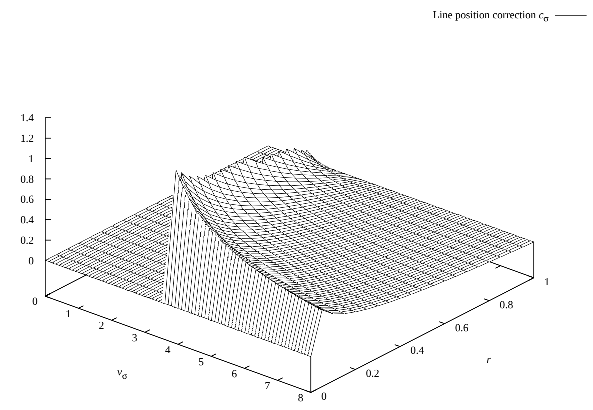
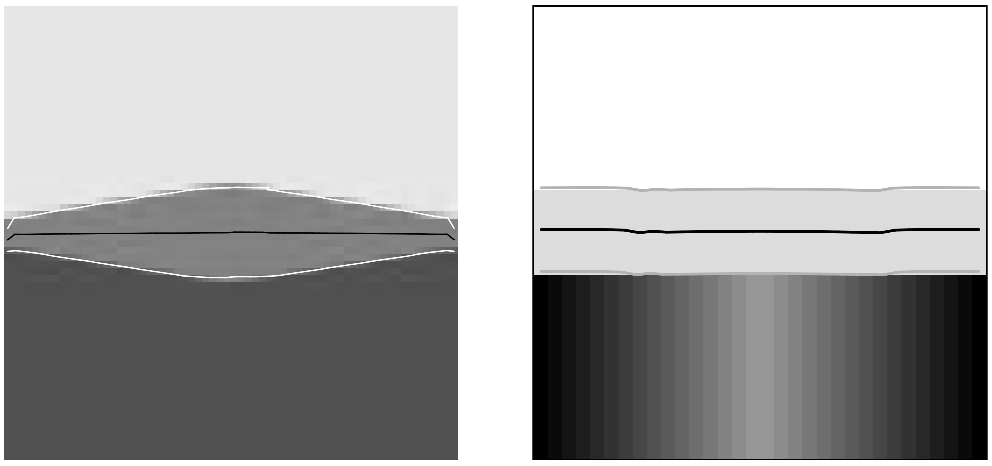


Figure 3.47: Line position correction for the staircase line $f_s$.



(a) Corrected lines (b) Corrected lines

Figure 3.48: Result of correcting the line positions and widths for the test images containing a line of varying width (a), varying asymmetry (b).

The only remaining question is how the algorithm should decide whether the true asymmetry is smaller or larger than 0.5. This can simply be done based on the gradient directions at one of the two edge points and on the direction perpendicular to the line. If the dot product of the two vectors is larger than zero, the bright side of the line is the right side, and vice versa. If the bright side of the line and the side with the weaker gradient are equal, the asymmetry must be chosen larger than 0.5, otherwise smaller.

The result of applying the line position and width correction to the test image of Figure 3.41(a) containing a line of varying width is displayed in Figure 3.48(a), while the result of applying it to the test image of Figure 3.40(a) showing a line of varying asymmetry is displayed in Figure 3.48(b), and that of the test image of Figure 3.43(a) containing an annular line in Figure 3.49. As can be seen, in each case the extracted line position corresponds to the true line position, while the extracted line width is very close to the true line width.

To conclude this section, Figure 3.50(a) displays an aerial image containing a line of differ-

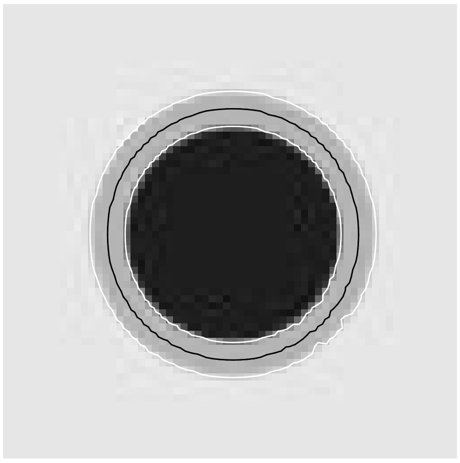


Figure 3.49: Result of correcting the line positions and widths for the test images containing an annular line.

ent polarity in the part marked by the square. This part is shown enlarged in Figure 3.50(b). The results of extracting lines of different polarity with $\sigma = 1.4$ and thresholds of 2 and 7 without and with applying the bias removal are displayed in Figures 3.50(c) and (d), respectively. As can be seen, only the road corresponding to the staircase line has been extracted. Since the line has a fairly large asymmetry, the uncorrected line positions and widths are severely biased. This bias is removed by the correction step, as is clear from Figure 3.50(d). The extracted line corresponds closely to the true position and width of the road.

## 3.3 Edge Extraction

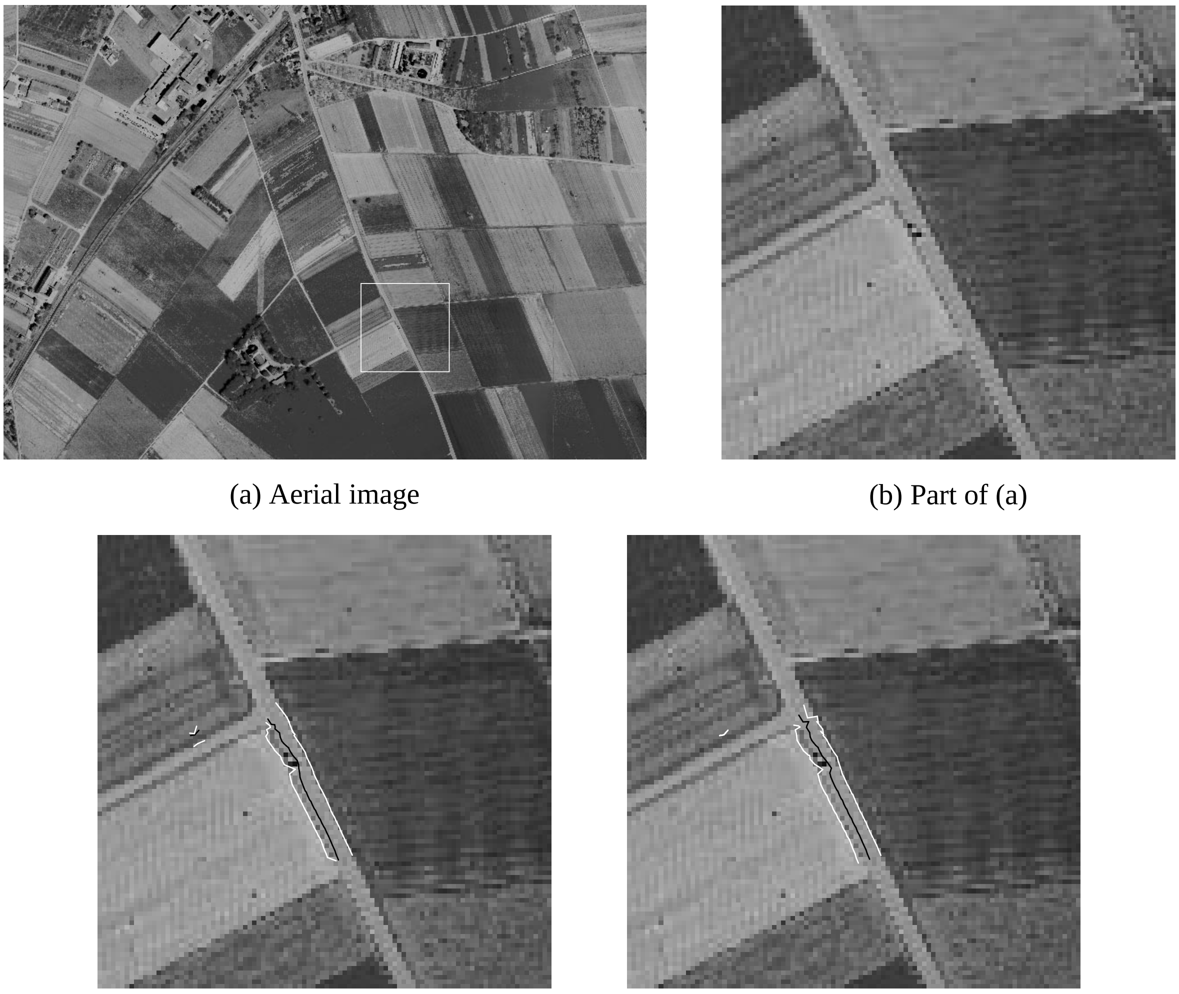
As discussed in Section 3.1.9, in an image containing the absolute value of the gradient, edges appear as bright lines. Hence, with the line extraction and linking algorithm developed in Section 3.1 it is possible to design an algorithm that turns an arbitrary gradient-based edge detector, such as the Canny [28, 29], (isotropic) Deriche [38, 39, 103], (isotropic) Shen [103, 163, 164, 165] or even the Sobel operator, into a sub-pixel edge detector.

### 3.3.1 Edge Detection Algorithm

Before the edge detection algorithm will be described, a few words on the model of the underlying edge are necessary. Most approaches use the step edge model [28, 29, 38, 103, 165], i.e., assume the edge of height $h$ to be given by

$$f_e(x) = \begin{cases} h, & x \geq 0 \\ 0, & x < 0 \ . \end{cases} \tag{3.62}$$

Other approaches assume the edge to be a Gaussian-blurred step edge, i.e., its profile to be given by $h\phi_\sigma(x)$ [114, 115]. While both these assumptions are useful, since they simplify the analysis of the operator considerably, for our purposes they are not strictly necessary. The main point


(a) Aerial image (b) Part of (a)

(c) Lines extracted without bias removal (d) Lines extracted with bias removal

Figure 3.50: Aerial image (a) containing a line with different polarity in the part marked by the square, displayed enlarged in (b). Lines extracted from (b) without (c) and with (d) bias removal.

about these step edge profiles is that they are symmetric with respect to the inflection point at $x = 0$. Therefore, if the absolute value of the first derivative of these profiles are taken, symmetrical line profiles are obtained. It is obvious from the discussion in Section 3.1.2 that under this condition, and the condition that the smoothing filter the edge operator uses is symmetrical with respect to the origin, isolated lines, and hence also edges, will be extracted in the correct sub-pixel position. Of course, if the edge profile is not symmetric to its inflection points, the edge locations will be biased [185]. If the correct edge model can be derived analytically, it is possible to model this bias with the techniques introduced in Sections 3.1.3 and 3.1.10, and hence to remove the bias.

As was discussed in Section 3.1.2, the edge points on a one dimensional profile can be extracted by the maxima of the convolution of the profile with an appropriate first derivative operator. For Canny's operator this is simply the first derivative of a Gaussian [28, 29], which he shows to be close to optimal for suitably chosen criteria, while for other approaches different filters which are optimal for different criteria are used [38, 165]. In any case, the response of convolving the edge profile by a first derivative operator is denoted by $r'_e(x, \alpha, h)$, where $\alpha$ is the filter parameter, e.g., $\sigma$ for the Gaussian, the edge points in the one-dimensional case are given by the maxima of $|r'_e(x, \alpha, h)|$, or equivalently by the zero crossings of $r''_e(x, \alpha, h)$, where additionally $r'''_e(x, \alpha, h) r'_e(x, \alpha, h) < 0$ is required. As can be seen, the first interpretation quite naturally leads to the conclusion that edges can be modeled as lines in the gradient image. In the two-dimensional case, these two edge definitions have been translated to the requirement that an edge is either given by the maxima of the first directional derivative in the direction of the gradient [73] or by the zero crossings of the Laplacian [125]. The first condition can be expressed as

$$Dr_e = \frac{r_x^2 r_{xx} + 2 r_x r_y r_{xy} + r_y^2 r_{yy}}{r_x^2 + r_y^2} = 0 \ , \tag{3.63}$$

while the second condition can be expressed as

$$\Delta r_e = r_{xx} + r_{yy} = 0 \ . \tag{3.64}$$

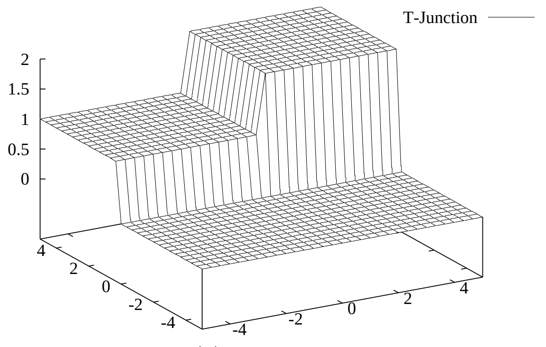
As is well known, these two definitions are equivalent if and only if the mean curvature of the image $z(x, y)$ vanishes [16]. This difference is especially noticeable for corners, where the first definition will move the edge positions to the inside of the corner, while the second definition will broaden the corner. In contrast, the interpretation of edges as lines in the gradient image leads again to the use of the De Saint-Venant condition to model the behavior of edges in 2D. In this case, edges are given by

$$Sr_e = \frac{e_x e_y (e_{xx} - e_{yy}) - (e_x^2 - e_y^2) e_{xy}}{e_x^2 + e_y^2} = 0 \ , \tag{3.65}$$

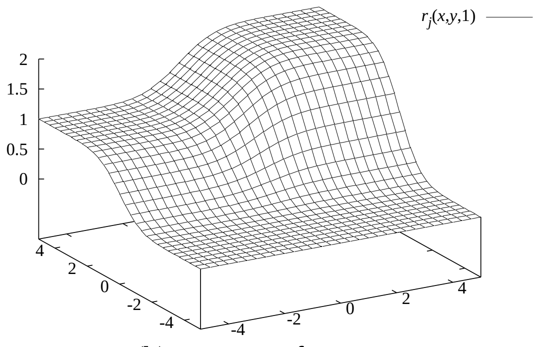
where $e_x, \dots, e_{yy}$ are calculated according to (3.42)–(3.46). These three definitions will be used below to explain the often counterintuitive failures of edge detectors to extract junctions, similar to the discussion in Section 3.1.8.

The result of the above discussion is that the facet model edge detection scheme used in Section 3.1.9 to extract the line width can be used to detect single edge points, and that the linking algorithm of Section 3.1.6 can be used to link the edge points into edges. However, two modifications must be made to make the algorithm usable. The first one is concerned with the thresholds used in the linking algorithm. As was discussed in Section 3.1.6, the second directional derivatives are used as the hysteresis thresholds. For edges, however, they have no immediately obvious connection to the edge height $h$. Therefore, once the line detector has determined all line points, the corresponding responses are substituted by the response of the edge operator at that point, and passed to the linking algorithm.

The second, and most important, change is how to implement the algorithm efficiently. It is obvious that the facet model line detector does not need to be run on the entire image since the lower threshold of the linking algorithm determines the points which cannot be edge points. Hence, running the line detector on the entire image is very wasteful. From this discussion it follows that the line detector only needs to be applied to the region of the image in which the response of the edge detector is higher than the lower hysteresis threshold. With these modifications, the resulting sub-pixel edge detector is only about 15% slower than the typical



(a) T-junction



(b) Response for $\sigma = 1$

Figure 3.51: A T-junction with $h = 2$ (a) and the response $r_j(x, y, 1)$ (b).

edge detection cycle of running the edge operator, doing a combined non-maximum suppression and hysteresis threshold operation, thresholding the resulting image, calculating a skeleton from this region, and linking the edge points into contours. This is a very small price to pay for the added bonus of sub-pixel accurate results. Furthermore, as will be discussed in the next section, the behavior of the operator at junctions is much improved compared to standard edge detectors.

### 3.3.2 Missing Junctions in Edge Detection

As is the case for lines, edge operators often fail to extract "obvious" junctions. This is especially relevant for a model-based interpretation of an image, since the knowledge that the edge operator sometimes fails to extract junctions requires additional modeling, and thus makes the knowledge base larger. As in Section 3.1.8, the reasons for these failures are explored by modeling junctions, and examining the results the above three edge definitions yield on a model junction.

Consider a T-junction where three areas of different intensity meet. A suitable model for this type of junction is

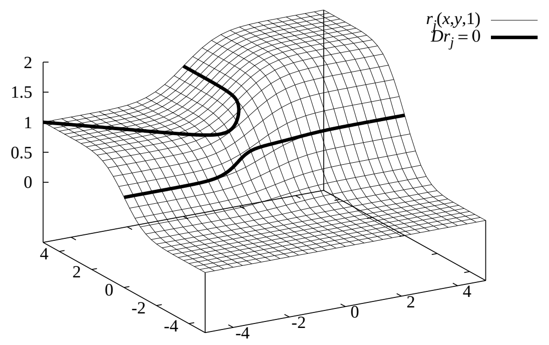
$$f_j(x,y) = \begin{cases} 0, & y < 0 \\ 1, & y \ge 0 \land x < 0 \\ h, & y \ge 0 \land x \ge 0 \ . \end{cases} \tag{3.66}$$

It is obvious that this models any possible configuration of intensities, since arbitrary intensities can always be scaled and translated such that one is zero, while another one is one. Hence the only free parameter is the third intensity. For a Gaussian smoothing filter, the scale-space description of this junction is given by

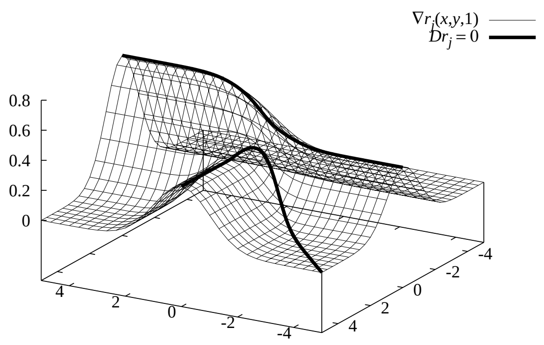
$$r_j(x,y,\sigma) = \phi_\sigma(y)((h-1)\phi_\sigma(x) + 1) \tag{3.67}$$

and its partial derivatives of first and second order. Figure 3.51 shows a T-junction with $h = 2$ and the corresponding response $r_j(x, y, 1)$ for $(x, y) \in [-5, 5] \times [-5, 5]$.

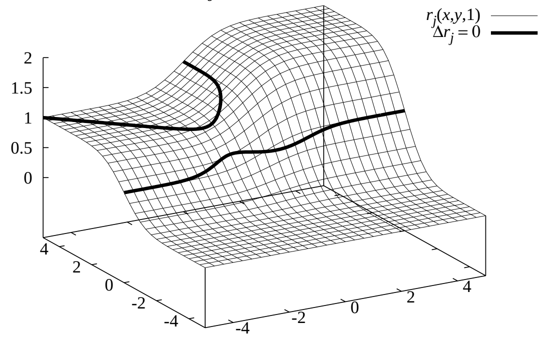
In order to analyze the behavior of the different edge definitions at junctions, the response $r_j$ can be plugged into (3.63)–(3.65). Figure 3.52 shows the edge positions extracted with the different edge definitions for the junction of Figure 3.51 with $h = 2$ mapped onto the



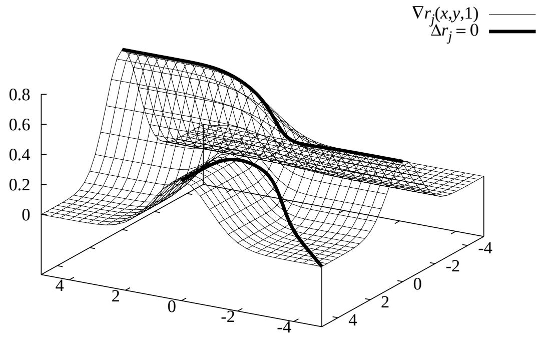
(a) $Dr_j = 0$ on response



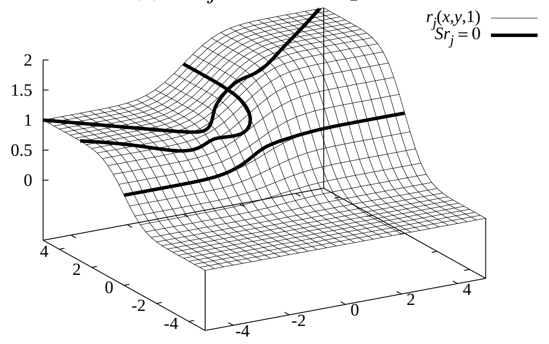
(b) $Dr_j = 0$ on gradient



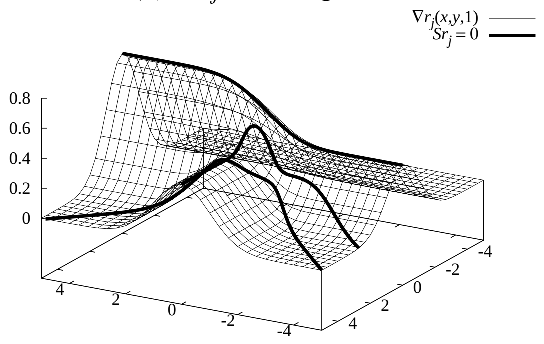
(c) $\Delta r_j = 0$ on response



(d) $\Delta r_j = 0$ on gradient



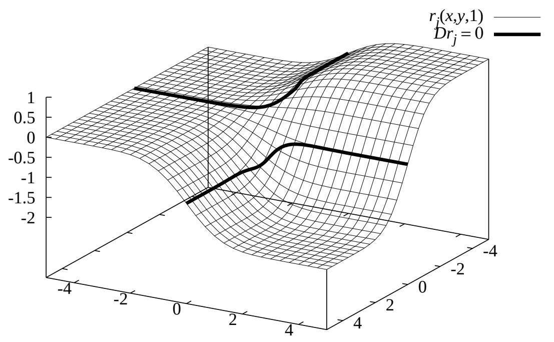
(e) $Sr_j = 0$ on response



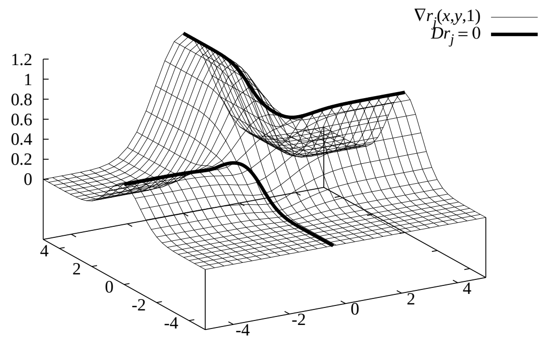
(f) $Sr_j = 0$ on gradient

Figure 3.52: Extracted edges for a T-junction with $h = 2$ by $Dr_j = 0$ mapped onto $r_j$ (a) and onto $\|\nabla r_j\|$ (b); by $\Delta r_j = 0$, (c) and (d); and by $Sr_j = 0$, (e) and (f). The graphs in the right column are rotated by $120°$ with respect to the graphs in the left column.

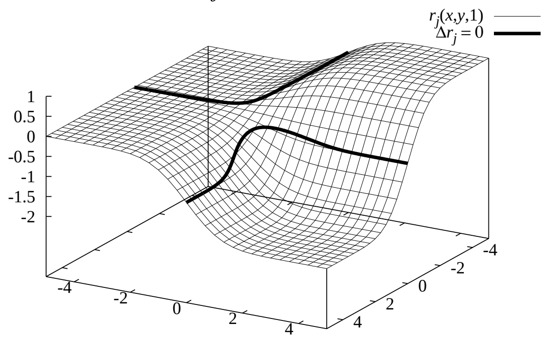
response $r_j$ and onto the absolute value of the gradient $\|\nabla r_j\|$, while Figure 3.53 shows the same for a junction with $h = -2$. As was the case for the analysis of the line junctions, the spurious edges for the three definitions have to be disregarded. For $Dr_j = 0$ these are the points where the absolute value of the gradient has a minimum in the direction of the gradient. For $\Delta r_j = 0$ they are the flat inflection points, while for $Sr_j = 0$ they are the points that are restricted minima in the direction of the maximum second directional derivative and the minima and maxima in the direction of the minimum second directional derivative. If this is kept in mind, the first interesting fact noticeable from these figures is that the edge positions extracted by $Sr_j = 0$ approach the junction much closer than the edges extracted by the other two definitions. Therefore, it can expected that the first two definitions will miss junctions much



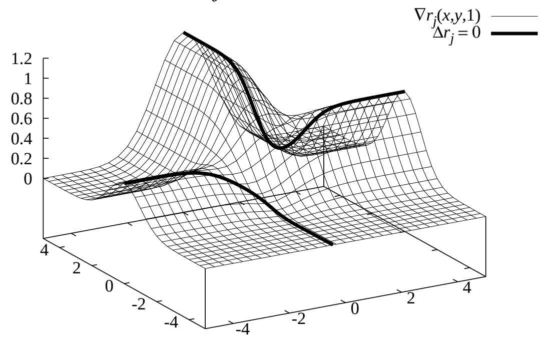
(a) $Dr_j = 0$ on response



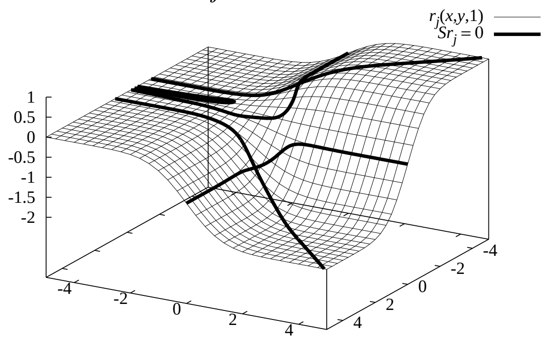
(b) $Dr_j = 0$ on gradient



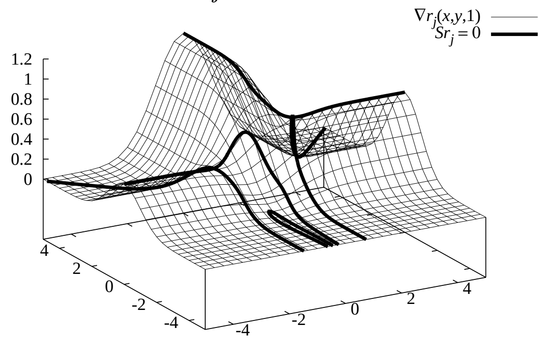
(c) $\Delta r_j = 0$ on response



(d) $\Delta r_j = 0$ on gradient



(e) $Sr_j = 0$ on response

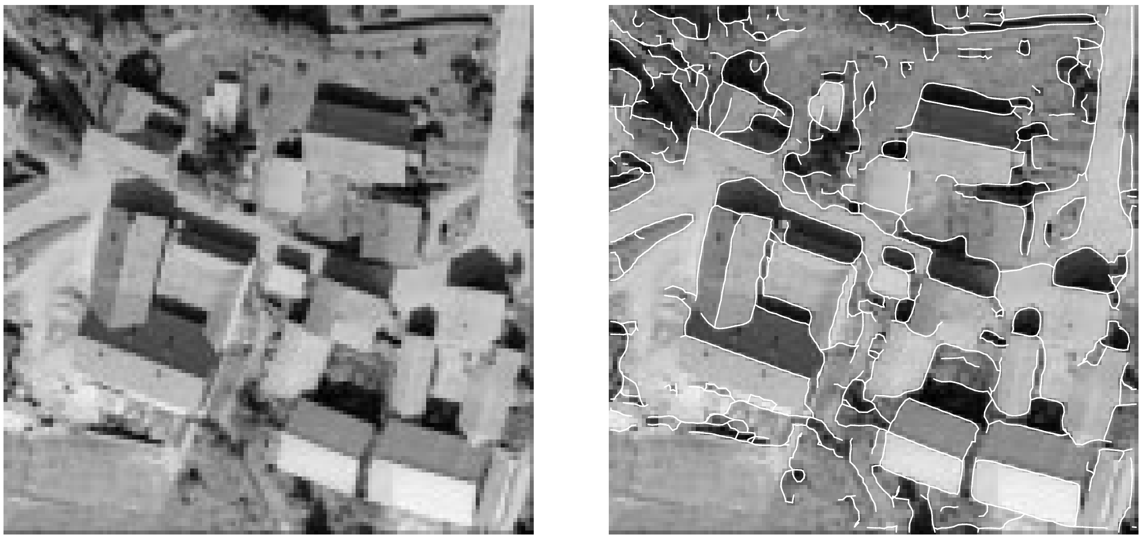


(f) $Sr_j = 0$ on gradient

Figure 3.53: Extracted edges for a T-junction with $h = -2$ by $Dr_j = 0$ mapped onto $r_j$ (a) and onto $\|\nabla r_j\|$ (b); by $\Delta r_j = 0$, (c) and (d); and by $Sr_j = 0$, (e) and (f). The graphs in the right column are rotated by $120°$ with respect to the graphs in the left column.

more frequently than the third definition, since for this definition the extracted edge often will approach the junction to within one pixel, so that the linking algorithm will be able to extract a junction. Furthermore, the direction of the edges extracted by the first two definitions in the vicinity of the junction are much worse than by the third definition. In general, as was also noted in Section 3.1.8, the edges extracted by $Sr_j = 0$ will point directly at the junction, whereas for the other two definitions this is not necessarily true, a fact that was also noted in [17, 36]. It is a surprising fact that the edge directions for $Sr_j = 0$ are much more stable than the gradient-based directions, since the former ones are based on the second derivatives, while the latter ones are based only on first derivatives.

This discussion gives another good reason to regard edges as bright lines in the gradient



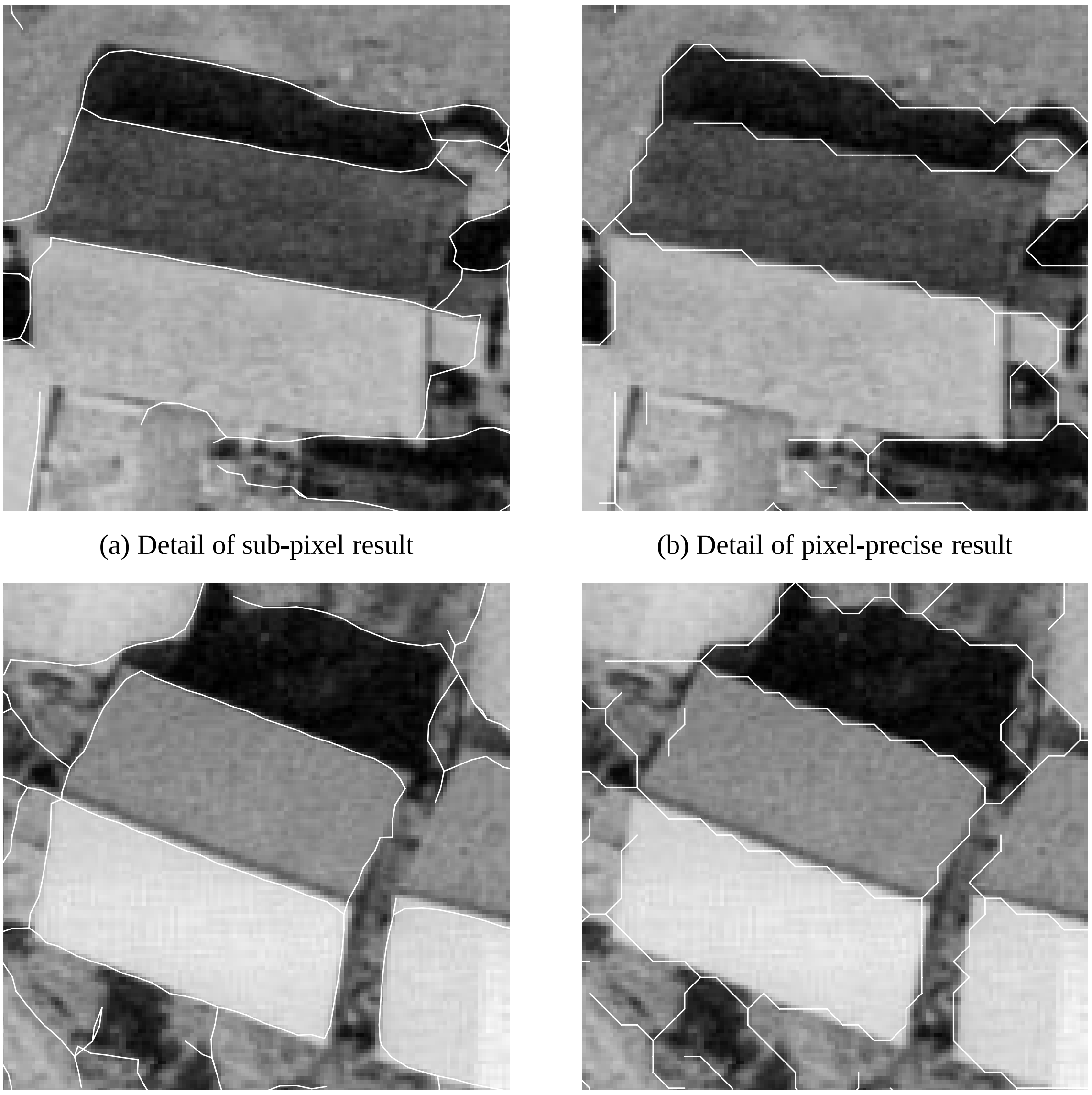
(a) Input image (b) Extracted edges

Figure 3.54: Edges (b) detected in an aerial image (a) of resolution 1 m with the proposed edge detection algorithm.

image. While the edge positions are equally good outside of junction areas, the line-based approach has a clear advantage in junction areas since the edge directions still point in the right direction in the vicinity of a junction for the edge that cannot join the junction. Therefore, the algorithm proposed for lines in Section 3.1.8 can be used to complete the missing junctions for edges as well.

### 3.3.3 Examples

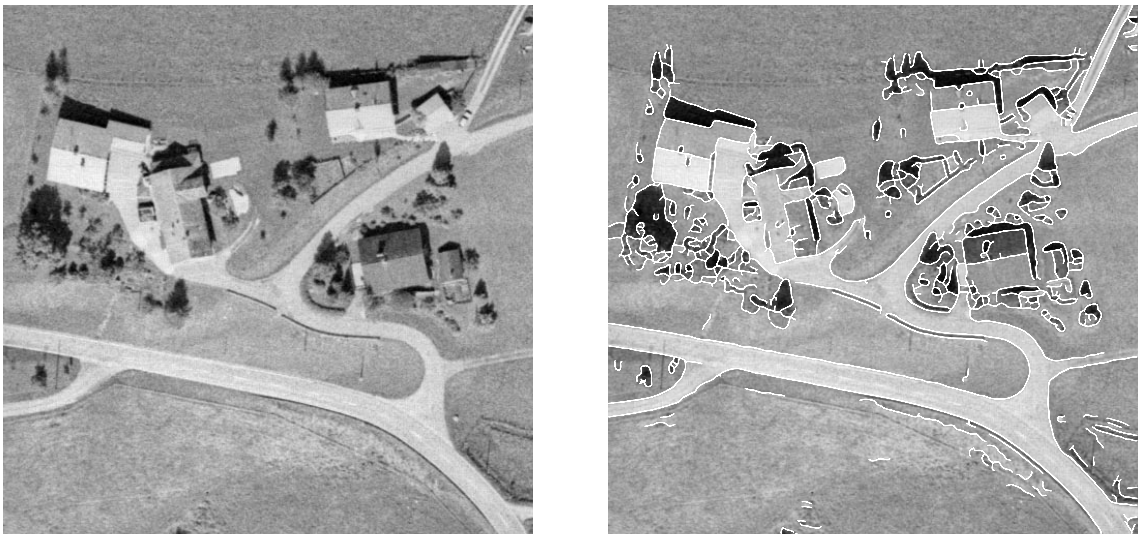
Similar to the tests on real images performed in Section 3.1.11, first the accuracy of the proposed edge detection algorithm is checked by testing it on scaled-down versions of high resolution images. In Figure 3.54(a) an aerial image of ground resolution 1 m containing several buildings is shown. The low-resolution image was obtained by uniform averaging of the original image. Figure 3.54(b) displays the result of the sub-pixel edge extraction using Gaussian derivatives with $\sigma = 0.7$ a lower threshold of 20, and an upper threshold of 40. Already in this low-resolution image the accuracy of the extracted edge positions is apparent. This is confirmed by the images in Figures 3.55(a) and (c), where the extracted edges have been mapped onto the image at the four times higher original resolution. Compare this to Figures 3.55(b) and (d), where edges have been extracted at pixel precision in the 1 m resolution image, and have also been mapped onto the original resolution image. Notice the dramatic improvement in positional accuracy of the edges. Figure 3.55(a) shows the upper-rightmost building of the input image. As can be seen, the extracted edges lie right on the gable of the roof, even in the four times higher original resolution of the image. This is also the case for the left and top edges of the darker part of the roof. This example shows that for symmetrical edges the edge locations can be recovered very precisely even from heavily scaled-down images, and thus justifies the sub-pixel approach.



(a) Detail of sub-pixel result
(b) Detail of pixel-precise result
(c) Detail of sub-pixel result
(d) Detail of pixel-precise result

Figure 3.55: Two four times enlarged parts of the extraction results on the aerial image of resolution 1 m projected onto the image at the original resolution of 0.25 m. (a) and (c) Edges extracted with the proposed sub-pixel algorithm. (b) and (d) Edges extracted with pixel resolution.

Compare this to Figure 3.55(c), which shows the second building from the right bottom corner of the image. The gable of the roof of this house contains a non-symmetrical edge due to a shadow on the upper side of the gable. Nevertheless, the extracted edge lies within one pixel of the true edge in the high-resolution image, i.e., within 0.25 pixels of the true edge in the low-resolution image. Note also that the remaining edges of the roof for the most part lie within the true edge pixel in the higher resolution image. Furthermore, it can be seen that the proposed algorithm yields much better results at junctions. See for example the left gable junction in

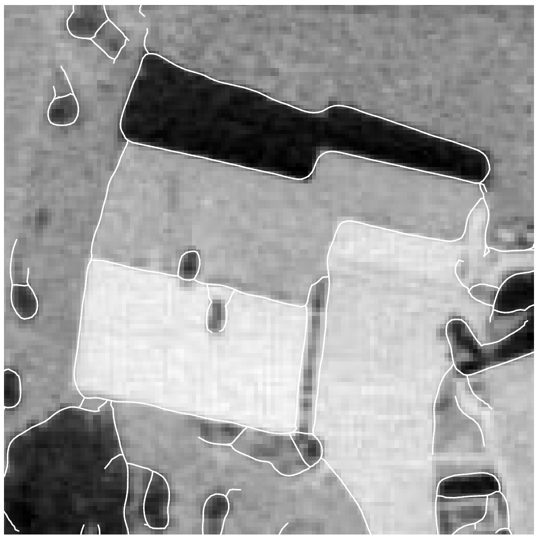


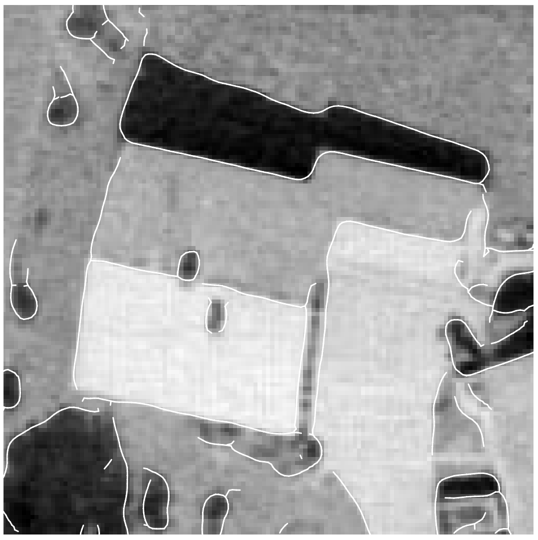
(a) Input image (b) Extracted edges

Figure 3.56: Edges (b) detected in an aerial image (a) of resolution 0.25 m with the proposed edge detection algorithm.

Figures 3.55(c) and (d) or the top left building corner in Figures 3.55(a) and (b). One notable exception is the left gable junction in the upper example, which has been completely missed by the sub-pixel algorithm, but surprisingly been extracted by the pixel-precise algorithm. Note also from the upper left and right corners of the roof in the lower images that the sub-pixel algorithm extracted the correct object boundaries of the roof, while the standard algorithm was confused into extracting two semantically meaningless junctions of background and roof edges.

Figure 3.56(a) displays an aerial image of ground resolution 0.25 m. The results of the edge extraction with the proposed algorithm using Gaussian filter masks with $\sigma = 2$ and hysteresis thresholds of 20 and 35 are shown in Figure 3.56(b). Already from this figure the precise location of the extracted edges is apparent. Figure 3.57 displays a four times enlarged detail of the upper left corner of the image of Figure 3.56 with three different extraction schemes to illustrate the results of the analysis of the behavior of the different edge definitions at junctions. In Figure 3.57(c), the results of the edge detection with pixel resolution using the definition in (3.63) are shown, i.e., the edges were computed by a Gaussian gradient operator, non-maximum suppression in the direction of the gradient, a hysteresis thresholding operation, computing the skeleton of the resulting region, and finally linking the edge points into contours. As was to be expected by the analysis above, this algorithm fails to detect almost all of the salient junctions of the building. If the edges are extracted to sub-pixel resolution by the definition in (3.65), some of the junctions are detected because the more robust estimation of the edge direction allows the edge to come within one pixel of the junction, as is shown in Figure 3.57(b). From this figure, also the accuracy of the extracted edge positions becomes readily observable. Especially the gable and perimeter edges of the roof are almost perfect straight lines. Finally, in Figure 3.57(a), the results of edge extraction with the additional step of completing the junctions is shown. It is apparent that the algorithm was successful in extracting all of the junctions in this part of the


(a) Extracted edges


(b) Edges without added junctions

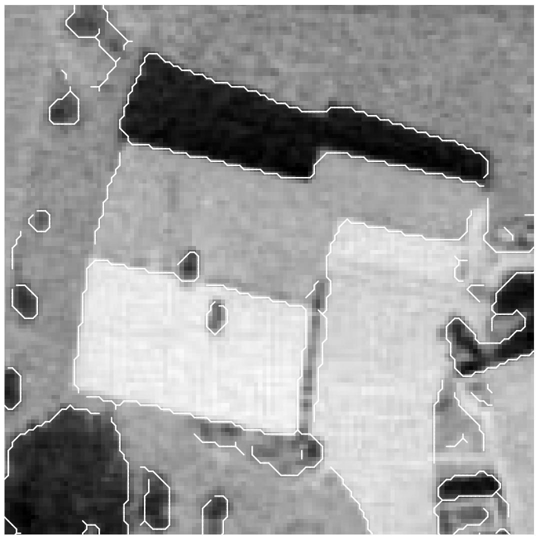
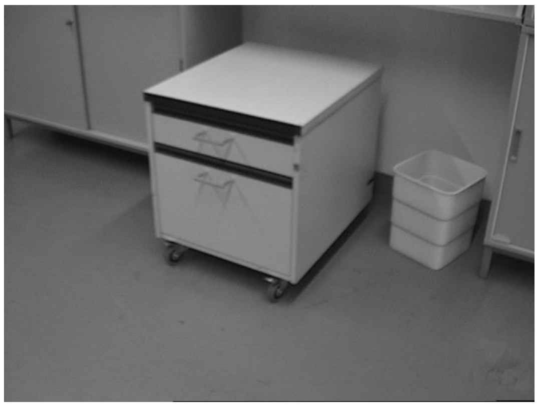

(c) Edges at pixel resolution

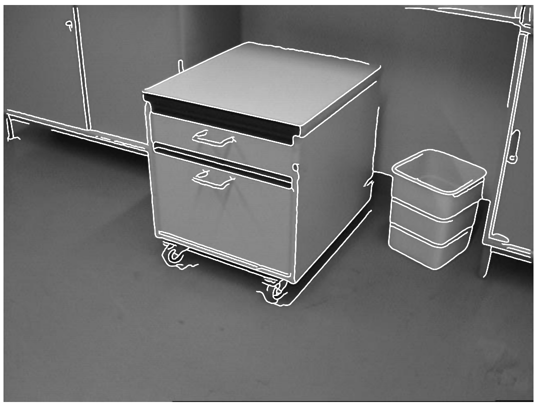
Figure 3.57: Four times enlarged part of the extraction results on the aerial image. (a) Edges extracted with the proposed algorithm. (b) Edges extracted without junction completion. (c) Edges extracted with pixel resolution.

image, and has therefore produced a complete description of the house roof.

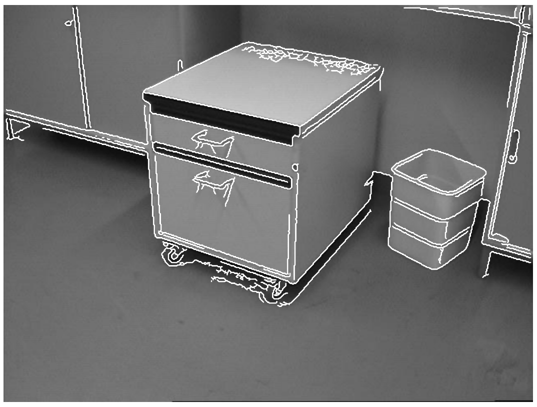
The final example is taken from the domain of autonomous mobile systems [107], where models are matched to image edges, and topological relations play a crucial role [105], and therefore a complete extraction of junctions by the edge detector is highly desirable. Figure 3.58(a) displays an image of a container and a waste paper basket which a mobile robot should be able to recognize from its model base. The image was taken from [102], and is used with kind permission. Figures 3.58(b) and (c) show the result of extracting edges with the pro-



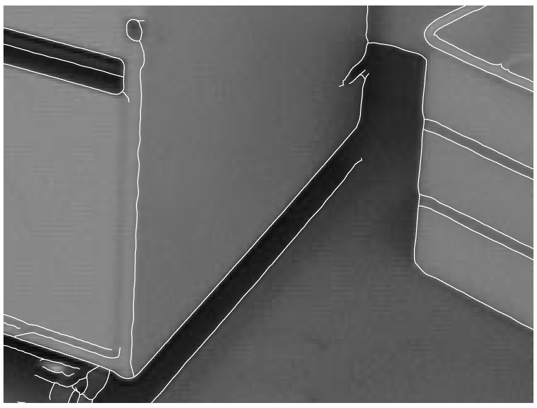
(a) Input image



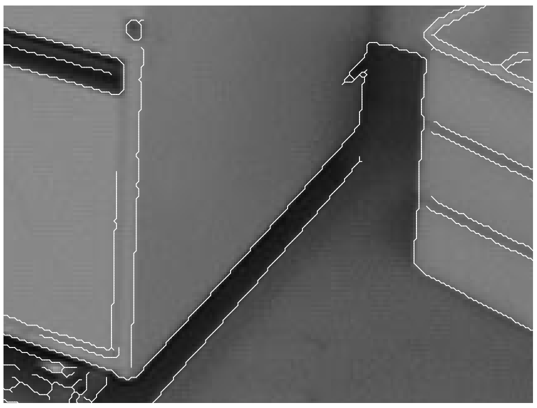
(b) Extracted sub-pixel edges



(c) Extracted pixel-precise edges



(d) Detail of sub-pixel result



(e) Detail of pixel-precise result

Figure 3.58: Edges extracted from an image (a) of a container and a waste paper basket using the sub-pixel algorithm (b) and the pixel-precise algorithm (c). A three times enlarged detail of both results is shown in (d) and (e).

posed sub-pixel algorithm and the pixel-precise algorithm, respectively, both using an isotropic Deriche edge operator [103] with $\alpha = 0.7$ (approximately equal to $\sigma = 2.5$), and thresholds of

8 and 19, as in [102]. The first thing that strikes the eye is that because Canny's original approach to hysteresis thresholding [28, 29] does not take into account the edge direction during the linking of edge points, it gets sidetracked into a maze of edges at the top of the container and in the shadow at the bottom of the container. This is undesirable since these edges have to be weeded out before or during the model matching. This problem does not occur with the sub-pixel algorithm because edge directions are taken into account during the linking stage. A similar effect can be observed at the two handles of the container, where the original linking approach also follows the shadow edges. Again, from Figure 3.58(b) the precise location of the edges is observable. This is made more obvious by Figure 3.58(d), where a three times enlarged detail of Figure 3.58(b) is shown. For comparison purposes, Figure 3.58(e) displays the same part of Figure 3.58(c). It can be seen that the sub-pixel algorithm was successful in extracting all relevant junctions on the waste paper basket and the container, while the pixel-precise algorithm missed all semantically meaningful junctions in this part of the image. In summary, the edges extracted by the proposed sub-pixel algorithm seem to be more suitable for model matching because they contain fewer wrong edges and possess more complete junction information.

# Chapter 4

# Line and Surface Extraction in 3D Images

If the line detection algorithm of the previous chapter is to be extended to 3D data, two different directions are possible. In three dimensions, the Hessian matrix of the gray value function is a $3 \times 3$ matrix, and hence has three eigenvalues and associated eigenvectors. Therefore, "line points" can be required to have a restricted one-dimensional extremum in the direction of the eigenvector corresponding to the eigenvalue of largest magnitude. This definition leads to two-dimensional manifolds of points having a line profile perpendicular to the manifold. These manifolds will henceforth be referred to simply as surfaces, although sometimes the term sheet is also used. Section 4.1 will describe an algorithm to extract such surface points as well as their width. Note that this definition explicitly excludes lines with different polarity. This was done since in the 3D case they never occur as interesting objects because the usual 3D imaging modalities, e.g., CT and MR, normally enhance objects of interest to be brighter than the background. If, on the other hand, the gray value function is required to exhibit a restricted two-dimensional maximum in the direction of the eigenvectors corresponding to the two eigenvectors of largest absolute value and equal sign, one-dimensional manifolds of line points will be extracted, which will simply called lines again, although a term like tube might be more appropriate. Note that again lines of different polarity have been excluded because they never occur as interesting objects. Since the extraction of the width of these lines is a non-trivial task, Section 4.2 will prepare the full 3D scheme by giving an algorithm for the extraction of two-dimensional extrema and their width, which is a full one-dimensional curve of perimeter points around the extremum. Finally, Section 4.3 will describe an algorithm to extract 3D lines and their widths using a modification of the 2D extrema algorithm.

One fundamental problem with the extraction of lines and surfaces in 3D images is that the two definitions necessarily are not disjunct, i.e., some surface points are also line points and vice versa. Figure 4.1 shows an example of what is meant by this statement. Imagine the gray value to be 1 within the object, and 0 outside of the object. Then, on the left side of the figure, the object can clearly be identified as a line, while on the right side one would more likely speak of a surface. However, if this image is convolved with a Gaussian smoothing kernel, it can be seen that all points lying on the $x$-axis will have a restricted 2D maximum, and hence will be labeled as line points. On the other hand, all points on the $x$-axis are, of course, also restricted 1D maxima, and therefore also surface points. Note that this is a problem only if lines and

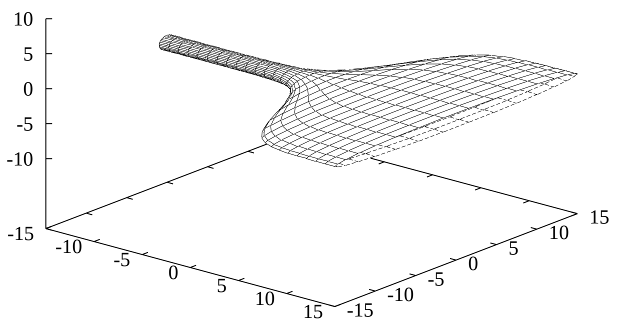


Figure 4.1: A 3D line which turns into a 3D surface. Imagine the gray value to be 1 within the object and 0 outside.

surfaces are extracted simultaneously. In this case, the decision on whether a point is a line or surface point can, for example, be based in the magnitude of the two largest eigenvalues. If their ratio is large a surface point has been detected, while if it is small a line point has been detected.

## 4.1 Surfaces with Line Profiles

Two-dimensional manifolds, i.e., surfaces, in $\mathbb{R}^3$ can be described in various ways, e.g., as zero crossings of a function $f : \mathbb{R}^3 \mapsto \mathbb{R}$ or as functions $f : \mathbb{R}^2 \mapsto \mathbb{R}^3$. Using the first definition, only closed surfaces can be described, which is undesirable since surfaces may terminate at arbitrary positions. Therefore, conceptually one can think of a surface sheet in 3D as a function $s(u, v)$, which exhibits a characteristic line profile in the direction perpendicular to $s(u, v)$. An explicit parameterization of $s(u, v)$ is often difficult to obtain, but fortunately also rarely necessary.

### 4.1.1 Surface Models

Since surfaces $s(u, v)$ are essentially described by their one-dimensional profile in the direction $n(u, v) = s^{\perp}(u, v)$, the analysis of their behavior is again a one-dimensional problem, as in Section 3.1.1. As can be seen from the MR image in Figure 3.31, surfaces in 3D will often exhibit the asymmetrical line profile $f_a(x)$ given by (3.2). Therefore, this profile is adopted as the most relevant profile in the 3D case as well. This means that all the relevant results of the scale-space analysis in Sections 3.1.2 and 3.1.3 hold for 3D surfaces, too. Hence, there is no need to spend time to discuss these matters anew, and one can proceed straight to the extraction of surface points in 3D images.

### 4.1.2 Detection of Surface Points

As mentioned above, surface points are given by restricted extrema in the direction of the eigenvector corresponding to the largest eigenvalue of the Hessian matrix. In order to obtain the relevant partial derivatives, the image is convolved with the corresponding partial derivatives of the Gaussian smoothing kernel:

$$\begin{aligned}
g_{x,\sigma}(x,y,z) &= g'_\sigma(x)g_\sigma(y)g_\sigma(z) && (4.1)\\
g_{y,\sigma}(x,y,z) &= g_\sigma(x)g'_\sigma(y)g_\sigma(z) && (4.2)\\
g_{z,\sigma}(x,y,z) &= g_\sigma(x)g_\sigma(y)g'_\sigma(z) && (4.3)\\
g_{xx,\sigma}(x,y,z) &= g''_\sigma(x)g_\sigma(y)g_\sigma(z) && (4.4)\\
g_{xy,\sigma}(x,y,z) &= g'_\sigma(x)g'_\sigma(y)g_\sigma(z) && (4.5)\\
g_{xz,\sigma}(x,y,z) &= g'_\sigma(x)g_\sigma(y)g'_\sigma(z) && (4.6)\\
g_{yy,\sigma}(x,y,z) &= g_\sigma(x)g''_\sigma(y)g_\sigma(z) && (4.7)\\
g_{yz,\sigma}(x,y,z) &= g_\sigma(x)g'_\sigma(y)g'_\sigma(z) && (4.8)\\
g_{zz,\sigma}(x,y,z) &= g_\sigma(x)g_\sigma(y)g''_\sigma(z)\ . && (4.9)
\end{aligned}$$

As in the 2D case, the responses of convolving the image with the respective kernels are denoted by $r_x, \ldots, r_{zz}$. The eigenvectors of the Hessian matrix

$$H(x,y,z) = \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{xy} & r_{yy} & r_{yz} \\ r_{xz} & r_{yz} & r_{zz} \end{pmatrix} \qquad (4.10)$$

could be computed by Jacobi rotations, as in the 2D case, but it is more efficient to reduce the matrix to tridiagonal form by using one Householder transformation, and then to compute the eigenvalues and eigenvectors by the QL algorithm with implicit shifts [152]. If the eigenvector of largest absolute value is denoted by $(n_x, n_y, n_z)$, and $\|(n_x, n_y, n_z)\|_2 = 1$ is required, the quadratic polynomial

$$z(x,y,z) = r + \begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} + \frac{1}{2}\begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{xy} & r_{yy} & r_{yz} \\ r_{xz} & r_{yz} & r_{zz} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad (4.11)$$

is obtained. As in the 2D case, $tn$ can be plugged into this equation to obtain the quadratic polynomial in the direction perpendicular to the surface:

$$\begin{aligned}
&z(tn_x, tn_y, tn_z) = \\
&= r + \begin{pmatrix} tn_x & tn_y & tn_z \end{pmatrix} \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} + \frac{1}{2}\begin{pmatrix} tn_x & tn_y & tn_z \end{pmatrix} \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{xy} & r_{yy} & r_{yz} \\ r_{xz} & r_{yz} & r_{zz} \end{pmatrix} \begin{pmatrix} tn_x \\ tn_y \\ tn_z \end{pmatrix} \\
&= r + tn_x r_x + tn_y r_y + tn_z r_z + \frac{1}{2}t^2 n_x^2 r_{xx} + t^2 n_x n_y r_{xy} + t^2 n_x n_z r_{xz} + \frac{1}{2} t^2 n_y^2 r_{yy} + \\
&\quad t^2 n_y n_z r_{yz} + \frac{1}{2} t^2 n_z^2 r_{zz}\ . \qquad (4.12)
\end{aligned}$$

The derivative of this expression is

$$\begin{aligned}
&\frac{\partial}{\partial t} z(tn_x, tn_y, tn_z) = \\
&= n_x r_x + n_y r_y + n_z r_z + tn_x^2 r_{xx} + 2tn_x n_y r_{xy} + 2tn_x n_z r_{xz} + tn_y^2 r_{yy} + \\
&\quad 2tn_y n_z r_{yz} + tn_z^2 r_{zz}\ . \qquad (4.13)
\end{aligned}$$

Hence, the maximum or minimum point is given by

$$(p_x, p_y, p_z) = (tn_x, tn_y, tn_z) \ , \tag{4.14}$$

where

$$t = -\frac{n_x r_x + n_y r_y + n_z r_z}{n_x^2 r_{xx} + 2n_x n_y r_{xy} + 2n_x n_z r_{xz} + n_y^2 r_{yy} + 2n_y n_z r_{yz} + n_z^2 r_{zz}} \ . \tag{4.15}$$

As in the 2D case, $(p_x, p_y, p_z) \in [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ is required in order for a point to be declared a surface point. Furthermore, as in the 2D case, the second directional derivative along $(n_x, n_y, n_z)$, i.e., the maximum eigenvalue, can be used to select salient surface points.

Before giving some examples of extracted surface points, it is discussed by what means an appropriate visualization of 3D lines and surfaces can be achieved. Since three-dimensional objects can be very complicated, the most important requirement for the visualization environment is that it allows objects to be manipulated, e.g., rotated or scaled, interactively as fast as possible, so that the human observer can easily analyze the extraction results. Therefore, standard image synthesis algorithms, e.g., ray tracing or radiosity methods [122, 155] cannot be used since they are by far too slow. Another requirement is that the extracted results should be easy to visualize on different operating systems and hardware platforms. For these reasons, the algorithms developed in this chapter write the extraction results in the file format of Open Inventor [189] or VRML 2.0 [77]. Both of these visualization toolkits are based on OpenGL [192], and thus offer adequate visualization options, such as Phong shading of surfaces or clipping of surfaces at arbitrary planes, and excellent visualization speed on hardware-supported OpenGL implementations. Open Inventor is available on many operating systems, while VRML 2.0 is the de-facto standard 3D visualization language of the World Wide Web. Thus, any WWW browser with a VRML 2.0 plug-in is able to display the extraction results.

Figure 4.2 shows the surface points extracted with the proposed approach on a 3D image of a spherical shell with a diameter of eight and a surface thickness of three. The surface points are visualized by displaying their local tangential plane. To effect this, a square with a diagonal length of one has been drawn at each surface point. The diagonals of the squares correspond to the two smallest eigenvectors of the Hessian matrix. This type of visualization aids the intuitive grasping of 3D shape much more than displaying the local coordinate frame, for example. Note how well the extracted points approximate the sphere.

In Figure 4.4 the surface points extracted from the CT image of a human head displayed in Figure 4.3 are shown. They correspond directly to the skull bones. This image is from the University of North Carolina at Chapel Hill (UNC) volumetric test dataset and was scaled by 0.5 in the row and column directions because the slice distance appeared to be twice the pixel distance, and also to reduce the size of the dataset. Here, the tangential planes have diagonals of length two since otherwise they would be too small to be clearly visible. Note that this rendering already gives a very good impression of the extracted surface, although individual surface points have not been linked into surfaces. What is interesting to note are the parts of the skull bone with high curvature, e.g., the chin. Here, some artifacts of the extraction can be observed. There are several surface points extracted perpendicular to the surface direction, which give the appearance of a “beard” on the chin. This effect comes from the fact that in high curvature areas the direction perpendicular to the surface can also have a surface-like appearance. This fact was noted for the case of 2D lines in [72]. Note also that the strange extraction results around the teeth are caused by artifacts in the dataset shown in Figure 4.3(d).

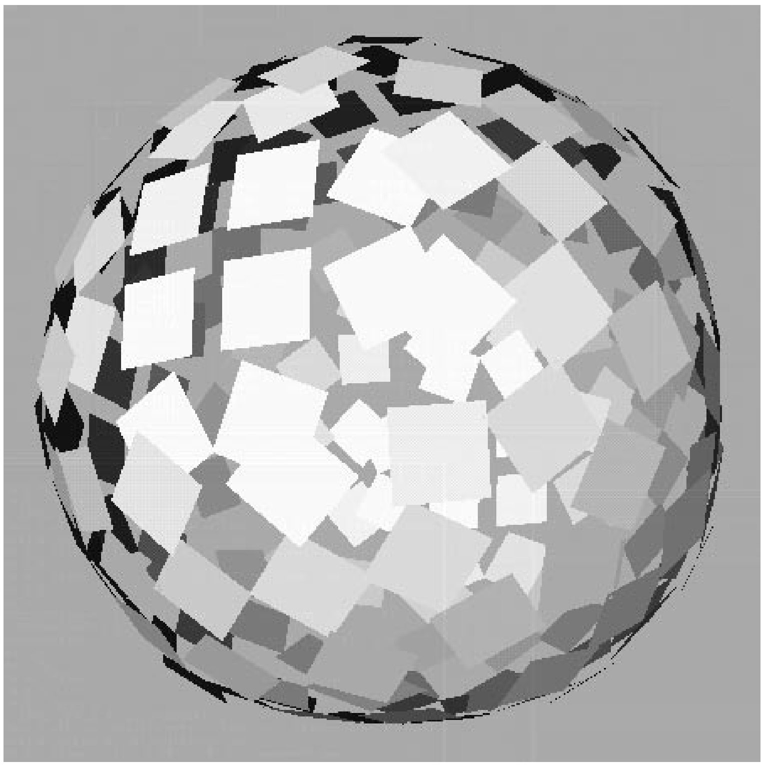
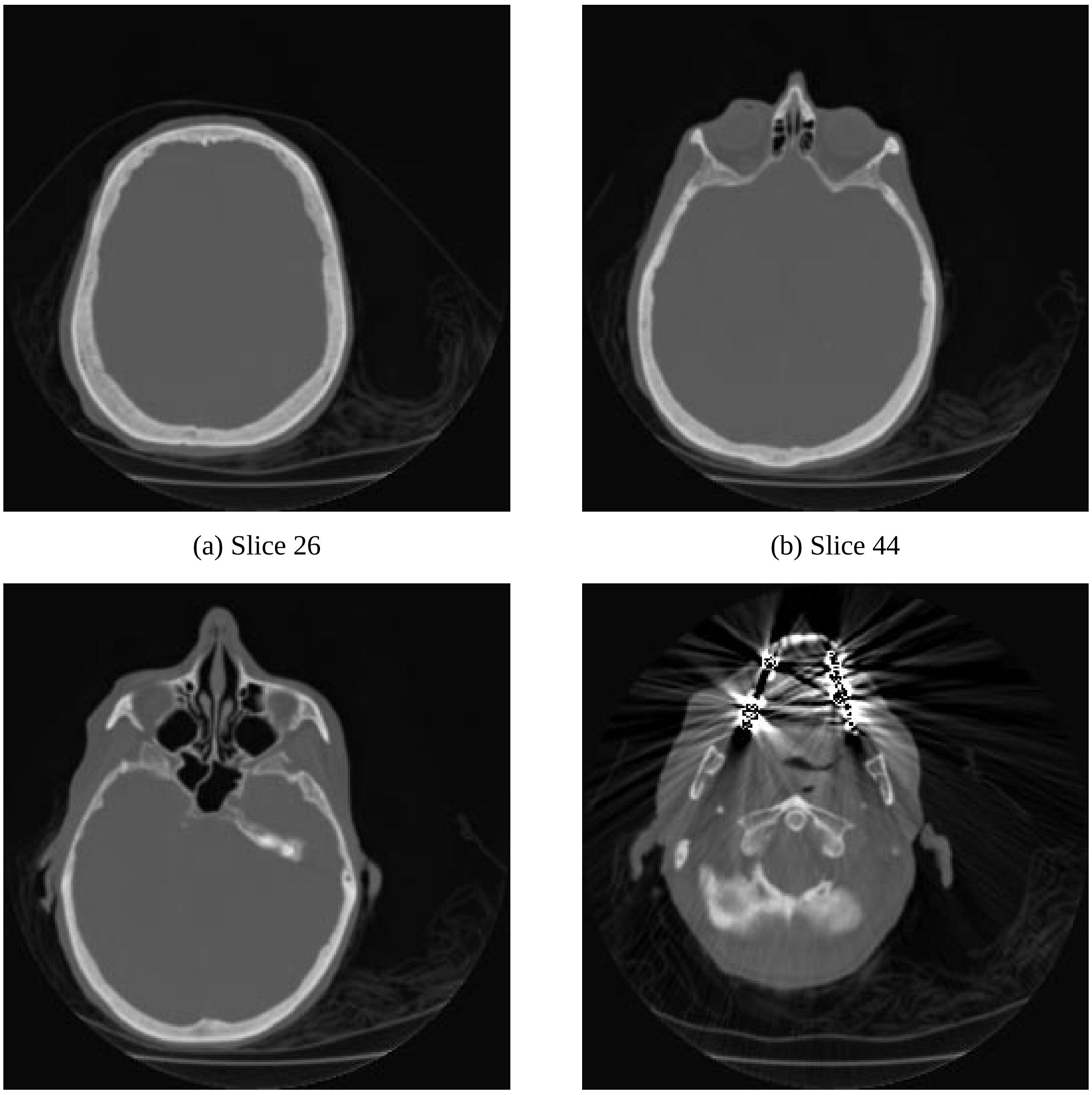


Figure 4.2: Surface points extracted from an image of a spherical shell with diameter 8 and surface thickness 3. Surface points are visualized by their local tangential planes.

Finally, Figure 4.6 shows the extracted surface points of the MR image of the UNC volumetric test dataset displayed in Figure 4.5. Since the slice distance in this dataset appeared to be about 1.5 times the pixel distance, it was scaled by 0.5 in the row and column directions, and by 0.66 in the slice direction. The extracted surface points in this image correspond closely to the skin of the head. Again, the tangential planes are visualized by squares of diameter two. Note that for this image artifacts of surface points extracted perpendicular to the surface direction are not present because there are no areas of very high curvature.

### 4.1.3 Linking of Surface Points

Linking surface points into surfaces is a much more complicated problem than linking line points into lines because each surface point has more than two neighbors. Therefore, the algorithm used in Section 3.1.6 to determine the optimal neighbor to add to the current line cannot be transfered to surface linking. However, the fundamentals of the algorithm, i.e., the hysteresis threshold operation, can be adapted to the case of 3D surfaces. This means that the surface linking algorithm should link all points into a surface that have a response larger than a lower threshold and are connected to at least one point having a response larger than an upper threshold. The term “connected” immediately suggests that the proper data structure to represent a surface internally is an undirected graph in which the individual surface points are the nodes and the edges determine the adjacency of two surface points.


(a) Slice 26 (b) Slice 44

(c) Slice 55 (d) Slice 77

Figure 4.3: Four slices from a CT image of a human head.

Thus, the surface linking algorithm constructs the graph beginning at the point with the maximum response, similarly to the line linking algorithm. To add appropriate neighbors to the current surface graph, all 26 neighboring voxels of the current surface point are examined whether they are compatible with the current point. Two criteria are used to determine compatibility: First, the surface normals of the neighboring point must not form an angle larger than a certain threshold, $30^\circ$ in the current implementation, with the normal of the current surface point. This heuristic removes most, but not all, of the extraneous surface points extracted perpendicularly to the surface mentioned in the previous section. Therefore, another criterion is also checked: the distance of the neighboring point to the plane spanned by the two smaller

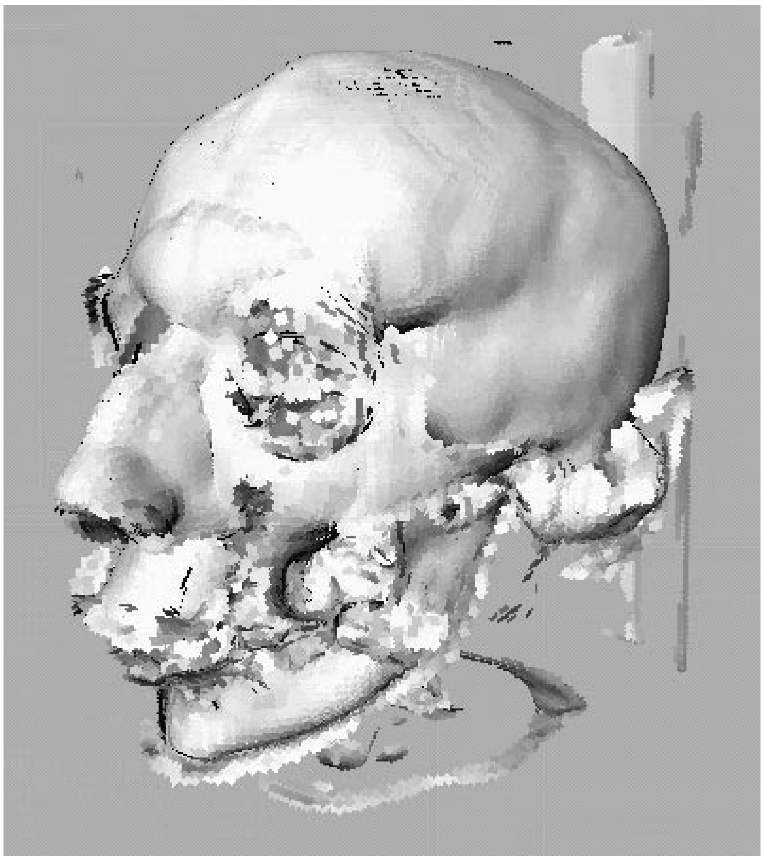
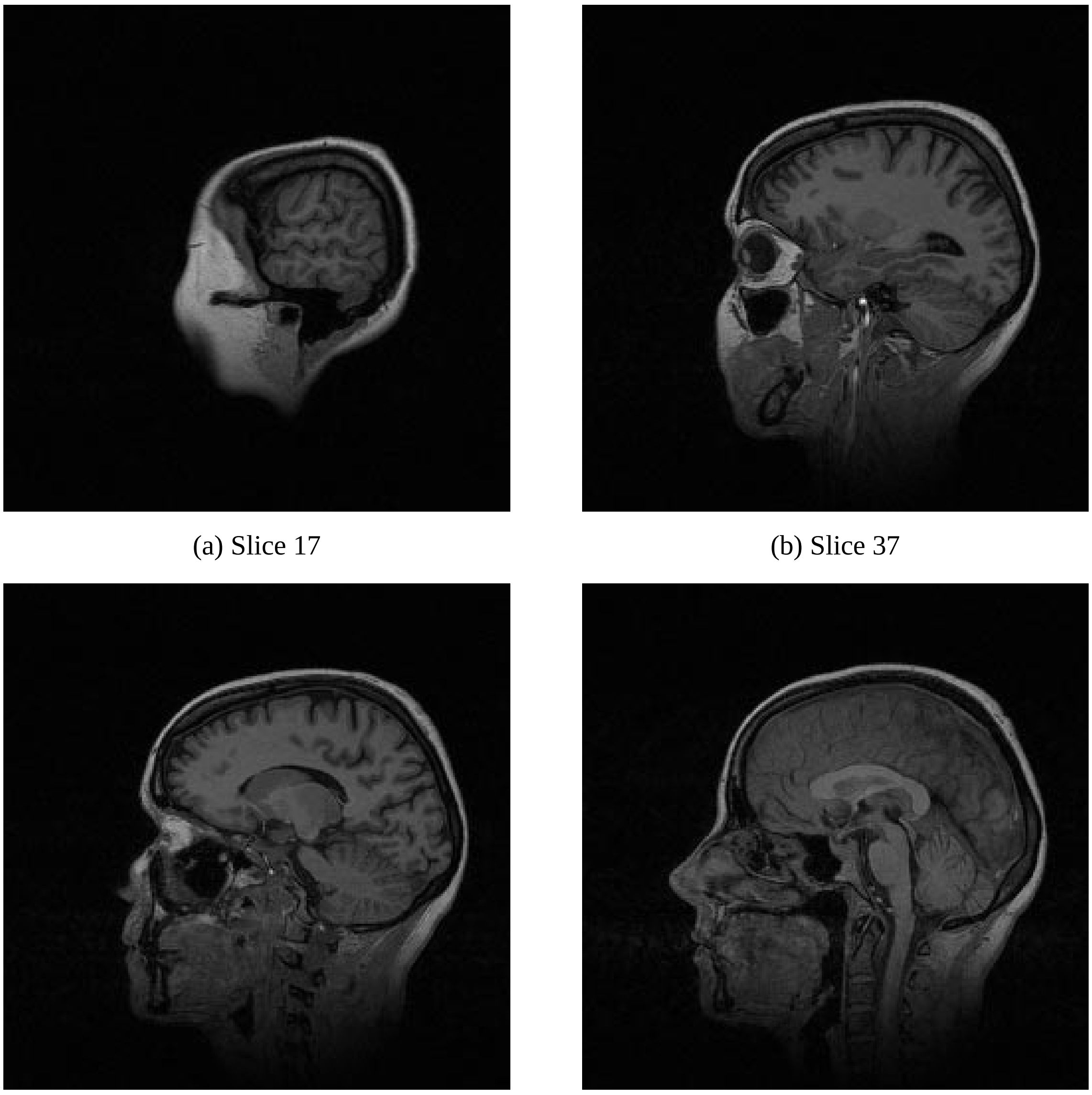


Figure 4.4: Surface points extracted from a CT image of a human head. Surface points are visualized by their local tangential planes.

eigenvectors of the current surface point. If $p$ denotes the current surface point, $n$ the surface normal, with $||n||_2 = 1$, and $q$ the neighboring point, this distance is given by [21]

$$d = |(q - p) \cdot n| \ . \tag{4.16}$$

In the current implementation the neighboring point must be closer than $\sqrt{2}$ to the plane. This second criterion was also used in [132] to link 3D edges. By combining the two criteria, almost all of the extraneous surface points are eliminated. After determining the appropriate neighbors, graph edges connecting the current surface point and the neighbors are created, the neighbors are inserted into a queue of points waiting to be processed if they are not marked as belonging to the current surface, and the current point is marked as processed and belonging to the current surface. This allows junction points to lie on multiple surfaces, and thus ensures that no gaps occur at junction areas. Thus, the surface graph is constructed in a breadth-first manner [160]. Additionally, the algorithm tries to orient the surface normals of the surface so that they all point to one side of the surface. To do so, the surface graph is again traversed breadth-first, and surface normals of neighboring points are oriented with respect to the current point by checking the sign of their dot product. If the dot product is negative the normals are negated. If necessary, the algorithm makes the neighboring coordinate frames right handed. Unfortunately, this algorithm sometimes fails to return a globally consistent orientation, as will be seen below.



(a) Slice 17 (b) Slice 37

(c) Slice 47 (d) Slice 56

Figure 4.5: Four slices from a MR image of a human head.

Figure 4.7 displays the surface graph resulting from linking the surface points of the spherical shell displayed in Figure 4.2. As can be seen, the graph is topologically complete with respect to the 26-neighborhood. This type of visualization is a direct representation of the graph structure, and therefore very hard to interpret for the human observer since one can look through the surface. For this reason, a complete triangulation of the surface is necessary for visualization purposes. This is a very difficult problem, even for the comparatively simple problem of thresholding a surface, where a solution called the marching cubes algorithm has been proposed [117, 183, 184]. Unfortunately, even for this relatively simple type of problem the marching cubes algorithm cannot ensure global topological completeness. Thus, a general solution for

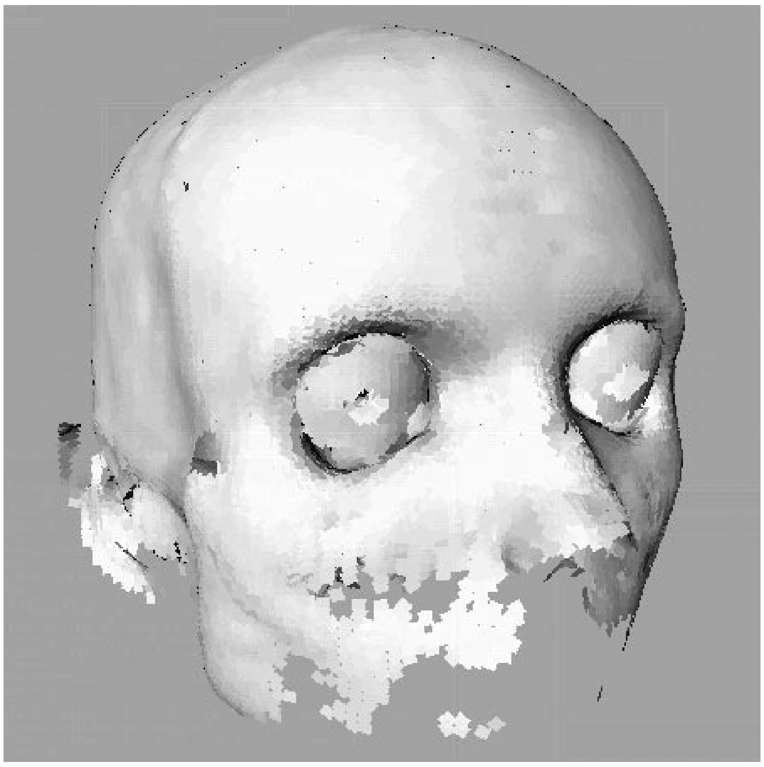


Figure 4.6: Surface points extracted from an MR image of a human head. Surface points are visualized by their local tangential planes.

the problem of producing a topologically complete, non-redundant triangulation of a surface from the surface graph is not attempted here. Since the sole purpose of the triangulation of the surface is its visualization, the triangulation algorithm will be allowed to produce overlapping triangles, essentially trading in programming complexity against visualization speed. A general solution of a non-redundant triangulation is left open for future research.

With the restriction of non-redundancy being dropped, a very simple algorithm suffices to produce a topologically sound triangulation. The basic approach is to look for cycles of length three in the surface graph. In order to ensure that only the minimum number of triangles is produced, i.e., that the same triangle does not occur up to six times with its vertices permuted, the following algorithm is used: The breadth-first search has produced a list of points in the order they were traversed and for each surface point a list of edges, each of which contains a reference to another surface point. Consequently, each surface point can be given an explicit number. Thus, surface points are traversed in order, and triangles are constructed from cycles in the graph involving the current surface point and two neighbors from the edge list which must have higher surface point numbers than the current point. Additionally, the edge leading to the second neighbor must come after the edge to the first neighbor in the edge list of the current point. This ensures that each cycle occurs exactly once in the resulting triangulation. Additionally, since this speeds up rendering, all triangles are oriented counterclockwise with respect to the surface normal. The resulting triangulation for the spherical shell example is shown in Figure 4.8. From Figure 4.7 it is clear that this triangulation contains many redundant overlapping

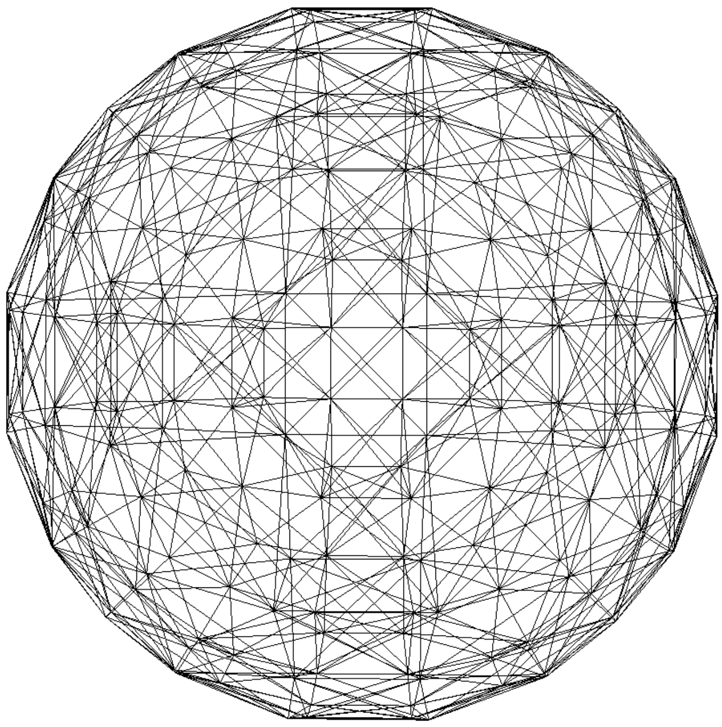


Figure 4.7: Surface points linked into the surface graph for the image of a spherical shell.

triangles. This surface is more or less a worst case since it is highly curved. For moderately curved surfaces the typical configuration is a quadrilateral which contains two diagonals in the graph and hence is triangulated into four triangles, although two would suffice. However, even for these cases no simple criterion has been found that is able to triangulate all of them correctly into two triangles globally. The main problem is how to connect the quadrilaterals correctly to the rest of the surface. Because for typical surfaces the above algorithm generates slightly more than two times the necessary number of triangles, it was deemed sufficient for the purposes of this dissertation.

Figure 4.9 displays the result of linking the surface points extracted from an MR image of a human head shown in Figure 4.6 into surfaces. For this image, $\sigma = 1.2$ and thresholds of 2 and 20 were used. As can be seen, the algorithm correctly extracted the center surface of skin and fatty tissue from this image. In order to see the surface structures extracted inside the skull a cut-away view using a clipping plane running roughly through the left eye of the head is shown in Figure 4.10. Several major anatomical features of the brain, e.g., the corpus callosum in the center of the brain and several of the gyri of the cerebral cortex, can immediately be identified. Thus, the algorithm seems to be able to extract anatomically interesting and relevant structures.

### 4.1.4 Determination of Surface Width

The extraction of surface width is completely analogous to the line width extraction for 2D lines. For each surface point the closest edge points to the right and left of the surface in the

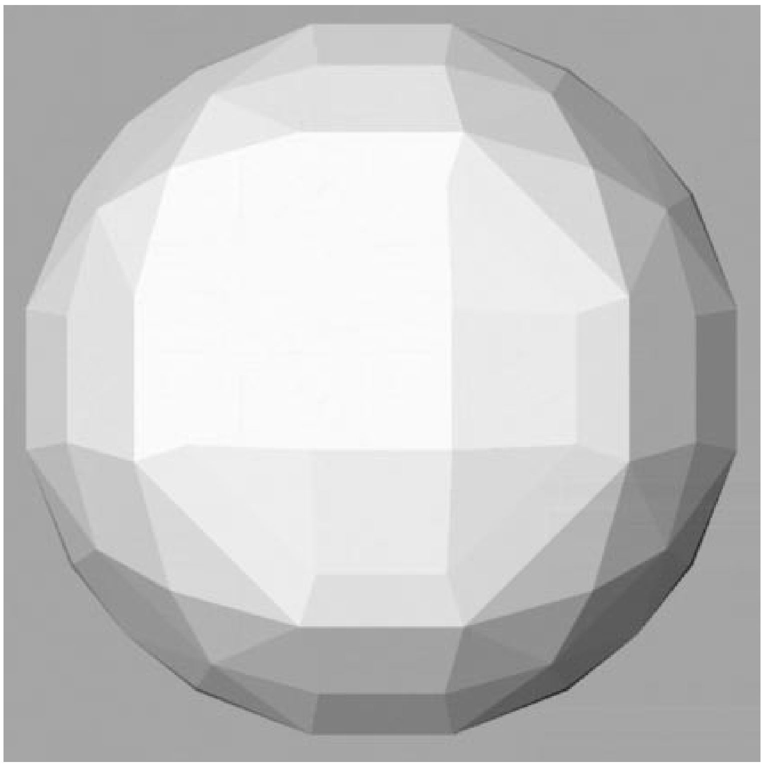


Figure 4.8: Triangulation produced from the surface graph of the image of a spherical shell.

direction perpendicular to the surface are determined. The major difference to the 2D case is that an appropriately modified 3D Bresenham algorithm that returns all voxels intersected by the search line is used. As in the 2D case, the search line length is $2.5\sigma$. To extract the edge points as bright surface points in the gradient image, a local second order polynomial $k_1 r + k_2 c + k_3 s + k_4 r^2 + k_5 rc + k_6 rs + k_7 c^2 + k_8 cs + k_9 s^2$ is calculated by convolving the image with the appropriate 3D facet model masks. Since the facet model masks are separable, they can be obtained by combining the appropriate combination of the following 1D facet model masks:

$$c_0 = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \tag{4.17}$$

$$c_1 = \frac{1}{2}\begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \tag{4.18}$$

$$c_2 = \frac{1}{2}\begin{pmatrix} 1 & -2 & 1 \end{pmatrix} . \tag{4.19}$$

For example, the convolution mask for $k_5$ is given by $c_1 * c_1 * c_0$, or explicitly by

$$k_5 = \frac{1}{12}\left(\begin{array}{ccc|ccc|ccc} 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \end{array}\right) , \tag{4.20}$$

where the slices of the convolution mask are displayed next to each other, with the slice with the smallest $s$-coordinate listed first. As in the 2D case, the Hessian matrix of this polynomial

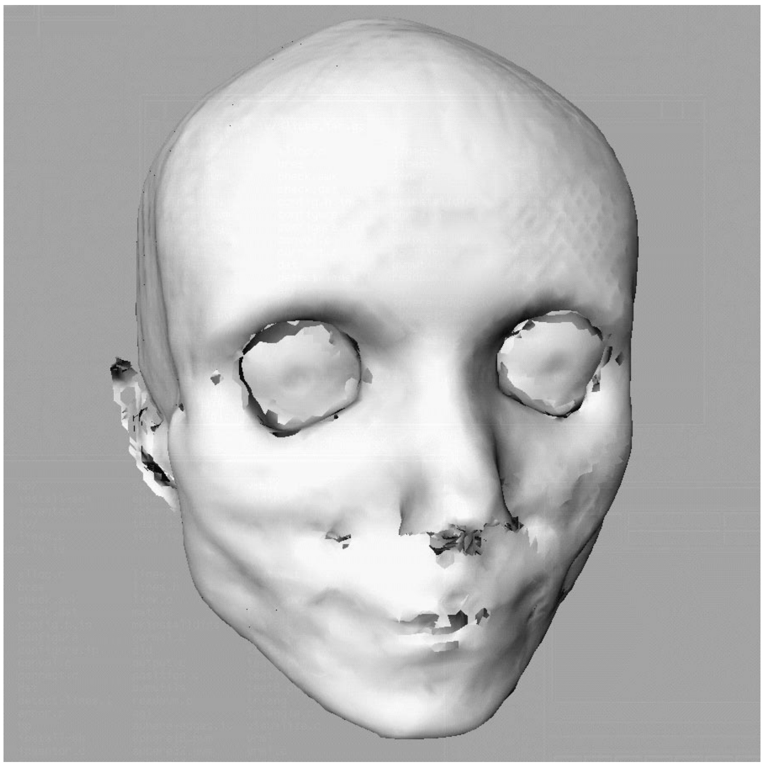


Figure 4.9: Linked surfaces extracted from an MR image of a human head.

is different from the Hessian matrix (4.10) obtained by Gaussian convolution. Therefore, (4.15) must be modified to

$$t = -\frac{n_x r_x + n_y r_y + n_z r_z}{2(n_x^2 r_{xx} + n_x n_y r_{xy} + n_x n_z r_{xz} + n_y^2 r_{yy} + n_y n_z r_{yz} + n_z^2 r_{zz})} \tag{4.21}$$

in order to obtain the correct edge location. As in the 2D case, the edge points thus extracted will in general not lie on the search line. Therefore, in a first try the edge location is projected onto the search line by intersecting the tangential plane of the edge with the search line. If the surface point is denoted by $s$, the direction normal to the surface by $n$, the edge location by $e$, and the two eigenvectors spanning the tangential plane of the edge by $u$ and $v$, the following linear equation system has to be solved, similar to (3.49):

$$s + \lambda n = e + \mu u + \nu v \ . \tag{4.22}$$

Analogously to (3.49), the edge point $p$ on the search line is given by $p = l + \lambda n$. If $p$ is more than 2 voxel units away from $e$, $e$ is projected perpendicularly onto the search line, as in the 2D case. The resulting equation is exactly the same as (3.50), the only difference being that the vectors involved are three-dimensional.

As in the 2D case, the above algorithm sometimes is unable to determine the surface width for some points of the surface. Because each of the surface points has more than two neighbors, the linear interpolation scheme of Section 3.1.9 that makes use of the arc length of the line cannot be extended to the 3D surface case. Instead, the surface points where width information

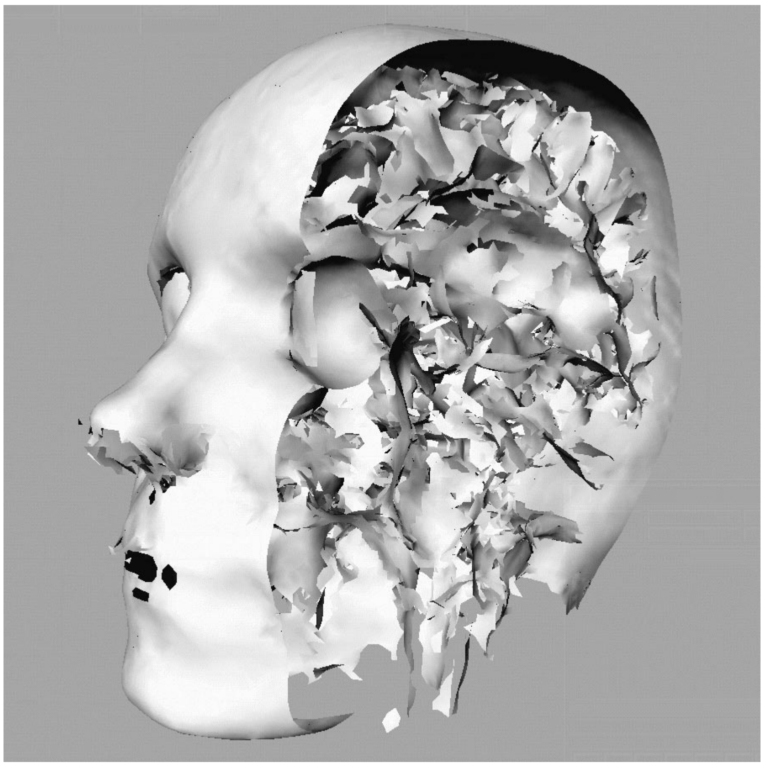


Figure 4.10: Cut-away view of the linked surfaces extracted from an MR image of a human head.

is missing are sorted according to how many neighbors with valid width information they have. The points with the most valid neighbors are processed first. Thus, width information propagates inwards from the borders of areas with missing widths. Assume that a surface point has $k$ neighbors with valid widths $w_i, i = 1, \ldots, k$, and that the distances of the neighbors to the current surface points are $d_i$. Then the width $w$ of the current point is interpolated according to

$$w = \sum_{i=1}^{k} \frac{w_i}{d_i} \Bigg/ \sum_{i=1}^{k} \frac{1}{d_i} \quad . \tag{4.23}$$

Since this scheme propagates widths from the boundary of areas with defined widths no special provision needs to be made for surface points at the border of the surface.

Figure 4.11 displays the result of extracting the width of the surfaces shown in Figure 4.9. The width is visualized as two parallel surfaces of the extracted surface. Therefore, the visible surface in this figure corresponds to the skin of the head. As can be seen, the extracted width makes the face appear very human, and thus is likely to correspond closely to the true surface of the head. However, there are numerous outliers at the ears resulting in a very jagged appearance of this part of the head. Figure 4.12 displays a cut-away view of these results, clearly showing the three parallel surfaces extracted for each surface in the image, i.e., the surface position and the two surfaces constructed from the surface width. Considering the example MR image used in Section 3.1.11, it was to be expected that some of the surfaces in this MR image will be extracted with asymmetries, and indeed some are clearly visible in the skin and fatty tissue at

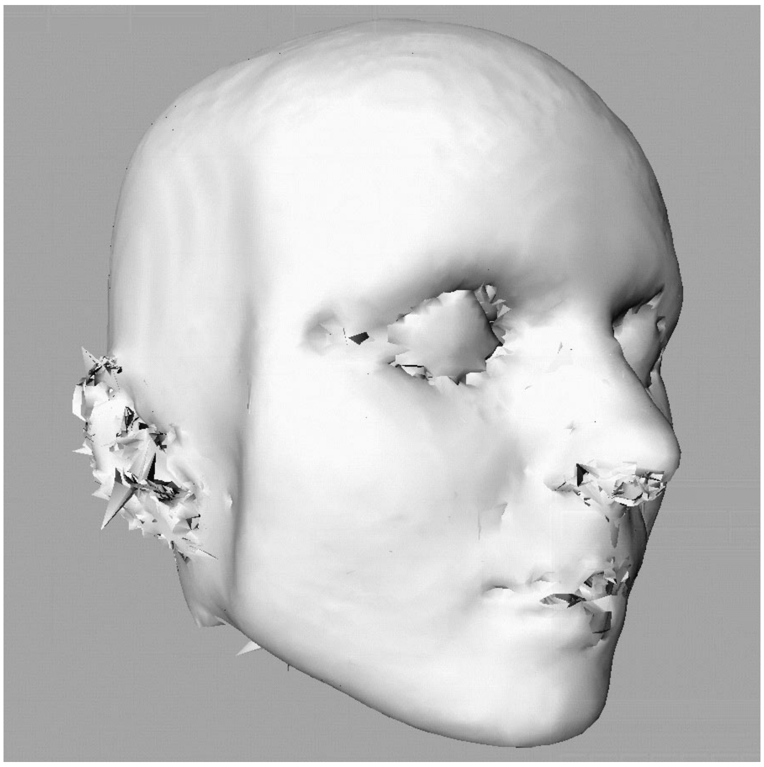


Figure 4.11: Surface widths extracted from an MR image of a human head.

the back of the head in Figure 4.12, manifesting themselves in different surface widths being extracted on each side of the surface. Therefore, bias removal is needed for surfaces as well.

### 4.1.5 Bias Removal for Surfaces

As discussed in Section 3.1.10, the problem of removing the bias from asymmetrical lines is a one-dimensional problem, and therefore can be applied to the problem of removing the bias from asymmetrical surfaces without change. Analogously to the 2D case, the total width $v_\sigma$ and the ratio $r$ of the gradients is extracted by the algorithm in the previous section. These two values are plugged into the inverted bias function $f^{-1}$ derived in Section 3.1.10 to obtain the true scale-normalized width $w_\sigma$ and the true asymmetry $a$. The true surface width is then given by $\sigma w_\sigma$. Exactly the same pre-computed table as in Section 3.1.10 is used to store the inverted bias function. Again it can happen that the extracted scale-normalized surface width $v_\sigma$ is smaller than 2. For these points, the interpolation algorithm given above is used to obtain meaningful surface widths and asymmetries. Finally, the true contrast $h$ of the surface is calculated using (3.52).

The results of applying the width correction to the surfaces extracted in Figure 4.11 are shown in Figure 4.13. Probably the most notable change induced by the width correction, which is visible in this part of the results, is that the ear is now clearly delineated, and that the facial muscles at the corner of the mouth are more pronounced. In Figure 4.14 the cut-away view of the head with bias removal is shown. Here, there are several noteworthy changes: First

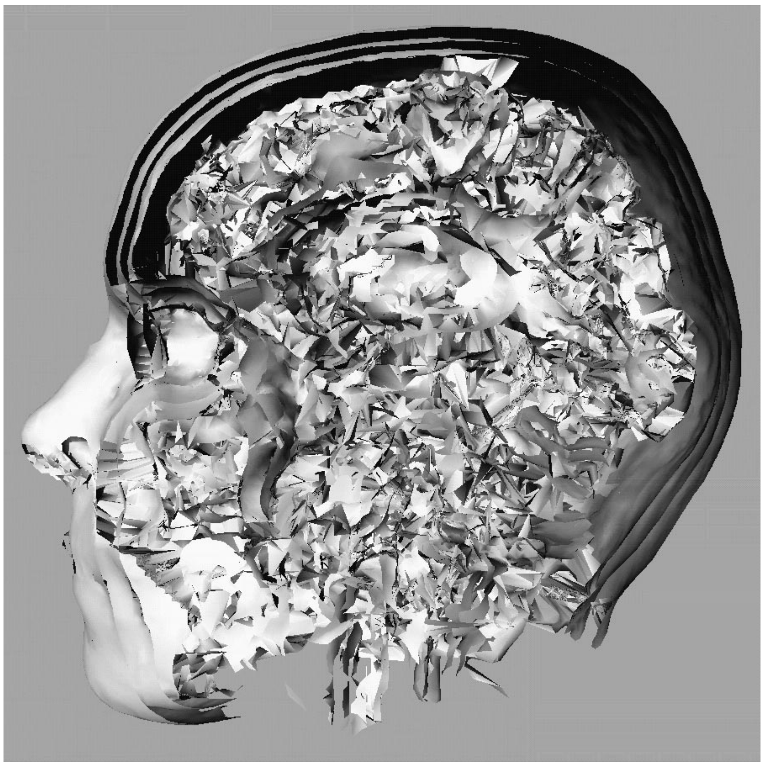


Figure 4.12: Cut-away view of the surface widths extracted from an MR image of a human head.

of all, clearly the skin and fatty tissue of the head now have symmetrical widths, indicating the width correction works successfully. Furthermore, the width of the gyri in the cerebral cortex can now be clearly seen. Additionally, the corpus callosum visible in Figure 4.5(c) and (d) is well delineated in the center of the image. Finally, the central part of the cerebellum is visible as a longer surface running down and to the right from the center of the image (see also Figure 4.5(b)). Thus, it is apparent that the algorithm is able to successfully extract features of important anatomical structures from MR images.

The visualization used in this example is in general very good at displaying the extraction results. However, sometimes it is relatively hard to interpret, especially if there are several surfaces close to each other. In these cases, a visualization of the surface width as a distinct color is better suited for human interpretation. Such a scheme has been incorporated in the surface extraction algorithm. Surface widths are visualized by using only saturated colors in order to allow shading of the surfaces for 3D visualization. However, this results in almost indiscernible gray values in a black and white print of this type of visualization. Therefore, no example of this type is given here.

### 4.1.6 Examples

This section is concluded by completing the discussion of the surface extraction results on the CT image of a human head given in Figure 4.3.

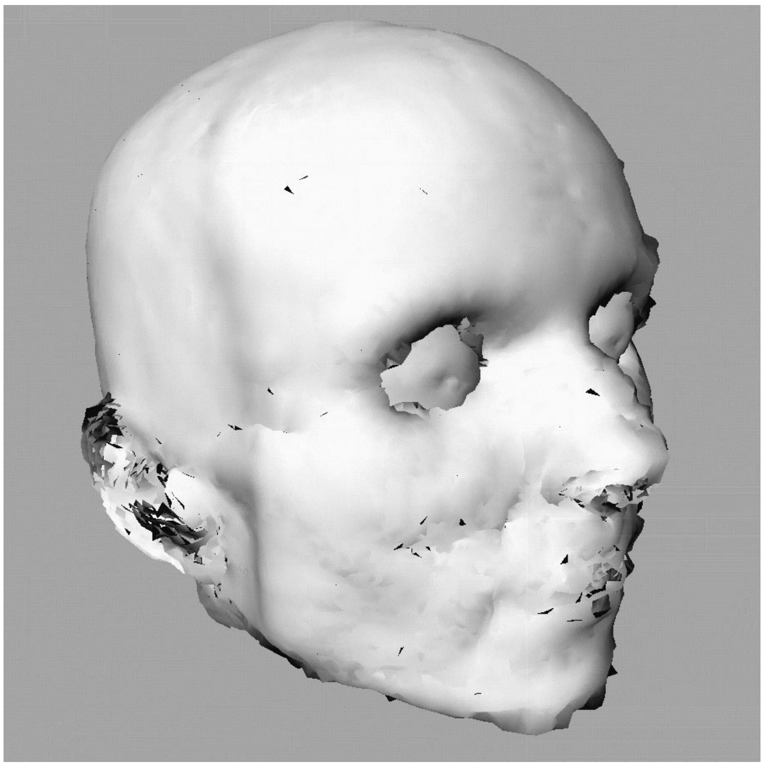


Figure 4.13: Corrected surface widths extracted from an MR image of a human head.

Figure 4.15 shows the surfaces extracted with $\sigma = 1.2$ and thresholds of 5 and 25 from the left side of the head. The surfaces give a precise delineation of the centers of the skull bones. All the important skull bones, including the vertebrae, have been extracted. Additionally, surface-like cartilage tissue has also been extracted, e.g., the tip of the nose or the ears. This is in distinct contrast to threshold-based segmentation approaches, where typically cartilage is too dark to be extracted, although it may have a surface-like structure. Most of the extraneous surface points visible in Figure 4.4 have been eliminated, e.g., the points above the eyes or at the ridge of the nose. However, the "beard" at the chin is still present because the surface there has such a high curvature that the points extracted perpendicular to the surface neither form a big enough angle nor are sufficiently far from the surface to be eliminated. Another example of this effect occurs at the tip of the teeth. Again, it should be pointed out that the strange segmentation result in the mouth are caused by the artifacts of the CT visible in Figure 4.3(d).

The results of extracting the corrected surface thickness for this example are displayed in Figure 4.16. As was to be expected, the outer surfaces of the bones are displayed in this image. Probably the most interesting effect occurring in this example is that the algorithm was unable to orient the surface normals globally. This can be seen from the line starting at the indentation at the temple of the skull and running diagonally towards the back of the head. The indentation violates the implicit assumption that the surface is relatively smooth used in the orientation algorithm, and consequently the algorithm has selected normals of opposite orientation for one surface point in the vicinity of the indentation. This is propagated across the skull by the breadth-first orientation algorithm. Unfortunately, configurations like these cannot be resolved

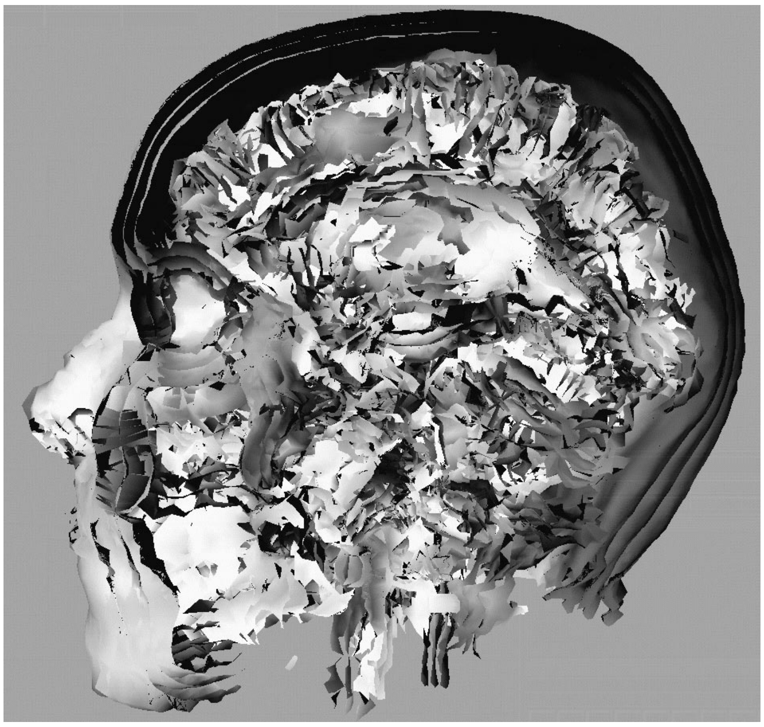


Figure 4.14: Cut-away view of the corrected surface widths extracted from an MR image of a human head.

locally by the simple orientation heuristic. For such hard cases, a global relaxation algorithm to orient the normals might yield better results.

Finally, Figure 4.17 shows a cut-away view of the corrected surface widths, with the clipping plane roughly at the center of the skull. This view shows that all the major bones at the base of the skull have been extracted in addition to the outer skull bones already visible in Figure 4.16. Furthermore, structures like the hard palate and the septal cartilage have also been precisely delineated. Additionally, several important cavities like the frontal sinus and the sella turcica, the bone containing the glandula pituitaria, are clearly visible. The area at the top of the head shows the flipping of the normals in a more vivid manner. Apart from this minor problem, the algorithm was very successful in extracting all the relevant surfaces from the example CT image.

## 4.2 Extraction of 2D Extrema and Their Perimeter

Before the full algorithm to extract lines and their width in 3D will be described, it is useful to consider the similar problem of extracting extrema and their "width" in 2D images since this will help to make the presentation simpler and clearer. Furthermore, the resulting algorithm is useful in itself because it has applications in some areas of computer vision, e.g., camera calibration [6, 104, 106] (extraction of calibration marks), cytology (extraction of cell nuclei), and dermatoscopy (extraction of brown globules).

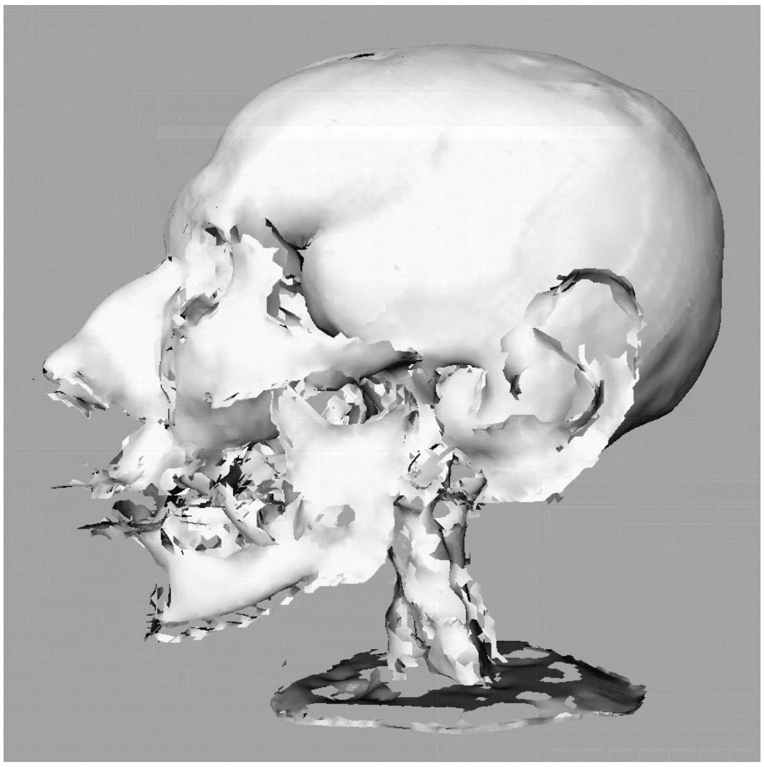


Figure 4.15: Linked surfaces extracted from a CT image of a human head.

### 4.2.1 Models for Extrema

Models for extrema in 2D images are not as easy to define as models for line profiles for 2D and 3D images. While the concept of the extremum itself is of course obvious, the notion of the width of the extremum needs some reflection. One approach to define the width immediately coming to mind would be to define extrema to have a line profile along the two principal directions of the extremum, i.e., along the directions of the eigenvectors of the Hessian matrix, provided the two eigenvalues have the same sign. This view of extrema essentially assumes that they are surrounded by edges which approximately form an ellipse. The widths along the two axes then correspond to the two major axes of the ellipse. This definition of the width of extrema may be sufficient for applications where it is known beforehand that only elliptical shapes can occur in the image, e.g., camera calibration. Furthermore, since it is well known that any shape evolves to an ovoid shape under Gaussian smoothing [92], this definition may even be desirable in some applications.

On the other hand, there are many applications where the objects characterized by extrema are not even close to being ellipses, e.g., cytological applications. Therefore, a more general model of extrema is needed. One such model is, for example, to assume that extrema are characterized by having a constant gray value within a certain region, which is either brighter or darker than the background. Otherwise, no assumption is made about the gray value distribution of the background. Furthermore, the region of constant gray values of the extremum is assumed to be star shaped around the extremum point. A region $R$ is said to be star-shaped around a point

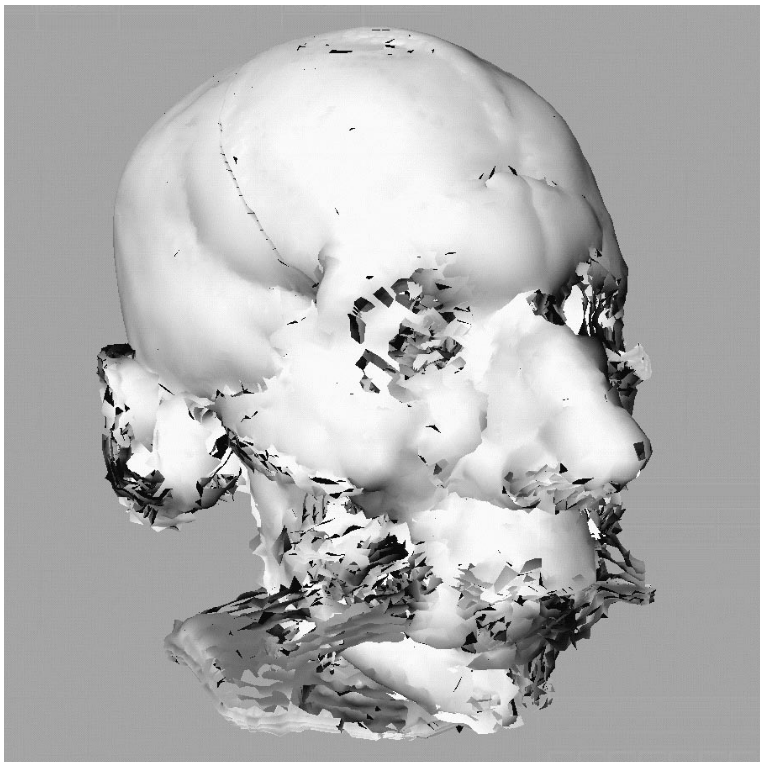


Figure 4.16: Surfaces and corrected surface widths extracted from a CT image of a human head.

$p$ if all straight lines from $p$ to a point in $R$ lie completely within $R$. While this assumption rules out some possible shapes of extrema, it still includes many useful shape types, which may even be concave. The advantage of this definition is that there is a well defined, non-self-intersecting perimeter curve around each extremum, which can be parameterized according to the angle to an arbitrarily chosen axis, e.g., the axis given by the eigenvector corresponding to the largest eigenvalue. Therefore, the width of an extremum, i.e., its perimeter curve, is given by

$$p(\alpha) = e + a_1 r(\alpha) \cos\alpha + a_2 r(\alpha) \sin\alpha \ \ , \tag{4.24}$$

where $e$ is the position of the extremum, $a_1$ and $a_2$ are the axes, i.e., eigenvectors, corresponding to the two eigenvectors, and $r(\alpha)$ is the width of the extremum in the direction $\alpha$.

## 4.2.2 Detection of Extrema

Extracting extrema in the continuous two-dimensional space is, of course, a simple matter. They are given by the points where the gradient of the image convolved with a Gaussian $\nabla r = (r_x, r_y)$ vanishes. As in the one-dimensional case, it is useful to select only salient extrema. These can be determined by the two eigenvectors of the Hessian matrix. Bright extrema will have $\lambda_1 \ll 0$ and $\lambda_2 \ll 0$, while dark extrema will have $\lambda_1 \gg 0$ and $\lambda_2 \gg 0$. Points where $\lambda_1$ and $\lambda_2$ have opposite sign are, of course, saddle points, and are of no interest here.

Since the model for extrema given in the previous section assumes that the gray value of the extremum within its perimeter is constant, the second derivatives of the extremum can be

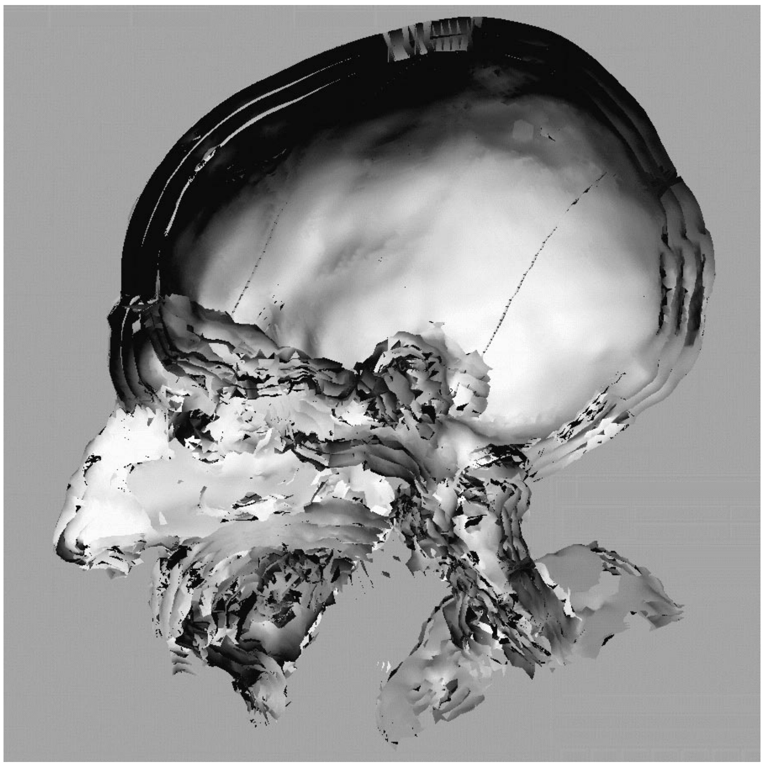


Figure 4.17: Cut-away view of the corrected surface widths extracted from a CT image of a human head.

expected to exhibit a similar behavior as the bar-shaped profiles of Section 3.1.2, i.e., the two eigenvalues will not take on their maximum negative values unless $\sigma$ is large enough. Ideally, one would like to obtain a similar restriction on $\sigma$ as in (3.15) for extrema with arbitrarily shaped perimeters. However, this is impossible since only very simple shapes can be analyzed analytically because the convolutions with the derivatives of Gaussians yield closed-form solutions only for them. Therefore, a simple rectangular shape is taken as the prototypical extremum and its behavior is analyzed. The results thus obtained will still be valid within limits for most shapes since the rectangle is convex, but has a rather low compactness, and therefore the results will at least hold for all ovoid shapes. A model of a rectangle of length $l$ and width $w$ is given by

$$f_q(x) = \begin{cases} 1, & |x| \le l \wedge |y| \le w \\ 0, & \text{otherwise} \end{cases} . \tag{4.25}$$

The scale-space response of this rectangle when convolved with a Gaussian obviously is given by

$$r_q(x, y, \sigma, l, w) = \Big(\phi_\sigma(x + l) - \phi_\sigma(x - l)\Big)\Big(\phi_\sigma(y + w) - \phi_\sigma(y - w)\Big) \tag{4.26}$$

and its partial derivatives. What is immediately obvious is the fact that the gradient $\nabla r$ vanishes at $(0, 0)$ for all $\sigma > 0$. This means that extrema can be detected for all $\sigma$ in the correct position. However, as in the 2D case, for low $\sigma$ the selection of salient extrema will be difficult. To see how large $\sigma$ must be for the two eigenvalues to be local minima, first note that $r_{xy}(0, 0, \sigma, l, w) = 0$ for all $\sigma$. Therefore, the problem reduces to two one-dimensional problems, of which the solution is already known. Hence, in order to ensure significant extrema can be selected according to their second derivatives, the image has to be convolved with a $\sigma$ obeying the following restriction:

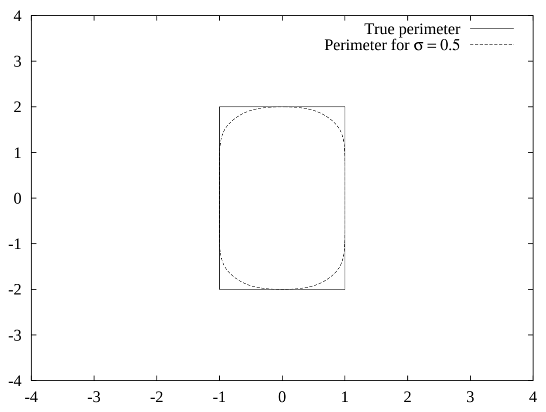
$$\sigma \geq \frac{\max(l, w)}{\sqrt{3}} \quad . \tag{4.27}$$

From this analysis it is clear that the $\sigma$ which should be chosen for arbitrarily shaped extrema is roughly determined by the diameter of the extremum.

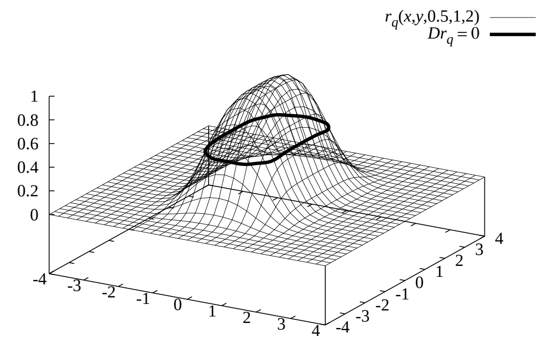
The next question is how the edges of the extremum behave in scale-space. As was discussed in Sections 3.1.2 and 3.1.9, edges are given by the maxima of the gradient in the gradient direction. From Section 3.3.1 it follows that the behavior of the edges in scale-space can be analyzed by either using the defining equations $Dr_q = 0$ or $Sr_q = 0$, which are equivalent in non-junction areas. For simplicity, $Dr_q = 0$ will be used. Figure 4.18 shows how the edges of a rectangle with $l = 1$ and $w = 2$ behave for $\sigma = 0.5$, $1$, and $2$. As can be seen, for very small $\sigma$ the extracted perimeter curve is contained entirely in the true perimeter rectangle. For moderate $\sigma$, the perimeter is partly inside and partly outside the true perimeter, while for large $\sigma$ it lies entirely outside of the true perimeter. Furthermore, it has evolved into an elliptical shape that has its larger axis aligned with the smaller axis of the rectangle. Therefore, contrary to the one-dimensional behavior, the width of an extremum may be extracted smaller than the true width. This is in distinct contrast to the behavior of the edges obtained by the Laplacian condition $\Delta r_q = 0$. For convex objects it was shown in [161] that the extracted edges will always lie outside of the convex object. For completeness sake, this is illustrated in Figure 4.19, where the perimeters extracted by the Laplacian are displayed for the same values of $\sigma$ as above. As can be seen, for small $\sigma$ the perimeter develops the familiar bulges at the corners of the rectangle. These are still visible at the longer sides of the rectangle for moderate $\sigma$. For large $\sigma$, again the perimeter has evolved into a large elliptical shape. When the results of the two edge definitions are compared, three things immediately become clear. One is the fact that using the Laplacian the convex rectangle evolves temporarily into a concave shape. Second, when looking at the edge positions mapped onto the smoothed rectangles, the edges extracted by the gradient maximum definition correspond more closely to where one intuitively would expect them to be since they all lie roughly on the same gray level, while when using the Laplacian definition they can lie on significantly lower gray values, especially for small $\sigma$. Finally, and most importantly, the perimeters extracted by the Laplacian exhibit a much larger bias for all $\sigma$ than the perimeters extracted by the gradient maximum definition. These three points give another reason to use the gradient maximum edge definition to extract the perimeter curves instead of the zero crossing of the Laplacian definition.

The discussion so far has assumed continuous images. For discrete images the extraction of extrema is not quite as simple. While, in principle, one could just smooth the image and extract extrema by comparing a pixel to its eight neighbors, this would give neither sub-pixel resolution nor the eigenvalues of the Hessian matrix. Recall from Section 3.1.5 that if the image is convolved with the derivatives of Gaussians of up to second order this results in an explicit Taylor polynomial

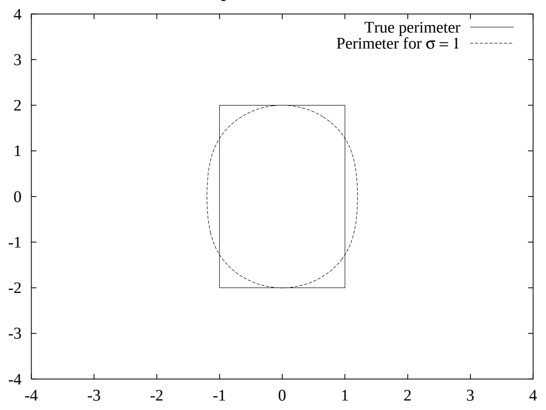
$$z(x, y) = r + \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} r_x \\ r_y \end{pmatrix} + \frac{1}{2} \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} r_{xx} & r_{xy} \\ r_{xy} & r_{yy} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.28}$$



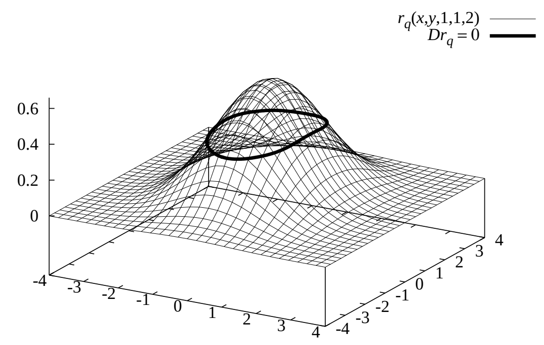
(a) $Dr_q = 0$ ($\sigma = 0.5$)



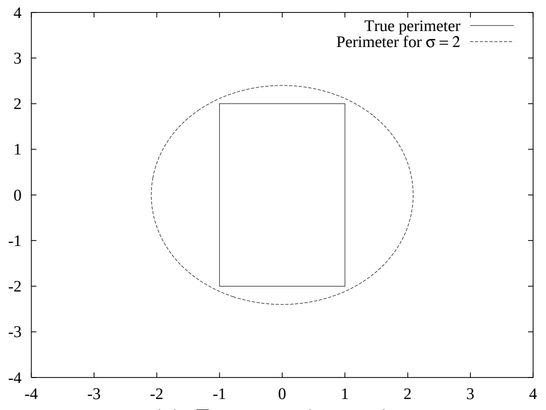
(b) $Dr_q = 0$ on response ($\sigma = 0.5$)



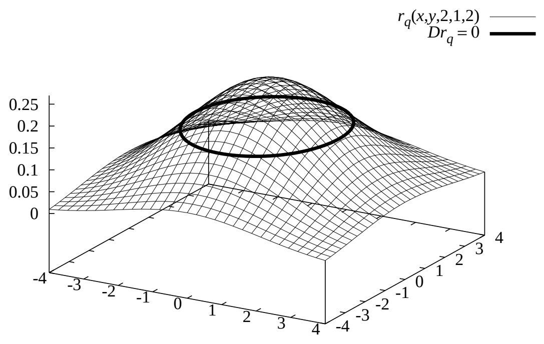
(c) $Dr_q = 0$ ($\sigma = 1$)



(d) $Dr_q = 0$ on response ($\sigma = 1$)



(e) $Dr_q = 0$ ($\sigma = 2$)



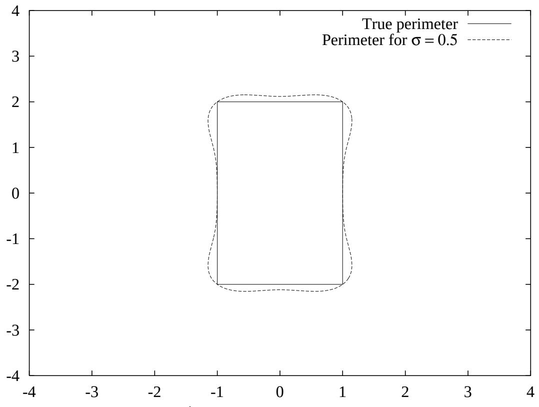
(f) $Dr_q = 0$ on response ($\sigma = 2$)

Figure 4.18: Extracted perimeter of an extremum with $l = 1$ and $w = 2$ using the maxima of the gradient in the gradient direction for a rectangular extremum for $\sigma = 0.5$ (a), (b), $\sigma = 1$ (c), (d), and $\sigma = 2$ (e), (f).

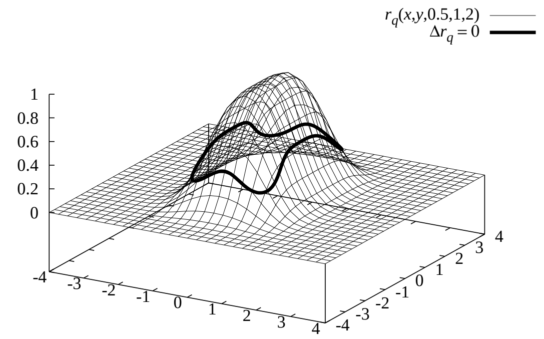
in each image point. To extract extrema, the gradient of this polynomial is calculated as

$$\nabla z(x,y) = \begin{pmatrix} r_x \\ r_y \end{pmatrix} + \begin{pmatrix} r_{xx} & r_{xy} \\ r_{xy} & r_{yy} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad . \tag{4.29}$$

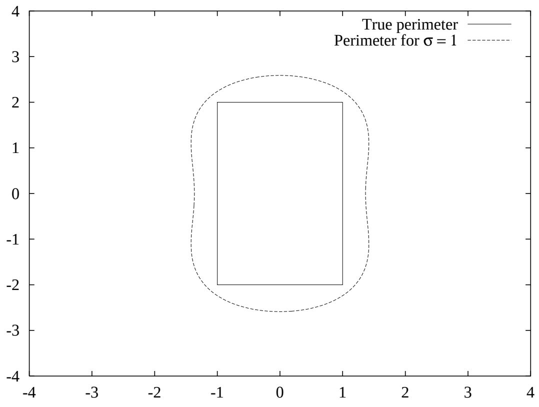
The locations of the extrema are given by the roots of this equation. This results in the following



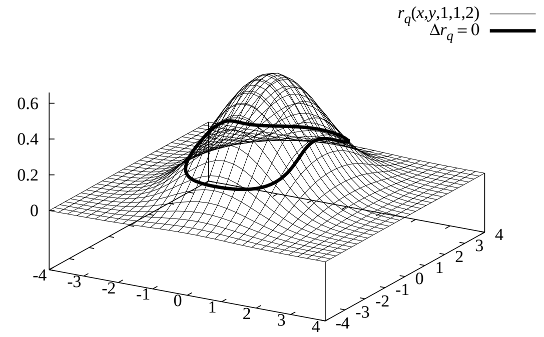
(a) $\Delta r_q = 0$ ($\sigma = 0.5$)



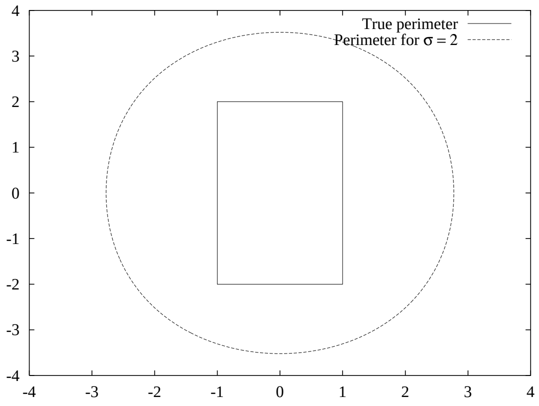
(b) $\Delta r_q = 0$ on response ($\sigma = 0.5$)



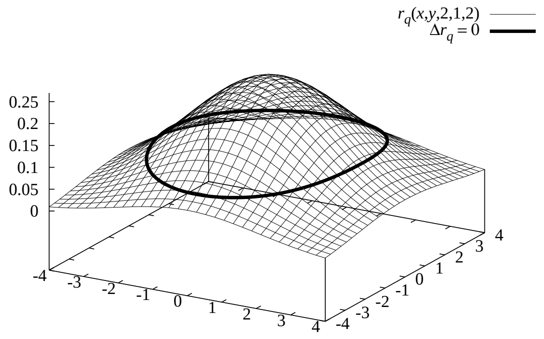
(c) $\Delta r_q = 0$ ($\sigma = 1$)



(d) $\Delta r_q = 0$ on response ($\sigma = 1$)



(e) $\Delta r_q = 0$ ($\sigma = 2$)



(f) $\Delta r_q = 0$ on response ($\sigma = 2$)

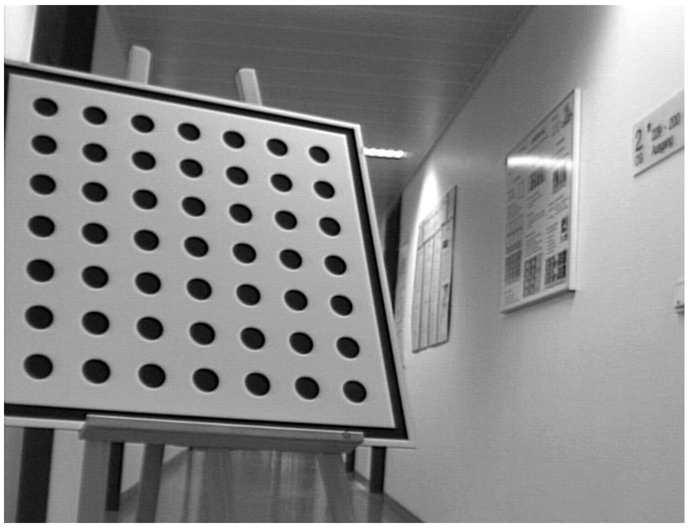
Figure 4.19: Extracted perimeter of an extremum with $l = 1$ and $w = 2$ using the zero crossings of the Laplacian for a rectangular extremum for $\sigma = 0.5$ (a), (b), $\sigma = 1$ (c), (d), and $\sigma = 2$ (e), (f).

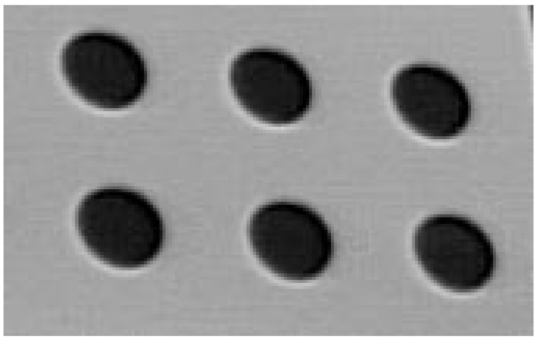
linear equation:

$$\begin{pmatrix} r_{xx} & r_{xy} \\ r_{xy} & r_{yy} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} -r_x \\ -r_y \end{pmatrix} . \tag{4.30}$$

As customary, $(p_x, p_y) \in [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ is required for an extremum to be accepted. Furthermore, as for 1D lines, the eigenvalues and eigenvectors of the Hessian matrix can be calculated by one Jacobi rotation. Salient extrema can the be selected if both of the eigenvalues have the same sign and are larger than a certain threshold.

Figure 4.20 displays an example of the extraction of extrema from an image of a calibration table. The entire input image is shown in Figure 4.20(a), while the six lower rightmost calibration targets are displayed in Figure 4.20(b). In this case, minima were extracted with $\sigma = 6$, which was estimated by the size of the ellipses in the image, and a threshold on the second directional derivatives of 0.4. The result is shown in Figure 4.20(c). It can be seen that despite the fact that a relatively large $\sigma$ was used for the extraction, the extracted minima correspond closely to the centers of the ellipses of the calibration targets.


(a) Calibration table


(b) Input image

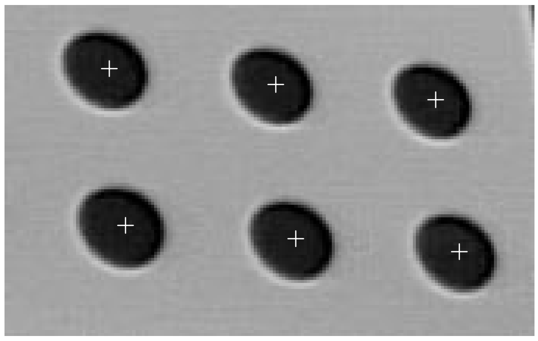
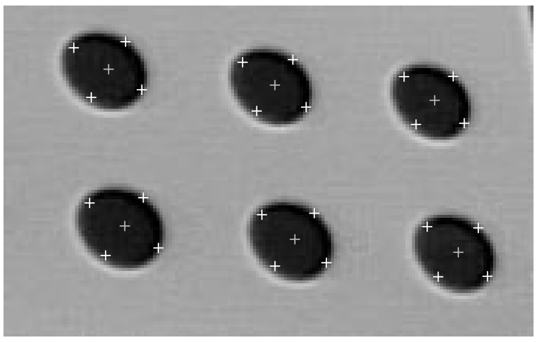

(c) Extracted minima

Figure 4.20: Extracted minima (c) on the lower right part (b) of the calibration table in (a).

### 4.2.3 Extraction of Extrema Perimeters

In order to extract the width of an extremum it is useful to first extract the widths along the two major axes. They are the “preferred” directions of the extremum in the sense that if the extremum is roughly elliptically shaped, then the principal directions will correspond closely to the major axes of the ellipse. Furthermore, since the gray value variation is largest along the direction corresponding to the larger eigenvalue the strongest, and hence best extractable, edges can be expected in this direction. The extraction of the extremum width along the axes can, of course, be done in a straightforward manner using the algorithm for line width extraction of Section 3.1.9. The result of this procedure is a set of four edge points on the two major axes, an example of which is displayed in Figure 4.21(a). As was discussed in Section 4.2.1, this view of extrema essentially assumes that they are approximately elliptical in shape. Therefore, the four edge points can be used to draw a quarter ellipse between each adjacent pair of edge points to obtain an approximation of the perimeter of the extremum. This was done in Figure 4.21(b), where the ellipse segments were approximated by Bezier splines in PostScript [1]. Note that this yields a quite good approximation of the shape of the calibration targets, and may be sufficient for some applications, but certainly not camera calibration, where the precise delineation of the calibration targets is of major importance.


(a) Edge points on major axes

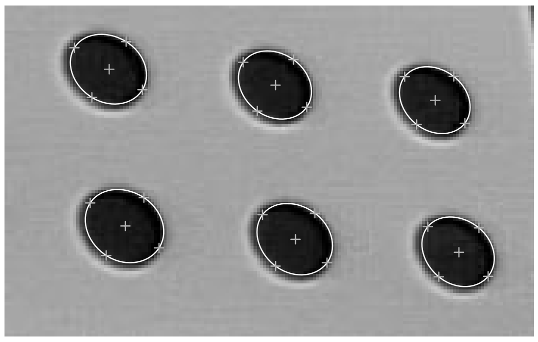

(b) Ellipse segments

Figure 4.21: Extracted extremum widths along the two major axes of the extremum (a) and approximate ellipse segments drawn between each adjacent pair of edge points along the major axes (b).

A first attempt at extracting the full perimeter curve might be to use an edge following algorithm starting at one of the four major edge points to track the edge curve around the extremum. However, this would have two major problems. First, the algorithm would have to ensure not to get sidetracked along the wrong edge curve at junctions on the perimeter curve. While this problem might be overcome, the second problem is not as easy to solve: The perimeter curve might possess points where the edge is too weak or broad to be detected. Therefore, an incomplete perimeter would result. Hence, to derive an algorithm to extract the full perimeter curve of an extremum, a good idea is to start from the definition (4.24). As was discussed in Section 4.2.1, the perimeter curve of an extremum is assumed to bound the star-shaped area around the extremum. Therefore, each angle $\alpha$ formed by a line through the extremum point $e$ with the major axis $a_1$ has exactly one associated width along this line. Hence the problem of finding the perimeter curve can be reduced to finding the width of the extremum at selected angles. While this certainly could be done for a fixed quantization of angles, which could even be calculated approximately from the four major widths, this procedure would not take the shape of the extremum into account. Therefore, a more flexible scheme to find a suitable set of angles

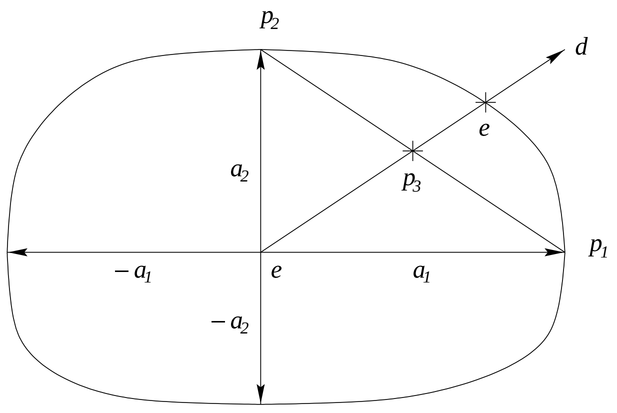
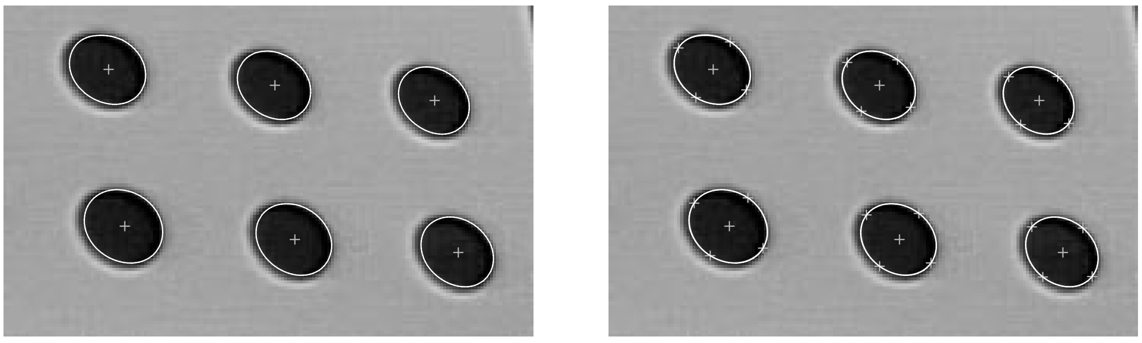


Figure 4.22: Subdivision of a segment of the perimeter curve during the extraction of the width of an extremum.

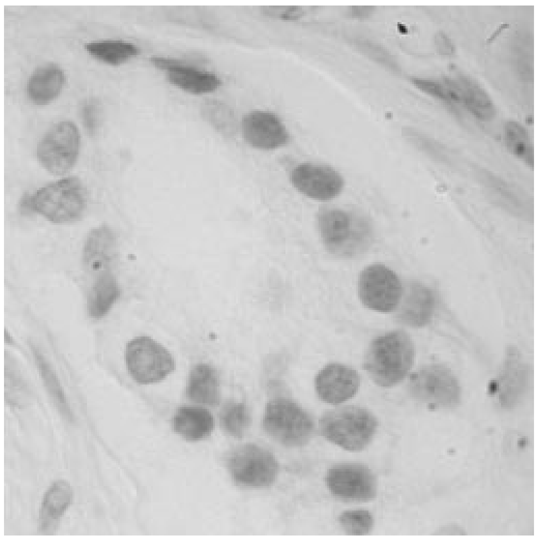


(a) Extrema and perimeters (b) Perimeters and ellipse segments

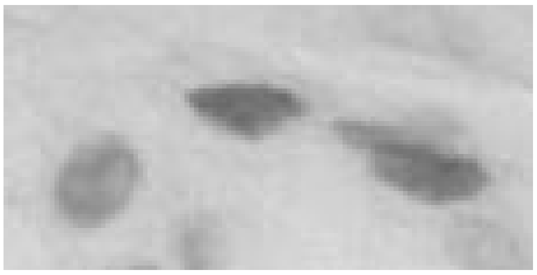
Figure 4.23: Fully extracted perimeters around the extrema in the calibration table example (a) and perimeter curves with approximate ellipse segments superimposed (b). Note that the ellipses are hardly visible because they match the extracted widths almost perfectly.

was implemented. The idea behind the scheme is to use a divide-and-conquer approach. The algorithm starts by taking two adjacent major edge points and constructs a straight line between them. Figure 4.22 demonstrates this. The two major edge points are labeled $p_1$ and $p_2$ there. Then, the point $p_3$ in the middle between $p_1$ and $p_2$ is calculated. This defines the new search direction $d$ and its associated angle. This direction is used to determine the edge point $e$ and its associated width. The procedure can be repeated recursively on the two sections given by $p_1$ and $e$, and by $e$ and $p_2$. This scheme is applied until the points $p_1$ and $p_2$ lie in 8-adjacent pixels or until the algorithm determines that the line given by $p_1$ and $p_2$ cannot be divided any further. This can happen, for example, if two very different widths are extracted at closely spaced angles. If this procedure is applied to all four segments between adjacent major edge points, the full perimeter curve is obtained.

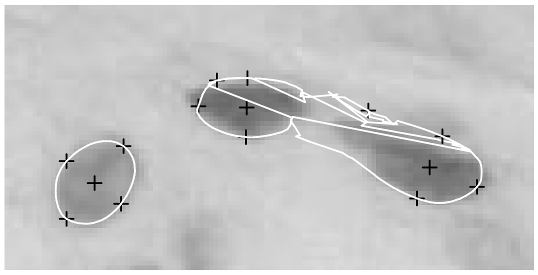
Figure 4.23(a) displays the full perimeter curves extracted from the part of the calibration table shown in Figure 4.21(b). Note that the perimeters correspond very well with the shape of the calibration targets. The edge locations in the upper right part of each calibration mark are



(a) Prostrate cells



(b) Input image



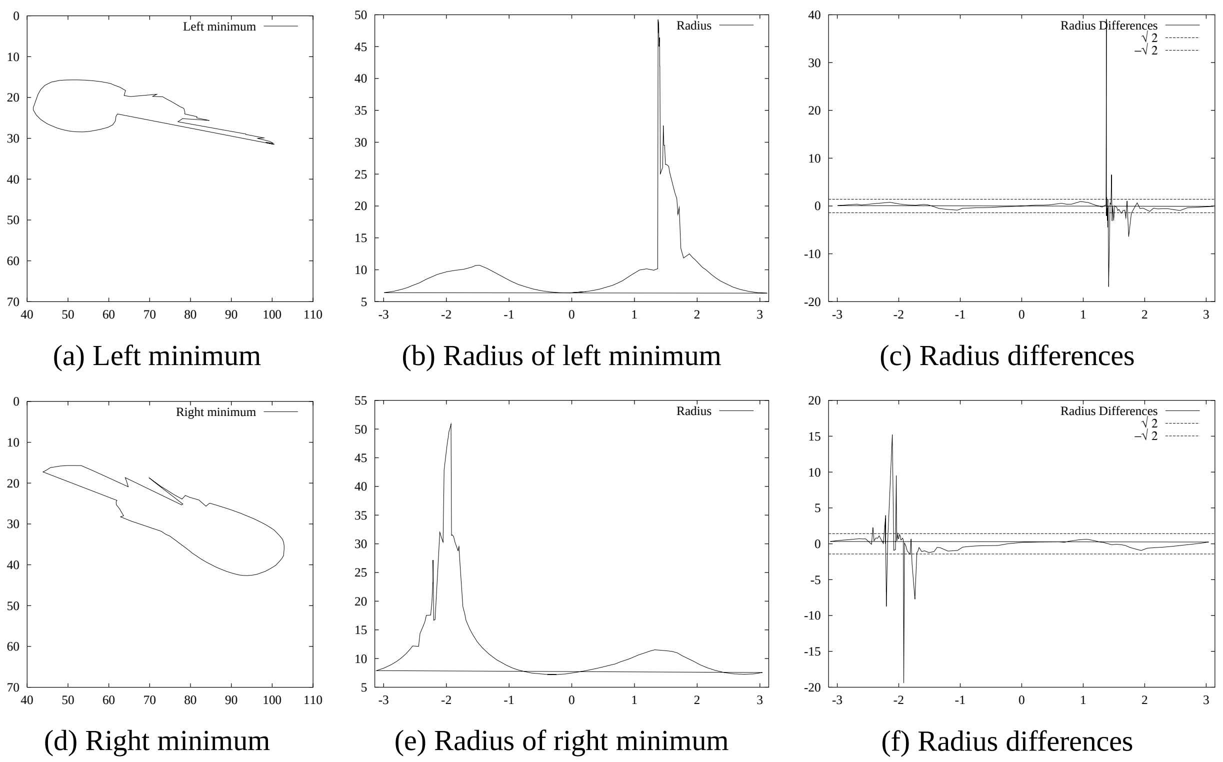
(c) Extracted minima

Figure 4.24: Extracted minima (c) on the upper left part (b) of the prostrate cell image in (a).

very precise, although a quite large $\sigma$ of 6 was used. In contrast, the lower left parts are roughly 1.5–2 pixels from their true locations, mainly because the edge exhibits strong non-symmetrical areas there. For comparison purposes, the approximating ellipse segments are displayed in light gray along with the extracted perimeter curves in Figure 4.23(b). As can be seen, since the calibration marks are almost perfect ellipses in this image, the two curves match almost exactly on all calibration marks.

The major problem with the width extraction scheme described so far is that for more complicated images missing edge points frequently occur. An example of this effect is shown in Figure 4.24(c), where minima and their perimeter have been extracted from a part of an image of prostrate cells displayed in Figure 4.24(a). There are very gradual and weak edges between the two topmost cells, and hence the perimeter curves each leak into the opposite cell, respectively. Because the perimeter curves of the two extrema overlap, they are shown separate in Figures 4.25(a) and (d). As can be seen, the width is extracted significantly too large in the areas where the edges are missing. This example shows that outlier removal is sometimes essential for a robust calculation of the perimeter curve.

Standard outlier detection schemes involve fitting a model function to the data and marking



(a) Left minimum (b) Radius of left minimum (c) Radius differences

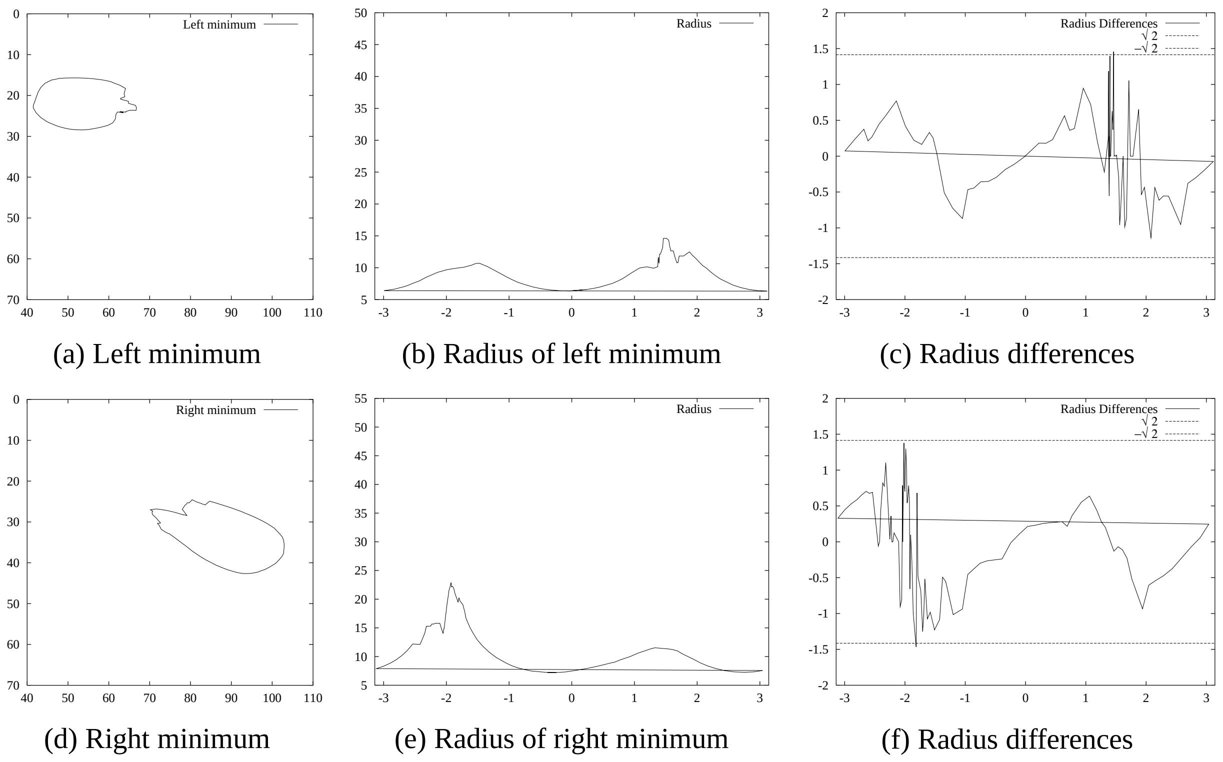
(d) Right minimum (e) Radius of right minimum (f) Radius differences

Figure 4.25: Detailed view of the left and right minimum (a), (d), as well as their radius functions (b), (e) and radius difference functions (c), (f) before outlier removal.

all data points as outliers that are further away from a robustly calculated mean than a certain constant times the robustly calculated standard deviation of the dataset [81, 152]. This algorithm can be iterated if after the first round of fitting not all outliers have been removed. However, this procedure does not appear very well suited for the task of outlier detection on perimeter curves for three reasons. First of all, a suitable model function for the perimeters has to be chosen. Since the only assumption is that the perimeter is star-shaped with respect to the extremum point, it is not at all clear which model function to choose. A suitable model function might be to use Fourier descriptors to describe the radius function $r(\alpha)$:

$$r(\alpha) = \sum_{i=0}^{n} \beta_i \sin(i\alpha + \phi_i) \quad , \tag{4.31}$$

where $\beta_i > 0$ and $\phi_i \in [0, 2\pi]$. However, it is not clear how to choose $n$ because the Fourier description can be made arbitrarily precise by choosing a large enough $n$, and in order to reject outliers $n$ must be chosen small enough so that the model is restrictive enough to detect outliers. The second problem with standard outlier rejection schemes is that the iterative procedure described above will often label and remove perfectly correct data points as outliers if the outliers are large enough to provide a very bad model fit because the fitting has to make a tradeoff between correct points and outliers [49]. Finally, as can be seen from Figures 4.25(a) and (d), the outliers on perimeter curves usually occur over a connected region of the perimeter curve. This will make the fitting of the model function even harder because instead of having to



(a) Left minimum (b) Radius of left minimum (c) Radius differences

(d) Right minimum (e) Radius of right minimum (f) Radius differences

Figure 4.26: Detailed view of the left and right minimum (a), (d) as well as their radius functions (b), (e) and radius difference functions (c), (f) after outlier removal.

make a tradeoff between correct points and outliers at isolated points, the tradeoff now has to be made in an extended region, and hence the model necessarily will fit better to the outlier points in this region. For example, in Figure 4.25(a), the fitted model function would lie somewhere in between the true curve and the spike on the right hand side of the perimeter curve. Therefore, some of the perfectly correct points on the perimeter before and after the spike would be labeled as outliers, and be removed.

For these reasons, a more specific outlier rejection scheme is needed for the task at hand. Figures 4.25(b) and (e) display the radius function $r(\alpha)$ of the two example extrema for $\alpha \in [-\pi, \pi]$. It can be seen that at the beginning and end of the outlier region on the perimeter curve there is a very large jump in the radius function, while in the rest of the curve the radius varies smoothly. This suggests to take a look at the differences in radii of consecutive points on the perimeter curve. These are displayed in Figures 4.25(c) and (f). As can be seen, the border of an outlier region corresponds to large differences in radii. Therefore, a suitable procedure to find the limits of an outlier region seems to be to label all points as outliers where the difference in radii is larger than a certain threshold. In Figures 4.25(c) and (f), $\pm\sqrt{2}$ is drawn as the threshold, and this corresponds very closely to the points which a human observer would intuitively label as the borders of outlier regions.

With the above procedure, the algorithm is able to detect the borders of outlier regions quite robustly. What should be done about them, however, is not quite clear. Note that the radius difference function and the radius function have a one-to-one correspondence: In analytical

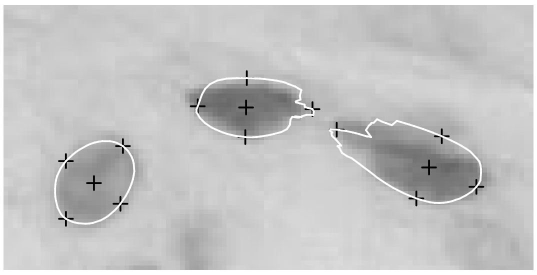


Figure 4.27: Prostrate cells after outlier removal.

space the following relations hold

$$d(\alpha) = \frac{\partial}{\partial\alpha} r(\alpha) \qquad \text{and} \qquad r(\alpha) = \int_0^\alpha d(t)\,\mathrm{d}t \ . \tag{4.32}$$

In the discrete formulation, the derivative is replaced by a difference and the integral by a sum. If the resulting radius function is required to be smooth, and hence the absolute value of the radius difference function to be bounded by a threshold, the following algorithm can be derived: Since the borders of the outlier regions correspond to large radius differences, and since it can be assumed that outlier regions always are delimited by radius differences of opposite sign, one way to make the radius difference function bounded is to search for the largest pair of radius differences of opposite sign. Suppose it were known which points actually belong to the true perimeter curve. If the largest radius differences are removed as a pair, the radius function in the outlier area will be drawn towards the true perimeter curve if the radius difference function is integrated. In order to ensure that the perimeter curve remains closed, it has to be ensured that the difference which is removed cancels, i.e., the removed outliers on $d(\alpha)$ must have the same magnitude, but opposite sign. To do this, the smaller difference can be set to zero, while this imbalance is subtracted from the larger difference. This procedure is iterated until only one or no radius differences lie above the threshold. After this, the perimeter curve can be obtained by summing up the radius differences from a suitable starting point. In order for this to work, it must be ensured that the starting point lies on the true perimeter curve. A robust estimate for this point is the point with the median value of the radius function. Figures 4.26(a) and (d) show the resulting perimeter curves if this procedure is applied, while the new radius and radius difference functions are displayed in Figures 4.26(b), (c), (e), and (f). Figure 4.27 exhibits the resulting perimeters on the original image. As can be seen, the algorithm was quite successful in drawing the perimeter curve back close to the true perimeter of the cells in this example. Of course, the perimeter curve in the areas where the outliers have been removed will not correspond to the true perimeter, and the shape of the object there corresponds to the shape of the perimeter curve in the outlier region because the radius differences are left untouched there if they are smooth. This seems justifiable in light of the fact that there is no useful perimeter information in this area anyway, so any smooth boundary may be valid. However, one could also argue that the perimeter in the outlier area should be required to be as smooth as possible, e.g., by minimizing the integral of the second derivatives there. This is undesirable, however, because a much more complicated algorithm that would have to label all outlier points explicitly would result. For this reason, and because the resulting perimeter curves seem to be smooth enough to be useful in the images the algorithm was tested on, such a procedure was not implemented.

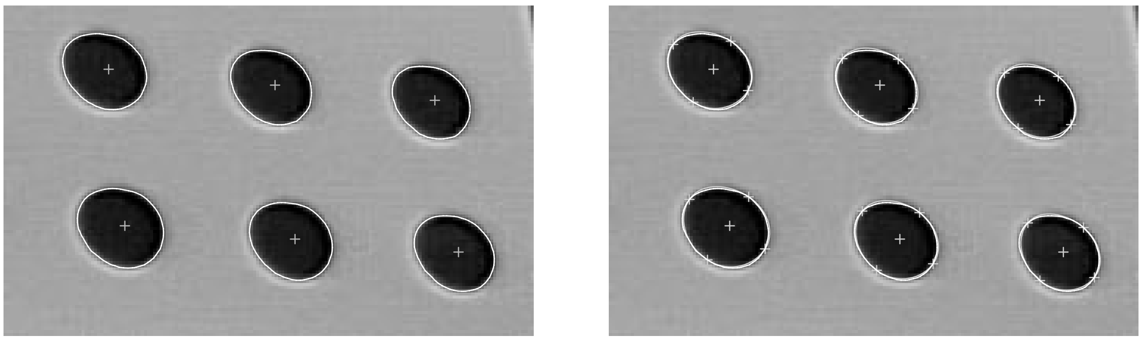
### 4.2.4 Bias Removal for Perimeters

From the discussion in Section 4.2.2 it can be seen that the extracted perimeter curves will generally be biased if $\sigma$ is chosen according to (4.27). Therefore, a bias removal algorithm similar to the one given in Section 3.1.10 is highly desirable. However, for extrema such a comparatively simple algorithm does not exist. There are several reasons for this. In general, the position of the extremum and the perimeter curve depends on the shape of the true perimeter curve and the distribution of the gray values along a region of the 2D space in the vicinity outside of the perimeter curve. Therefore, the bias at least depends on two functions with an infinite number of parameters: The perimeter curve $p(\alpha)$ and the contrast on the perimeter curve $h(p(\alpha))$. While it is at least conceivable that the bias map depending on these two functions may be invertible, it is certainly infeasible to proceed in this way. First of all, the integrals in the calculation of the predicted bias are not solvable analytically. Second, and more importantly, even if the bias function were invertible, it is impossible to pre-compute a table of corrections as in Section 3.1.10. Therefore, the inversion would have to be done numerically for each perimeter of every extremum, which is extremely expensive in terms of computational resources.

For these reasons, the only feasible manner to remove the bias from the perimeter curves seems to be to extract them again with a lower $\sigma$, because for small $\sigma$ the bias induced by Gaussian smoothing will be small. This is similar in spirit to the edge focusing algorithms for edge detection described in [15, 31, 162]. The bias removal algorithm therefore calculates the gradient image anew with a small $\sigma$ for localization given by the user. This $\sigma$ should generally be selected as low as possible while taking into account the noise in the image and the breadth of the edges. After this, edges are extracted along all the angles $\alpha_i$ of the perimeter curve obtained in the detection phase. As shown in Section 4.2.2, the edges of the perimeter curve can also move closer to the extremum as $\sigma$ gets larger. Therefore, the search space for the localization step is chosen to be 1.2 times the width of the perimeter at $\alpha_i$ obtained from the detection step. This enlargement has proved to be sufficient from tests done with several images. One problem with this approach is that there may be several edge points lying on the search line, which have been annihilated into a single edge point extracted in the detection stage. In order to determine the improved width estimate, the edge point on the search line having the largest gradient is chosen because this edge point has most likely made the largest contribution to the single, merged edge point from the detection stage. Another possibility would be to select the edge point lying closest to the edge point from the detection stage.

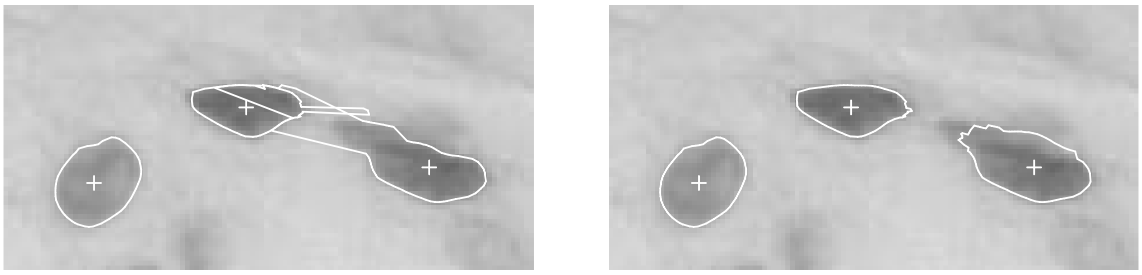
Figure 4.28(a) shows the corrected perimeter curve for the part of the calibration table displayed in Figures 4.21 and 4.23. Since the edges are very sharp and the noise level in the image is low, a very low localization $\sigma$ of 0.7 was chosen. As can be seen, the corrected perimeter curve lies exactly on the true perimeter. For completeness sake, approximating ellipse segments are displayed in Figure 4.28(b) along with the corrected perimeter curves. Although the ellipses do not approximate the localized perimeters as closely as in the detection stage, the match is still very good.

In Figure 4.29(a) the corrected perimeters of the prostrate cell example of Figures 4.24 and 4.26 are shown. Here, a localization $\sigma$ of 1.5 was used because the edges are comparatively



(a) Extrema and perimeters (b) Perimeters and ellipse segments

Figure 4.28: Corrected perimeters around the extrema in the calibration table example (a) and perimeter curves and approximate ellipse segments superimposed (b). Note that the ellipses are hardly visible because they match the extracted widths almost perfectly.



(a) Corrected perimeters (b) Corrected perimeters without outliers

Figure 4.29: Corrected perimeters around the extrema in the prostrate cell example without (a) and with outlier removal (b).

broad. The perimeter curves correspond closely to the true object boundaries for the most part. However, again the outliers spoil the extraction result. The corrected perimeters with outliers removed are shown in Figure 4.29(b). It can be seen that the corrected results now lie on the true object perimeters almost everywhere.

### 4.2.5 Examples

This section is concluded by showing the extrema extraction results on the full calibration table image of Figure 4.20(a). Figure 4.30(a) displays the extracted minima for $\sigma = 6$ with a threshold of 0.4. From the previous examples it could be expected that the perimeter curves correspond closely to the calibration marks on the upper right hand side of the marks, while they lie on the inside of the marks on the lower left part, and this is indeed the case. All calibration marks have been extracted successfully, while only four extraneous minima have been found. These can easily be removed based on their shape properties, e.g., their moments. A formula to derive arbitrary moments of a closed polygon is given in [170]. Figure 4.30(b) displays

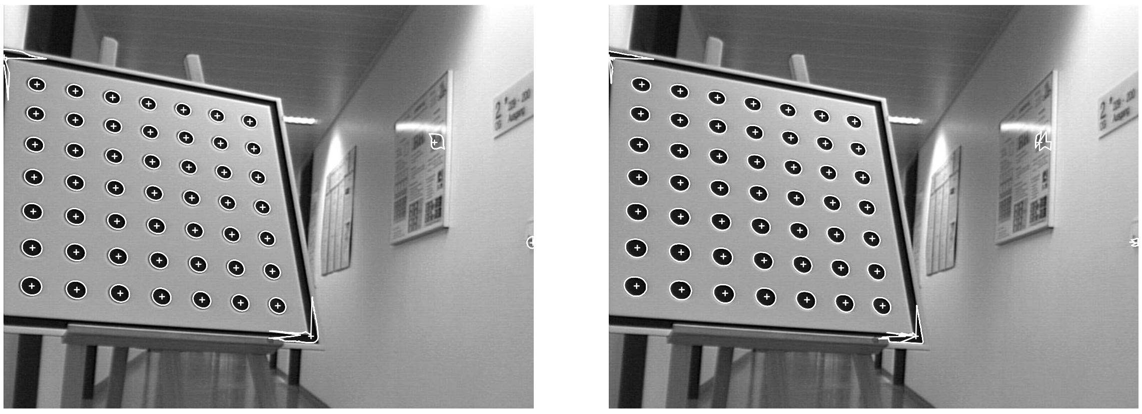

(a) Extrema and perimeters (b) Extrema and corrected perimeters

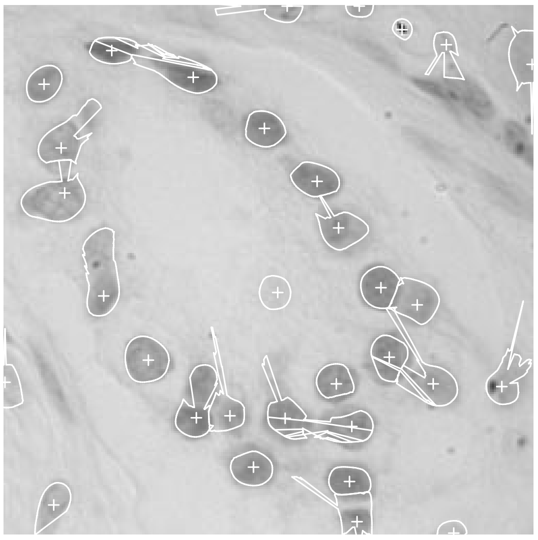
Figure 4.30: Extracted minima and perimeters on the full image of the calibration table (a) and corrected perimeters (b).

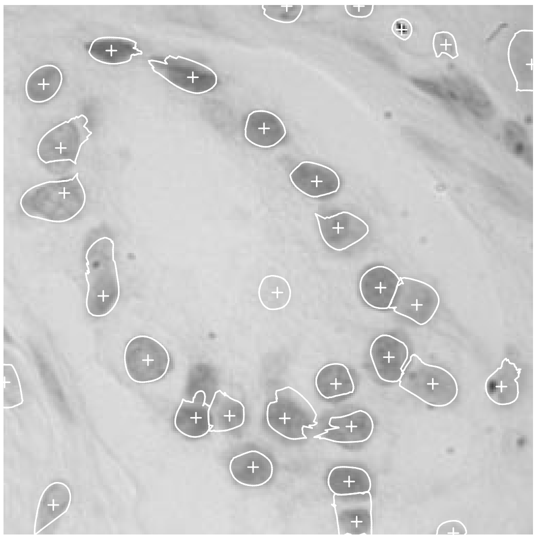
the corrected perimeter curves obtained with a localization $\sigma$ of 0.7. As was to be expected, the extracted perimeters lie exactly on the calibration marks' boundaries.

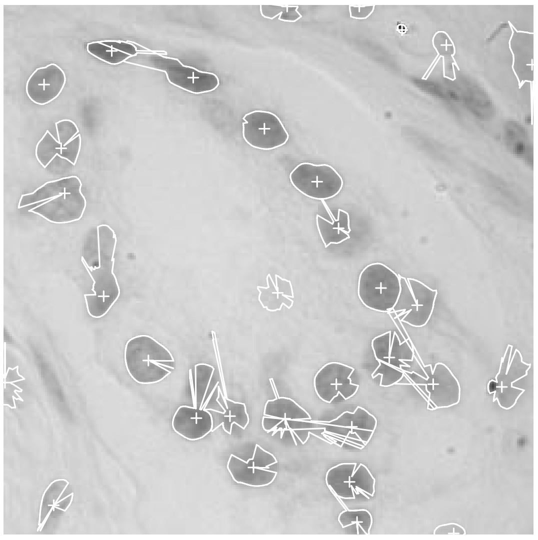
As a final example, Figure 4.31 shows the extraction of minima on the full image of prostrate cells given above in Figure 4.24(a). The extraction results with $\sigma = 5$ and a threshold of 0.1 are displayed in Figure 4.31(a). Most of the cells have been detected correctly. However, there are numerous outliers on several of the cells. These have been successfully removed in Figure 4.31(b). However, for several cells the localization of the perimeters are far from being optimum. Note, for example, the cell at the lower image boundary about one third from the right border. Figure 4.31(c) displays the results of correcting the perimeter curves with $\sigma = 1.5$. The perimeter curves now correspond much better to the objects' boundaries, although outliers are again a major problem in many cells. As is to be expected, most of the outliers now lie towards the extremum side of the perimeter curve, because the edge points with the largest gradient are selected. Finally, Figure 4.31(d) shows the corrected perimeters with outliers removed. The results are very good for most of the cells, where the perimeters delineate the cells' boundaries precisely. One notable exception is the cell immediately below the two touching cells at the right of the center of the image, where the perimeter has been extracted significantly too large because of the outlier removal. All in all, the extraction results are quite satisfactory on this difficult image.

## 4.3 Lines in 3D Images

After considering the problem of extracting 2D extrema and their width, the foundation is laid for the extraction of 3D lines and their width. Analogously to 2D lines, 3D lines conceptually are given by their center line, which is a one-dimensional manifold, or curve, $s(t)$ in $\mathbb{R}^3$. Obviously, the entire line is not a one-dimensional manifold since it also possesses a width, and hence is a full three-dimensional object.


(a) Extrema and perimeters


(b) Extrema and perimeters with outlier removal


(c) Corrected perimeters

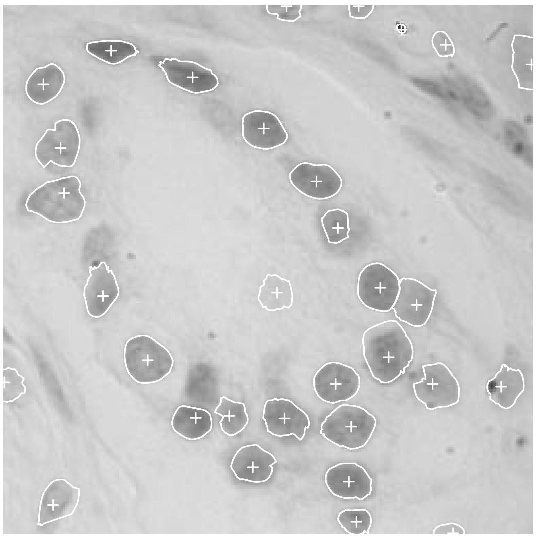

(d) Corrected perimeters with outlier removal

Figure 4.31: Extracted minima and perimeters on the full prostrate cell image without (a) and with outlier removal (b). Corrected perimeters without (c) and with outlier removal (d).

### 4.3.1 Line Models

As was discussed in the introduction to this chapter, line points are given by restricted extrema in the plane spanned by the two eigenvectors of the Hessian matrix corresponding to the largest eigenvalues. A suitable model for a three-dimensional line is then given by assuming it to have a constant gray value in a two-dimensional star-shaped region around the restricted extremum in the plane perpendicular to the line direction, analogously to the 2D extrema discussed in the previous section. Again, no assumption is made about the gray value distribution outside of the line. Therefore, the width of a three-dimensional line can be described by a perimeter curve

$$p(\alpha) = e + a_1 r(\alpha)\cos\alpha + a_2 r(\alpha)\sin\alpha \tag{4.33}$$

for each line point. This is, of course, the same formula as (4.24). The difference is that for 3D lines, $e$, $a_1$, and $a_2$ are 3D points and vectors, respectively.

## 4.3.2 Detection of Line Points

To extract line 3D points, the relevant partial derivatives of the image are needed. As for 3D surfaces, they are obtained by convolution with the appropriate partial derivatives of the Gaussian kernel (4.1)–(4.9). The corresponding Hessian matrix and local Taylor polynomial are given by (4.10) and (4.11), respectively. If the two eigenvectors of largest absolute value are denoted by $(n_x, n_y, n_z)$ and $(m_x, m_y, m_z)$, and $\|(n_x, n_y, n_z)\|_2 = \|(m_x, m_y, m_z)\|_2 = 1$ is required, $tn + sm$ can be plugged into (4.11) to obtain:

$$\begin{aligned}
&z(tn_x + sm_x, tn_y + sm_y, tn_z + sm_z) = \\
&= r + \begin{pmatrix} tn_x + sm_x & tn_y + sm_y & tn_z + sm_z \end{pmatrix} \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} + \\
&\quad \frac{1}{2}\begin{pmatrix} tn_x + sm_x & tn_y + sm_y & tn_z + sm_z \end{pmatrix} \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{xy} & r_{yy} & r_{yz} \\ r_{xz} & r_{yz} & r_{zz} \end{pmatrix} \begin{pmatrix} tn_x + sm_x \\ tn_y + sm_y \\ tn_z + sm_z \end{pmatrix} \\
&= r + (tn_x + sm_x)r_x + (tn_y + sm_y)r_y + (tn_z + sm_z)r_z + \frac{1}{2}(tn_x + sm_x)^2 r_{xx} + \\
&\quad (tn_x + sm_x)(tn_y + sm_y)r_{xy} + (tn_x + sm_x)(tn_z + sm_z)r_{xz} + \\
&\quad \frac{1}{2}(tn_y + sm_y)^2 r_{yy} + (tn_y + sm_y)(tn_z + sm_z)r_{yz} + \frac{1}{2}(tn_z + sm_z)^2 r_{zz} \ .
\end{aligned} \tag{4.34}$$

Reordering the terms with respect to $s$ and $t$ results in

$$\begin{aligned}
&z(tn_x + sm_x, tn_y + sm_y, tn_z + sm_z) = \\
&= r + t(n_x r_x + n_y r_y + n_z r_z) + s(m_x r_x + m_y r_y + m_z r_z) + \\
&\quad t^2(\frac{1}{2}n_x^2 r_{xx} + n_x n_y r_{xy} + n_x n_z r_{xz} + \frac{1}{2}n_y^2 r_{yy} + n_y n_z r_{yz} + \frac{1}{2}n_z^2 r_{zz}) + \\
&\quad ts(n_x m_x r_{xx} + n_x m_y r_{xy} + n_y m_x r_{xy} + n_x m_z r_{xz} + n_z m_x r_{xz} + \\
&\qquad n_y m_y r_{yy} + n_y m_z r_{yz} + n_z m_y r_{yz} + n_z m_z r_{zz}) + \\
&\quad s^2(\frac{1}{2}m_x^2 r_{xx} + m_x m_y r_{xy} + m_x m_z r_{xz} + \frac{1}{2}m_y^2 r_{yy} + m_y m_z r_{yz} + \frac{1}{2}m_z^2 r_{zz}) \\
&= r + \begin{pmatrix} t & s \end{pmatrix} \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \frac{1}{2}\begin{pmatrix} t & s \end{pmatrix} \begin{pmatrix} c_{xx} & c_{xy} \\ c_{xy} & c_{yy} \end{pmatrix} \begin{pmatrix} t \\ s \end{pmatrix} \ ,
\end{aligned} \tag{4.35}$$

where

$$\begin{aligned}
c_x &= n_x r_x + n_y r_y + n_z r_z && (4.36)\\
c_y &= m_x r_x + m_y r_y + m_z r_z && (4.37)\\
c_{xx} &= n_x^2 r_{xx} + 2n_x n_y r_{xy} + 2n_x n_z r_{xz} + n_y^2 r_{yy} + 2n_y n_z r_{yz} + n_z^2 r_{zz} && (4.38)\\
c_{xy} &= n_x m_x r_{xx} + n_x m_y r_{xy} + n_y m_x r_{xy} + n_x m_z r_{xz} + n_z m_x r_{xz} + \\
&\quad n_y m_y r_{yy} + n_y m_z r_{yz} + n_z m_y r_{yz} + n_z m_z r_{zz} && (4.39)\\
c_{yy} &= m_x^2 r_{xx} + 2m_x m_y r_{xy} + 2m_x m_z r_{xz} + m_y^2 r_{yy} + 2m_y m_z r_{yz} + m_z^2 r_{zz}\ . && (4.40)
\end{aligned}$$

Equation (4.35) is, as was to be expected, a two-dimensional quadratic polynomial $z(t, s)$. Completely analogously to the case of 2D extrema in Section 4.2.2, the line position, i.e., the restricted 3D extremum, can be extracted by calculating the gradient of the local Taylor polynomial

$$\nabla z(t,s) = \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \begin{pmatrix} c_{xx} & c_{xy} \\ c_{xy} & c_{yy} \end{pmatrix} \begin{pmatrix} t \\ s \end{pmatrix} \tag{4.41}$$

and setting it to zero. This results in the following linear equation:

$$\begin{pmatrix} c_{xx} & c_{xy} \\ c_{xy} & c_{yy} \end{pmatrix} \begin{pmatrix} t \\ s \end{pmatrix} = \begin{pmatrix} -c_x \\ -c_y \end{pmatrix}\ . \tag{4.42}$$

After solving this linear equation, the line position is given by

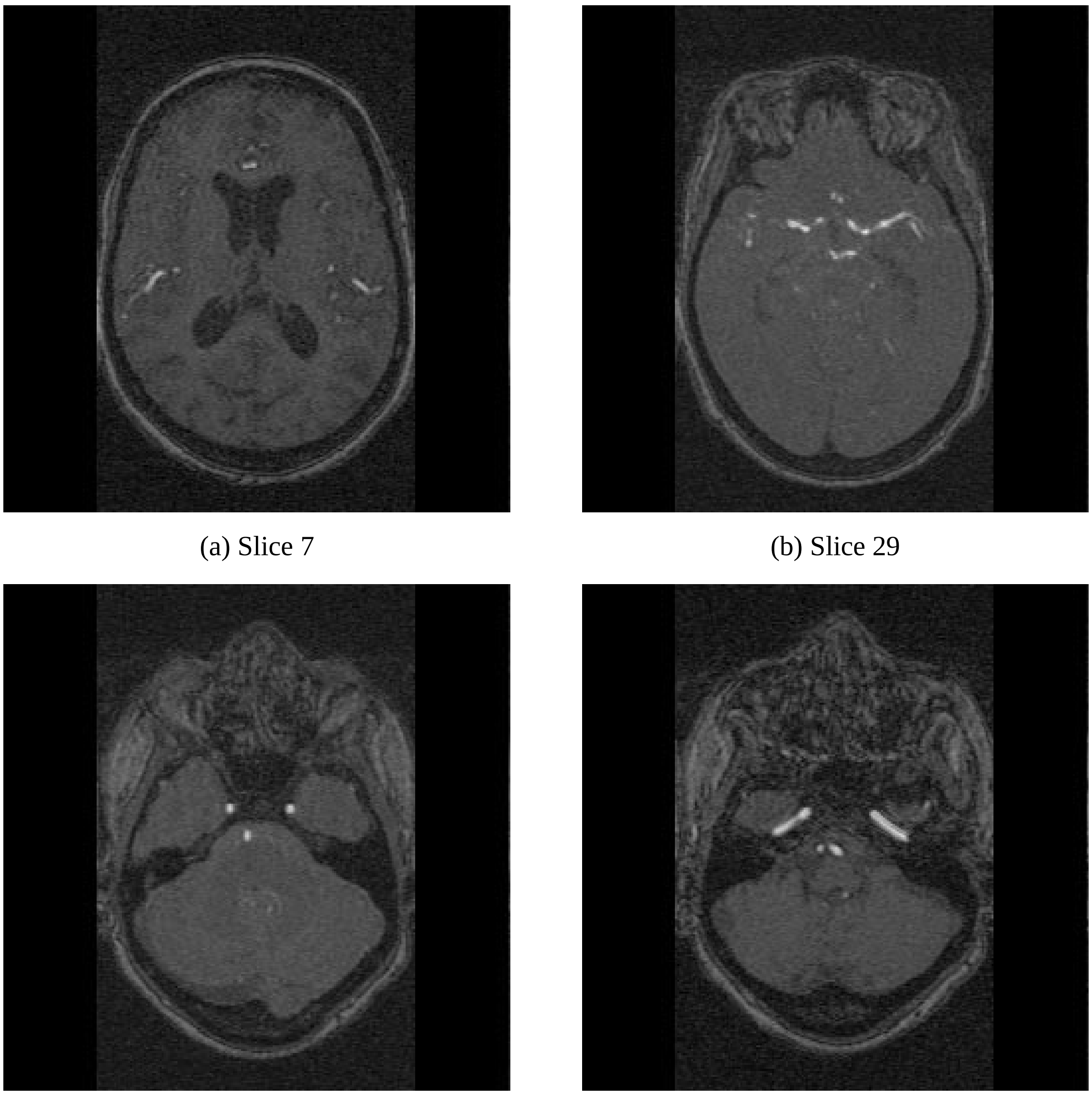
$$(p_x, p_y, p_z) = (tn_x + sm_x, tn_y + sm_y, tn_z + sm_z)\ . \tag{4.43}$$

As always, $(p_x, p_y, p_z) \in [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ is required in order for a point to be declared a line point. Furthermore, analogously to the extraction of 2D extrema, salient line points can the be selected if both largest eigenvalues have the same sign and are larger than a certain threshold.

Figure 4.32 shows four slices of an MR angiogram of a human head. This image was kindly provided by Dr. Stephen Aylward of the Department of Radiology at the University of North Carolina, Chapel Hill. It consists of 72 slices of dimension $256 \times 256$ each. Because the inter-slice distance of the original dataset is 1.2 times the pixel size, it was rescaled along the slice axis to be isotropic. Furthermore, the black borders of the image were cropped off to reduce the size of the dataset, resulting in an image of dimensions $190 \times 240 \times 87$. Figure 4.33 displays the results of extracting bright line points from this dataset with $\sigma = 2$, while Figure 4.34 displays a close-up view around the junction in the central part of Figure 4.33. Similar to Section 4.1.2, the line points are visualized by their local tangential lines of length 1. As can be seen, the line positions are very smooth, indicating good accuracy. As was the case for surfaces, this type of visualization already gives a very good impression of line continuity, although the close-up view Figure 4.34 reveals the line points not to be linked. Also interesting is the fact that several line points were extracted perpendicular to the line direction in areas of high curvature. Therefore, a linking step is essential to remove these spurious line points.

## 4.3.3 Linking Line Points into Lines

The linking algorithm for 3D lines is an almost straightforward adaption of the linking algorithm for 2D lines described in Section 3.1.6. Again, the goals of the linking algorithm are to produce



(a) Slice 7 (b) Slice 29

(c) Slice 50 (d) Slice 55

Figure 4.32: Four slices from an MR angiogram of a human head.

a topologically sound data structure of lines and junctions, to eliminate double responses, and to orient the local coordinate frames of each line point with respect to one another. The latter point is done more for aesthetical reasons because, contrary to 2D lines, no interpolation of missing edge information along the line direction is done so far in the algorithm, and hence the orientation of the normals of the line does not play as crucial a role as for 2D lines.

As in the 2D case, the algorithm starts from the line point with the largest response, and adds appropriate neighbors to the current line in both line directions. The line direction is given by the smallest eigenvector at each line point. Again, only a restricted set of neighbors is checked for possible line continuation based on the line direction. Basically, a set of voxels around the

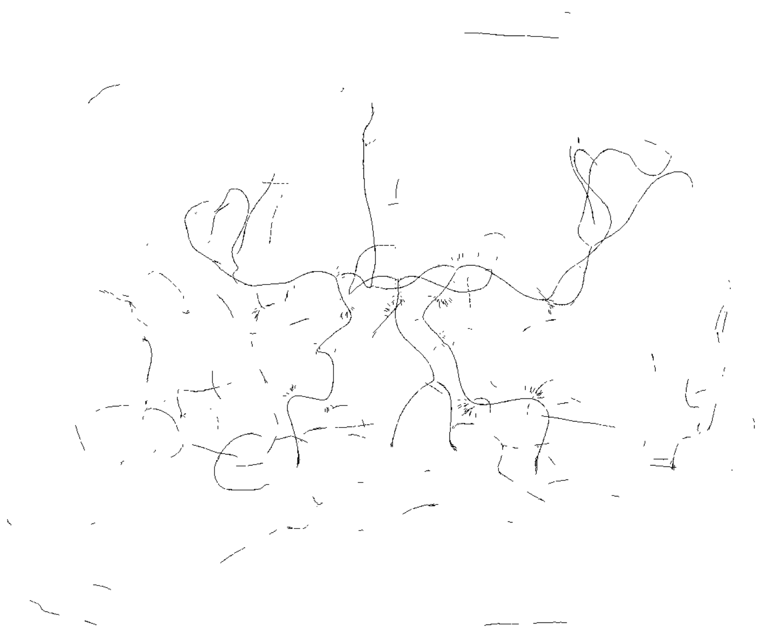


Figure 4.33: Line points extracted from an MR angiogram of a human head. Line points are visualized by their local tangential lines.

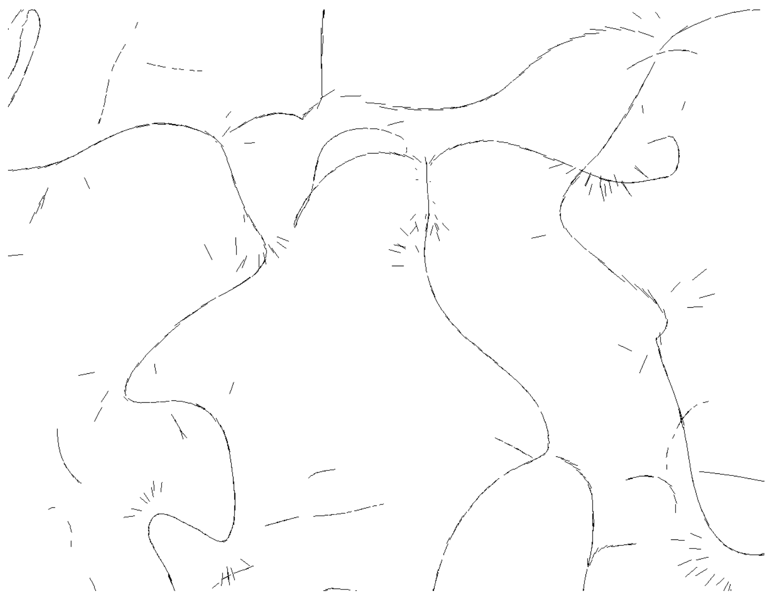


Figure 4.34: Close-up view of line points extracted from an MR angiogram of a human head.

line direction is examined. Hence, there are 26 possible configurations, of which only three are topologically different. These are displayed in Figure 4.35, where a $3 \times 3$ cube of neighbors is shown from two opposite directions. Voxels checked for line continuation are displayed as light gray cubes. The arrow indicates the line direction. As in the 2D case, another set of neighbors is checked for double responses. These voxels are approximately perpendicular to the line direction, and are displayed as dark gray cubes in Figure 4.35. The choice of the appropriate neighbor to add to the current line is based on exactly the same criteria as in the 2D linking algorithm of Section 3.1.6, i.e., the distance of the line points and the angle the two respective line directions form. The rest of the linking algorithm logic, i.e., the hysteresis threshold condition and the junction detection algorithm remains unchanged.

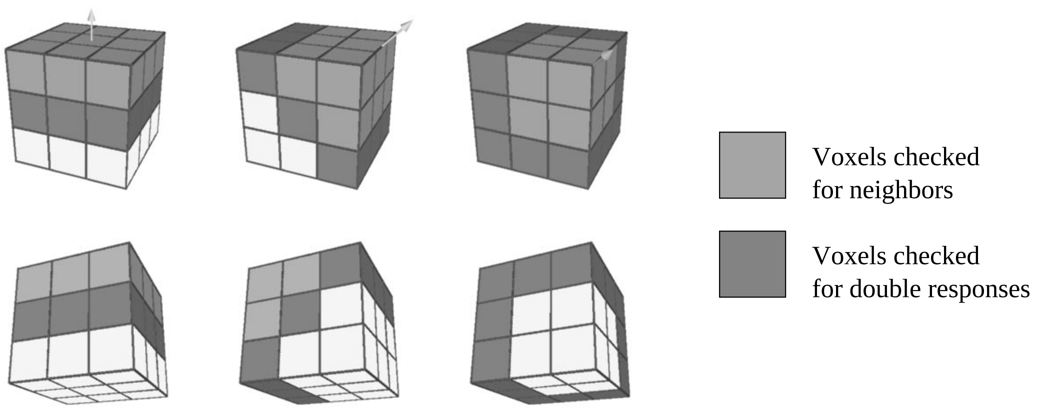

Voxels checked for neighbors
Voxels checked for double responses

Figure 4.35: Three topologically different of the twenty-six possible neighborhoods examined during the linking process.

The problem of orienting the coordinate frames of the line points is handled as follows: For the maximum point, a right-handed coordinate frame is chosen such that the line direction is aligned with the sequential order of the line points. All other coordinate frames are made right-handed, and are aligned with their neighbors on the line by possibly negating the line direction vector, and by possibly negating and reordering the two vectors corresponding to the normal plane of the line direction in such a way that the angles between the pairs of consecutive normal vectors is minimized.

Figure 4.36 displays the result of linking the line points of Figure 4.33 into lines along with the oriented coordinate frames. Figure 4.37 shows a detailed view looking down along the line in the upper central part of Figure 4.36 to give a better impression of how the coordinate frames are aligned. As can be seen from these figures, the algorithm successfully selected the salient lines, and linked the individual line points into lines, while eliminating almost all of the spurious responses visible in Figure 4.33.

## 4.3.4 Extraction of Line Perimeters

The extraction of line perimeters for 3D lines is a straightforward extension of the algorithm to extract 2D extrema perimeters discussed in Section 4.2.3. First, four 3D edge points are extracted along the two major axes of each line point. They serve as the starting points for the divide-and-conquer approach to extract the entire perimeter curve. By construction, all the extracted perimeter points lie in the plane spanned by the two major axes of the line point, as desired.

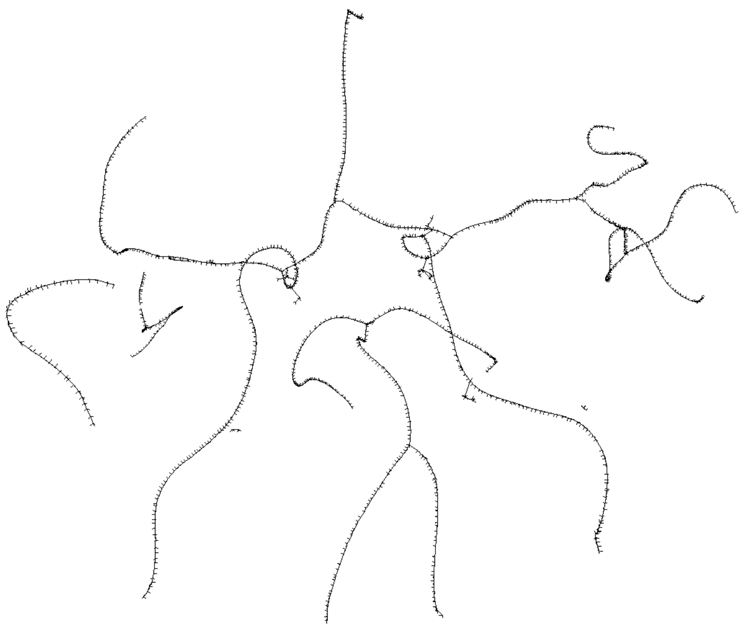


Figure 4.36: Linked lines and oriented coordinate frames extracted from an MR angiogram of a human head.

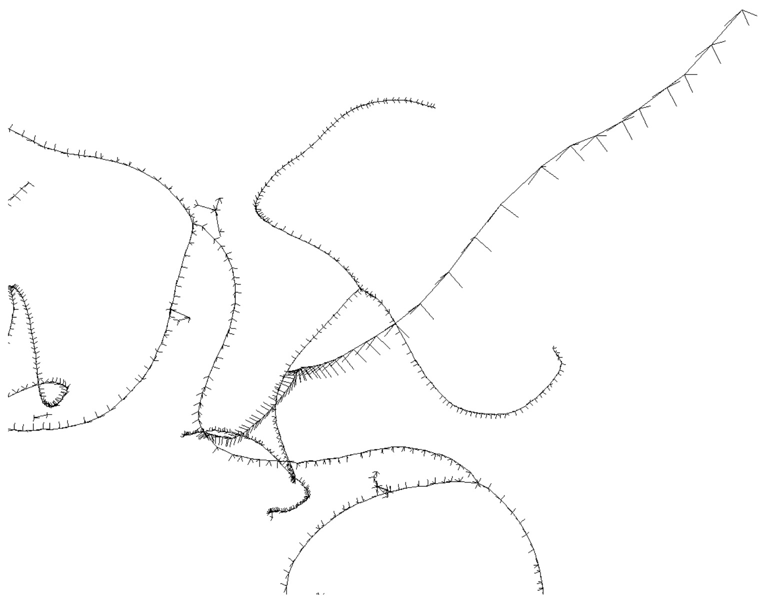


Figure 4.37: Close-up view of linked lines and oriented coordinate frames extracted from an MR angiogram of a human head. The view is from the topmost line downwards.

Figure 4.38 displays the results of extracting the line perimeters from the example MR angiogram of a human head, while Figure 4.39 shows a close-up view from within the line extending from the center of Figure 4.38 toward the lower left corner, looking towards the

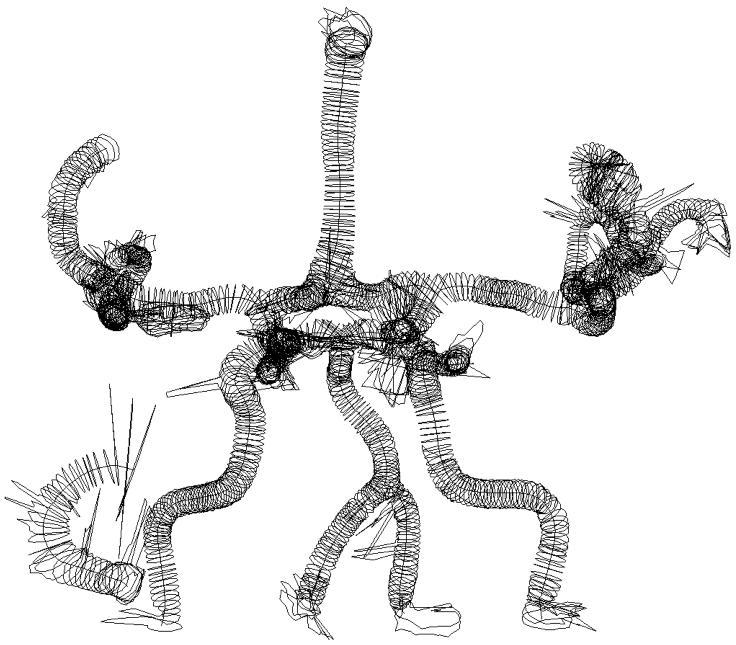


Figure 4.38: Full perimeter curves extracted from an MR angiogram of a human head.

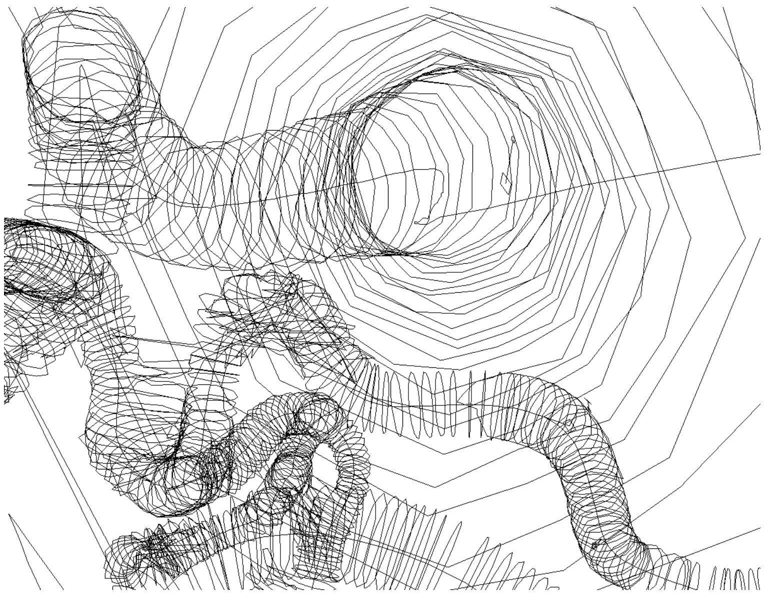


Figure 4.39: Close-up view of extracted perimeter curves from an MR angiogram of a human head.

center of the image. The extraction was done with $\sigma = 2$ and hysteresis thresholds of 1 and 2. No outlier removal has been performed. The resulting perimeter curves are very smooth almost everywhere, except for some outliers on a limited number of perimeter curves, indicating that the algorithm works very well on this image.

The type of visualization used in Figures 4.38 and 4.39 is an immediate representation of the final output data structure of the algorithm, since it displays the linked lines and the perimeter curves for each line point, and hence gives the observer an intuitive impression of the results. Unfortunately, this type of visualization does not give a good three-dimensional impression of the extracted lines. What is needed instead is a surface representation of the line perimeters as tubular objects. To obtain such a visualization, the line perimeter curves have to be triangulated, i.e., adjacent perimeter curves have to be connected by triangles. Some algorithms have been proposed for the task of reconstructing a surface representation from slices [19, 59], which is a similar problem to the one being faced here. However, they are very complicated algorithms taking into account special cases which cannot occur for the problem of triangulating line perimeters. Therefore, a different scheme was implemented, which is fast and much easier to implement. This scheme constructs the triangulation between two adjacent line perimeter curves, and can be described as follows: First, the two closest points on the two perimeter curves are calculated. The line between them forms the "fringe" of the current triangulation. Then, the algorithm takes the line on the fringe, examines the next point in counterclockwise direction on each of the two perimeter curves, and constructs a triangle from the fringe points and the point on the perimeter curve which has the shorter distance to the fringe point on the opposite perimeter curve. These two points are then put onto the fringe. The algorithm terminates if the triangulation is complete or if a maximum number of $n_1 + n_2$ triangles have been constructed, where $n_1$ and $n_2$ are the number of points on the respective perimeter curves. If in the latter case the perimeter is not complete, i.e., if the current fringe line is different from the initial fringe line, the algorithm fails to produce a complete triangulation. In the experiments carried out on real images this happens extremely rarely, e.g., if the two perimeter curves have very different widths. Since the triangulation algorithm is only intended for visualization purposes and does not change the results, this was deemed acceptable for the purposes of this work. If a guaranteed complete triangulation is necessary, one of the algorithms given in [19, 59] could be used.

Figure 4.40 shows the results of triangulating the line perimeters of Figure 4.38. As can be seen, this type of visualization gives the observer a very good impression of the three-dimensionality of the lines. The extracted line perimeters appear very smooth. In this image, the triangulation algorithm has failed in four cases, most notably in the area of the large outliers on the line running from the center to the lower left part of the image. Figure 4.41 displays a detailed view from within a line from roughly the same position as Figure 4.39 to give an impression of the quality of the extracted line positions and widths.

From the above examples it can be seen that the results suffer from outliers, similar to the extrema extracted in Section 4.2.3. Therefore, an outlier detection and removal scheme is necessary for the 3D algorithm, too. Since the outlier removal algorithm proposed for the 2D extrema only operates on the perimeter curve parameterized by its angle, it can be used unchanged for 3D lines.

Figure 4.42 shows the results of removing the outliers from the results of Figure 4.40. As can be seen, the algorithm works successfully in most of the image. However, some line widths are corrected to extremely large widths. Because the outlier removal algorithm reconstructs the perimeter starting at the point with the median width, this essentially means that more than 50% of the points are considered outliers. This may happen, for instance, if the perimeter has a strongly elliptical shape and only few outlier points. It results in the median point having

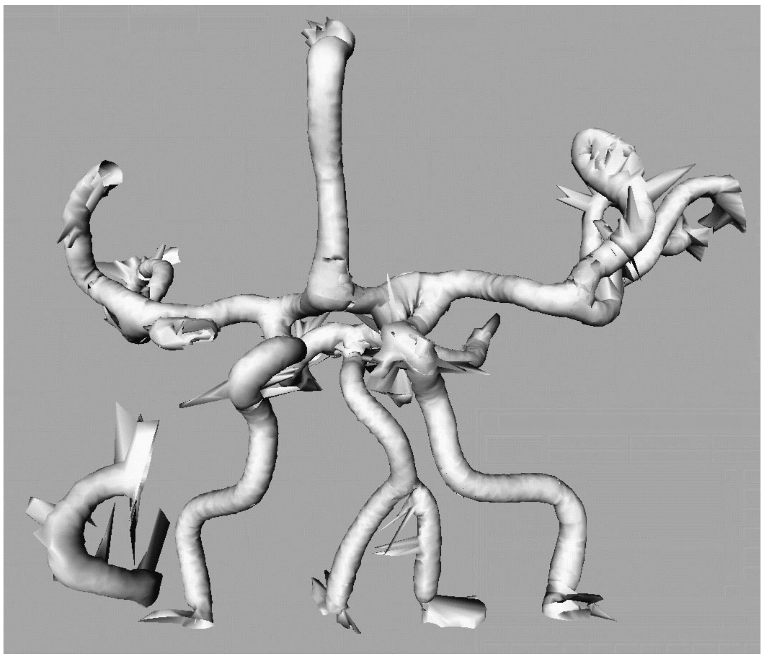


Figure 4.40: Triangulated perimeter curves extracted from an MR angiogram of a human head.

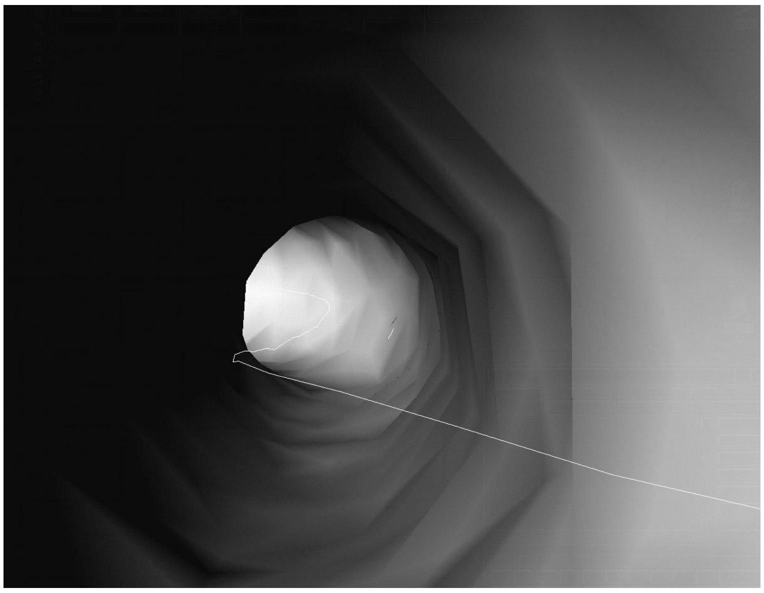


Figure 4.41: Close-up view of triangulated perimeter curves extracted from an MR angiogram of a human head.

a relatively large width, and hence yields incorrect results. Fortunately, such cases are very rare. One way to improve this results would be to additionally enforce continuity of the line perimeters in the direction of the line. This could, for example, be done by using tubular snakes to extract the line widths [32]. However, at present such a scheme has not been implemented.

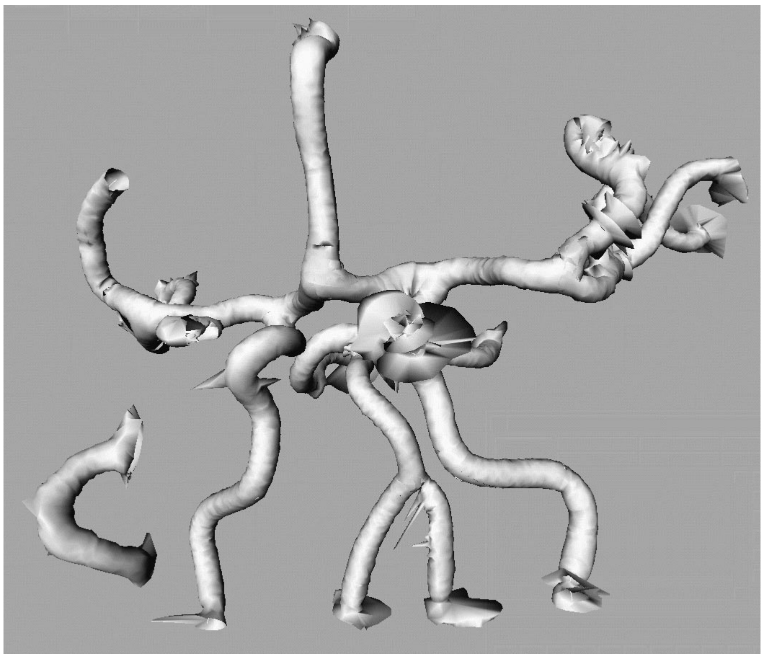


Figure 4.42: Perimeter curves with outliers removed, extracted from an MR angiogram of a human head.

### 4.3.5 Bias Removal for Perimeters of 3D Lines

The results of the previous section, in addition to the bias analysis for extrema in Section 4.2.2, indicate that the extracted line perimeters are biased. This can be seen from Figure 4.42, where all extracted lines appear to have roughly the same width, whereas from common anatomical knowledge one would expect the lines in the upper and back parts of the image to have smaller widths. Therefore, width correction is essential to obtain meaningful line widths. As was the case for 2D extrema perimeters, the only feasible way to correct the line widths appears to be to extract the perimeter curves once more with a smaller value of $\sigma$. Fortunately, the algorithm given in Section 4.2.4 extends naturally to the 3D case.

The results of line width extraction with $\sigma = 2$, hysteresis thresholds of 1 and 2, and a localization $\sigma$ of 1 are displayed in Figure 4.43. To further exhibit the effects of the outlier removal algorithm, no outliers were suppressed in this example. With these parameters, the extracted line widths correspond much more closely to the true widths of the blood vessels. As expected, the lines in the lower part of the image are much wider than in the middle and back parts of the image. Figure 4.44 displays a close-up view of the line junction immediately to the right of the central junction of Figure 4.43. As can be seen, there are numerous outliers connecting the two lines to the left and below the junction. Note also how well the remaining junctions have been extracted.

Figure 4.45 exhibits the results of removing the outliers in Figure 4.43. There are only three noticeable remaining outliers, where the line width has been corrected to excessively large values by the outlier removal algorithm. Furthermore, all junction areas have been correctly extracted and the triangulation algorithm has produced no visible artifacts in this image. Figure 4.46 shows the same junction area as Figure 4.44. As can be seen, the outlier removal

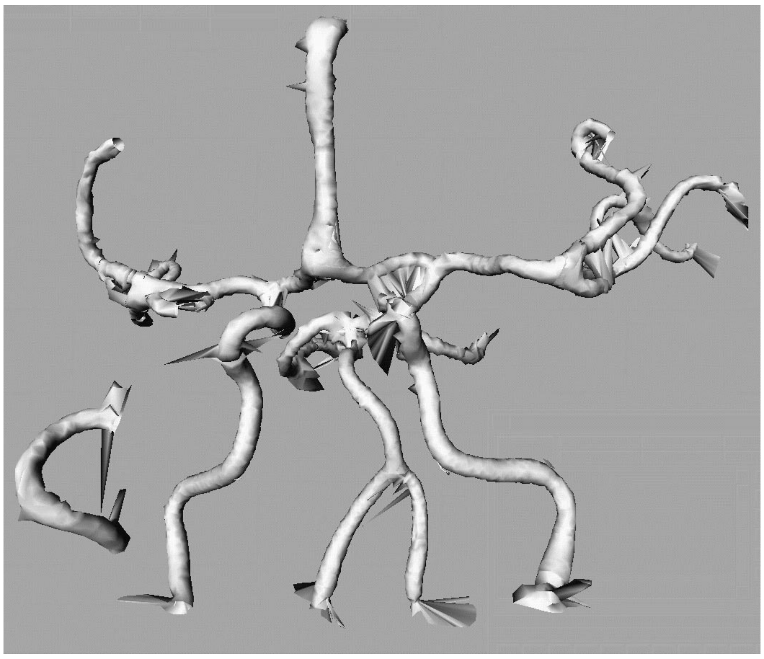


Figure 4.43: Corrected perimeter curves without outlier removal extracted from an MR angiogram of a human head.

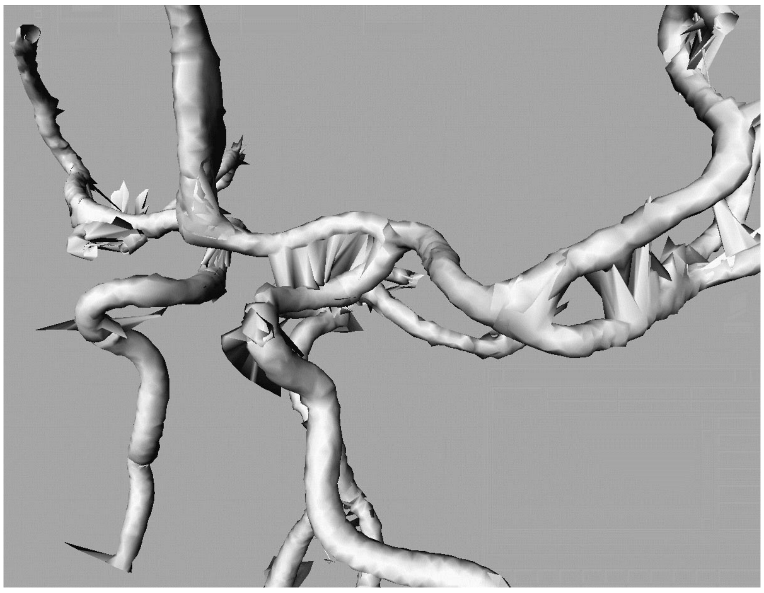


Figure 4.44: Close-up view of corrected perimeter curves without outlier removal from an MR angiogram of a human head.

algorithm was very successful in separating the two lines that seemed to have merged in Figure 4.44. In summary, the algorithm has produced a very good result on this image.

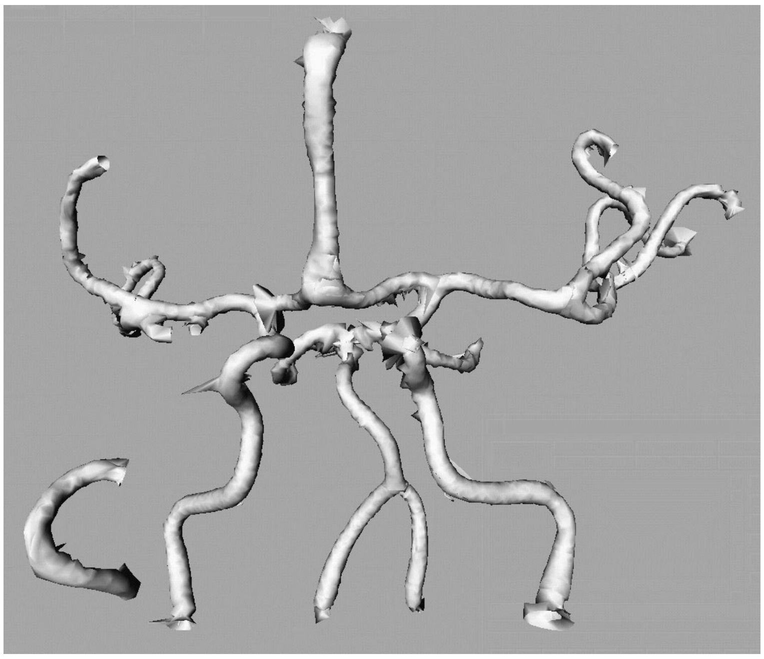


Figure 4.45: Corrected perimeter curves with outlier removal extracted from an MR angiogram of a human head.

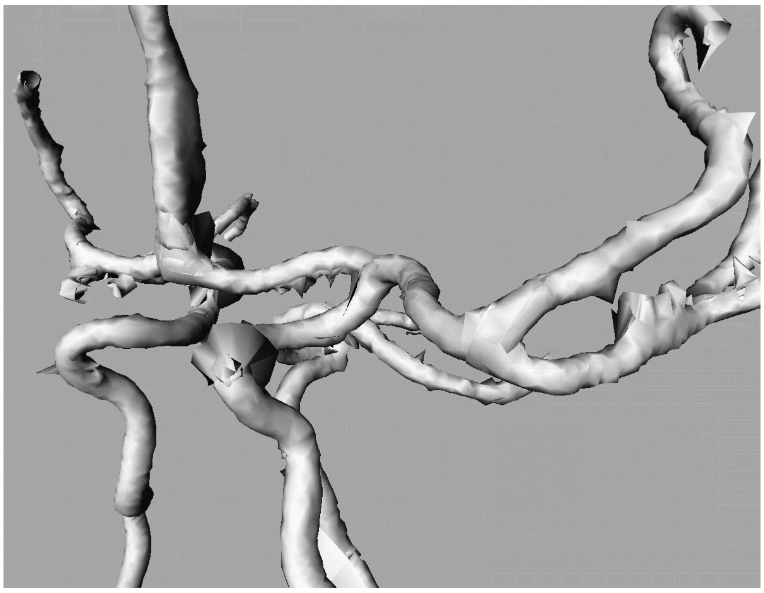
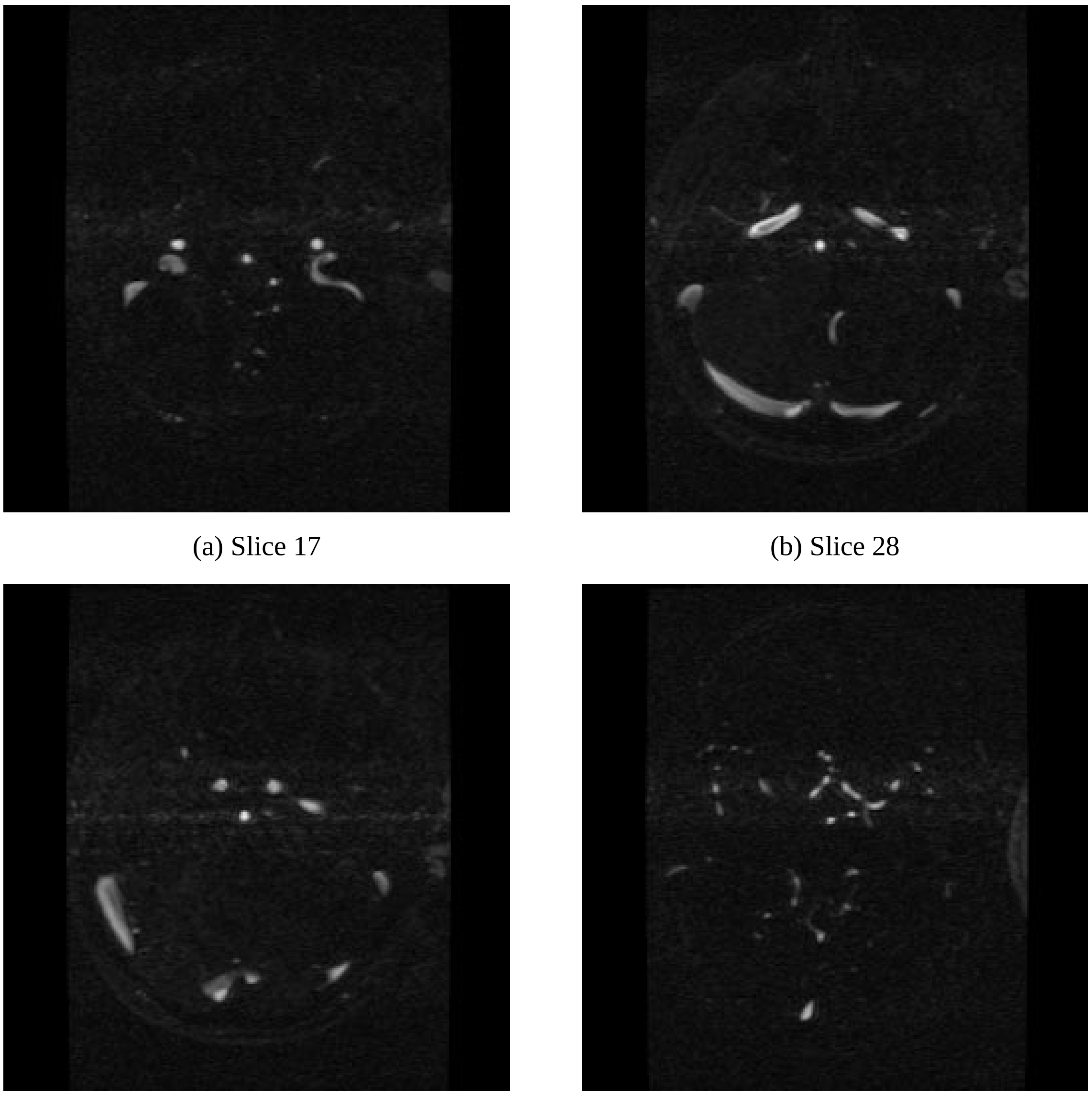


Figure 4.46: Close-up view of corrected perimeter curves with outlier removal from an MR angiogram of a human head.

### 4.3.6 Examples

This section is concluded by giving two more examples of the results obtainable with the proposed approach to 3D line extraction.


(a) Slice 17 (b) Slice 28

(c) Slice 33 (d) Slice 61

Figure 4.47: Four slices from an MR angiogram of a human head.

Figure 4.47 shows four slices of an MR angiogram of a human head. This image was obtained from the image archive of the United Medical and Dental Schools of Guy's and St. Thomas' Hospitals' Image Processing Group, London (UMDS-IPG). The extent of the dataset is $256 \times 256 \times 124$. As was done for the previous example, the dark borders of the image were cropped of to reduce the size of the dataset. Since no information was given about the inter-slice distance of the image, the aspect ratio of the image was left unchanged.

The results of extracting bright lines with $\sigma = 2$, thresholds of 0.5 and 1.75, and outlier removal are displayed in Figure 4.48. Since in this image arteries at the front of the head as well as veins at the back of the head are extractable, the resulting lines are shown from the left side

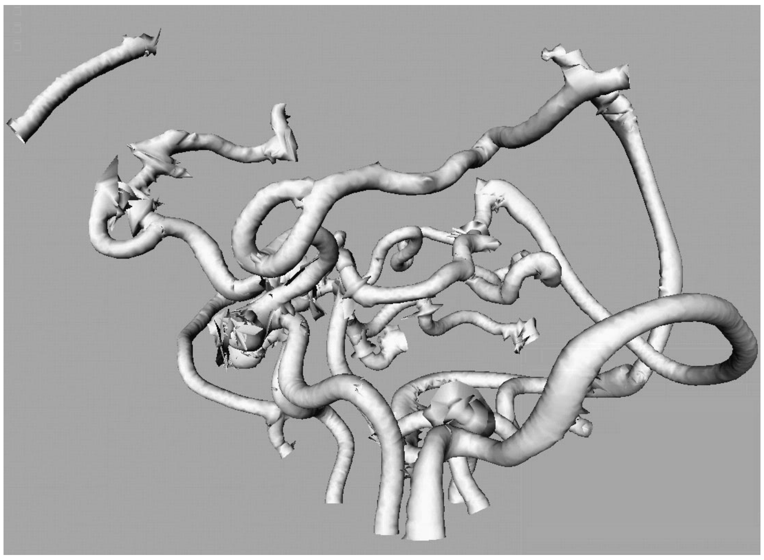


Figure 4.48: Lines and their widths extracted from an MR angiogram of a human head as seen from the left side of the head.

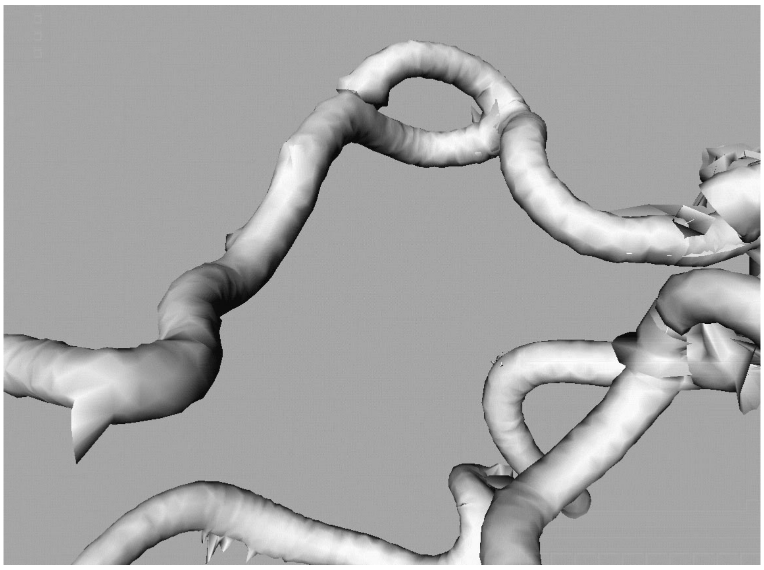


Figure 4.49: Close-up view of lines extracted from an MR angiogram of a human head.

of the head. The results are very good for most of the image. However, several outliers can still be observed in the central and upper left part of the image. Figure 4.49 displays the double line junction configuration slightly above and to the left of the center of Figure 4.48 from inside of the head looking forward and down. As can be seen, this complex junction configuration has been extracted in a topologically sound manner. Again, there is little variation in the extracted line widths, indicating that the estimated line widths are biased.

To get a better width estimation, the line perimeters were corrected using a localization $\sigma = 1$ with outlier removal. The resulting lines are exhibited in Figures 4.50 and 4.51. With

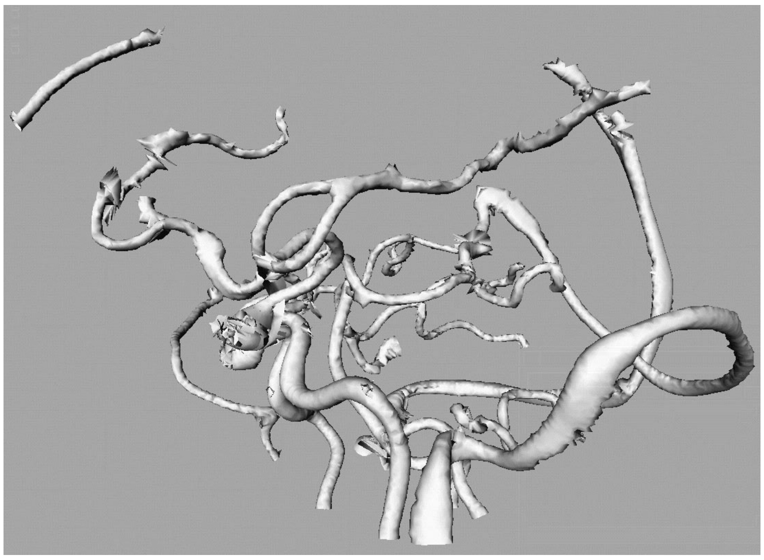


Figure 4.50: Lines and their corrected widths extracted from an MR angiogram of a human head.

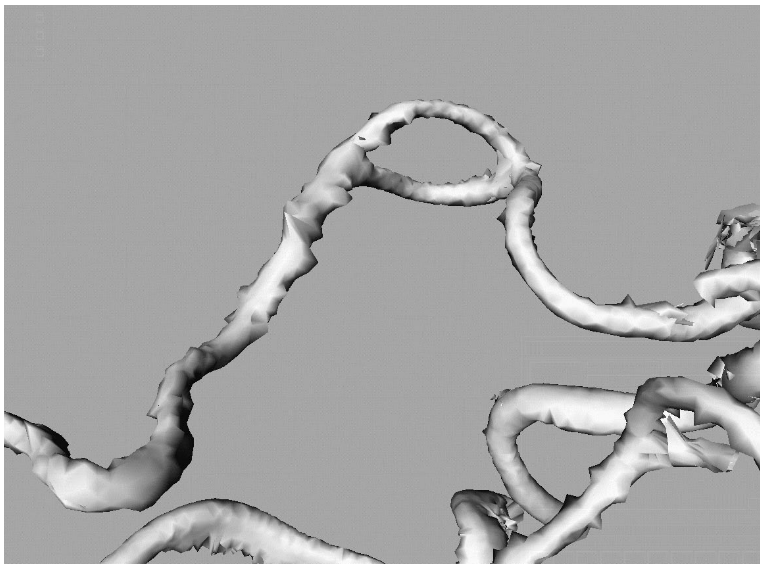
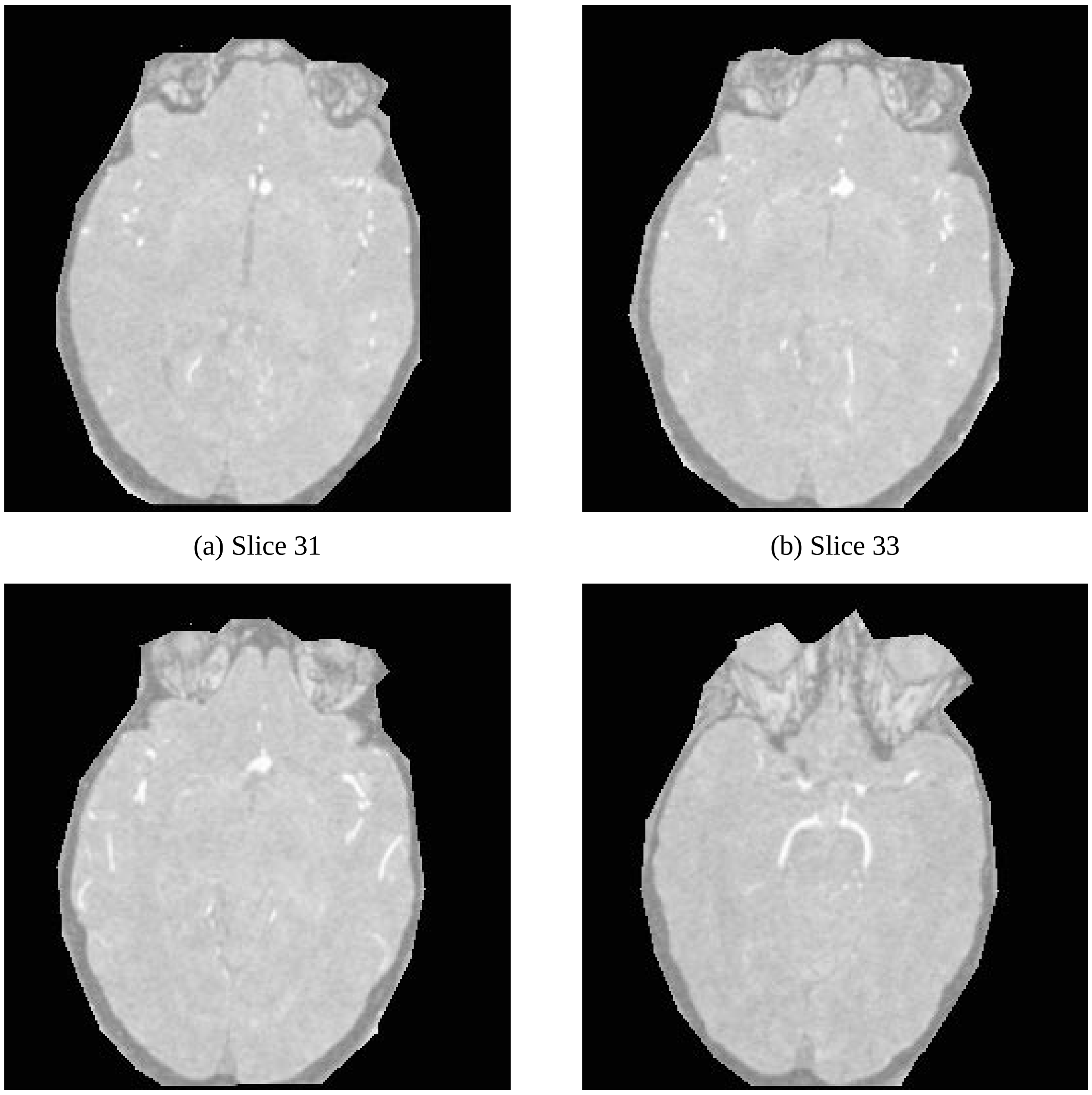


Figure 4.51: Close-up view of lines with corrected widths extracted from an MR angiogram of a human head.

this parameter, the line widths correspond very well with what they can be expected to be. Furthermore, there are relatively few remaining outliers. The complex junction configuration is again represented very soundly and precisely. Although the results are slightly worse in terms of smoothness and outliers than the ones given in Figure 4.45, they can still be rated very good.

Four slices of the input image of the final example are shown in Figure 4.52. This image was kindly provided by Dr. Stephen Aylward of the Department of Radiology at the University of North Carolina, Chapel Hill. It consists of $48$ slices of dimension $256 \times 256$ each, with the


(a) Slice 31 (b) Slice 33

(c) Slice 35 (d) Slice 44

Figure 4.52: Four slices from an MR angiogram of a human head. Note the aneurysm visible as large circular blobs in the upper central part of slices 31–35.

inter-slice distance being 1.6 times the pixel distance. Therefore, the dataset was scaled by this amount in the slice direction and the borders of the image were cropped off. Note that a coarse hand segmentation has been performed on the dataset by the originators of the image to remove the skull. This only has a slight impact on the extraction results. What is interesting about the dataset is the aneurysm which is clearly visible as large circular blobs in slices 31–35 displayed in Figure 4.52. It is highly desirable to extract this aneurysm very precisely for diagnostic purposes and surgical planning.

Figure 4.53 displays the results with $\sigma = 2$, thresholds of 0.5 and 1.75, and outlier removal.

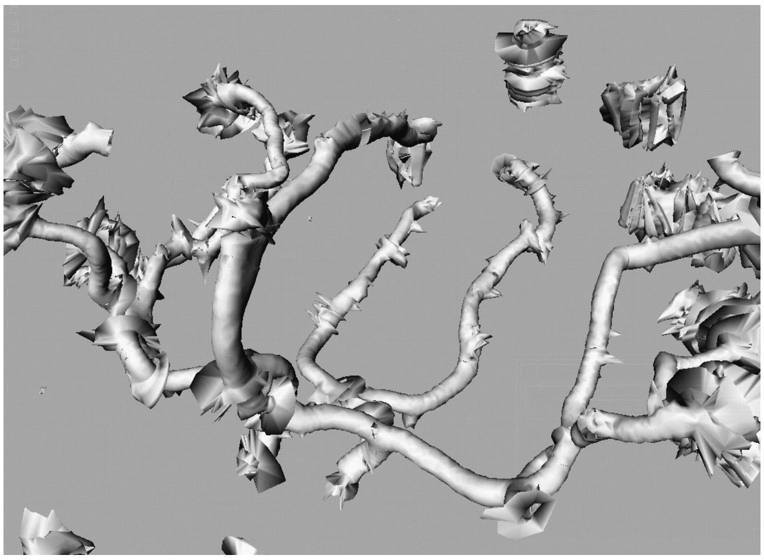


Figure 4.53: Lines and their widths extracted from an MR angiogram of a human head. The view is from the front left part of the head.

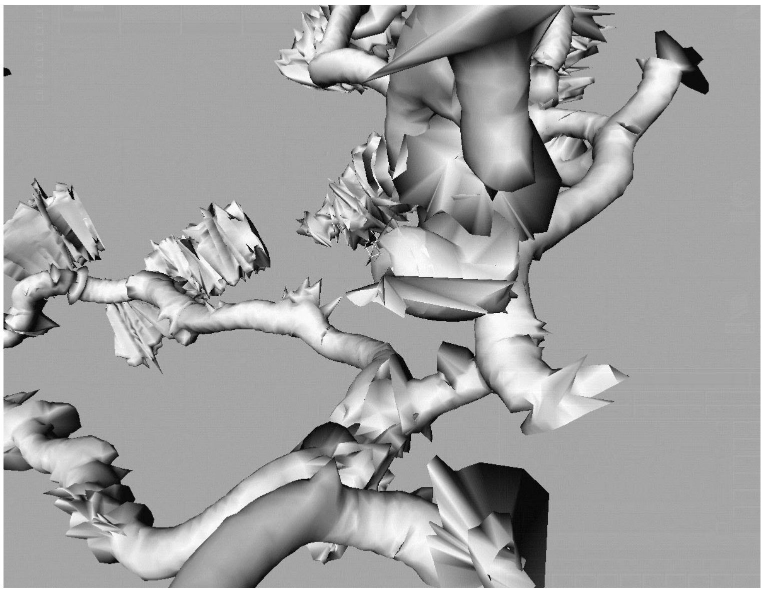


Figure 4.54: Close-up view of the aneurysm extracted from an MR angiogram of a human head. The view is from the right side of the head.

The view is from the front left part of the head looking downward. Because of the hand segmentation performed on the image, there are many areas in the dataset where parts of the bone and skin are visible next to parts where they have been cropped off, resulting in line-like objects in these parts of the image. Therefore, there is a great number of erroneously extracted lines at the borders of the image. These should be ignored while evaluating the results. Keeping this in mind, the extraction results are fairly good. The extracted lines and perimeters are very smooth

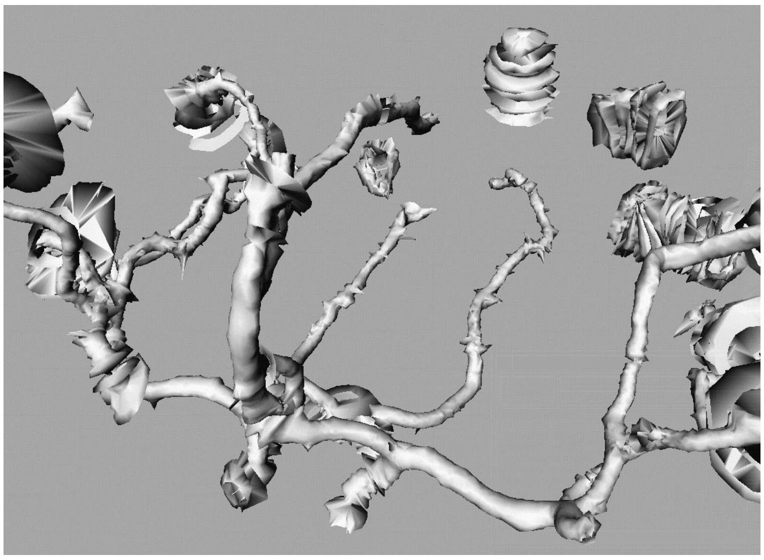


Figure 4.55: Lines and their corrected widths extracted from an MR angiogram of a human head. The view is from the front left part of the head.

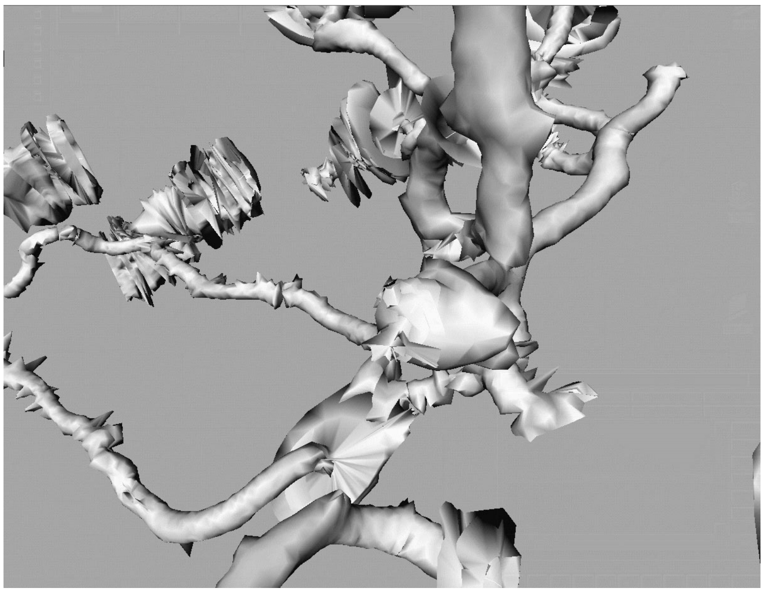


Figure 4.56: Close-up view of the aneurysm extracted with width correction from an MR angiogram of a human head. The view is from the right side of the head.

throughout the central part of the image. However, there are quite a few remaining outliers present. A view of the aneurysm from the right side of the brain looking towards the left side is given in Figure 4.54. Although the aneurysm has been extracted fairly well, the outliers in this area make it hard to discern its precise location.

The results of correcting the line widths using a localization $\sigma = 1$ are shown in Figure 4.55. There are relatively few remaining outliers, enabling a clear view of the aneurysm at the bottom of the artery in the left central part of the image. A detailed view of the aneurysm is displayed in Figure 4.56, where it can clearly be seen in the right center part of the image. Although the aneurysm has been extracted fairly successfully, the most notable failure of the line detection is that the artery emanating upwards from the aneurysm has not been connected with the aneurysm into a junction. This probably stems from the fact that the line width increases very sharply in this area, and hence the second derivatives of the image in this area are very low. Therefore, the restrictive thresholds set in this example in order to reduce the number of erroneous lines extracted at the border of the image hindered a completely successful extraction of the aneurysm. Nevertheless, the extraction results are fairly good overall, and might be used in surgical planning.

# Chapter 5

# Analysis of the Localization Quality

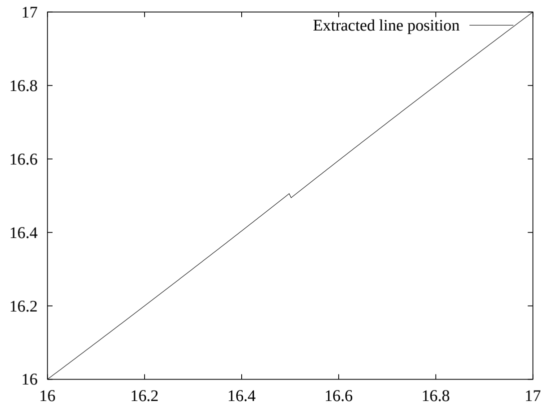
The analysis of the performance characteristics of a feature extraction algorithm is very important [56]. First, it makes an algorithm comparable to other algorithms, thus helping users in selecting the appropriate method for the task they have to solve. Second, it helps to identify breakdown points of the algorithm, i.e., areas where the algorithm cannot be used because some of the assumptions it makes are violated. Therefore, in this chapter an attempt is made to characterize the performance of the proposed line detection algorithm. Most of the experiments will be done using the two-dimensional version of the line extractor presented in Section 3.1.1, although some will involve the edge detector of Section 3.3. The derived results carry over immediately to the three-dimensional surface extraction algorithm, while it can be expected that they will, at least qualitatively, hold for three-dimensional line extraction as well. Since the line extractor is a very complex algorithm, Section 5.1 first shows the unbiasedness of the extraction results for noise-free synthetic images. In Section 5.2, the behavior of the line positions and widths for noisy synthetic images is analyzed and an analytical formula for the variance of the line positions under the influence of noise is derived. Finally, the quality of the line positions is tested for real images in Section 5.3.

## 5.1 Experimental Tests on Unbiasedness on Synthetic Images

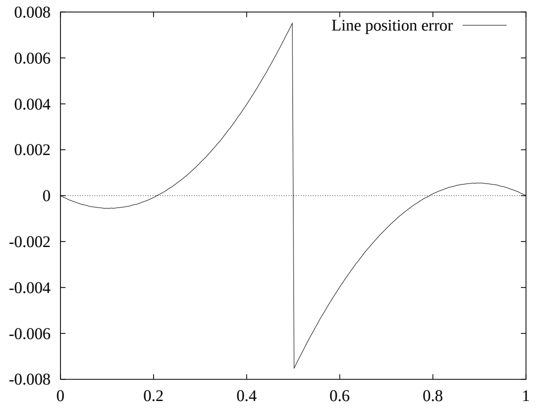
The line extractor proposed in Section 3.1 is a very complex algorithm, which has been derived for continuous images, but is applied to discrete images, which are quantized to a finite number of gray levels. For these reasons, it is not immediately obvious that the analytically derived unbiasedness of the algorithm also holds for discrete images. Therefore, in this section the performance of the algorithm will be tested on synthetically generated images to derive limits of applicability under controlled, best-case conditions.

### 5.1.1 Sub-Pixel Accuracy of Line and Edge Positions

First of all, the sub-pixel accuracy of the algorithm is checked. In order to isolate the influences of the various processing stages of the algorithm, this is first done without applying the width extraction and correction. The basic procedure adopted to test the sub-pixel accuracy is as



(a) Extracted line position



(b) Line position error

Figure 5.1: Extracted line position (a) and position error (b) for a line of total width 5.

follows: Lines of constant total width $w$ and contrast $h = 255$ are drawn into the center part of images of size $32 \times 32$. The lines are shifted in small increments by a total of one pixel along the direction perpendicular to the line. In order to allow sub-pixel shifts, the imaging device is assumed to integrate the incoming light intensity across the square area of each sensor, i.e., to possess a square aperture function. This is a frequently made assumption [121, 176], which often, at least approximately, holds in practice. Therefore, a line shifted by $x$ from a pixel position can be drawn by painting $w - 1$ pixels with the intensity $255$, one pixel to one side of the line with intensity $255x$, and one pixel to the other side of the line with intensity $255(1 - x)$.

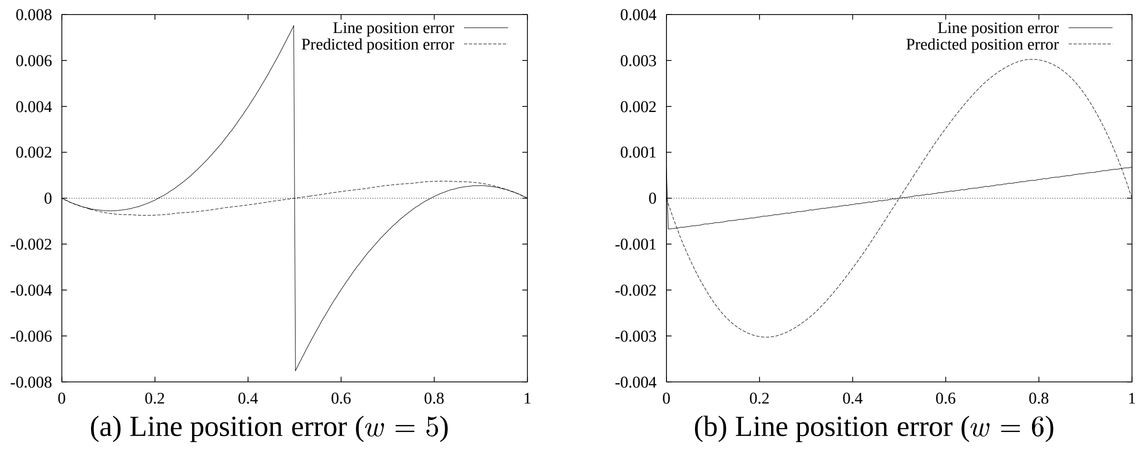
Figure 5.1(a) shows the extracted absolute line position for a line of total width 5, while Figure 5.1(b) shows the position error of the line. The value of $\sigma$ used for the extraction was the optimal $\sigma$ chosen according to (3.15), i.e, $5/(2\sqrt{3})$. There is a maximum position error of about 0.0075 pixels, i.e., less than one hundredth of a pixel. The error seems to contain a sinusoidal component. This might indicate that the error may be caused by the way the line shift is modeled. In order to test this hypothesis, the techniques developed in Section 3.1 can be used to model the discrete, sub-pixel shifted line in continuous space. From the above discussion it follows that a model for the discrete line is given by

$$f_d(x) = \begin{cases} 1, & |x| \le w \\ 1 - a, & -w - 1 \le x < -w \\ a, & w < x \le w + 1 \\ 0, & |x| > w + 1 \ . \end{cases} \tag{5.1}$$

In order to determine the line position of this line as $a$ varies, the first derivative of the model line convolved with a Gaussian kernel is needed. It is given by

$$r'_d(x, \sigma, a, w) = (1 - a)(g_\sigma(x + w + 1) - g_\sigma(x - w)) + a(g_\sigma(x + w) - g_\sigma(x - w - 1)) \ . \tag{5.2}$$

Figure 5.2(a) shows the predicted and extracted line position error superimposed in one diagram. As can be seen, the sinusoidal component of the position error can be explained from the discrete approximation of the sub-pixel-shifted line. What remains is a roughly linear error which is largest for sub-pixel line positions lying close to the boundary of two pixels. This error stems from the fact that the algorithm extrapolates the line position from the Taylor polynomial at the pixel centers. Since the model line convolved with a Gaussian cannot be approximated exactly by a quadratic polynomial this error is to be expected.



(a) Line position error ($w = 5$) (b) Line position error ($w = 6$)

Figure 5.2: Extracted and predicted line position error for a line of total width 5 (a) and 6 (b).

The next question that arises is how the localization error depends on the line width. Experiments with total line widths varying from 2 to 9 have been carried out to test this. For odd line widths the results of these tests are qualitatively similar to Figure 5.2(a). The maximum position error increases for smaller line widths, while it decreases for larger line widths. This is to be expected since for small widths the extrapolation is done from positions far away from the line position relative to the line width, therefore increasing the extrapolation error, and vice-versa for large line widths. For $w = 3$ the maximum error is approximately 0.05 pixels, while for $w = 9$ it is roughly 0.0008 pixels, i.e., less than one thousandth of a pixel, and therefore less than the 255 gray levels used to model the position shift. For even line widths an interesting behavior occurs. Figure 5.2(b) shows the line position error for a line of width 6. As can be seen, the sinusoidal error term expected from the model is absent in the extraction results. Only the linear extrapolation error is present. Note that for even line widths the line positions are at the pixel boundaries if the line is not shifted. Therefore, the maximum extrapolation error occurs at the borders of the diagram. A similar behavior occurs for all other even line widths, with the maximum error decreasing as $w$ increases. Here, the inexplicable fact that the extraction results are better in the discrete space than the analytical model predicts occurs.

Another question is the influence of the choice of $\sigma$ on the positional error. In order to model this behavior, line extraction was done by multiplying the optimal $\sigma$ according to (3.15) with a sigma factor $\sigma_f$. The results of choosing $\sigma$ too small, i.e., $\sigma_f < 1$, are shown in Figure 5.3(a), while the effects of selecting $\sigma$ too large are displayed in Figure 5.3(b). As can be seen, the maximum position error increases sharply if $\sigma$ is chosen 20% too small, leading roughly to a sevenfold increase of the position error. If $\sigma$ is chosen moderately too large, the maximum position error increases. The worst increases are achieved for $\sigma$ 30–50% too large, where nevertheless the maximum error is never more than twice the error at the optimal $\sigma$. After this, the maximum error decreases again, eventually being smaller than the error at the optimal $\sigma$. These results give another very good reason to choose $\sigma$ according to (3.15).

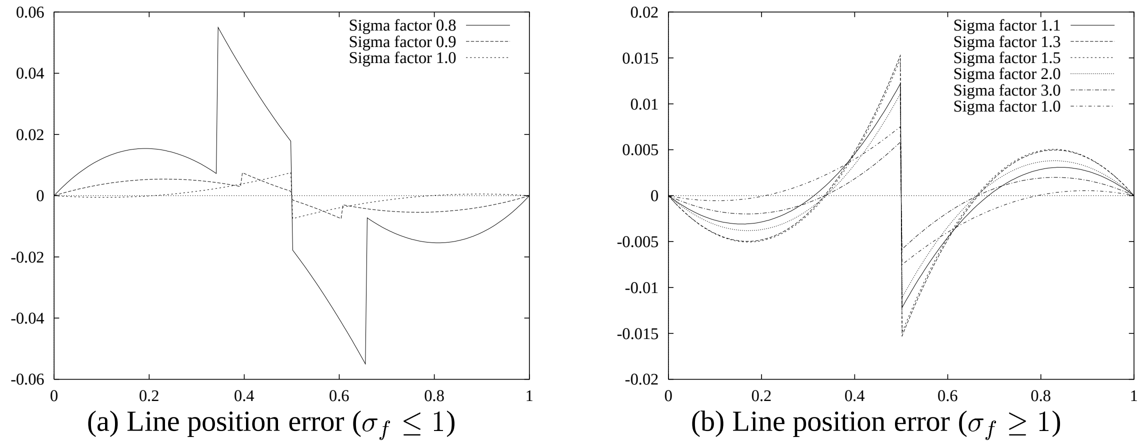


Figure 5.3: Extracted and predicted line position error for a line of total width 5 with $\sigma$ chosen too small (a) and too large (b).

The final question is the influence of the width correction on the position error. Figure 5.4 displays the line position error for a line of width 5 if the extraction is done using the line width and position correction described in Section 3.1.10. As can be seen, the correction introduces another small error. This error is largest for line positions close to the center of a pixel, and therefore has no effect on the maximum positional error. The maximum correction error is approximately 20% of the maximum overall positional error, as can be seen by comparing Figures 5.2(a) and 5.4.

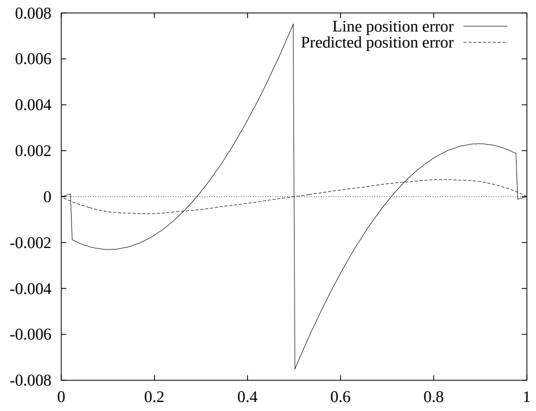
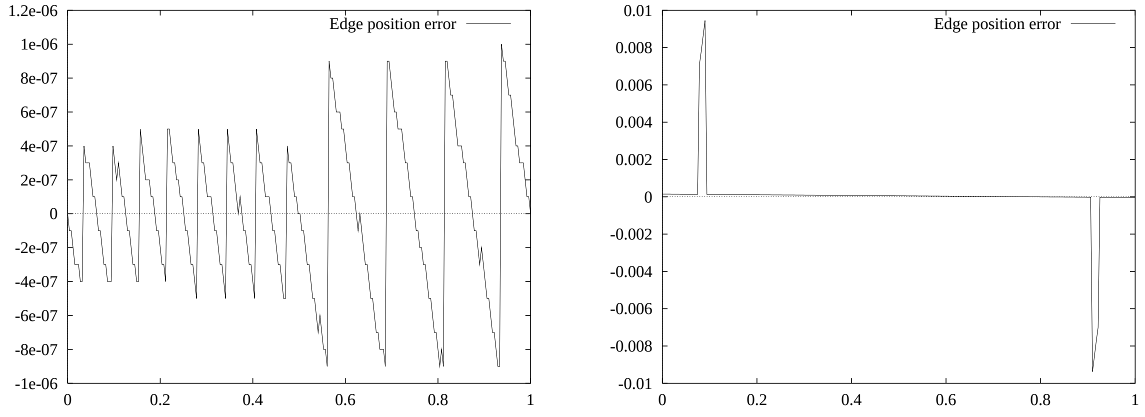


Figure 5.4: Extracted and predicted line position error for a line of total width 5. The extraction included the line width and position correction.

While discussing the subject of position errors, the edge location errors occurring for the edge extraction algorithm proposed in Section 3.3 should be analyzed. As was done above for lines, the sub-pixel displacement will be modeled by appropriately modifying gray values. More specifically, the test images are generated by drawing the upper half of a $32 \times 32$ image with gray value 0, the lower half with gray value 255, and the central row with a gray value varying from 0 to 255. Figure 5.5(a) displays the edge location error if edges are extracted using Gaussian derivatives with $\sigma = 1$. As can be seen, the errors are of the magnitude of the floating point representation used to store the edge locations. The results of extracting edges using the isotropic version of the Deriche operator [103] with $\alpha = 0.565$, corresponding to $\sigma = 1$, are shown in Figure 5.5(b). Surprisingly, there are two small areas where the error climbs to almost one hundredth of a pixel. This may be caused by the numerical problems in the recursive implementation of the filter. In any case, for the synthetic images used here the edge positions can be considered perfect.



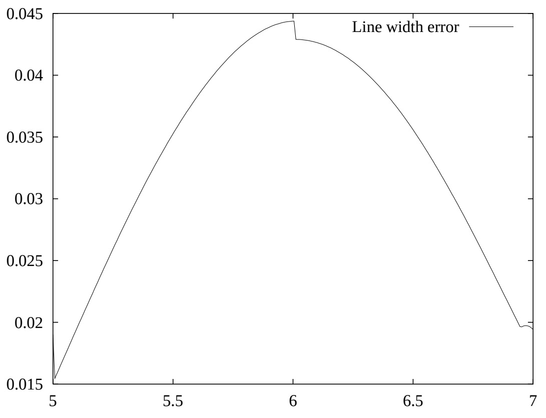
(a) Edge position error (Gauss) (b) Edge position error (Deriche)

Figure 5.5: Edge location errors for edges extracted with Gaussian derivatives (a) and the isotropic Deriche operator (b).

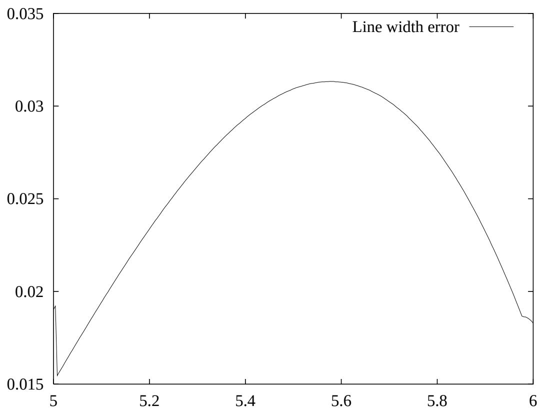
### 5.1.2 Sub-Pixel Accuracy of Line Width

The second most important feature the line extraction algorithm returns is the line width. Since the line width is generally biased, it is only useful to check the extraction results for sub-pixel accuracy after applying the line width and position correction. In order to do so, lines of total width $w$ are gradually widened to a total width of $w + 2$ by increasing the gray value on each side of the $w$ pixels wide line from 0 to 255. As was done for the assessment of the accuracy of the line position, the line extraction is done with the optimal $\sigma$, which for this test increases linearly from $w/(2\sqrt{3})$ to $(w+2)/(2\sqrt{3})$. The error of the corrected line width results for a line with $w = 5$ is shown in Figure 5.6(a). As can be seen, the width error is less than one twentieth of a pixel for this range of line widths. As was the case for the line position, the maximum width error increases as $w$ decreases, up to a maximum error of approximately 0.17 pixels for $w \approx 2.7$. For increasing $w$, the width error again decreases, but contrary to the line position error, the width error asymptotically seems to level out at one thirtieth of a pixel for large $w$.

In the above test, the line width was enlarged symmetrically on both sides of the line. To check whether an asymmetrical enlargement influences the accuracy of the line width, in a second test the line width was enlarged on one side of the line from $w$ to $w + 1$, and lines were extracted using the optimal value of $\sigma$. The resulting line width errors are shown in Figure 5.6(b). It can be seen that the asymmetry causes no new error to be introduced. In fact,



(a) Line width error (two-sided)



(b) Line width error (one-sided)

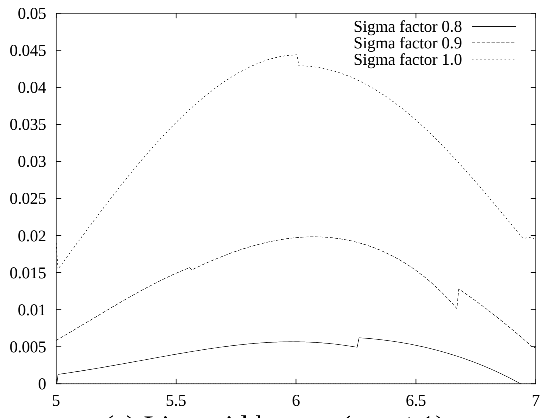
Figure 5.6: Extracted line width error for a line of total width varying from 5 to 7 with two-sided enlargement of the line width (a) and for a line of total width varying from 5 to 6 with one-sided enlargement of the line width (b). The extraction included the line width and position correction.

the errors are smaller than for the two-sided enlargement. As was the case for the line position, the width error is largest if the edge positions lie exactly between two pixel positions.

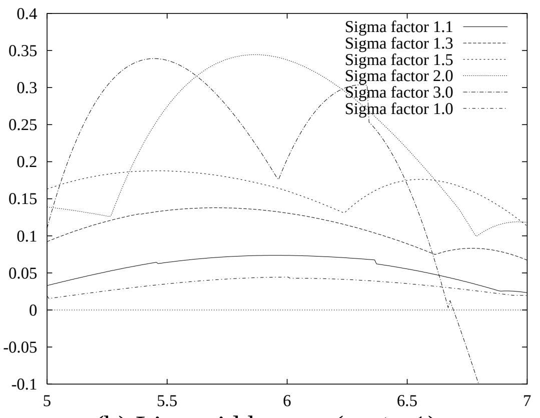
As was the case for the line position error analysis, it is important to determine the influence of choosing $\sigma$ too small or too large for a given width. This is only done for the two-sided enlargement. As for the line positions, this is modeled by extracting the width with the optimal $\sigma$ scaled by a factor $\sigma_f$. The results for choosing $\sigma$ too small are displayed in Figure 5.7(a). As was to be expected from the discussion in Section 3.1.10, the line width error decreases when $\sigma$ is too small because the inputs to the correction step are in the linear part of the correction functions displayed in Figure 3.27, and because the two line edges have less influence on each other for small values of $\sigma$. If $\sigma$ is chosen too large, the line width errors increase steadily up to a maximum error of 0.35 for $\sigma_f = 3$, as Figure 5.7(b) shows. This can be attributed to the fact that by choosing $\sigma$ too large, the inputs to the correction step are moved to the relatively non-linear part of the correction table. Therefore, the error incurred by the bilinear interpolation used to interpolate the values in the correction table leads to a larger error in the corrected line widths. Note, however, that the width error is never larger than 6.5% for this example. Qualitatively similar results can also be observed for different line widths $w$.

### 5.1.3 Line Orientation and Asymmetry

One feature which requires a thorough analysis is the extracted line orientation. Because all modeling in Section 3.1 was done for one-dimensional line profiles it is not immediately obvious that the line orientation extracted from a two-dimensional discrete image will return meaningful results. In order to test the accuracy of the extracted line orientations, lines of total width $w$ forming an angle of $\alpha$ with the coordinate frame are drawn as if they were viewed through an optical system with a square aperture function, i.e., the intensity of a pixel is proportional to the area of the pixel covered by the line. As customary, lines are extracted with the optimal $\sigma$ for the respective line width. The resulting error of the extracted line orientations is shown in degrees in Figure 5.8 for a line of total width $w = 5$. For this line width the angle error is



(a) Line width error ($\sigma_f \leq 1$)



(b) Line width error ($\sigma_f \geq 1$)

Figure 5.7: Extracted line width error for a line of total width varying from 5 to 7 with $\sigma$ chosen too small (a) and too large (b).

never larger than one quarter of a degree, i.e., 15 minutes of arc. For smaller line widths the maximum angle error can be expected to increase, and this is indeed the case. Even for lines of total width 2 the error is rarely larger than one degree. For increasing line widths, the maximum orientation error decreases. For example, for lines with $w = 9$ it is approximately one tenth of a degree.

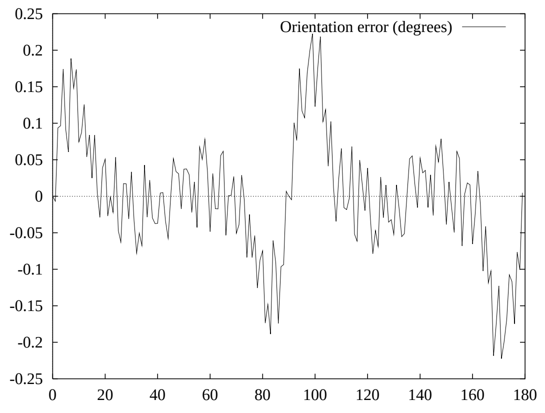


Figure 5.8: Extracted line orientation error in degrees for a line of total width 5. The extraction included the line width and position correction.

After having analyzed the accuracy of the extracted line orientation, it is important to know whether the line orientation influences the extracted line positions and widths. Figure 5.9(a) displays the line position error as a function of the line orientation. Clearly, there is no systematic dependency of these two features. The maximum line position error incurred by rotating the line is less than one fortieth of a pixel. For smaller line widths the maximum error can increase to one twentieth of a pixel, while for large line widths it decreases steadily. For example, the maximum error for a line of total width 9 is less than 0.003 pixels. For the line widths there is also no systematic dependency on the line orientation, as Figure 5.9(b) indicates. The extracted widths are on average too large by approximately 0.03 pixels, or 0.6%. For small line widths the maximum width error can increase to one seventh of a pixel, while for large $w$ it seems to level out at roughly one fiftieth of a pixel.

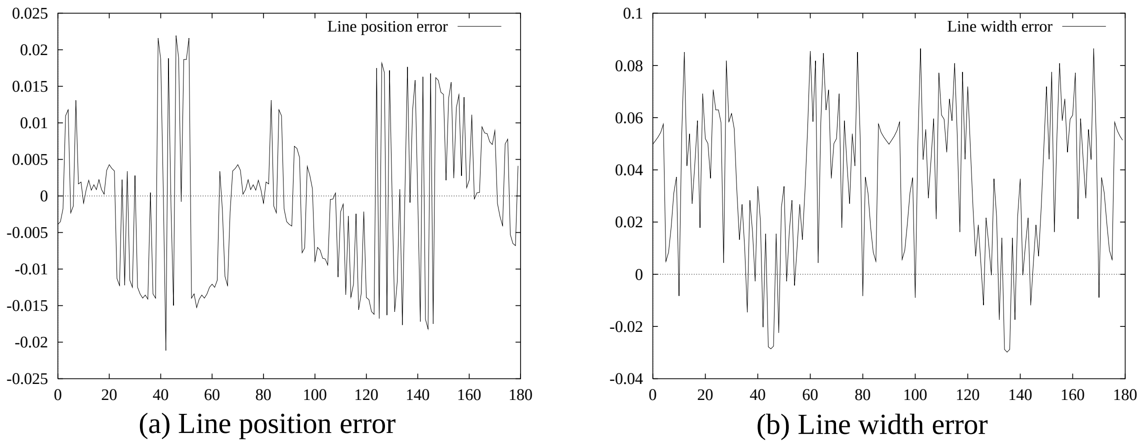
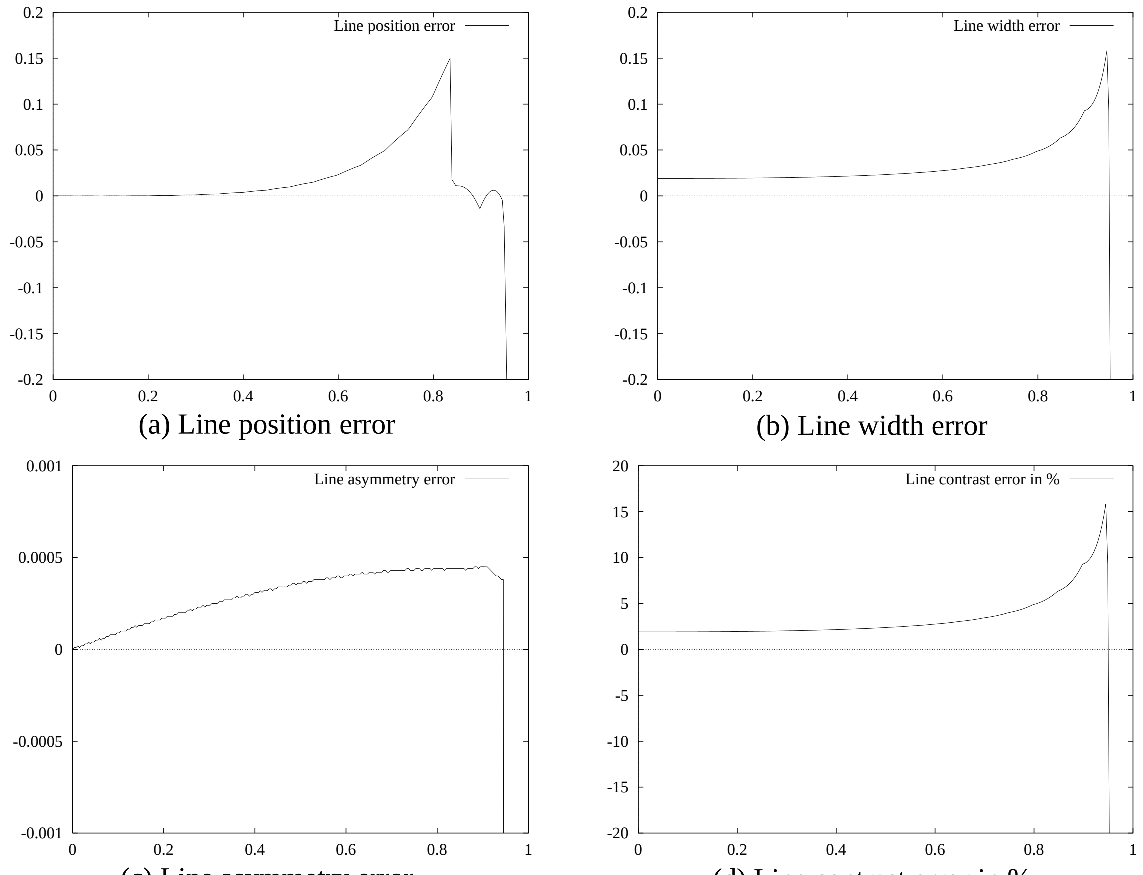


Figure 5.9: Extracted line position (a) and width error (b) for a line of total width 5 as a function of line orientation.

Up to now, only symmetrical lines have been considered. Therefore, the final test of the accuracy of the line extraction algorithm is to check the dependency of various extraction errors on the asymmetry of a line. In order to perform this check, lines of constant width $w$ are drawn with intensity 255. The gray value on one side of the line is kept constant at 0, while the gray value on the other side of the line is varied from 0 to 255. The total width $w$ is restricted to integer values because the analysis above shows that only very small errors are introduced by sub-pixel variations of line position and width. With this being said, Figure 5.10(a) displays the line position error as a function of the true asymmetry of the line. The error is less than 0.15 pixels for asymmetries up to 0.95, less than one tenth of a pixel for asymmetries up to 0.75, and less than one twentieth of a pixel for asymmetries up to approximately 0.65. Very similar results hold for the line width error, as Figure 5.10(b) shows. Again, the line width error is less than 0.15 pixels for all asymmetries less than 0.95, less than one tenth of a pixel for asymmetries less than 0.85, and less than one twentieth of a pixel for asymmetries up to 0.75. The extracted asymmetry of the line is never more than 0.05% different from the true asymmetry for asymmetries less than 0.95, as displayed in Figure 5.10(c). Finally, the contrast extracted according to (3.52) is also less than 15% away from the true contrast of the line for almost all asymmetries, as can be seen from Figure 5.10(d). Interestingly enough, the distribution of the errors across the range of asymmetries is roughly the same as for the line widths. As $w$ is varied, the following observations can be made: The extracted line position error is less than one tenth of a pixel for all asymmetries less than 0.8 for all $w$, with the range of asymmetries for which this is the case increasing for increasing $w$. The width estimation error is less than one tenth of a pixel for all $w \geq 3$ and for all asymmetries smaller than 0.85. Again, the asymmetry range increases slightly for larger $w$. For $w = 2$ the width error is less than 0.2 pixels for all asymmetries smaller than 0.8. The extracted asymmetry error is less than 0.01 for all line widths and true asymmetries smaller than 0.95, while it is less than 0.001 for $w \geq 5$. The extracted



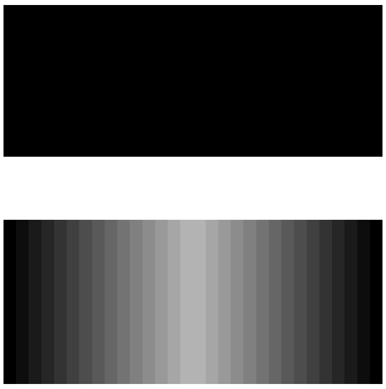
(a) Line position error (b) Line width error

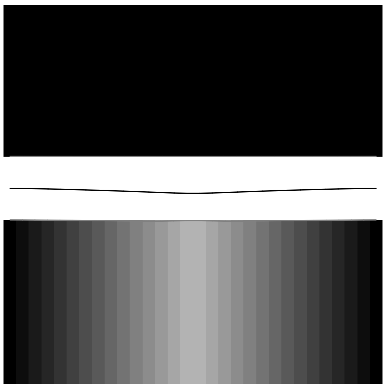
(c) Line asymmetry error (d) Line contrast error in %

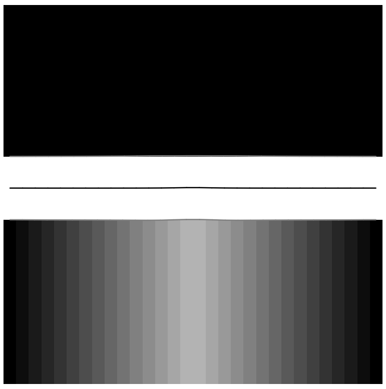
Figure 5.10: Extracted line position (a), width (b), asymmetry (c), and contrast error (d) for a line of total width 5 as a function of true line asymmetry.

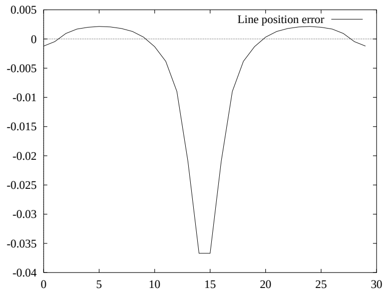
contrast is within 10% of the true contrast for asymmetries smaller than 0.75 and $w \geq 4$. For smaller line widths it is within 20% of the true contrast in this range of asymmetries.

The tests so far have assumed that the asymmetry is constant along the length of the line. If this is not the case, the line position and orientation will vary along the length of the line. Since this may have consequences on the extracted and corrected line positions, the concluding experiment of this section consists of an image in which the asymmetry of the line is varied from 0 to 0.7 and back to 0 along the line. This image is shown in Figure 5.11(a), while the uncorrected and corrected line extraction results are displayed in Figure 5.11(b) and (c), respectively. The corrected extraction results are very good, and no bias is apparent. From Figure 5.11(d) it can be seen that the maximum position error is less than one twenty-fifth of a pixel. However, as Figure 5.11(e) shows, the varying asymmetry along the line leads to biased estimations of the line orientation. The maximum angle error is approximately $1.5°$. Since the line orientation cannot be corrected in the process of correcting the line position and width, this error leads to the line width being returned too large because it is measured in a direction which is not perpendicular to the true line. From Figure 5.11(f) it can be seen that the maximum error incurred is approximately 0.11 pixels. This leads to the paradoxical situation that the extracted line widths are returned with errors although the corresponding edge positions are located very

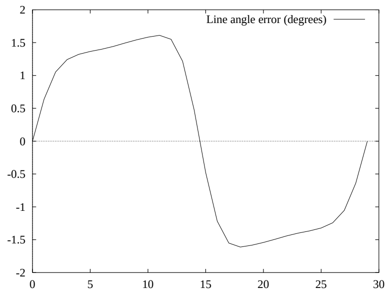

(a) Input image


(b) Uncorrected lines

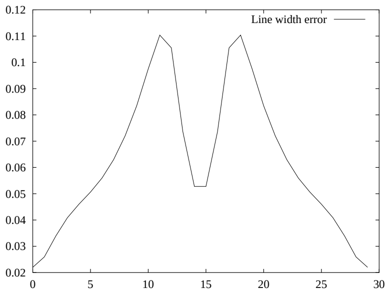

(c) Corrected lines



(d) Line position error



(e) Line angle error



(f) Line width error

Figure 5.11: Line with asymmetry varying from 0 to 0.7 (a). Extracted uncorrected (b) and corrected lines (c). Corrected line position (d), orientation (e), and width error (f). The width error is measured in the direction of the slightly wrong line orientation leading to seemingly erroneous line width estimations.

accurately. Therefore, in applications where precision is of major importance, the edge locations should be used as an additional criterion to measure the width of an object if the asymmetry can be expected to vary greatly in relatively short segments of the line.

In summary, from the analysis presented in this section it can be seen that the line extraction results, namely the position, width, angle, asymmetry, and contrast, can be said to be unbiased for a very large range of asymmetries and line widths. If one tenth of a pixel is allowed as the maximum position and width error, lines with an asymmetry of up to 0.75 are extracted unbiased for all line widths. If the maximum error allowed is increased to one fifth of a pixel, almost all combinations of widths and asymmetries of up to 0.95 lead to unbiased extraction results. In summary, the algorithm can be given the label “unbiased” in good faith for the synthetic, noise-free test images used in this section.

## 5.2 Quality of the Localization of Line and Edge Points

The tests in the previous section have been done on synthetic, noiseless images. For real images, however, noise, i.e., random fluctuations of the image intensity, play a significant role. Noise enters the image in different stages of the imaging process, e.g., through the photon flux on the sensor or the quantization of the image intensity to a finite number of gray levels. The noise

$n(x)$ can be modeled as a random process in one dimension and as a random field in higher dimensions. Thus, the observed version of the image is given by

$$i(x, y) = f(x, y) + n(x, y) \ . \tag{5.3}$$

The noise component $n$ is assumed to be wide-sense stationary, statistically independent of the image content $f$, and white, i.e., its expected value $E(n) = 0$ and its autocorrelation $R(\tau) = \sigma_n^2 \delta(\tau)$, where $\sigma_n^2$ is the variance of the noise [146].

Obviously, if noise is added to an image the extracted line or edge positions, and in general any extraction result, such as the line widths, will not be the correct values that would have been obtained from the uncorrupted version of the image. Therefore, one of the major performance criteria of an algorithm is the dependence of the extracted features on the input noise level. Put into different terms, what needs to be determined is how the variance of the line position and width and the variance of the edge position varies as the variance of the noise in the image varies. Ideally, if this dependency has been determined the algorithm could return an estimate of the variance of the respective feature, e.g., the line position, if given an estimate of the variance of the image noise. The variance $\sigma_n$ of the image noise can, for example, be determined by automatic procedures from an input image [23, 54]. Such a variance estimate can serve as an aid for self-diagnosis [56], e.g., to determine if certain requirements of the output data can be met. For example, if an application requires one tenth of a pixel accuracy in the line or edge position, and the returned standard deviation of the line position is larger than some fraction of this amount, an algorithm can alert the user to the fact that the lines cannot be extracted with the required precision. The user could then try to fix the problem, e.g., by changing the lighting conditions or by using a different sensor to reduce the noise level in the images.

### 5.2.1 Previous Work on Localization Quality

Over the years, several attempts have been made to characterize the quality of the extracted feature positions. Most of them were made for edge detectors. Often, these quality measures are made in connection with deriving an "optimal" operator with respect to several criteria, one of which often is a term that tries to measure the goodness of edge locations. One of the earliest approaches to this problem is introduced in [29], where an optimal one-dimensional edge detector with respect to three criteria is derived: good detection, i.e., the algorithm should have a high probability of extracting true edge points, while having a low probability of extracting false edge points, good localization, i.e., the extracted edge points should be as close as possible to the true edge points, and finally having only a single response to an edge. The optimal operator is described by a convolution of the image with a kernel of finite width $2w$. It is assumed to yield a maximum response at the edge location, i.e., the first derivative of the filter response at the edge is zero. With this, the localization criterion can be derived as follows: Assume the response of convolving the image $i$ with the edge operator $e$ to be given by

$$r(x) = h(x) * i(x) = h(x) * f(x) + h(x) * n(x) = r_f(x) + r_n(x) \ . \tag{5.4}$$

Then, due to the image noise $n$, the extracted edge location will lie at a point $x_e$, where $r'(x_e) = 0$. Thus, $r'_n(x_e) + r'_f(x_e) = 0$. If this equation is expanded into a Taylor series of order 1 about the origin, one obtains $r'(x_e) = r'(0) + r''(0)x_e + O(x_e^2)$. Since the model edge is located at 0, by assumption $r'(0) = 0$, and hence $x_e \approx -r'_n(x_e)/r''(0)$. In [29] it is now claimed that $r'_n(x_e)$ is a Gaussian random variable with variance

$$E(r'_n(x_e)^2) = \sigma_n^2 \int_{-w}^{w} f'(x)^2 \, dx \ . \tag{5.5}$$

Thus, the variance of the edge position is given by

$$E(x_e^2) = \frac{\sigma_n^2 \int_{-w}^{w} f'(x)^2 \, dx}{\left( \int_{-w}^{w} h'(-x) f'(x) \, dx \right)^2} \ . \tag{5.6}$$

These continuous criteria for an optimal edge detector are adapted for pixel-precise discrete edge operators in [37], where mainly the formulation of the single response criterion changes.

This definition of the localization quality measure was later criticized for being incorrect [178]. The major objection put forth there is that (5.5) only holds if $x_e$ is constant for all realizations, which, of course, is not the case. Another objection is that the localization criterion in [29] only takes into account the first zero crossing of $r'(x)$, whereas it should take into account all zero crossings of the response. Therefore, the physical density of the zero crossings, i.e., the expected number of zero crossings per unit time, of the edge detector applied to a noisy step edge is derived in [178]. This density is given by

$$\mu(x) = \frac{1}{2\pi} \sqrt{-\frac{R''_{n'}(0)}{R_{n'}(0)}} \exp(-f(x)^2 / 2\sigma_{n'}^2) \ . \tag{5.7}$$

The authors note that $1 - \mu(x)$ is a measure for the suppression of the density of zero crossings by the edge operator, and go on to prove that the first derivative of the Gaussian kernel is the unique function that optimizes this criterion. With this criterion, in effect the localization and single response criteria used in [29] are combined into a single criterion, yielding a better optimality criterion for edge operators. Of course, $\mu(x)$ is not very useful to characterize the dependency of the localization of edges on the image noise, precisely for the fact that all zero crossings are taken into account.

For this reason, an attempt is made in [96] to derive the true variance of the edge position. The authors note that in addition to the problems mentioned in [178], i.e., the wrong derivation of $E(r'(x)^2)$, the Taylor expansion of the edge location should be done differently. The authors claim that the correct Taylor expansion is given by $r'(x_e) = r'_n(0) + (r''_f(0) + r''_n(0))x_e$, and hence $x_e \approx -r'_n(0)/(r''_f(0) + r''_n(0))$. They then go on to derive the probability density of $x_e$ and arrive at the result that it is given by a sum of two Cauchy densities. Therefore, they note, the variance of $x_e$ does not exist. Of course, in this case the mean of $x_e$ also would not exist. What this obviously erroneous result implies is that if even the tiniest amount of noise is added to an image, the edge locations would fluctuate wildly all over the image, which is, of course, not supported by experimental evidence. If this were true, edges could in effect not be extracted at all because their positions would be meaningless.

A different approach is given in [5], where the imaging process is modeled in several steps. First, the ideal continuous image is assumed to be blurred by the imaging device with a Gaussian kernel with standard deviation $\sigma_b$. Then, the image is sampled and noise with standard deviation $\sigma_n$ is added to the image. Finally, edge extraction is done by smoothing the image with a Gaussian kernel with standard deviation $\sigma_e$ and calculating the gradient of the resulting image. The authors derive the following formula for the variance of an edge of height $h$, if the edge extraction is performed in a direction forming an angle $\alpha$ with the direction perpendicular to the edge:

$$E(x_e^2) = \frac{3\sigma_n^2(\sigma_b^2 + \sigma_e^2)^3}{8h^2\sigma_b^6 \cos^6 \alpha} \; . \tag{5.8}$$

Although the authors claim that their formula is supported by experimental evidence, this would imply that the variance of the edge locations increases as the standard deviation of the Gaussian used for edge extraction is increased. However, as is obvious from the discussion in Section 2.1, larger values of $\sigma_e$ lead to less noise being present in the smoothed image. Therefore, one should expect the edge location variance not to increase as $\sigma_e$ is increased, at least for the ideal step edges assumed in [5], and hence it is questionable whether (5.8) correctly describes the variance of the edge positions.

Contrary to this, edge detection is regarded as template matching in [56]. In this case, the variance of the edge position is given by

$$\sigma_e^2 = \frac{\sigma_n^2}{\displaystyle\sum_{r=-w}^{w} \sum_{c=-w}^{w} f_u^2(r,c)} \; , \tag{5.9}$$

where the template is of size $(2w+1) \times (2w+1)$ and $f_u(r,c)$ is the derivative of the template in the direction perpendicular to the edge. Furthermore, the variance of the edge direction can be derived as

$$\sigma_\phi^2 = \frac{\sigma_n^2}{\displaystyle\sum_{r=-w}^{w} \sum_{c=-w}^{w} v^2(r,c) f_u^2(r,c)} \; , \tag{5.10}$$

where $v(r,c)$ is the coordinate of the pixel along the edge. Unfortunately, since the edge and line detection algorithms proposed in this dissertation do not use template matching, these results do not apply directly to them.

### 5.2.2 Variance of the Zero Crossing of an Arbitrary Function

Because none of the above derivations of the variance of the edge position are applicable to the problem of determining the variance of the line position and width, and because some of them are erroneous even for the problem of determining the variance of the edge position, there is a need for an, at least qualitatively, correct formula that also is supported by experimental evidence. The algorithm of determining the line position in the one-dimensional continuous case can be broken down into two steps: the input signal is first convolved with the derivative of a Gaussian kernel, and then the zero crossings of the resulting response function $r'(x)$ are determined, where additionally $r''(x) < 0$ is required. For edges, the only change is that the zero crossings of the second derivative are determined. Since convolution with a Gaussian kernel is a linear operation, the response $r(x)$ of the line detection operator to the corrupted signal $i(x)$ is given by

$$r(x) = g_\sigma(x) * i(x) = g_\sigma(x) * f(x) + g_\sigma(x) * n(x) = r_f(x) + r_n(x) \tag{5.11}$$

and analogously for the corresponding derivatives. Thus, the response of the line detection operator to a model line profile $f(x)$ corrupted by wide-sense stationary white noise $n(x)$ can be broken down into a deterministic part $r_f(x)$ and a stochastic process $r_n(x)$. The deterministic part can, of course, be determined analytically, e.g., by (3.16)–(3.18) for the asymmetrical bar-shaped profile. Note that the deterministic part of the response has exactly one zero crossing of $r'(x)$ in $[-\infty, \infty]$. The random part of the response shifts this zero crossing and may introduce additional zero crossings of $r'(x)$. In order to determine the localization quality of the line detection algorithm, it is therefore useful to consider the variance of the zero crossing of $r'(x)$ closest to the zero crossing of the deterministic part $r'_f(x)$ of the response.

Since the noise $n(x)$ is assumed to be wide-sense stationary and white, the statistics of the response of the line detector to the noise can be calculated easily [14, 146, 175]. Because the mean value $E(n(x))$ is assumed to be zero, convolution of $n(x)$ with an arbitrary derivative of the Gaussian kernel is also a stationary stochastic process with mean zero. Furthermore, the autocorrelation function $R_r(\tau)$ of the noise smoothed with a Gaussian kernel is given by [146]

$$R_r(\tau) = \sigma_n^2 g_{\sqrt{2}\sigma}(\tau) \ , \tag{5.12}$$

while the autocorrelation function of the noise convolved with the first derivative of a Gaussian kernel is

$$R_{r'}(\tau) = -R''_r(\tau) = -\sigma_n^2 g''_{\sqrt{2}\sigma}(\tau) \ . \tag{5.13}$$

Thus, the variances of the respective stochastic processes are

$$\sigma_r^2 = R_r(0) = \frac{\sigma_n^2}{2\sqrt{\pi}\sigma} \tag{5.14}$$

$$\sigma_{r'}^2 = R_{r'}(0) = \frac{\sigma_n^2}{4\sqrt{\pi}\sigma^3} \ . \tag{5.15}$$

With this, the problem of determining the variance of the line position can be transformed to the equivalent problem of determining the variance of the zero crossing of an arbitrary function with exactly one zero crossing, e.g., (3.17), corrupted by a the addition of a stationary stochastic process with zero mean and a given autocorrelation function, e.g., (5.15). Without loss of generality, the zero crossing of the function can be assumed to be at the origin.

Before an approximation of the variance of a zero crossing with added noise will be derived, it is useful to look at this problem in a purely geometrical manner to get an intuitive notion of the dependency. Assume for the moment that the uncorrupted signal is simply the linear function $f(x) = hx$, and that the distribution of the noise $n(x)$ is uniform in some interval $[-m, m]$ for all $x$. Then, conceptually one can think of the corrupt signal $i(x)$ to be contained entirely within a "tube" of diameter $dy = 2m$ around $f(x)$, as shown in Figure 5.12. Therefore, the zero crossings must all lie in an interval of diameter $dx$ around 0. Obviously, $dy/dx = h$, and therefore the width of this interval is proportional to $1/h = 1/f'(0)$. Although the distribution of the zero crossings within the interval is unknown, the variance of the zero crossings can be expected to be proportional to $1/f'(0)^2$. Thus, it can be conjectured that for general functions the variance of the zero crossings is proportional to $\sigma_n^2/f'(0)^2$.

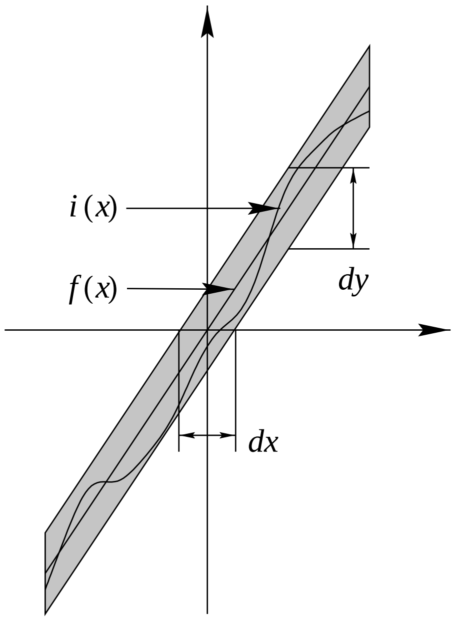


Figure 5.12: If the noise $n(x)$ is assumed to be uniformly distributed in $[-m, m]$ for all $x$ and $f(x) = hx$, the zero crossings of $i(x) = f(x) + n(x)$ must be contained in an interval of width $dx = dy/h$.

In order to prove this conjecture, a Taylor series expansion of order one of (5.11) can be performed, where for the moment it should be assumed that $r_f(x)$ and $r_n(x)$ are not obtained by convolution with a Gaussian, but are an arbitrary function and noise with a given autocorrelation function, respectively. Thus, $r(x_0) = 0$ for some $x_0$, and hence

$$r(x_0) \approx r(0) + r'(0)x_0 = r_f(0) + r_n(0) + (r'_f(0) + r'_n(0))x_0 \ . \tag{5.16}$$

By assumption, $r_f(0) = 0$, and hence

$$x_0 = -\frac{r_n(0)}{r'_f(0) + r'_n(0)} \ . \tag{5.17}$$

The calculation of the variance of $x_0$ from (5.17) is extremely difficult. However, if one assumes that the variance $\sigma_{r'}^2$ of $r'_n(0)$ is small with respect to the first derivative of the signal $r'_f(0)$, the term $r'_n(0)$ can be omitted from the denominator of (5.17):

$$x_0 \approx -\frac{r_n(0)}{r'_f(0)} \ . \tag{5.18}$$

With this simplification it is a simple matter to calculate the variance of $x_0$. The denominator is a known deterministic function, while the nominator is a stationary stochastic process with known autocorrelation $R_r(\tau)$, and therefore with known variance $\sigma_r^2 = R_r(0)$. Thus, the variance $\sigma_0^2$ of the location $x_0$ of the zero crossing is given by

$$\sigma_0^2 = \frac{\sigma_r^2}{r_f'(0)^2} \quad . \tag{5.19}$$

It is useful to check whether this result also holds in practice. To do this, various functions $r_f(x)$ with exactly one zero crossing were sampled at discrete points $x_i$, $i = -32, \ldots, 32$. For the tests the following functions were used with varying signal levels $h$: $f(x) = hx$, $f(x) = h \arctan x$, $f(x) = h\sqrt{2\pi}(\phi_1(x) - 0.5)$, and $h \tanh(x)$. Note that all these functions have $r_f'(0) = h$. The noise $r_n(x)$ was generated by convolving Gaussian white noise of standard deviation 1 with a Gaussian smoothing kernel of width $\sigma$. Thus, it has the autocorrelation $R_r(\tau) = g_{\sqrt{2}\sigma}(\tau)$, i.e., $\sigma_r = 1/(2\sqrt{\pi}\sigma)$. For the experiments, $\sigma = 2$ was used. From the sampled values $r_i = r_f(x_i) + r_n(x_i)$, zero crossings are extracted to sub-pixel accuracy by linear interpolation. Since the relation $r_f'(0) = h$ does not hold for the sampled $r_i$, care is taken to choose $h$ in such a way that the discrete approximation of the first derivative has the value $h$, i.e., $(r_1 - r_{-1})/2 = h$. For each function, 100000 experiments were performed, and the variance of the closest zero crossing to 0 was calculated from these experiments. The results are shown in Figure 5.13. As can be seen, for high signal to noise ratios, i.e., for high $h$, the variance of the zero crossings predicted by (5.19) and the experimentally determined variance match almost perfectly. If the signal level $h$ is less than 1, i.e., if the standard deviation of the noise is larger than approximately 40% of the amplitude of the signal, the term $r_n'(0)$ neglected in (5.18) becomes important. As can be seen, in this range of signal to noise ratios, the variance of the zero crossings is estimated too small by (5.19). Fortunately, such low signal to noise ratios occur extremely rarely in real images. One notable exception are Synthetic Aperture Radar (SAR) images, where the speckle effect can create very low signal to noise ratios [79, 80]. Thus, (5.19) in most cases gives a very accurate estimate of the true variance of the zero crossings of a function corrupted by noise. If a more precise estimate of the variance is needed, one should note that this problem is intimately related to the problem of determining the first-passage density of a stochastic process to an arbitrary boundary, for which a solution was presented in [43, 158]. However, since the first-passage density derived in these papers is rather complex and can only be calculated numerically, it is not applied here.

### 5.2.3 Variance of Line and Edge Positions

With these preliminaries, the predicted variance of the line and edge positions can be determined easily. Since the edge model is much simpler than the line model it is considered first. To derive the variance of the edge locations in the two-dimensional case, without loss of generality it can be assumed that the edge is straight and lies on the $y$-axis. Therefore, the edge position is given by the zero crossings of the model edge convolved with the second directional derivative along the $x$-axis of a Gaussian kernel. Thus, the edge location is given by

$$r_{xx}(x, y) = r_{f,xx}(x, y) + r_{n,xx}(x, y) = 0 \quad . \tag{5.20}$$

The two-dimensional autocorrelation function $R_{r_{n,xx}}$ of $r_{n,xx}(x, y)$ is given by [175]

$$R_{r_{n,xx}}(\tau_x, \tau_y) = \sigma_n^2 g''''_{\sqrt{2}\sigma}(\tau_x) g_{\sqrt{2}\sigma}(\tau_y) \quad , \tag{5.21}$$

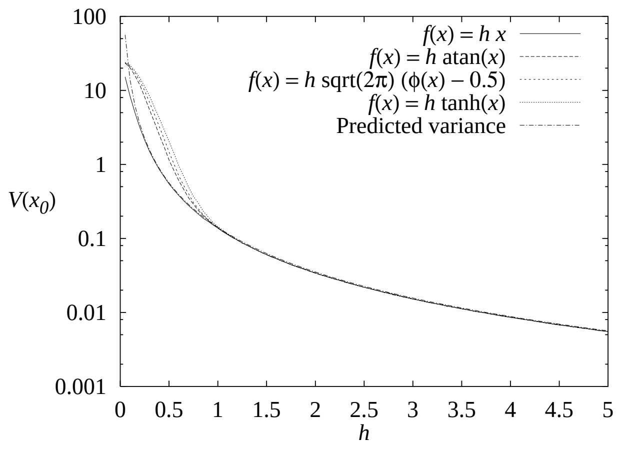


Figure 5.13: Variance of the zero crossing of different test functions when corrupted with noise of known autocorrelation and analytically predicted zero crossing variance. Note the logarithmic scale of the $y$-axis.

where $\sigma_n^2$ is the variance of the noise in the original image. Thus, the variance of the random noise field is

$$\sigma^2_{r_{n,xx}} = \frac{3\sigma_n^2}{16\pi\sigma^6} \ , \tag{5.22}$$

where $\sigma$ is the standard deviation of the Gaussian smoothing kernel. For the response of the filter to the model edge needed in (5.19), obviously

$$r_{f,xxx}(x,y) = hg''_\sigma(x) \ , \tag{5.23}$$

where $h$ is the contrast of the edge. Thus,

$$r_{f,xxx}(0,0)^2 = \frac{h^2}{2\pi\sigma^6} \ . \tag{5.24}$$

Therefore, the variance of the location of an edge in a two-dimensional image is given by

$$\sigma_e^2 = \frac{\sigma^2_{r_{n,xx}}}{r_{f,xxx}(0,0)^2} = \frac{\dfrac{3\sigma_n^2}{16\pi\sigma^6}}{\dfrac{h^2}{2\pi\sigma^6}} = \frac{3}{8}\frac{\sigma_n^2}{h^2} \ . \tag{5.25}$$

This result is quite surprising since it signifies that the variance $\sigma_e^2$ of the edge locations is independent of the choice of the smoothing parameter of the Gaussian kernel used to calculate the gradient of the image. This counterintuitive result should be verified by experiments. To do so, test images of size $32 \times 32$ were generated with exactly one edge in the center of the image. Edges were extracted with Gaussian derivatives of varying standard deviation $\sigma$. The input images were corrupted with Gaussian noise of varying standard deviation $\sigma_n$. For each combination of $\sigma$ and $\sigma_n$, 1000 experiments were performed, resulting in roughly 30000 edge points from which the mean value and variance of the edge positions were computed. Figure 5.14 shows that the mean value of the edge positions is unbiased. The maximum deviation

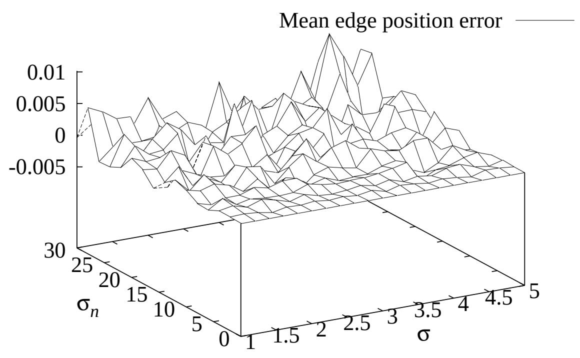


Figure 5.14: Mean value of the edge location as a function of the standard deviation $\sigma$ of the Gaussian smoothing kernel and the standard deviation $\sigma_n$ of the input noise.

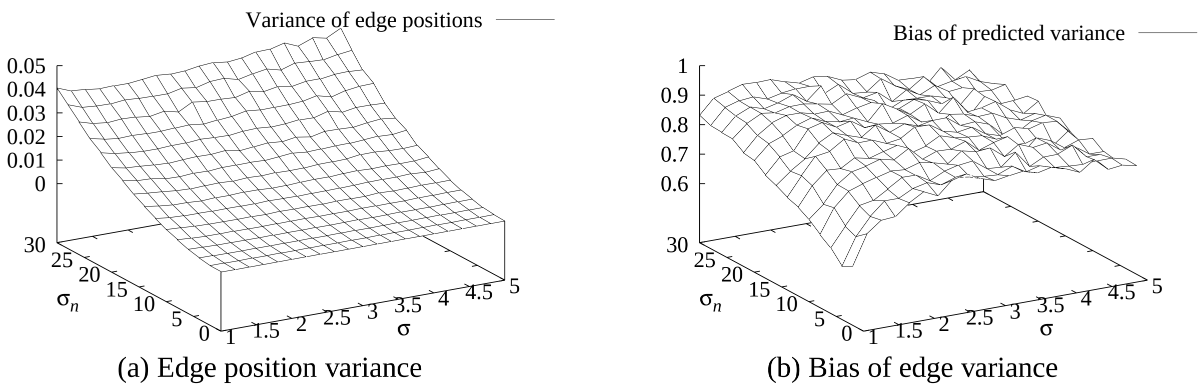


Figure 5.15: Variance of the edge location as a function of the standard deviation $\sigma$ of the Gaussian smoothing kernel and the standard deviation $\sigma_n$ of the input noise (a). Ratio of the predicted and extracted edge position variances, i.e., bias of the prediction.

of the edge positions in this series of experiments is roughly one hundredth of a pixel. More importantly, Figure 5.15(a) shows the calculated variances of the edge positions. As can be seen, the results do not depend on the degree of smoothing. To check the quality of the edge position variance predicted by (5.25), Figure 5.15(b) displays the ratio of the predicted and extracted edge position variances. This ratio is a measure of the bias of (5.25). As can be seen, it is very close to 1 for most combinations of $\sigma$ and $\sigma_n$. The maximum error occurs if $\sigma$ and $\sigma_n$ are close to zero. Overall, (5.25) gives an excellent estimate of the edge position variance. It should be noted that this analysis has only been done for edges extracted by Gaussian smoothing. The reason for this is that because the smoothing filter of the Deriche edge detector is of the form $e^{-a|x|}$, while the derivative filter is of the form $-xe^{-a|x|}$, the analytical integrations required to obtain the autocorrelation of the noise used in (5.25) is much harder. Nevertheless, one can expect the variance of the edge positions returned by the Deriche edge detector to be in the same order of magnitude as the edges obtained by Gaussian derivatives.

In order to predict the variance of the line position in two-dimensional images if the width and position correction is not applied, the same technique can be used. As above, without loss of generality the line can be assumed to lie on the $y$-axis. Thus, the line position is given by the

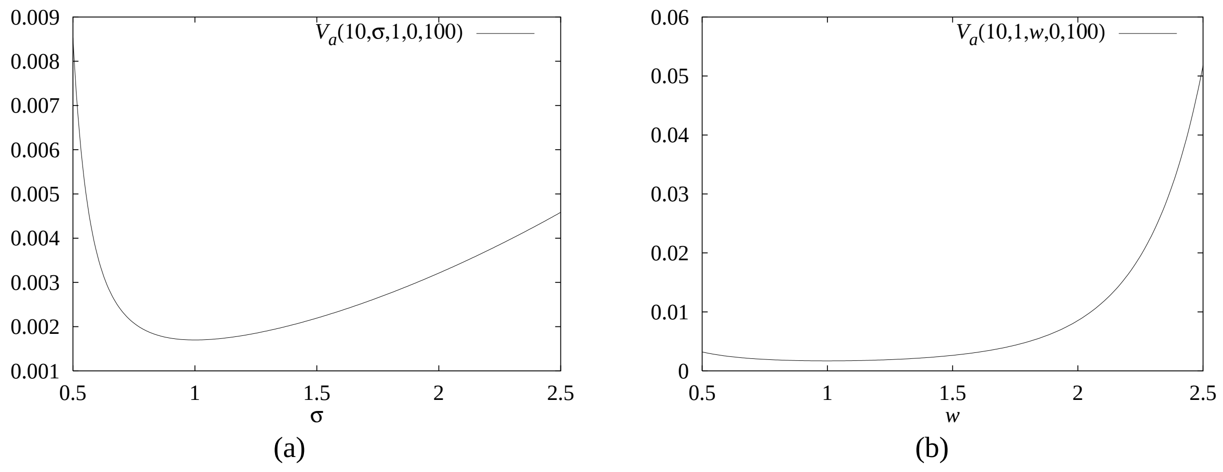


Figure 5.16: Predicted variance of the line location as a function of the standard deviation $\sigma$ of the Gaussian smoothing kernel for a line of width $w = 1$, contrast $h = 100$, and noise of standard deviation $\sigma_n = 10$ (a), and as a function of the line width $w$ for $\sigma = 1$, $h = 100$, and $\sigma_n = 10$ (b).

zero crossings of the model line convolved with the first directional derivative along the $x$-axis of a Gaussian kernel. Hence, the line location is given by

$$r_x(x, y) = r_{f,x}(x, y) + r_{n,x}(x, y) = 0 \ . \tag{5.26}$$

The two-dimensional autocorrelation function $R_{r_{n,x}}$ of $r_{n,x}(x, y)$ is

$$R_{r_{n,x}}(\tau_x, \tau_y) = -\sigma_n^2 g''_{\sqrt{2}\sigma}(\tau_x) g_{\sqrt{2}\sigma}(\tau_y) \ , \tag{5.27}$$

and hence the variance of the random noise field is

$$\sigma_{r_{n,x}}^2 = \frac{\sigma_n^2}{8\pi\sigma^4} \ . \tag{5.28}$$

Since the asymmetry of a line causes the line position to shift, the denominator of (5.19) has to be evaluated at the true line position $l$ given by (3.19). With this, the variance of the position of a two-dimensional line is given by

$$\sigma_l^2 = \frac{\sigma_{r_{n,x}}^2}{r_{f,xx}(0,0)^2} = \frac{\sigma_n^2}{8\pi\sigma^4} \frac{1}{h^2 r''_a(l, \sigma, w, a)^2} \ , \tag{5.29}$$

where $r''_a(l, \sigma, w, a)$ is given by (3.18). Unfortunately, this expression cannot be simplified very much. Therefore, to get an impression of the dependency of the line position variance on different parameters, Figure 5.16(a) shows the predicted variance as a function of the standard deviation $\sigma$ of the Gaussian smoothing kernel, while Figure 5.16(b) shows the predicted variance as a function of the line width $w$. As can be seen, the line position variance is smallest for $\sigma = w$. Furthermore, if $\sigma$ is chosen much too small for a given line width $w$, the line position variance increases rapidly.

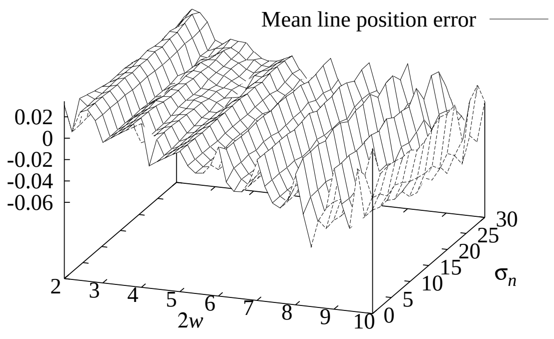
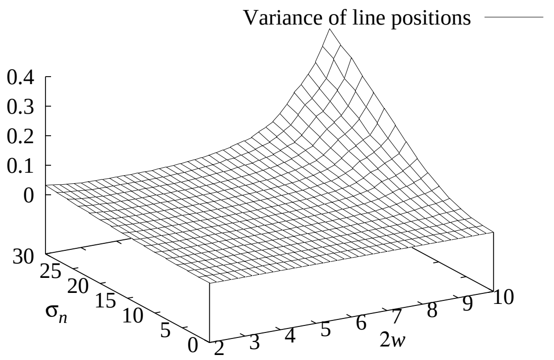
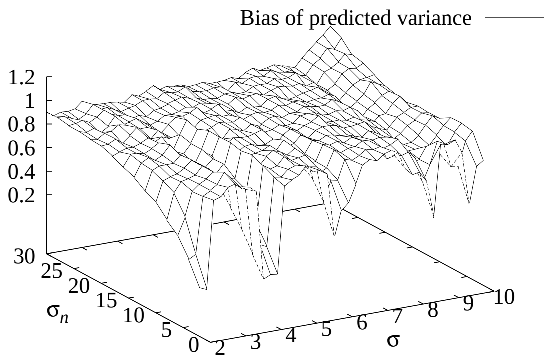


Figure 5.17: Mean value of the line position as a function of the total line width $2w$ and the standard deviation $\sigma_n$ of the input noise.



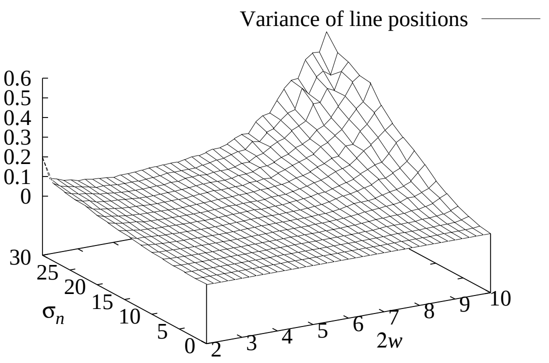
(a) Line position variance



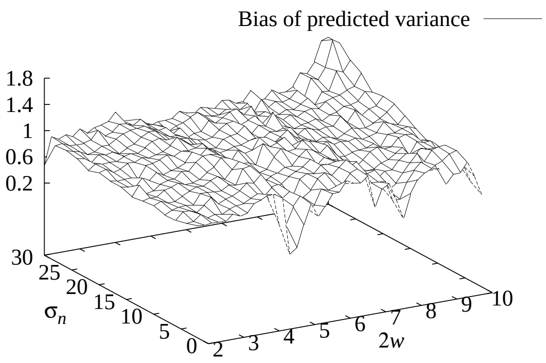
(b) Bias of line variance

Figure 5.18: Variance of the line location as a function of the total line width $2w$ and the standard deviation $\sigma_n$ of the input noise (a). Ratio of the predicted and extracted line position variances, i.e., bias of the prediction (b).

Again, it has to be checked whether (5.29) holds for synthetic test images. To do so, images of size $32 \times 32$ were generated with exactly one line lying in the center of the image. To simulate the typical application case, the $\sigma$ used for smoothing was kept constant, while the total width $2w$ of the line was varied in the interval $[\sigma, 5\sigma]$. Furthermore, the asymmetry of the line was varied in the interval $[0, 0.75]$. The input images were corrupted with Gaussian noise of varying standard deviation $\sigma_n$. For each combination of $a$, $w$, and $\sigma_n$, 1000 experiments were performed, resulting in roughly 30000 edge points from which the mean value and variance of the edge positions were computed. For the experiments, lines of contrast $h = 100$ were extracted with $\sigma = 2$. Figure 5.17 displays the mean line position error, i.e., the difference between the mean value of the line positions extracted from the image and the true line positions for $a = 0$. As can be seen, for all odd integer line widths the mean error is very close to zero. For non-integer line widths the mean may lie up to 0.07 pixels from the true line position. This should come as no surprise if the discussion in Section 5.1.1 is taken into account, since in these cases the line position does not exactly lie in the center of the pixels. Thus, noise introduces no new bias of the line positions, and therefore, the line detector can now be regarded as unbiased in the presence of noise as well. Figure 5.18(a) shows the extracted line position variance. As can be



(a) Line position variance

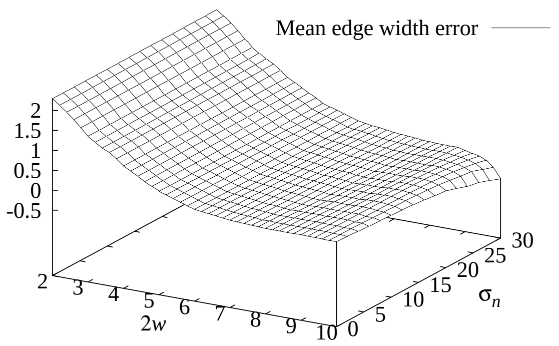


(b) Bias of line variance

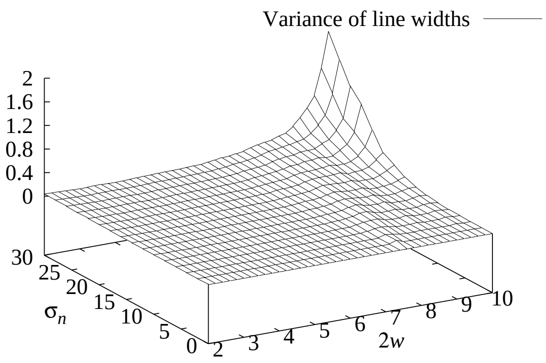
Figure 5.19: Variance of the line location for $a = 0.5$ (a). Bias of the variance prediction for $a = 0.5$ (b).

seen, the cross sections of the surface in this figure closely resemble the shape of the predicted variance displayed in Figure 5.16(b). To check the validity of (5.29), Figure 5.18(b) shows the ratio of the predicted and extracted line position variances. This measure of the bias of (5.29) is very close to 1 for almost all combinations of $w$ and $\sigma_n$. For even integer line widths and small noise variances $\sigma_n^2$, the extracted line position variance is larger than the predicted variance by up to a factor of five because the line positions lie at the borders of a pixel, and thus from the discussion in Section 5.1.1 one should expect the variance to be larger in these cases. Similar results are also obtained for all other values of the asymmetry $a$. Figure 5.19 exemplifies this by showing the line position variance and its bias for $a = 0.5$. As can be seen, the measure of the bias is again very close to 1 for almost all combinations of $w$ and $\sigma_n$. Overall, (5.29) gives an excellent estimation of the line position variance for uncorrected line extraction results.

Another very important performance measure for lines is the variance of the extracted line widths. Since the total line width in case the width correction is not applied is given by the distance of the two edges of the line, in principle it seems possible to describe the variance of the line width using the approach introduced in this section. For example, one could calculate the variances of the two edge positions, assume they are independent, and obtain the variance of the line width as the sum of the two edge position variances. However, this has several problems. First, the edge locations at which (5.19) must be evaluated can only be calculated numerically. Therefore, no analytical result is possible. Instead, the width variance would have to be tabulated. More importantly, however, the edge positions, and thus their variances, are not independent of one another, as is clear from the discussion in Sections 3.1.3 and 3.1.10. Finally, since the first edge point found on each side of the line is used to calculate the width of the line without taking the edge strength into account, it can be expected that for high noise levels the edge position is much more affected by noise than predicted by an analytical model because then the first edge point found may be caused by noise instead of the true edge point. For these reasons, the approach of adding the variances of the two edge locations cannot return very good results, and this was verified on the synthetic examples used above for the line position variance. From this discussion it follows that a good analytical prediction of the variance of the line width seems very hard to obtain. Therefore, this topic has not been pursued further in this dissertation. Instead, only the experimentally obtained means and variances of the line width



(a) Mean line width error



(b) Line width variance

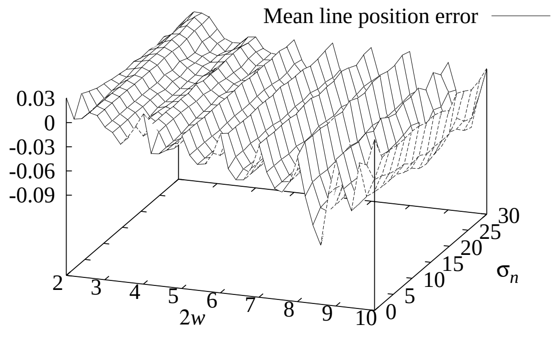
Figure 5.20: Mean (a) and variance (b) of the line width as a function of the total line width $2w$ and the standard deviation $\sigma_n$ of the input noise.

will be discussed. These are shown for $a = 0$ in Figures 5.20(a) and (b), respectively. As can be seen from Figure 5.20(a), the noise has almost no influence on the extracted mean width error, i.e., the difference of the mean line width and the true line width. Of course, the mean line width is biased by an amount which corresponds to the example of Figure 3.5. Figure 5.20(b) displays the experimentally obtained variance $\sigma_w^2$ of the line widths. As can be seen, it again increases proportionally to $\sigma_n^2$. Furthermore, the width variance is relatively small for small $w$. It increases significantly only for large $w$. Again, similar results hold for all asymmetries $a$.

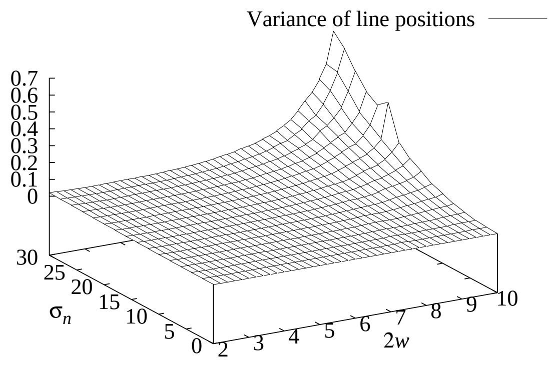
The final important question is how the position and width correction influences the variances of the line position and width. Remember from Section 3.1.10 that the bias removal conceptually is given by a function $f : (v_\sigma, r) \mapsto (w_\sigma, a)$. Thus, standard error propagation schemes could be used to propagate the variances $\sigma_l^2$ and $\sigma_w^2$ through $f$. If the covariance $\sigma_{l,w}$ of the line position and width were known, the covariance matrix $\Sigma_u$ of the uncorrected could be calculated, and the covariance matrix $\Sigma_c$ of the corrected line position and width would simply be given by [74]

$$\Sigma_c = Df \Sigma_u Df^T \quad , \tag{5.30}$$

where $Df$ is the Jacobian matrix, i.e., the first derivative, of $f$. Since $\sigma_w^2$ is only known empirically, since an explicit estimation for $\sigma_{l,w}$ is very hard to obtain, and since $f$ is only given in tabulated form, no attempt has been made to derive an explicit formula for $\Sigma_c$. However, from the shape of the bias inversion function displayed in Figure 3.27 it can be estimated that the variances of the corrected line position and width will be slightly larger than the uncorrected variances because the partial derivatives of $f$ are larger than 1 almost everywhere. Figure 5.21 displays the mean and variance of the corrected line position, while Figure 5.22 shows the mean and variance of the corrected line width. From Figure 5.21(a) it can be seen that the line position correction slightly increases the mean error in areas of non-integer line width, i.e., in areas where the line position does not lie close to the center of a pixel. The maximum error increases from approximately 0.07 to 0.09 pixels. Furthermore, from Figure 5.22(a) it can be seen that the corrected line widths are much closer to the true line widths. The corrected line widths exhibit a small bias only if the line width is very small compared to $\sigma$. This fact was also observed for noiseless images in Section 5.1.2, and thus should come as no surprise. The variances of the line position are slightly decreased for small $w$, while they are slightly increased for large

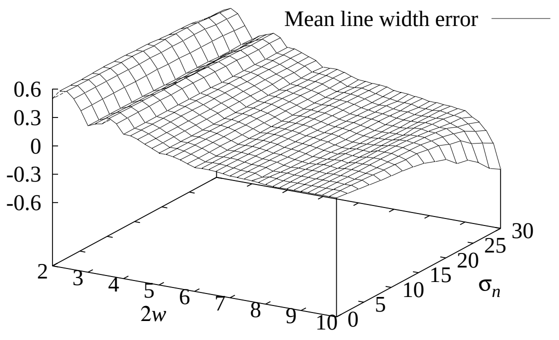


(a) Mean line position error

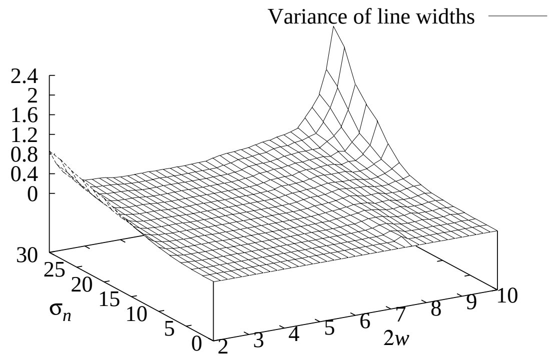


(b) Line position variance

Figure 5.21: Mean (a) and variance (b) of the line position as a function of the total line width $2w$ and the standard deviation $\sigma_n$ of the input noise if the line width and position correction is applied.



(a) Mean line width error



(b) Line width variance

Figure 5.22: Mean (a) and variance (b) of the line width as a function of the total line width $2w$ and the standard deviation $\sigma_n$ of the input noise if the line width and position correction is applied.

$w$, as can be seen when Figure 5.21(b) is compared to Figure 5.18(a). On the other hand, the line width variance increases by a large amount for small $w$, while the increase is small for large $w$, as can be seen by comparing Figures 5.22(b) and 5.20(b). Essentially, for small $w$ the correction seems to trade better line position variance for worse line width variance. As above, qualitatively similar results hold for all other values of the asymmetry $a$.

In summary, from the discussion in this section it can be concluded that the extracted line positions and widths are unbiased in the presence of noise, and that the variances of the line positions and widths are very small. Thus, sub-pixel accuracy can definitely be achieved for noisy synthetic images, while it seems very likely that this kind of accuracy may also be achievable in real images.

To conclude this section, it should be noted that sometimes other performance measures apart from the localization quality also play an important role. The most often used measure is the detectability of a feature in the presence of noise. Essentially, this can be modeled by the signal to noise ratio, i.e., the ratio of magnitude squared of the derivative of the model feature in which the thresholding is done and the variance of the noise in this derivative. For edge detection, the relevant derivative is the first derivative, while for line detection it is the second derivative. Thus, for edge detection the signal to noise ratio $r$ is given by

$$r = \frac{r_e'(0,\sigma,h)^2}{\sigma_{r_{n,x}}^2} = \frac{4h^2\sigma^2}{\sigma_n^2} \ , \tag{5.31}$$

while for lines it is given by

$$r = \frac{h^2 r_a''(l,\sigma,w,a)^2}{\sigma_{r_{n,xx}}^2} = \frac{16\pi\sigma^6 h^2 r_a''(l,\sigma,w,a)^2}{3\sigma_n^2} \ . \tag{5.32}$$

This leads to the fairly obvious conclusion that features become easier to detect as the standard deviation $\sigma$ of the Gaussian used to smooth the image is increased. Of course, this only holds if there are no other nearby features complicating the extraction.

## 5.3 Experiments on Real Images

All the experiments so far have been done on synthetic test images. Since for real imaging conditions various assumptions made in the test so far may not hold, it is important to perform experiments on real images. For example, the camera and framegrabber used to acquire an image may not have the ideal square aperture function or the camera may not exhibit a linear increase in gray values as the scene intensity increases linearly.

### 5.3.1 Experimental Set-Up

In order to test the sub-pixel accuracy of the line detection algorithms the following strategy was adopted: A test set containing five lines of widths 2 mm–10 mm were printed on a high-quality laser printer. The resulting print was mounted on a table which can be shifted in one direction in 10 $\mu$m increments. A camera with a 12 mm lens was mounted perpendicular to the table at a distance of approximately 50 cm. A standard analog framegrabber was used to acquire the images, since this is by far the most common setup in industrial applications, where sub-pixel measurements are extremely important. For edge detection, the test set contained a single edge in the center of the print. Figure 5.23 shows an image of the line and edge test sets. Note that although the lens used in this test is fairly good for computer vision standards, there is nevertheless a significant radial distortion. To test the sub-pixel accuracy, each of the test sets was shifted by 1 mm in increments of 50 $\mu$m, resulting in a shift of approximately 2 pixels in total in the image. In each position of the measurement table, 20 images were taken. Thus, for each test set a total of 420 images were taken. The goal of the test is to determine the repeatability, i.e., the variance, of the line and edge position measurements and the absolute position shift of the line and edge positions in order to decide whether it is possible to detect sub-pixel shifts of 50 $\mu$m or, equivalently, approximately one tenth of a pixel.

Before this can be done, a few words on the architecture of standard analog framegrabbers are necessary since the technology used for them will influence the quality of the acquired

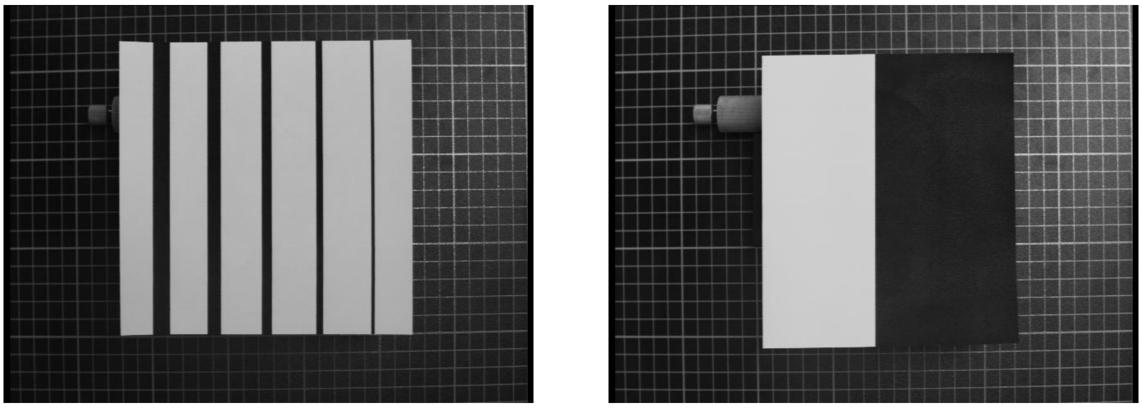

(a) Line test set (b) Edge test set

Figure 5.23: Test sets used to check the sub-pixel accuracy of the line and edge detection algorithms.

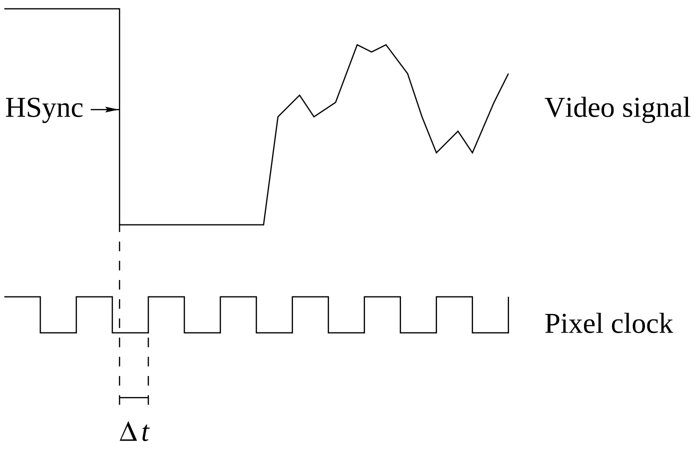
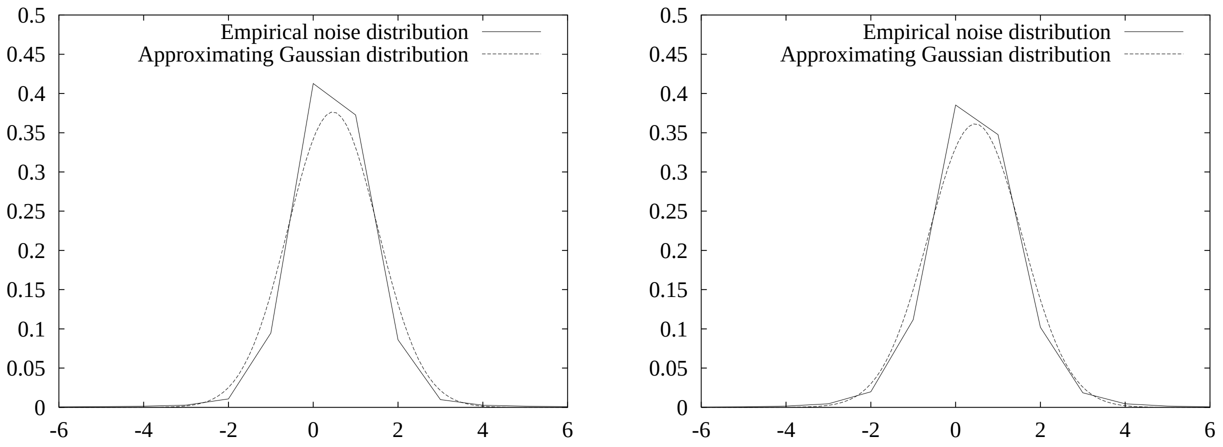


Figure 5.24: Scheme to start the digitization of one line of a video signal in current framegrabber architectures: Digitization is started at the first rising flank of the unsynchronized pixel clock after the horizontal sync signal has occurred, resulting in an offset of $\Delta t$.

images. The main implementation issue for framegrabbers is when to start digitizing a line of the video signal. To do this, most analog framegrabbers have a clock chip running at several times the pixel resolution, typically 4–10 times. They start the digitization process on the first upward flank of the pixel clock signal after the falling flank of the horizontal synchronization (HSync) signal of the video signal has occurred, as shown in Figure 5.24. Thus, for each line there is a random offset of $\Delta t$ by which the line is shifted with respect to the true HSync signal, resulting in a random shift by up to one fourth to one tenth of a pixel, depending on the frequency of the pixel clock. This seems to prevent sub-pixel accuracy of better than one tenth of a pixel. Fortunately, however, the random offset is independent for each line of the video signal. Thus, one can expect to achieve better accuracy by the smoothing done in the line or edge detector. In effect, the random shifts in the digitized video signal can be modeled as an additional noise source. From this discussion it follows that the noise should be larger at the



(a) Noise distribution (lines) (b) Noise distribution (edges)

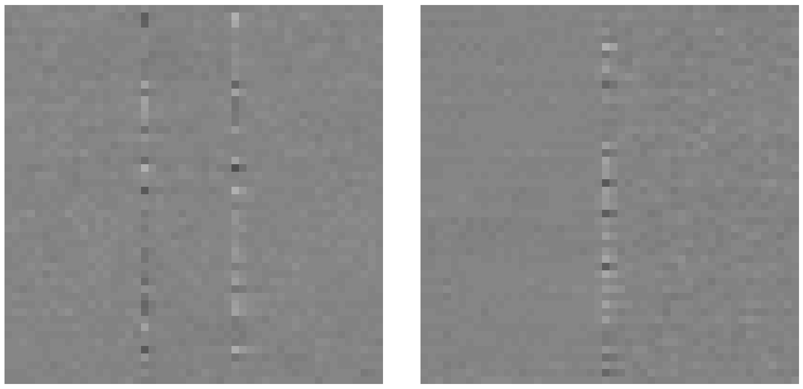
Figure 5.25: Experimentally determined noise distributions and Gaussian distributions with the same mean and variance for the line test set (a) and the edge test set (b).

border of the features of interest since the sub-pixel shifts result in larger gray value variations there, while they have very little influence in regions of approximately constant gray value. Of course, this problem does not occur if the framegrabber is synchronized with the pixel clock of the camera or if a digital camera is used.

In order to test the hypothesis that the random shifts in each line of the video signal lead to increased noise at edge positions and to obtain the noise statistics of the test images, the average of all 20 images within each shift can be calculated. Then, the individual images can be subtracted from the mean image to obtain the noise distribution for that particular image. If the distributions over all 420 images are averaged, a very good approximation of the noise distribution is obtained. Figure 5.25 shows the noise distributions thus obtained for the line and edge test set, superimposed by a Gaussian distribution of the same mean and variance. As can be seen, the Gaussian distribution yields a very good approximation of the noise distribution in both cases. The standard deviations of the noise were estimated as approximately 1.06 and 1.10, respectively, which is very small. Figure 5.26 shows small parts of test images subtracted from their corresponding mean image. As can be seen, the random shifts lead to significantly increased noise at the edges of the line in Figure 5.26(a) and at the edge in Figure 5.26(b). Therefore, the line and edge locations may not be as accurate as one would expect from the discussion in the previous section.

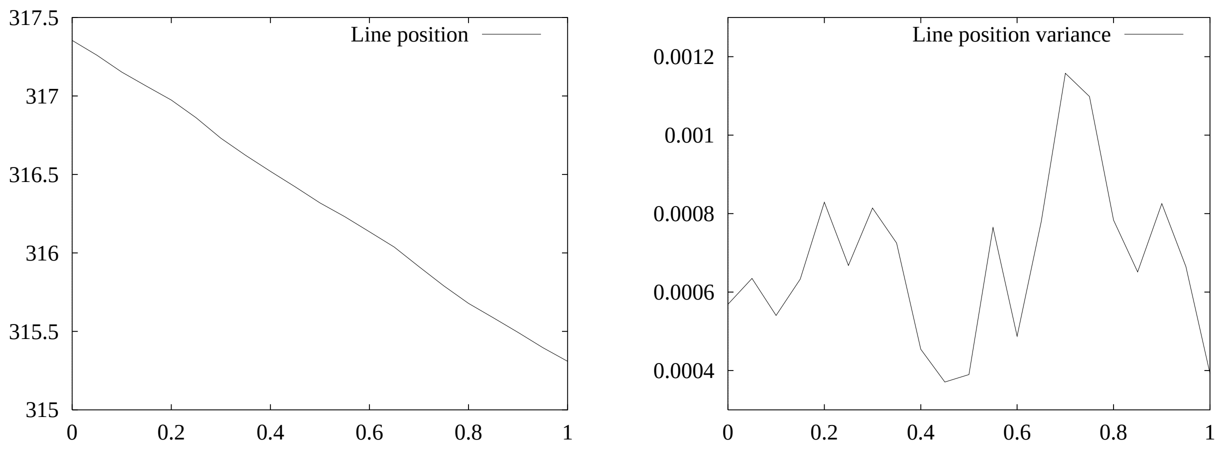
### 5.3.2 Sub-Pixel Accuracy of Line Position and Width

To test the sub-pixel accuracy of the line detection algorithm, only the center line of the test set of width 6 mm was used since it turned out that the lens distortion was too large to obtain meaningful results for the other four lines. Although the center line was carefully aligned to be as vertical as possible, even for this line the lens distortion influences the extraction results, if statistics are calculated over the entire line. Therefore, small windows of height 25 were used to calculate the mean line position and width, and their corresponding variances, because for these small windows the lens distortion played only a small role. Since the means and variances are



(a) Noise in a line image (b) Noise in an edge image

Figure 5.26: Examples of a test image subtracted from its corresponding mean image for the line test set (a) and the edge test set (b). The gray values in the images have been scaled by a factor of 5 for clarity.



(a) Line position (b) Line position variance

Figure 5.27: Line position and variance as a function of the shift of the line in mm.

calculated over 20 images, a total of 500 points are used to obtain the statistics. The experiments showed that the resulting statistics are independent of the position of the window. Because the center line of the test images has a total width of approximately 12 pixels, $\sigma = 6/\sqrt{3}$ was used for the extraction. Figure 5.27(a) shows the extracted line position as a function of the shift of the line in mm. As can be seen, the resulting curve is almost a perfect straight line. The repeatability of the line position, i.e., its variance, is displayed in Figure 5.27(b), from which it can be seen that the maximum variance is approximately 0.00115, i.e., the maximum standard deviation is roughly one thirtieth of a pixel. From (5.29) the line position variance can be estimated as $1.84 \cdot 10^{-5}$. Thus, the extracted position variance is larger than the predicted variance by a factor of 62.5 mainly due to the random line shifts induced by the framegrabber, the still noticeable lens distortion in this small part of the image, and the fact that the line is not

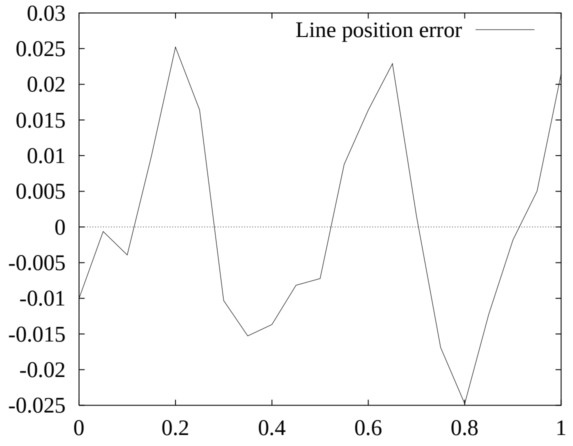


Figure 5.28: Absolute error of the line position calculated as the difference of the extracted line positions and their regression line as a function of the shift of the line in mm.

perfectly aligned with the vertical axis. However, since the position variance is still very small it can be expected that the goal of these experiments to detect sub-pixel shifts of one tenth of a pixel can be achieved. This can easily be verified by testing the hypothesis that the line positions corresponding to two adjacent shifts are equal [51]. These tests have been performed and the hypothesis that the line positions are equal can be rejected with a probability greater than 99.9% for all line positions. Therefore, relative shifts of one tenth of a pixel can definitively be detected in real images.

Another interesting point is the absolute position accuracy of the line. Since the camera was not calibrated, this is very hard to obtain. If, however, it is assumed that the linear shift of the object in the real world corresponds to a linear shift of the line in the image, a straight line can be fitted through the line positions extracted from the image. The fitting should, of course, take the extracted position variances into account. The equation of the line obtained for the experiment shown in Figure 5.27 is $-2.07633x + 317.364$. Therefore, it is likely that 1 mm in the real world corresponds to 2.07633 pixels in this part of the image. With this, the absolute position error of the line can be calculated as the difference of the extracted line position and the regression line. The results are shown in Figure 5.28. As can be seen, the absolute position errors are less than one fortieth of a pixel. The remaining errors show a systematic sinusoidal component, which may be caused by the mapping of the scene intensity to the gray values in the image performed by the framegrabber. Since they are very small, the line extraction algorithm yields very good absolute accuracy on real images as well. Of course, to get meaningful positions in real world coordinates the camera must be calibrated.

The final question for the line detection algorithm is the accuracy of the extracted line widths. The extracted line width and its variance are shown in Figure 5.29. The line width seems to be centered around 12 pixels. If the above discussion on the correspondence between 1 mm to 2.07633 pixels is taken into account, this would imply that the width of the 6 mm wide line is underestimated by approximately 3.7%. Therefore, the extracted line width is very close to the true line width. Furthermore, it can be seen by the regression line superimposed in Figure 5.29(a), that the line width increases as the shift increases. This can be attributed to the fact that for increasing shifts the line moves towards the center of the image where lens distortions

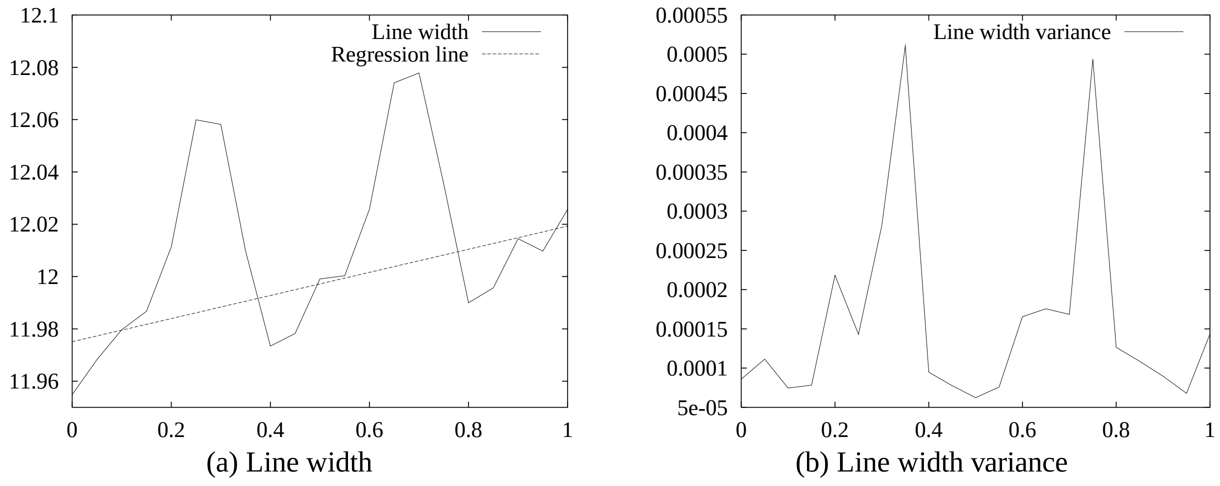


Figure 5.29: Line width and variance as a function of the shift of the line in mm.

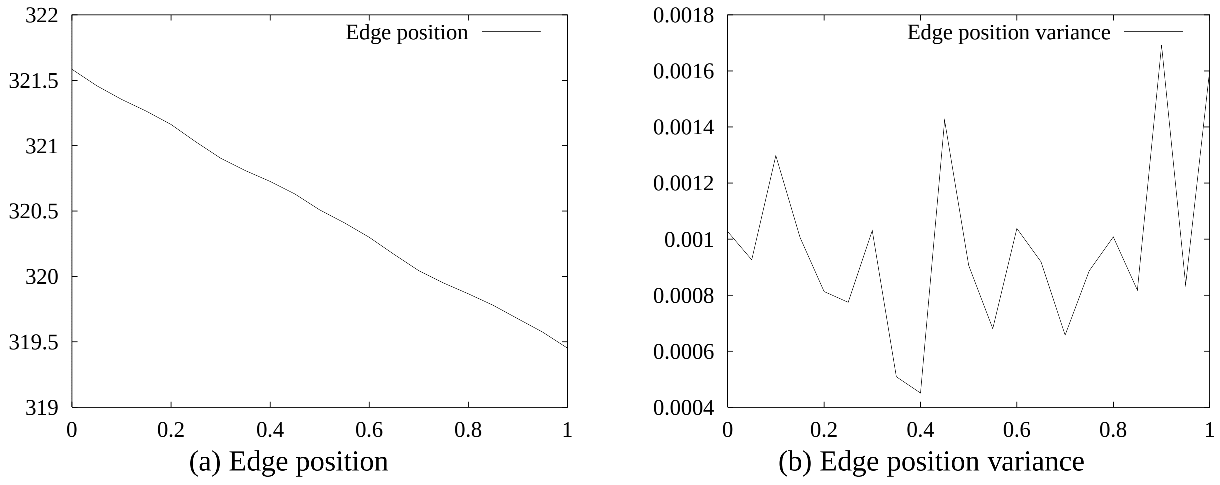


Figure 5.30: Edge position and variance as a function of the shift of the edge in mm.

are not as severe. Finally, Figure 5.29(b) displays the line width variance obtained from this experiment. As can be seen, the variance is less than 0.0005 almost everywhere, i.e., the standard deviation of the extracted line widths is less than one fortieth of a pixel. Thus, line widths can be extracted with very high sub-pixel accuracy.

### 5.3.3 Sub-Pixel Accuracy of Edge Position

To conclude this section, the results of performing the same kind of experiment for the sub-pixel edge detection algorithm are discussed. The edges were extracted using Gaussian derivatives with $\sigma = 1$. The resulting edge positions are exhibited in Figure 5.30(a). As was the case for lines, the extracted edge positions lie very close to a perfect straight line. Figure 5.30(b) displays the corresponding variances of the edge positions. The maximum variance is approximately 0.0017. The variance predicted by (5.25) is $1.57 \cdot 10^{-5}$, i.e., the extracted variance is larger

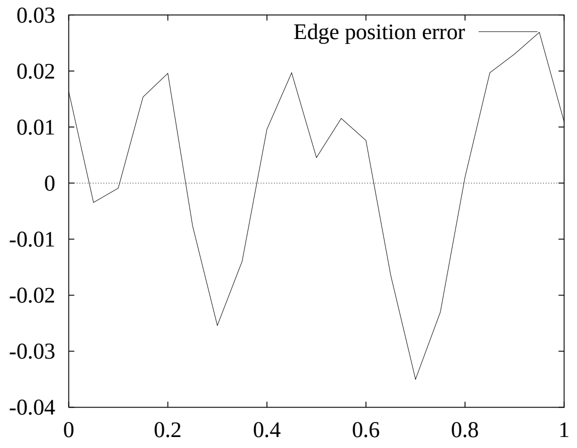


Figure 5.31: Absolute error of the edge position calculated as the difference of the extracted edge positions and their regression line.

than the predicted variance by a factor of 108. Again, this is caused by the random shifts of the discretized lines of the video signal, the lens distortion, and the fact that the edge is not perfectly aligned with the vertical axis. However, the standard deviation of the edge positions is still very small, being approximately one twenty-fifth of a pixel. Therefore, with the same hypothesis test as used above, it can be shown that edge shifts of one tenth of a pixel can be detected with better than 99.9% probability. Finally, as above, the absolute position error can be estimated by fitting a straight line through the extracted line positions. In this case, the equation of the line is $-2.12664x + 321.568$. The resulting absolute position errors are depicted in Figure 5.31. The maximum absolute error is approximately one thirtieth of a pixel. Therefore, edges can be extracted with very good absolute sub-pixel accuracy. Qualitatively similar results were also obtained if edges were extracted using the isotropic Deriche operator. If the remaining errors of up to one thirtieth of a pixel are too large for an application, a scheme as the one given in [147] could be used to estimate and remove the remaining errors. However, since the remaining errors are not generic, i.e., depend on the camera, lens, and framegrabber used, they are not modeled in this dissertation.

The analysis in this chapter shows that the 2D line and edge extraction algorithms presented in this dissertation return unbiased estimates of the line position, width, angle, asymmetry, and contrast, and of the edge position and orientation with very high sub-pixel precision and accuracy. Because of the similarity of the 3D feature extraction algorithms proposed in this dissertation, similar results can also be expected to hold for their extraction results.

# Chapter 6

# Conclusions

In this dissertation, a novel approach to extract lines and their width with sub-pixel resolution in two-dimensional images is proposed. Lines are assumed to exhibit a characteristic asymmetrical profile perpendicular to the line direction. Individual line points are extracted by using partial derivatives of the image. The partial derivatives are obtained by convolutions with the respective derivatives of the Gaussian smoothing kernel. This results in a scale-space description of the line position and width. It is shown that the line position is generally biased if the line exhibits different lateral contrast, while the line width is always biased. From the scale-space analysis a conceptually elegant and efficient method to remove the bias from the line position and width is derived. Additionally, a new algorithm to link individual line points into lines, which includes all points of a line, i.e., does not remove line point information by some sort of skeletonization, and which returns a topologically sound data structure of lines and junctions, is proposed. Furthermore, the behavior of the line detector in junction areas is analyzed thoroughly. This analysis shows the reasons why junctions are often extracted incompletely. From this analysis, a new and efficient method to complete missing junctions is derived.

As a by-product, a novel edge detection algorithm with sub-pixel resolution is derived from the 2D line extraction algorithm by regarding edges as lines in the gradient image. Furthermore, the behavior of various edge definitions at junctions is analyzed, and it is shown that regarding edges as a special kind of lines leads to improved extraction results at junctions since the method to complete line junctions can then also be applied to edge detection.

The line detection algorithm is extended to 3D images, where it naturally leads to two different kinds of features: lines and surfaces. For surfaces, the determination of the surface position and width, and the bias removal algorithm are straightforward extensions of the algorithms developed for the 2D line detector. The major difference is the algorithm to link individual surface points into surfaces. An algorithm to construct a surface graph from individual surface points is proposed. However, this representation is not adequate for visualization purposes. Therefore, a simple algorithm to triangulate the surface graph is proposed as well.

For 3D lines, the determination of the line position can easily be derived from the 2D algorithm. The major difference for 3D lines is that the line width is given by an entire perimeter curve. A solution for this problem is developed first by extracting the perimeter curve of 2D extrema, resulting in an algorithm which has applications in limited domains. A scale-space analysis of a model extremum shows that the perimeter curves exhibit a similar bias as the line widths of 2D lines. Since the bias removal algorithm for 2D lines cannot be transfered to the problem of removing the bias from the perimeter curves of 2D extrema and 3D lines, a solution for this problem by edge focusing is proposed, which improves the estimation of the line width considerably. Furthermore, it is noted that sometimes the perimeter curves contain gross outliers due to missing edge information. Therefore, a simple and efficient outlier removal algorithm for individual perimeter curves is proposed.

Finally, a thorough performance analysis is carried out for the 2D line and edge extraction algorithms. Noiseless synthetic test images are used to prove the unbiasedness of the extracted line position, width, orientation, asymmetry, and contrast, as well as the unbiasedness of the edge position. Furthermore, the quality of the line extraction results is evaluated for noisy images. Analytical formulas predicting the variance of the line and edge positions in case the variance of the image noise is known are derived and shown to correspond very well to experimentally determined line and edge position variances. Additionally, the variance of the line width in the presence of noise is determined experimentally. Finally, the quality of the line and edge positions and of the line widths is analyzed for real images. The results of this performance analysis indicate that the extraction results are unbiased for noisy synthetic and real images, that the variances of the extracted features are very small, i.e., the sub-pixel precision is very high, and that the absolute positions and widths can be determined to a very high degree of sub-pixel accuracy. For real images it is shown that position shifts of one tenth of a pixel can be detected with a probability of more than 99.9%, indicating that much better sub-pixel accuracy than one tenth of a pixel can be achieved for real images. Thus, it is shown that the line and edge extraction algorithms not only achieve sub-pixel resolution, but also sub-pixel precision and accuracy.

From this discussion it can be seen that the proposed approach fulfills all the requirements discussed in Section 2.7 and achieves all the goals stated in Chapter 1. Thus, from an application point of view, the major result of this dissertation is that the proposed line and edge detection algorithms can be used as high-accuracy low-level feature detectors for many different applications. For example, the 2D line extraction algorithm is used in two different multi-scale road extraction systems to generate road hypotheses in low-resolution images [10, 127]. Furthermore, it is used in a dermatoscopic system to classify skin lesions into benign and malignant [52]. Further applications in which the line extraction algorithm could be used are document analysis, e.g., map interpretation and OCR, and cytological applications. The 3D line extraction algorithm shows great potential to aid the diagnosis of aneurysms and stenoses of blood vessels, while the 3D surface extraction algorithm may have applications in image registration tasks. Finally, the proposed sub-pixel accurate edge detector can be used in any system that uses a traditional, pixel-precise edge detector in order to improve the accuracy of the extracted edges and to gain complete junction information. For these reasons, the proposed algorithms seem to be highly relevant for many applications.

The paradigm of explicitly modeling a feature and analyzing its behavior in linear scale-space introduced in this dissertation can be applied to many other features to improve precision and accuracy. For example, if the sensor characteristics lead to the conclusion that edges will usually be asymmetrical in images, the induced bias could be modeled by the technique proposed in Sections 3.1 and 3.2, and could therefore be removed. One application area where this typically happens is industrial inspection, where the lighting conditions sometimes lead to specular highlights on the edges of an object, and hence to asymmetrical edges. Since the sub-pixel precision and accuracy of the results is very important in industrial inspection, explicit modeling of the bias of the edges is essential under such circumstances.

From a performance point of view, one might ask whether the proposed line and edge detectors are optimal. This question is intentionally left unanswered in this dissertation because it depends strongly on the requirements of the individual application. This means that although a set of globally valid criteria might be defined, it would have to be the lowest common denominator of the requirements of any conceivable application, i.e., consist of few criteria, and would therefore not be very useful for a particular application. Furthermore, it is not obvious what the choice of the global optimality criteria should be. Thus, different sets of global optimality criteria could be used, which will lead to different "optimal" feature detectors. What is more, even if one could agree on a single set of criteria, the choice of admissible functions, e.g., recursive infinite impulse response filters versus finite impulse response filters, will lead to different optimal detectors, as was shown for edge detection filters in [29, 38, 142]. Therefore, a useful assessment of the applicability of an algorithm for an application can only be done by checking whether the particular algorithm fulfills the specific requirements of the application. For the proposed line extraction algorithm, such a performance evaluation has been done for extracting linear objects from SAR images in [79]. The results of the proposed line detector are compared to those of a line detector developed especially for SAR images, which takes into account the special properties of this class of sensors, most notably the speckle effect. This evaluation shows that the line detector proposed here yields results of almost the same quality as the specialized operator, although it was not designed for the special noise conditions in SAR images.

Probably the major inherent problem remaining to be addressed is that features are assumed to be isolated, i.e., it is assumed that there are no other edges or lines in the vicinity of an extracted line. As the example in Figure 3.30 shows, the edges of a line cannot move as far outward as the model predicts in this case. While these configurations could be modeled explicitly, the high number of parameters necessary in an explicit model, e.g., six for a line with a close edge on both sides of the line instead of the two necessary to model an isolated line, would result in an enormously large correction table, and therefore make this approach infeasible. Therefore, the only feasible approach to solve the problem of close features seems to be the edge focusing approach used in the 3D line extraction algorithm, i.e., to extract the line width using a smaller $\sigma$. However, this would create problems of a different kind: since disturbing objects, e.g., cars on a road, are not smoothed away using the smaller $\sigma$ as they are when the line position is extracted using the larger $\sigma$, the problem of selecting the correct edge belonging to a line must be solved. This could only be done using an ad-hoc heuristic, and therefore the line width is not guaranteed to be consistent with the line position. The same problem can, of course, occur for the extraction of 2D extrema and 3D lines, where this approach is used. This is not a big problem for most applications interested in 3D lines, however, since the imaging modalities usually enhance 3D lines in a manner that few disturbances occur in the vicinity of the line.

The individual algorithms proposed in this dissertation exhibit only very few remaining problems. The main problem of the 3D surface extractor lies in its triangulation algorithm: the resulting triangulation contains about two times as many triangles as necessary. Future research in this area could focus on developing a triangulation algorithm that contains the minimum number of triangles. For the 3D line extractor the main problem is that the outlier removal algorithm does not take into account the continuity of the line widths in the direction of the line.

Therefore, one possible extension of the 3D line extraction algorithm is to impose continuity restrictions on the line widths in the direction of the line in the outlier removal algorithm. Furthermore, on rare occasions also the results of the 2D line extractor could be improved by an outlier removal scheme, which has not been implemented so far.

One of the conceptual problems which remains to be solved is that the bias correction only returns precise results for a single specific line model, e.g., an asymmetrical bar-shaped or an asymmetrical parabolic profile. If it can be expected that the images of interest contain several distinct types of line profiles, high accuracy will be achieved for only one of them. However, this problem could be solved by fitting each of the model profiles, after they are convolved with a Gaussian with the $\sigma$ used for the extraction, to the image data, and selecting the model profile that yields the smallest residuals. Of course, the fit must either take the curvature of the line in the image into account or be done over a segment of the line short enough that the curvature can be neglected. If the correction of the best matching profile is applied, high accuracy can expected to be achieved for all types of line profiles.

Another unsolved problem is how to extract lines from color and multispectral images. The proposed differential geometric approach does not extend easily to these cases. However, one could extract lines in each spectral band simultaneously. Since the extracted line positions and widths are unbiased, they can be expected to be very close in each spectral band. Therefore, matching the lines across the individual spectral bands does not pose a complicated problem, and lines could be extracted if they occur in at least one band.

One obvious possibility to extend the work of this dissertation is to use the approach of Section 3.3 to implement a sub-pixel accurate 3D edge detector. Furthermore, it should be investigated whether the analysis of the problem of missing junctions for lines and edges and the proposed junction completion algorithm can be extended to 3D lines, surfaces, and edges.

Another major goal for future research could be to combine line and edge extraction into a single operator. This is very useful for applications that require lines as well as edges as low-level primitives. By extracting both types of features simultaneously, the description of a scene will be simplified since edges corresponding to lines are not returned as edges anymore, but instead are included in the line width. Thus, a later reasoning step can be simplified since it will not have to sort out edges corresponding to lines.

Finally, it must be emphasized that the extracted lines are low-level features and hence do not describe semantic objects. For the interpretation of an image a domain model is essential. A performance evaluation of the line extraction algorithm and the two above mentioned model-based road extraction systems [10, 127] is carried out in [78], where the extracted lines and roads are compared to a manually generated reference road network. The results of this analysis indicate that the line extractor yields only slightly worse results than the two road extraction systems in terms of completeness of the extracted roads, indicating that a road is extracted as a line as long as it is visible in the image. This result is to be expected since the model-based approaches are able to bridge gaps where roads are occluded, e.g., due to shadows. However, in terms of correctness the model-based systems usually yield better performance. This is not surprising because many other linear objects exist in aerial images, e.g., house roofs or narrow fields. These are removed by the model-based approaches, resulting in fewer false hypotheses for roads. Therefore, the line extraction algorithm is useful for generating road hypotheses, but a model-based interpretation is essential to obtain semantically meaningful road objects.

# Bibliography

[1] Adobe Systems Incorporated. *PostScript Language Reference Manual*. Addison-Wesley Publishing Company, Reading, MA, 2nd edition, 1990.

[2] S. Airault, R. Ruskoné, and O. Jamet. Road detection from aerial images: a cooperation between local and global methods. In J. Desachy, editor, *Image and Signal Processing for Remote Sensing*, Proc. SPIE 2315, pages 508–518, 1994.

[3] N. Armande, P. Montesinos, and O. Monga. A 3D thin nets extraction method for medical imaging. In *13th International Conference on Pattern Recognition*, volume I, pages 642–646, 1996.

[4] N. Armande, P. Montesinos, and O. Monga. Thin nets extraction using a multi-scale approach. In B. ter Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, editors, *Scale-Space Theory in Computer Vision*, volume 1252 of *Lecture Notes in Computer Science*, pages 361–364, Berlin, 1997. Springer-Verlag.

[5] K. Åström and A. Heyden. Stochastic modelling and analysis of sub-pixel edge detection. In *13th International Conference on Pattern Recognition*, volume II, pages 86–90, 1996.

[6] K. B. Atkinson, editor. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, Caithness, 1996.

[7] G. Aumann. *Aufbau qualitativ hochwertiger digitaler Geländemodelle aus Höhenlinien*. Number 411 in Reihe C. Deutsche Geodätische Kommission, München, 1994.

[8] S. Aylward, E. Bullitt, S. Pizer, and D. Eberly. Intensity ridge and widths for tubular object segmentation and description. In *Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 131–138, 1996.

[9] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):26–33, January 1986.

[10] A. Baumgartner, C. Steger, H. Mayer, and W. Eckstein. Multi-resolution, semantic objects, and context for road extraction. In Förstner and Plümer [57], pages 140–156.

[11] A. Baumgartner, C. Steger, C. Wiedemann, H. Mayer, W. Eckstein, and H. Ebner. Update of roads in GIS from aerial imagery: Verification and multi-resolution extraction. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXI, part B3, pages 53–58, 1996.

[12] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition*, pages 1000–1006, 1997.

[13] G. Bellaire. Hashing with a topological invariant feature. In *2nd Asian Conference on Computer Vision*, volume 2, pages 598–602, 1995.

[14] J. S. Bendat. *Principles and Applications of Random Noise Theory*. John Wiley & Sons, Inc., New York, NY, 1958.

[15] F. Bergholm. Edge focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):726–741, November 1987.

[16] V. Berzins. Accuracy of Laplacian edge detectors. *Computer Vision, Graphics, and Image Processing*, 27:195–210, 1984.

[17] D. J. Beymer. Finding junctions using the image gradient. A.I. Memo 1266, MIT Artificial Intelligence Laboratory, Cambridge, MA, December 1991.

[18] F. Bignone, O. Henricsson, P. Fua, and M. Stricker. Automatic extraction of generic house roofs from high resolution aerial imagery. In B. Buxton and R. Cipolla, editors, *Fourth European Conference on Computer Vision*, volume 1064 of *Lecture Notes in Computer Science*, pages 85–96, Berlin, 1996. Springer-Verlag.

[19] J.-D. Boissonnat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, 44:1–29, 1988.

[20] G. Bordes, G. Giraudon, and O. Jamet. Road modeling based on a cartographic database for aerial image interpretation. In Förstner and Plümer [57], pages 123–139.

[21] I. N. Bronstein and K. A. Semendjajew. *Taschenbuch der Mathematik*. B. G. Teubner Verlagsgesellschaft, Leipzig, 22nd edition, 1985.

[22] J. W. Bruce and P. J. Giblin. *Curves and singularities: A geometrical introduction to singularity theory*. Cambridge University Press, Cambridge, 2nd edition, 1992.

[23] R. Brügelmann and W. Förstner. Noise estimation for color edge extraction. In W. Förstner and S. Ruwiedel, editors, *Robust Computer Vision: Quality of Vision Algorithms*, pages 90–106, Karlsruhe, 1992. Wichmann.

[24] A. Busch. Ein Verfahren zur Erkennung linienhafter Objekte in digitalen Bildern. In S. J. Pöppl and H. Handels, editors, *Mustererkennung*, Informatik aktuell, pages 616–623, Berlin, 1993. Springer-Verlag.

[25] A. Busch. Fast recognition of lines in digital images without user-supplied parameters. In *International Archives of Photogrammetry and Remote Sensing*, volume XXX, part 3/1, pages 91–97, 1994.

[26] A. Busch. A common framework for the extraction of lines and edges. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXI, part B3, pages 88–93, 1996.

[27] A. Busch. Extraction of roads and built-up areas from satellite imagery. In Leberl et al. [108], pages 277–292.

[28] J. Canny. Finding edges and lines in images. Technical Report 720, MIT Artificial Intelligence Laboratory, Cambridge, MA, June 1983.

[29] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, June 1986.

[30] N. Cerneaz and M. Brady. Finding curvilinear structures in mammograms. In N. Ayache, editor, *Computer Vision, Virtual Reality and Robotics in Medicine*, volume 905 of *Lecture Notes in Computer Science*, pages 372–382, Berlin, 1995. Springer-Verlag.

[31] J. S. Chen and G. Medioni. Detection, localization, and estimation of edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):191–198, February 1989.

[32] L. D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, November 1993.

[33] A. C. F. Colchester. Network representation of 2-d and 3-d images. In K. H. Höhne, H. Fuchs, and S. M. Pizer, editors, *3D Imaging in Medicine*, pages 45–62, Berlin, 1990. Springer-Verlag.

[34] C. Coppini, M. Demi, R. Poli, and G. Valli. An artificial vision system for X-ray images of human coronary trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):156–162, February 1993.

[35] M. E. de Gunst. *Knowledge-based Interpretation of Aerial Images for Updating of Road Maps*. PhD thesis, Delft University of Technology, Delft, 1996.

[36] E. de Micheli, B. Caprile, P. Ottonello, and V. Torre. Localization and noise in edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1106–1117, October 1989.

[37] D. Demigny and T. Kamlé. A discrete expression of Canny's criteria for step edge detector performances evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1199–1211, November 1997.

[38] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1:167–187, 1987.

[39] R. Deriche. Fast algorithms for low-level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–87, January 1990.

[40] R. Deriche. Recursively implementing the Gaussian and its derivatives. Rapport de Recherche 1893, INRIA, Sophia Antipolis, April 1993.

[41] R. Deriche and G. Giraudon. A computational approach for corner and vertex detection. *International Journal of Computer Vision*, 10(2):101–124, April 1993.

[42] M. R. Dobie, P. H. Lewis, and M. S. Nixon. Curvilinear network extraction from remotely sensed images. In *MVA '94 IAPR Workshop on Machine Vision Applications*, pages 123–126, 1994.

[43] J. Durbin. The first-passage density of a continuous Gaussian process to a general boundary. *J. Appl. Prob.*, 22:99–122, 1985.

[44] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. Technical Report TR93–055, Department of Computer Science, University of North Carolina, Chapel Hill, NC, 1993.

[45] P. H. Eichel, E. J. Delp, K. Koral, and A. J. Buda. A method for a fully automatic definition of coronary arterial edges from cineangiograms. *IEEE Transactions on Medical Imaging*, 7(4):313–320, December 1988.

[46] A. Fischer, T. H. Kolbe, and F. Lang. Integration of 2d and 3d reasoning for building reconstruction using a generic hierarchical model. In Förstner and Plümer [57], pages 159–180.

[47] M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf. Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique. *Computer Graphics and Image Processing*, 15:201–223, 1981.

[48] M. A. Fischler. The perception of linear structure: A generic linker. In *Image Understanding Workshop*, pages 1565–1579, San Francisco, CA, 1994. Morgan Kaufmann Publishers.

[49] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[50] M. A. Fischler and H. C. Wolf. Linear delineation. In *Computer Vision and Pattern Recognition*, pages 351–356, 1983.

[51] M. Fisz. *Wahrscheinlichkeitstheorie und mathematische Statistik*. VEB Deutscher Verlag der Wissenschaften, Berlin, 11th edition, 1989.

[52] M. G. Fleming, C. Steger, and A. B. Cognetta. Structural analysis of dermatoscopic imagery. *Skin Research and Technology*, 3:192, 1997.

[53] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Scale and the differential structure of images. *Image and Vision Computing*, 10(6):376–388, July 1992.

[54] W. Förstner. *Statistische Verfahren für die automatische Bildanalyse und ihre Bewertung bei der Objekterkennung und -vermessung.* Number 370 in Reihe C. Deutsche Geodätische Kommission, München, 1991.

[55] W. Förstner. A framework for low level feature extraction. In J.-O. Eklundh, editor, *Third European Conference on Computer Vision*, volume 801 of *Lecture Notes in Computer Science*, pages 383–394, Berlin, 1994. Springer-Verlag.

[56] W. Förstner. 10 pros and cons against performance characterization of vision algorithms. In H. I. Christensen, W. Förstner, and C. B. Madsen, editors, *Workshop on Performance Characteristics of Vision Algorithms*, pages 13–29, April 1996.

[57] W. Förstner and L. Plümer, editors. *Semantic Modeling for the Acquisition of Topographic Information from Images and Maps*, Basel, 1997. Birkhäuser Verlag.

[58] J. M. Gauch and S. M. Pizer. Multiresolution analysis of ridges and valleys in grey-scale images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):635–646, June 1993.

[59] B. Geiger. *Three-dimensional modeling of human organs and its application to dignosis and surgical planning.* PhD thesis, INRIA, Sophia Antipolis, April 1993.

[60] D. Geman and B. Jedynak. Shape recognition and twenty questions. Rapport de Recherche 2155, INRIA, Rocquencourt, November 1993.

[61] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):1–14, January 1996.

[62] G. Gerig, T. Koller, G. Székely, C. Brechbühler, and O. Kübler. Symbolic description of 3-d structures applied to cerebral vessel tree obtained from MR angiography volume data. In H. H. Barrett and A. F. Gmitro, editors, *Information Processing in Medical Imaging*, volume 687 of *Lecture Notes in Computer Science*, pages 94–111, Berlin, 1993. Springer-Verlag.

[63] G. Gerig, G. Székely, and T. Koller. Line-finding in 2-d and 3-d by multi-valued nonlinear diffusion of feature maps. In S. J. Pöppl and H. Handels, editors, *Mustererkennung*, Informatik aktuell, pages 289–296, Berlin, 1993. Springer-Verlag.

[64] G. Gerig, R. Kikinis, and F. A. Jolesz. Image processing of routine spin-echo MR images to enhance vascular structures: Comparison with MR angiography. In K. H. Höhne, H. Fuchs, and S. M. Pizer, editors, *3D Imaging in Medicine*, pages 121–132, Berlin, 1990. Springer-Verlag.

[65] F. Glazer. Curve finding by ridge detection and grouping. In W. Kropatsch and H. Bischof, editors, *Mustererkennung*, Informatik Xpress 5, pages 109–116, Wien, 1994. Springer-Verlag.

[66] L. D. Griffin, A. C. F. Colchester, and G. P. Robinson. Scale and segmentation of grey-level images using maximum gradient paths. In A. C. F. Colchester and D. J. Hawkes, editors, *Information Processing in Medical Imaging*, volume 511 of *Lecture Notes in Computer Science*, pages 256–272, Berlin, 1991. Springer-Verlag.

[67] L. D. Griffin and A. C. F. Colchester. Superficial and deep structure in linear diffusion scale space: isophotes, critical points and separatrices. *Image and Vision Computing*, 13(7):543–557, September 1995.

[68] A. Gruen, O. Kuebler, and P. Agouris, editors. *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, Basel, 1995. Birkhäuser Verlag.

[69] A. Gruen and H. Li. Semi-automatic road extraction by dynamic programming. In *International Archives of Photogrammetry and Remote Sensing*, volume XXX, part 3/1, pages 324–332, 1994.

[70] N. Haala and M. Hahn. Data fusion for the detection and reconstruction of buildings. In Gruen et al. [68], pages 211–220.

[71] G. Hake and D. Grünreich. *Kartographie*. De Gruyter, Berlin, 1994.

[72] R. M. Haralick. Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing*, 22:28–38, 1983.

[73] R. M. Haralick. Digital step edges from zero crossings of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):58–68, January 1984.

[74] R. M. Haralick. Propagating covariance in computer vision. In H. I. Christensen, W. Förstner, and C. B. Madsen, editors, *Workshop on Performance Characteristics of Vision Algorithms*, pages 1–12, April 1996.

[75] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, volume I. Addison-Wesley Publishing Company, Reading, MA, 1992.

[76] R. M. Haralick, L. T. Watson, and T. J. Laffey. The topographic primal sketch. *International Journal of Robotics Research*, 2(1):50–72, 1983.

[77] J. Hartman and J. Wernecke. *The VRML 2.0 Handbook: Building Moving Worlds on the Web*. Addison-Wesley Publishing Company, Reading, MA, 1996.

[78] C. Heipke, H. Mayer, C. Wiedemann, and O. Jamet. Evaluation of automatic road extraction. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXII, part 3–4W2, pages 151–160, 1997.

[79] O. Hellwich. *Linienextraktion aus SAR-Daten mit einem Markoff-Zufallsfeld-Modell*. Number 487 in Reihe C. Deutsche Geodätische Kommission, München, 1997.

[80] O. Hellwich, H. Mayer, and G. Winkler. Detection of lines in synthetic aperture radar (SAR) scenes. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXI, part B3, pages 312–320, 1996.

[81] P. J. Huber. *Robust Statistical Procedures*. SIAM, Philadelphia, PA, 2nd edition, 1996.

[82] A. Huertas and G. Medioni. Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):651–664, September 1986.

[83] L. A. Iverson and S. W. Zucker. Logical/linear operators for image curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10):982–996, October 1995.

[84] P. T. Jackway and M. Deriche. Scale-space properties of the multiscale morphological dilation-erosion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):38–51, January 1996.

[85] A. Jain, L. Hong, and R. Bolle. On-line fingerprint verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):302–314, April 1997.

[86] B. Jedynak and J.-P. Rozé. Tracking roads in satellite images by playing twenty questions. In Gruen et al. [68], pages 243–253.

[87] K. Jensen and D. Anastassiou. Subpixel edge localization and the interpolation of still images. *IEEE Transactions on Image Processing*, 4(3):285–295, March 1995.

[88] V. Juhan, B. Nazarin, K. Malkani, R. Bulot, J. M. Bartoli, and J. Sequeira. Geometrical modelling of abdominal aortic aneurysms. In J. Troccaz, E. Grimson, and R. Mösges, editors, *Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery*, volume 1205 of *Lecture Notes in Computer Science*, pages 243–252, Berlin, 1997. Springer-Verlag.

[89] B. B. Kimia and K. Siddiqi. Geometric heat equation and non-linear diffusion of shapes and images. Technical Report LEMS-124, Laboratory For Engineering Man/Machine Systems, Division of Engineering, Brown University, Providence, RI, July 1993.

[90] M. Kisworo, S. Venkatesh, and G. West. Modeling edges at subpixel accuracy using the local energy approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):405–410, April 1994.

[91] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[92] J. J. Koenderink. *Solid Shape*. MIT Press, Cambridge, MA, 1990.

[93] J. J. Koenderink and A. J. van Doorn. Two-plus-one-dimensional differential geometry. *Pattern Recognition Letters*, 15(5):439–443, May 1994.

[94] T. M. Koller, G. Gerig, G. Székely, and D. Dettwiler. Multi-scale detection of curvilinear structures in 2-d and 3-d image data. Technical Report BIWI-TR-153, Communication Technology Laboratory, Image Science, ETH Zürich, December 1994.

[95] T. M. Koller, G. Gerig, G. Székely, and D. Dettwiler. Multiscale detection of curvilinear structures in 2-d and 3-d image data. In *Fifth International Conference on Computer Vision*, pages 864–869, 1995.

[96] J. Koplowitz and V. Greco. On the edge location error for local maximum and zero-crossing edge detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1207–1212, December 1994.

[97] U. Köthe. Local appropriate scale in morphological scale-space. In B. Buxton and R. Cipolla, editors, *Fourth European Conference on Computer Vision*, volume 1064 of *Lecture Notes in Computer Science*, pages 219–228, Berlin, 1996. Springer-Verlag.

[98] U. Köthe. Local appropriate scale in morphological scale-space. Technical Report 96i001-FEGD, Fraunhofer-Institut für Graphische Datenverarbeitung, Rostock, January 1996.

[99] R. Kumar and A. R. Hanson. Robust methods for pose determination. In *NSF/ARPA Workshop on Performance versus Methodology in Computer Vision*, pages 41–57, 1994.

[100] I. S. Kweon and T. Kanade. Extracting topographic terrain features from elevation maps. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 59(2):171–182, March 1994.

[101] V. Lacroix and M. Acheroy. Constrained gradient: low-level tool for feature extraction. In Leberl et al. [108], pages 89–101.

[102] S. Lanser. *Modellbasierte Lokalisation gestützt auf monokulare Videobilder*. PhD thesis, Forschungs- und Lehreinheit Informatik IX, Technische Universität München, 1997.

[103] S. Lanser and W. Eckstein. A modification of Deriche's approach to edge detection. In *11th International Conference on Pattern Recognition*, volume III, pages 633–637, 1992.

[104] S. Lanser and C. Zierl. Robuste Kalibrierung von CCD-Sensoren für autonome, mobile Systeme. In R. Dillmann, U. Rembold, and T. Lüth, editors, *Autonome Mobile Systeme*, Informatik aktuell, pages 172–181, Berlin, 1995. Springer-Verlag.

[105] S. Lanser and C. Zierl. On the use of topological constraints within object recognition tasks. In *13th International Conference on Pattern Recognition*, volume I, pages 580–584, 1996.

[106] S. Lanser, C. Zierl, and R. Beutlhauser. Multibildkalibrierung einer CCD-Kamera. In G. Sagerer, S. Posch, and F. Kummert, editors, *Mustererkennung*, Informatik aktuell, pages 481–491, Berlin, 1995. Springer-Verlag.

[107] S. Lanser, C. Zierl, O. Munkelt, and B. Radig. MORAL — a vision-based object recogntion system for autonomous mobile systems. In *Computer Analysis of Images and Patterns '97*, Lecture Notes in Computer Science, pages 33–41, Berlin, 1997. Springer-Verlag.

[108] F. Leberl, R. Kalliany, and M. Gruber, editors. *Mapping Buildings, Roads and other Man-Made Structures from Images*, Wien, 1996. R. Oldenbourg Verlag.

[109] L. M. Lifshitz and S. M. Pizer. A multiresolution hierarchical approach to image segmentation based on intensity extrema. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):529–540, June 1990.

[110] C. Lin, A. Huertas, and R. Nevatia. Detection of buildings using perceptual grouping and shadows. In *Computer Vision and Pattern Recognition*, pages 62–69, 1994.

[111] T. Lindeberg. *Discrete scale-space theory and the scale-space primal sketch*. PhD thesis, Department of Numerical Analysis and Computing Science, KTH (Royal Institute of Technology), Stockholm, May 1991.

[112] T. Lindeberg. Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction. *Journal of Mathematical Imaging and Vision*, 3(4):349–376, 1993.

[113] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Dordrecht, 1994.

[114] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Computer Vision and Pattern Recognition*, pages 465–470, 1996.

[115] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. Technical Report ISRN KTH/NA/P–96/06–SE (CVAP 191), Department of Numerical Analysis and Computing Science, KTH (Royal Institute of Technology), Stockholm, 1996.

[116] A. M. López and J. Serrat. Tracing crease curves by solving a system of differential equations. In B. Buxton and R. Cipolla, editors, *Fourth European Conference on Computer Vision*, volume 1064 of *Lecture Notes in Computer Science*, pages 241–250, Berlin, 1996. Springer-Verlag.

[117] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface reconstruction algorithm. *Computer Graphics*, 21:163–169, 1987.

[118] C. Lorenz, I.-C. Carlsen, T. M. Buzug, C. Fassnacht, and J. Weese. A multi-scale line filter with automatic scale selection based on the hessian matrix for medical image segmentation. In B. ter Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, editors, *Scale-Space Theory in Computer Vision*, volume 1252 of *Lecture Notes in Computer Science*, pages 152–163, Berlin, 1997. Springer-Verlag.

[119] C. Lorenz, I.-C. Carlsen, T. M. Buzug, C. Fassnacht, and J. Weese. Multi-scale line segmentation with automatic estimation of width, contrast and tangential direction in 2d and 3d medical images. In J. Troccaz, E. Grimson, and R. Mösges, editors, *Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery*, volume 1205 of *Lecture Notes in Computer Science*, pages 233–242, Berlin, 1997. Springer-Verlag.

[120] D. G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992.

[121] E. P. Lyvers, O. R. Mitchell, M. L. Akey, and A. P. Reeves. Subpixel measurements using a moment-based edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1293–1309, December 1989.

[122] N. Magnenat-Thalmann and D. Thalmann. *Image Synthesis: Theory and Practice*. Springer-Verlag, Berlin, 1987.

[123] J. B. A. Maintz, P. A. van den Elsen, and M. A. Viergever. Evaluation of ridge seeking operators for multimodality medical image matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):353–365, April 1996.

[124] D. Maio and D. Maltoni. Direct gray-scale minutiae detection in fingerprints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):27–40, January 1997.

[125] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London, Series B*, 207:187–217, 1980.

[126] Y. Masutani, T. Kurihara, M. Suzuki, and T. Dohi. Quantitative vascular shape analysis for 3d MR-angiography using mathematical morphology. In N. Ayache, editor, *Computer Vision, Virtual Reality and Robotics in Medicine*, volume 905 of *Lecture Notes in Computer Science*, pages 449–454, Berlin, 1995. Springer-Verlag.

[127] H. Mayer, I. Laptev, A. Baumgartner, and C. Steger. Automatic road extraction based on multi-scale modeling, context, and snakes. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXII, part 3–2W3, pages 106–113, 1997.

[128] H. Mayer and C. Steger. A new approach for line extraction and its integration in a multi-scale, multi-abstraction-level road extraction system. In Leberl et al. [108], pages 331–348.

[129] D. M. McKeown Jr. and J. L. Denlinger. Cooperative methods for road tracking in aerial imagery. In *Computer Vision and Pattern Recognition*, pages 662–672, 1988.

[130] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38:113–125, 1994.

[131] O. Monga, N. Armande, and P. Montesinos. Thin nets and crest lines: Application to satellite data and medical images. Rapport de Recherche 2480, INRIA, Rocquencourt, February 1995.

[132] O. Monga, R. Deriche, and J.-M. Rocchisani. 3d edge detection using recursive filtering: Application to scanner images. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(1):76–87, January 1991.

[133] O. Monga, R. Langagne, R. Deriche, and S. D. Ma. A multi-scale approach for crest line extraction in 3d medical images. In *MVA '94 IAPR Workshop on Machine Vision Applications*, pages 34–37, 1994.

[134] P. Montesinos and L. Alquier. Perceptual organization of thin networks with active contour functions applied to medical and aerial images. In *13th International Conference on Pattern Recognition*, volume I, pages 647–651, 1996.

[135] B. S. Morse, S. M. Pizer, and A. Liu. Multiscale medial analysis of medical images. In H. H. Barrett and A. F. Gmitro, editors, *Information Processing in Medical Imaging*, volume 687 of *Lecture Notes in Computer Science*, pages 112–131, Berlin, 1993. Springer-Verlag.

[136] B. S. Morse, S. M. Pizer, and A. Liu. Multiscale medial analysis of medical images. *Image and Vision Computing*, 12(6):327–338, July 1994.

[137] B. S. Morse, S. M. Pizer, D. T. Puff, and C. Gu. Zoom-invariant vision of figural shape: Effects on cores of image disturbances. Technical Report TR96-005, Department of Computer Science, University of North Carolina, Chapel Hill, NC, 1996.

[138] L. R. Nackman. Two-dimensional critical point configuration graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(4):442–450, April 1984.

[139] L. Najman and M. Schmitt. Watershed of a continuous function. *Signal Processing*, 38:99–112, 1994.

[140] V. S. Nalwa and T. O. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):699–714, November 1986.

[141] P. Neskovic and B. B. Kimia. Geometric smoothing of 3d surfaces and non-linear diffusion of 3d images. Technical Report LEMS-144, Laboratory For Engineering Man/Machine Systems, Division of Engineering, Brown University, Providence, RI, November 1995.

[142] M. Nielsen, L. Florack, and R. Deriche. Regularization, scale-space, and edge detection filters. In B. Buxton and R. Cipolla, editors, *Fourth European Conference on Computer Vision*, volume 1065 of *Lecture Notes in Computer Science,* pages 70–81, Berlin, 1996. Springer-Verlag.

[143] O. F. Olsen. Multi-scale segmentation of grey-scale images. Technical Report DIKU–96/30, Datalogisk Institut, Københavns Universitet, Copenhagen, September 1996.

[144] O. F. Olsen and M. Nielsen. Generic events for the gradient squared with application to multi-scale segmentation. In B. ter Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, editors, *Scale-Space Theory in Computer Vision*, volume 1252 of *Lecture Notes in Computer Science*, pages 101–112, Berlin, 1997. Springer-Verlag.

[145] I. Overington and P. Greenway. Practical first-difference edge detection with subpixel accuracy. *Image and Vision Computing*, 5(3):217–224, August 1987.

[146] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, Inc., New York, NY, 3rd edition, 1991.

[147] F. Pedersini, A. Sarti, and S. Tubaro. Estimation and compensation of subpixel edge localization error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1278–1284, November 1997.

[148] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.

[149] S. M. Pizer, D. Eberly, B. S. Morse, and D. Fritsch. Zoom-invariant vision of figural shape: The mathematics of cores. Technical Report TR96-004, Department of Computer Science, University of North Carolina, Chapel Hill, NC, 1996.

[150] B. Plietker. Semiautomatic revision of street objects in ATKIS database DLM 25/1. In *International Archives of Photogrammetry and Remote Sensing*, volume XXX, part 4, pages 311–317, 1994.

[151] I. R. Porteous. *Geometric differentiation for the intelligence of curves and surfaces*. Cambridge University Press, Cambridge, 1994.

[152] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2nd edition, 1992.

[153] V. Prinet, O. Monga, C. Ge, S. L. Xie, and S. D. Ma. Thin network extraction in 3d images: Application to medical angiograms. In *13th International Conference on Pattern Recognition*, volume III, pages 386–390, 1996.

[154] J. H. Rieger. Topographical properties of generic images. *International Journal of Computer Vision*, 23(1):79–92, January 1997.

[155] D. F. Rogers. *Procedural Elements for Computer Graphics*. McGraw-Hill, Inc., New York, NY, 1985.

[156] R. Rothe. Zum Problem des Talwegs. In *Sitzungsberichte der Berliner Mathematischen Gesellschaft*, volume XIV, pages 51–68, Leipzig, 1915. B. G. Teubner Verlag.

[157] R. Ruskoné, S. Airault, and O. Jamet. Road network interpretation: A topological hypothesis driven system. In *International Archives of Photogrammetry and Remote Sensing*, volume XXX, part 3/2, pages 711–717, 1994.

[158] I. Rychlik. A note on Durbin's formula for the first-passage density. *Statistics & Probability Letters*, 5(6):425–428, October 1987.

[159] Y. Sato, S. Nakajima, H. Atsumi, T. Koller, G. Gerig, S. Yoshida, and R. Kikinis. 3d multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. In J. Troccaz, E. Grimson, and R. Mösges, editors, *Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery*, volume 1205 of *Lecture Notes in Computer Science*, pages 213–222, Berlin, 1997. Springer-Verlag.

[160] R. Sedgewick. *Algorithms in C.* Addison-Wesley Publishing Company, Reading, MA, 1990.

[161] S. Sengupta and S. C. Sahasrabudhe. Scale space displacement of zero-crossings of $\nabla^2 G$ operated images for convex bodies and convex sets. *Signal Processing*, 47:279–285, 1995.

[162] M. Shah, A. Sood, and R. Jain. Pulse and staircase edge models. *Computer Vision, Graphics, and Image Processing*, 34:321–343, 1986.

[163] J. Shen. On multi-edge detection. *Graphical Models and Image Processing*, 58(2):101–114, March 1996.

[164] J. Shen and S. Castan. An optimal linear operator for step edge detection. In *Computer Vision and Pattern Recognition*, pages 109–114, 1986.

[165] J. Shen and S. Castan. An optimal linear operator for step edge detection. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 54(2):112–133, March 1992.

[166] K. Siddiqi and B. B. Kimia. A shock grammar for recognition. In *Computer Vision and Pattern Recognition*, pages 507–513, 1996.

[167] C. Steger. Extracting curvilinear structures: A differential geometric approach. In B. Buxton and R. Cipolla, editors, *Fourth European Conference on Computer Vision*, volume 1064 of *Lecture Notes in Computer Science*, pages 630–641, Berlin, 1996. Springer-Verlag.

[168] C. Steger. Extracting lines using differential geometry and Gaussian smoothing. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXI, part B3, pages 821–826, 1996.

[169] C. Steger. Extraction of curved lines from images. In *13th International Conference on Pattern Recognition*, volume II, pages 251–255, 1996.

[170] C. Steger. On the calculation of arbitrary moments of polygons. Technical Report FGBV–96–05, Forschungsgruppe Bildverstehen (FG BV), Informatik IX, Technische Universität München, September 1996.

[171] C. Steger. An unbiased detector of curvilinear structures. Technical Report FGBV–96–03, Forschungsgruppe Bildverstehen (FG BV), Informatik IX, Technische Universität München, July 1996.

[172] C. Steger. Removing the bias from line detection. In *Computer Vision and Pattern Recognition*, pages 116–122, 1997.

[173] C. Steger, C. Glock, W. Eckstein, H. Mayer, and B. Radig. Model-based road extraction from images. In Gruen et al. [68], pages 275–284.

[174] W. Stolz, O. Braun-Falco, P. Bilek, M. Landthaler, and A. B. Cognetta. *Color Atlas of Dermatoscopy*. Blackwell Science, 1994.

[175] A. A. Sveshnikov. *Applied Methods of the Theory of Random Functions*. Pergamon Press, Oxford, 1966.

[176] A. J. Tabatabai and O. R. Mitchell. Edge location to subpixel values in digital imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):188–201, March 1984.

[177] S. Tabbone and D. Ziou. Subpixel positioning of edges for first and second order operators. In *11th International Conference on Pattern Recognition*, volume III, pages 655–658, 1992.

[178] H. D. Tagare and R. J. P. deFigueiredo. On the localization performance measure and optimal edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1186–1190, December 1990.

[179] L. Tang. *Einsatz der Rasterdatenverarbeitung zum Aufbau digitaler Geländemodelle*. PhD thesis, Fakultät für Bauingenieurwesen, Technische Universität Graz, 1991.

[180] B. M. ter Haar Romeny. Introduction to scale-space theory: Multiscale geometric image analysis. Technical Report ICU-96-21, Utrecht University, Utrecht, September 1996.

[181] B. M. ter Haar Romeny, L. M. Florack, J. J. Koenderink, and M. A. Viergever. Scale space: Its natural operators and differential invariants. In A. C. F. Colchester and D. J. Hawkes, editors, *Information Processing in Medical Imaging*, volume 511 of *Lecture Notes in Computer Science*, pages 239–255, Berlin, 1991. Springer-Verlag.

[182] B. M. ter Haar Romeny, L. M. Florack, A. H. Salden, and M. A. Viergever. Higher order differential structure of images. *Image and Vision Computing*, 12(6):317–325, July/August 1994.

[183] J.-P. Thirion and A. Gourdon. The marching lines algorithm: New results and proofs. Rapport de Recherche 1881, INRIA, Sophia Antipolis, April 1993.

[184] J.-P. Thirion and A. Gourdon. The 3d marching lines algorithhm. *Graphical Models and Image Processing*, 58(6):503–509, November 1996.

[185] F. Ulupinar and G. Medioni. Refining ridges detected by a LoG operator. *Computer Vision, Graphics, and Image Processing*, 51:275–298, 1990.

[186] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, June 1991.

[187] L. Wang and T. Pavlidis. Detection of curved and straight segments from gray scale topography. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 58(3):352–365, November 1993.

[188] L. Wang and T. Pavlidis. Direct gray-scale extraction of features for character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1053–1067, October 1993.

[189] J. Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley Publishing Company, Reading, MA, 1994.

[190] J. Williams and L. Wolff. Analysis of the pulmonary vascular tree using differential geometry based vector fields. *Computer Vision and Image Understanding*, 65(2):226–236, February 1997.

[191] A. P. Witkin. Scale-space filtering. In *Eigth International Joint Conference on Artificial Intelligence*, volume 2, pages 1019–1022, 1983.

[192] M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.1*. Addison-Wesley Publishing Company, Reading, MA, 2nd edition, 1997.

[193] P. Wunsch. *Modellbasierte 3-D Objektlageschätzung für visuell geregelte Greifvorgänge in der Robotik*. PhD thesis, Forschungs- und Lehreinheit Informatik IX, Technische Universität München, 1997.

[194] I. T. Young and L. J. van Vliet. Recursive implementation of the Gaussian filter. *Signal Processing*, 44:139–151, 1995.

[195] A. L. Yuille and T. A. Poggio. Scaling theorems for zero crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):15–25, January 1986.

[196] J. Zerubia and N. Merlet. Classical mechanics and roads detection in SPOT images. Rapport de Recherche 1889, INRIA, Sophia Antipolis, April 1993.

[197] A. Zlotnick and P. Carnine, Jr. Finding road seeds in aerial images. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 57(2):243–260, March 1993.