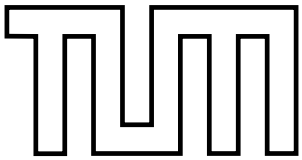
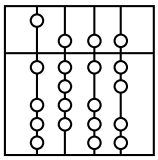




An Unbiased Detector of Curvilinear Structures

Carsten Steger

Technical Report FGBV–96–03

July 1996

Forschungsgruppe Bildverstehen (FG BV), Informatik IX
Technische Universität München

Technical Report FGBV–96–03

Address: Carsten Steger
Forschungsgruppe Bildverstehen, Informatik IX
Technische Universität München
Orleansstraße 34
81667 München
Germany
WWW: http://wwwradig.informatik.tu-muenchen.de/forschung/fgbv/
E-mail: stegerc@informatik.tu-muenchen.de
fgbv@informatik.tu-muenchen.de

An Unbiased Detector of Curvilinear Structures

Carsten Steger

***Abstract*—*The extraction of curvilinear structures is an important low-level operation in computer vision that has many applications. Most existing operators use a simple model for the line that is to be extracted, i.e., they do not take into account the surroundings of a line. This leads to the undesired consequence that the line will be extracted in the wrong position whenever a line with different lateral contrast is extracted. In contrast, the algorithm proposed in this paper uses an explicit model for lines and their surroundings. By analyzing the scale-space behaviour of a model line profile, it is shown how the bias that is induced by asymmetrical lines can be removed. Furthermore, the algorithm not only returns the precise sub-pixel line position, but also the width of the line for each line point, also with sub-pixel accuracy.*

1 Introduction

Extracting curvilinear structures, often simply called lines, in digital images is an important low-level operation in computer vision that has many applications. In photogrammetric and remote sensing tasks it can be used to extract linear features, including roads, railroads, or rivers, from satellite or low resolution aerial imagery, which can be used for the capture or update of data for geographic information systems [1, 2]. In addition it is useful in medical imaging for the extraction of anatomical features, e.g., blood vessels from an X-ray angiogram [3] or the bones in the skull from a CT or MR image [4].

The published schemes for line detection can be classified into three categories. The first approach detects lines by considering the gray values of the image only [5, 6, 7] and uses purely local criteria, e.g., local gray value differences. Since this will generate many false hypotheses for line points, elaborate and computationally expensive perceptual grouping schemes have to be used to select salient lines in the image [8, 9, 10, 7]. Furthermore, lines cannot be extracted with sub-pixel accuracy.

The second approach is to regard lines as objects having parallel edges [11, 12, 13]. In a first step, the local direction of a line is determined for each pixel. Then two edge detection filters are applied in the direction perpendicular to the line, where each filter is tuned to detect either the left or right edge of the line. The responses of each filter are combined in a non-linear way to yield the final response of the operator [11]. The advantage of this approach is that since the edge detection filters are based on the derivatives of Gaussian kernels, the procedure can be iterated over the scale-space parameter σ to detect lines of arbitrary widths. However, because special directional edge detection filters that are not separable have to be constructed, the approach is computationally expensive.

The final approach is to regard the image as a function $z(x, y)$ and extract lines from it by using various differential geometric properties of this function. The basic idea behind these algorithms is to locate the positions of ridges and ravines in the image function. These methods can be further divided according to which property they use.

The first sub-category defines ridges as the point on a contour line of the image, often also called isohypse or isophote, where the curvature of the contour line has a maximum [4, 14, 15]. One way to do this is to extract the contour lines explicitly, to find the points of maximum curvature on them, and to then link the extracted points into ridges [14]. However, this scheme suffers from two main drawbacks. Firstly, since no contour lines will be found for perfectly flat ridges, such ridges will be labeled as an extended peak. Furthermore, 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 maxima in a curvature image [4, 15]. However, this procedure will also fail for perfectly flat ridges. While Sard's theorem [16] 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, the ridge positions found by this operator will often be in wrong positions due to the nature of the differential geometric property used, even for images without noise [4, 17].

In the second sub-category, ridges are found at points where one of the principal curvatures of the image assumes a local maximum [18, 15], which is analogous to the approach taken to define ridges in advanced differential geometry [19]. For lines with a flat profile it has the problem that two separate points of maximum curvature symmetric to the true line position will be found [15]. This is clearly undesirable.

In the third sub-category, ridges and ravines are detected by locally approximating the image function by its second or third order Taylor polynomial. The coefficients of this polynomial are usually determined by using the facet model, i.e., by a least squares fit of the polynomial to the image data over a window of a certain size [20, 21, 22, 23, 24, 25]. The direction of the line is determined from the Hessian matrix of the Taylor polynomial. Line points are then found by selecting pixels that have a high second directional derivative perpendicular to the line direction. The advantage of this approach is that lines can be detected with sub-pixel accuracy without having to construct specialized directional filters. However, because the convolution masks that are used to determine the coefficients of the Taylor polynomial are rather poor estimators for the first and second partial derivatives, this approach usually leads to multiple responses to a single line, especially when masks larger than 5×5 are used to suppress noise. Therefore, the approach does not scale well and cannot be used to detect lines that are wider than the mask size. For these reasons, a number of line detectors have been proposed that use Gaussian masks to detect the ridge points [26, 27, 15]. These have the advantage that they can be tuned for a certain line width by selecting an appropriate σ. It is also possible to select the appropriate σ for each image point by iterating through scale space [26]. However, since the surroundings of the line are not modeled the extracted line position becomes progressively inaccurate as σ increases.

Evidently, very few approaches to line detection consider the task of extracting the line width along with the line position. Most of them do this by an iteration through scale-space while selecting the scale, i.e., the σ, that yields the maximum value to a certain scale-normalized response as the line width [11, 26]. However, this is computationally very expensive, especially if one is only interested in lines in a certain range of widths. Furthermore, these approaches will only yield a relatively rough estimate of the line width since, by necessity, the scale-space is quantized in rather rough intervals. A different approach is given in [28], where, lines and edges are extracted in one simultaneous operation. For each line point two corresponding edge

points are matched from the resulting description. This approach has the advantage that lines and their corresponding edges can in principle be extracted with sub-pixel accuracy. However, since a third order facet model is used, the same problems that are mentioned above apply. Furthermore, since the approach does not use an explicit model for a line, the location of the corresponding edge of a line is often not meaningful because the interaction between a line and its corresponding edges is neglected.

In this paper, an approach to line detection is presented that uses an explicit model for lines, and various types of line profile models of increasing sophistication are discussed. A scale-space analysis is carried out for each of the models. This analysis is used to derive an algorithm in which lines and their widths can be extracted with sub-pixel accuracy. The algorithm uses a modification of the differential geometric approach described above to detect lines and their corresponding edges. Because Gaussian masks are used to estimate the derivatives of the image, the algorithm scales to lines of arbitrary widths while always yielding a single response. Furthermore, since the interaction between lines and their corresponding edges is explicitly modeled, the bias in the extracted line and edge position can be predicted analytically, and can thus be removed. Therefore, line position and width will always correspond to a semantically meaningful location in the image.

The outline of the paper is as follows. In Section 2 models for lines in 1D and 2D images are presented and algorithms to extract individual line points are discussed. Section 3 presents an algorithm to link the individual line points into lines and junctions. The extraction of the width of the line is discussed in Section 4. Section 5 describes an algorithm to correct the line position and width to their true values. Finally, Section 6 concludes the paper.

2 Detection of Line Points

2.1 Models for Line Profiles in 1D

Many approaches to line detection consider lines in 1D to be bar-shaped, i.e., the ideal line of width $2w$ and height h is assumed to have a profile given by

$$f_b(x) = \begin{cases} h, & |x| \leq w \\ 0, & |x| > w \end{cases} . \tag{1}$$

However, due to sampling effects of the sensor lines often do not have this profile. Figure 1 shows a typical profile of a line in an aerial image, where no flat bar profile is apparent. Therefore, let us first consider lines with a parabolically shaped profile because it will make the derivation of the algorithm clearer and provide us with criteria that should be fulfilled for arbitrary line profiles. The ideal line of width $2w$ and height h is given by

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

The line detection algorithm will be developed for this type of profile, but the implications of applying it to bar-shaped lines will be considered later on.

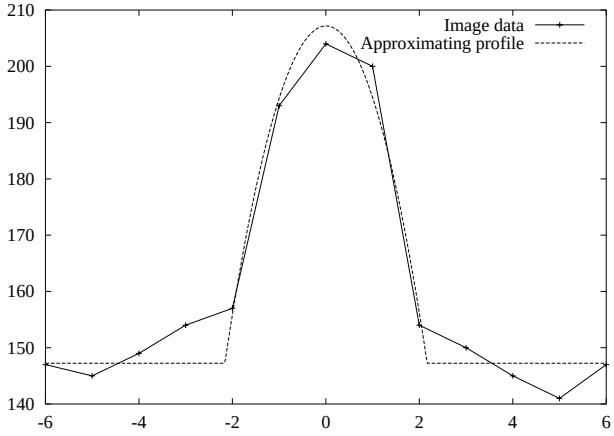


Figure 1: Profile of a line in an aerial image and approximating parabolic line profile.

2.2 Detection of Lines in 1D

In order to detect lines with a profile given by (2) 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, 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, assumtions, it is the only kernel that makes the inherently ill-posed problem of estimating the derivatives of a noisy function well-posed [29, 30]. The Gaussian kernels are given by:

$$g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{3}$$

$$g'_\sigma(x) = \frac{-x}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2}{2\sigma^2}} \tag{4}$$

$$g''_\sigma(x) = \frac{x^2 - \sigma^2}{\sqrt{2\pi}\sigma^5} e^{-\frac{x^2}{2\sigma^2}} \quad . \tag{5}$$

The responses, i.e., the estimated derivatives, will then be:

$$\begin{aligned} r_p(x,\sigma,w,h) &= g_\sigma(x) * f_p(x) \\ &= \frac{h}{w^2}\Big((w^2 - x^2 - \sigma^2)(\phi_\sigma(x+w) - \phi_\sigma(x-w)) - \\ &\qquad 2\sigma^2 x(g_\sigma(x+w) - g_\sigma(x-w)) - \\ &\qquad \sigma^4(g'_\sigma(x+w) - g'_\sigma(x-w))\Big) \\ r'_p(x,\sigma,w,h) &= g'_\sigma(x) * f_p(x) \end{aligned} \tag{6}$$

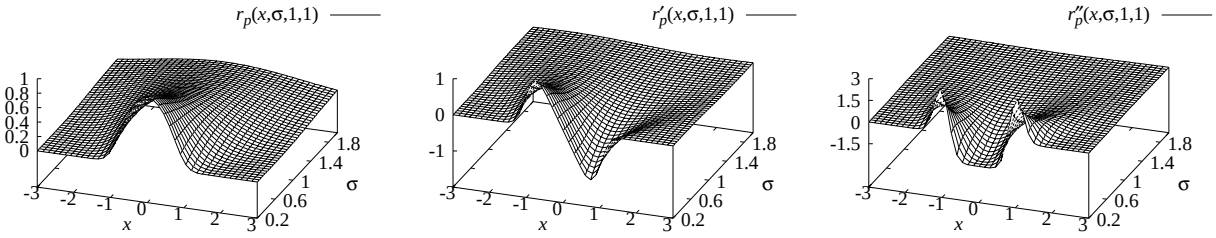


Figure 2: Scale-space behaviour of the parabolic line f_p when convolved with the derivatives of Gaussian kernels for $x \in [-3, 3]$ and $\sigma \in [0.2, 2]$.

$$
\begin{aligned}
&= \frac{h}{w^2}\Big(-2x(\phi_\sigma(x+w) - \phi_\sigma(x-w)) + \\
&\quad (w^2 - x^2 - 3\sigma^2)(g_\sigma(x+w) - g_\sigma(x-w)) - \\
&\quad 2\sigma^2 x(g'_\sigma(x+w) - g'_\sigma(x-w)) - \\
&\quad \sigma^4(g''_\sigma(x+w) - g''_\sigma(x-w))\Big) \qquad (7)
\end{aligned}
$$

$$
\begin{aligned}
r''_p(x,\sigma,w,h) &= g''_\sigma(x) * f_p(x) \\
&= \frac{h}{w^2}\Big(-2(\phi_\sigma(x+w) - \phi_\sigma(x-w)) - \\
&\quad 4x(g_\sigma(x+w) - g_\sigma(x-w)) + \\
&\quad (w^2 - x^2 - 5\sigma^2)(g'_\sigma(x+w) - g'_\sigma(x-w)) - \\
&\quad 2\sigma^2 x(g''_\sigma(x+w) - g''_\sigma(x-w)) - \\
&\quad \sigma^4(g'''_\sigma(x+w) - g'''_\sigma(x-w))\Big) \qquad (8)
\end{aligned}
$$

where

$$\phi_\sigma(x) = \int_{-\infty}^{x} e^{-\frac{t^2}{2\sigma^2}} dt \ . \qquad (9)$$

Equations (6)–(8) 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 Gaussian kernels. Figure 2 shows the responses for an ideal line with $w = 1$ and $h = 1$, i.e., a bright line 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 σ. Furthermore, $r''_p(x, \sigma, w, h)$ takes on its maximum negative value at $x = 0$ for all σ. Hence it is possible to determine the precise location of the line for all σ. In addition it can be seen that the ideal line will be flattened out as σ increases as a result of smoothing. This means that if large values for σ are used, the threshold to select salient lines will have to be set to an accordingly smaller value.

Let us now consider the more common case of a bar-shaped profile. 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 will be:

$$r_b(x,\sigma,w,h) = g_\sigma(x) * f_b(x)$$

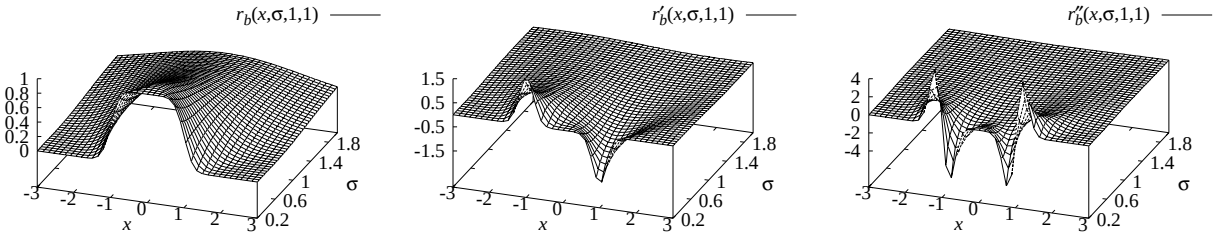


Figure 3: Scale-space behaviour of the bar-shaped line f_b when convolved with the derivatives of Gaussian kernels for $x \in [-3, 3]$ and $\sigma \in [0.2, 2]$.

$$= h\Big(\phi_\sigma(x+w) - \phi_\sigma(x-w)\Big) \tag{10}$$

$$\begin{aligned} r'_b(x,\sigma,w,h) &= g'_\sigma(x) * f_b(x) \\ &= h\Big(g_\sigma(x+w) - g_\sigma(x-w)\Big) \end{aligned} \tag{11}$$

$$\begin{aligned} r''_b(x,\sigma,w,h) &= g''_\sigma(x) * f_b(x) \\ &= h\Big(g'_\sigma(x+w) - g'_\sigma(x-w)\Big)\ . \end{aligned} \tag{12}$$

Figure 3 shows the scale-space behaviour 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 $g_\sigma(x)$. However, the second derivative $r''_b(x,\sigma,w,h)$ will not take on its maximum negative value for small σ. In fact, for $\sigma \leq 0.2w$ it will be very close to zero. Furthermore, there will be two distinct minima in the interval $[-w, w]$. It is, however, desirable for $r''_b(x,\sigma,w,h)$ to exhibit a clearly defined minimum at $x = 0$ since salient lines are detected by this value. After some lengthy calculations it can be shown that

$$\sigma \geq \frac{w}{\sqrt{3}} \tag{13}$$

has to hold for this. Furthermore, it can be shown 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 on σ must be observed.

In addition to this, (11) and (12) can be used to derive how the edges of a line will behave in scale-space. Since this analysis involves equations which cannot be solved analytically, the calculations must be done using a root finding algorithm [31]. Figure 4 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 (12) it is apparent that the edges of a line can never move closer than σ to the real line, and thus the width of the line will be estimated significantly too large for narrow lines. However, since it is possible to invert the map that describes the edge position, the edges can be localized very precisely once they are extracted from an image.

The discussion so far has assumed that lines have the same contrast on both sides, which is rarely the case for real images. For simplicity, only asymetrical bar-shaped lines

$$f_a(x) = \begin{cases} 0, & x < -w \\ 1, & |x| \leq w \\ a, & x > w\ . \end{cases} \tag{14}$$

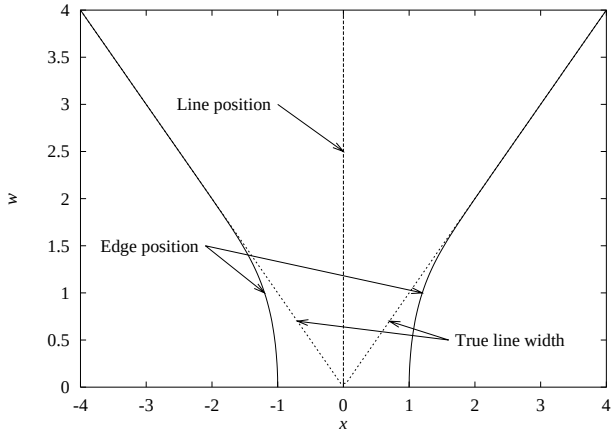


Figure 4: Location of a line with width $w \in [0, 4]$ and its edges for $\sigma = 1$.

are considered ($a \in [0, 1]$). General lines of height h can be obtained by considering a scaled asymmetrical profile, i.e., $hf_a(x)$. However, this changes nothing in the discussion that follows since h cancels out in every calculation. The corresponding responses are given by:

$$r_a(x, \sigma, w, a) = \phi_\sigma(x + w) + (a - 1)\phi_\sigma(x - w) \quad (15)$$

$$r'_a(x, \sigma, w, a) = g_\sigma(x + w) + (a - 1)g_\sigma(x - w) \quad (16)$$

$$r''_a(x, \sigma, w, a) = g'_\sigma(x + w) + (a - 1)g'_\sigma(x - w) \ . \quad (17)$$

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

$$l = -\frac{\sigma^2}{2w} \ln(1 - a) \ . \quad (18)$$

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

$$a \leq 1 - e^{-\frac{2w^2}{\sigma^2}} \ . \quad (19)$$

The location of the corresponding edges can again only be computed numerically. Figure 5 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 posseses the smaller edge gradient.

Note that (18) gives an explicit formula for the bias of the line extractor. Suppose that we knew w and a 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 5 will describe the solution to this problem.

Because the asymmetrical line case is by far the most likely case in any given image it is adopted as the basic model for a line in an image. It is apparent from the analysis above 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 will fail to return very meaningful results.

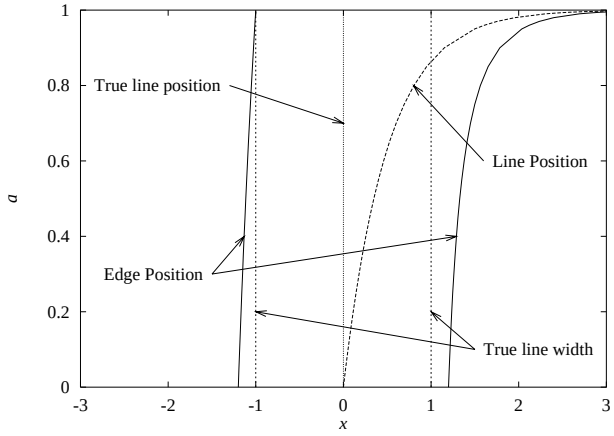


Figure 5: Location of an asymmetrical line and its corresponding edges with width $w = 1$, $\sigma = 1$, and $a \in [0, 1]$.

2.3 Lines in 1D, Discrete Case

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 were chosen as convolutions masks, mainly because the scale-space analysis of Section 2.2 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 will be given by:

$$g_{n,\sigma} = \phi_\sigma(n + \tfrac{1}{2}) - \phi_\sigma(n - \tfrac{1}{2}) \quad (20)$$

$$g'_{n,\sigma} = g_\sigma(n + \tfrac{1}{2}) - g_\sigma(n - \tfrac{1}{2}) \quad (21)$$

$$g''_{n,\sigma} = g'_\sigma(n + \tfrac{1}{2}) - g'_\sigma(n - \tfrac{1}{2}) \ . \quad (22)$$

For the implementation the approximation error is set to 10^{-4} in each case because for images that contain gray values in the range $[0, 255]$ this precision is sufficient. Of course, other schemes, like discrete analogon of the Gaussian [32] or a recursive computation [33], are suitable for the implementation as well. However, for small σ the scale-space analysis will have to be slightly modified because these filters have different coefficients compared to the integrated Gaussian.

The second problem that must be solved is the determination of 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

$$p(x) = r + r'x + \frac{1}{2}r''x^2 \ . \quad (23)$$

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

$$x = -\frac{r'}{r''} \quad . \tag{24}$$

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. Please 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 will be deferred to Section 4.

2.4 Detection of Lines in 2D

Curvilinear structures in 2D can be modeled as curves $s(t)$ that exhibit a characteristic 1D line profile, i.e., f_a, in the direction perpendicular to the line, i.e., perpendicular to $s'(t)$. Let this direction be $n(t)$. 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 be made about the derivatives in the direction of $s'(t)$. For example, let $z(x, y)$ be an image that results from sweeping the profile f_a along a circle $s(t)$ of radius r. When this image is convolved with the derivatives of a Gaussian kernel, the second directional derivative perpendicular to $s'(t)$ will have a large negative value, as desired. However, the second directional derivative along $s'(t)$ will also be non-zero.

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 will have to be estimated, and this can be done by convolving the image with the following kernels

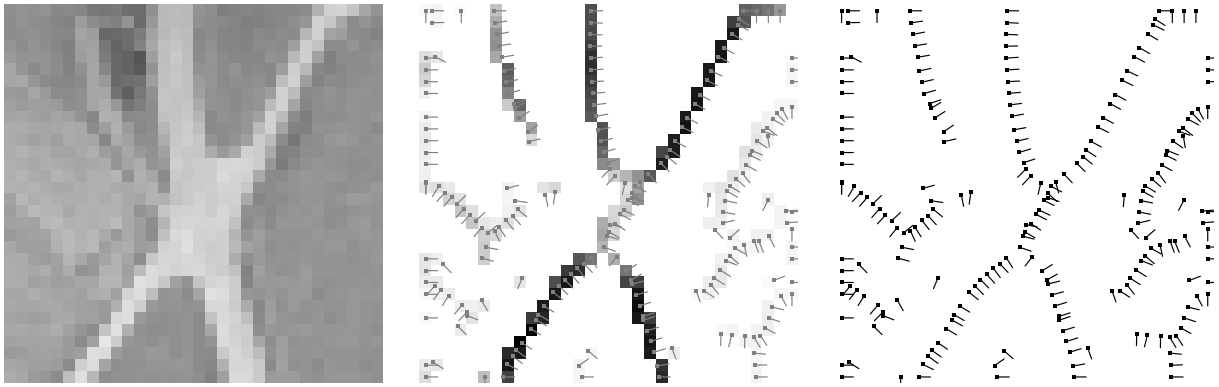
$$\begin{aligned}
g_{x,\sigma}(x, y) &= g_\sigma(y) g'_\sigma(x) && (25)\\
g_{y,\sigma}(x, y) &= g'_\sigma(y) g_\sigma(x) && (26)\\
g_{xx,\sigma}(x, y) &= g_\sigma(y) g''_\sigma(x) && (27)\\
g_{xy,\sigma}(x, y) &= g'_\sigma(y) g'_\sigma(x) && (28)\\
g_{yy,\sigma}(x, y) &= g''_\sigma(y) g_\sigma(x) \quad . && (29)
\end{aligned}$$

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

$$H(x, y) = \begin{pmatrix} r_{xx} & r_{xy} \\ r_{xy} & r_{yy} \end{pmatrix} \quad . \tag{30}$$

The calculation can be done in a numerically stable and efficient way by using one Jacobi rotation to annihilate the r_{xy} term [31]. 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 will be used to determine whether the first directional derivative along (n_x, n_y) vanishes within the current pixel. This point will be given by

$$(p_x, p_y) = (tn_x, tn_y) \tag{31}$$



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

Figure 6: 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.

where

$$t = -\frac{r_x n_x + r_y n_y}{r_{xx} n_x^2 + 2 r_{xy} n_x n_y + r_{yy} n_y^2} \quad . \tag{32}$$

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.

2.5 Example

Figures 6(b) and (c) give an example of the results obtainable with the presented approach. Here, bright line points were extracted from the input image given in Fig. 6(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 6 it might appear, if an 8-neighborhood is used, that the proposed approach returns multiple responses to each line. However, when the sub-pixel location of each line point 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 prefectly. Therefore, linking will be considerably easier than in approaches that yield multiple responses, e.g., [27, 21, 22], and no thinning operation is needed [34].

3 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 right after the extraction of the line points because the later stages of determining 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 Fig. 6, 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.

3.1 Linking Algorithm

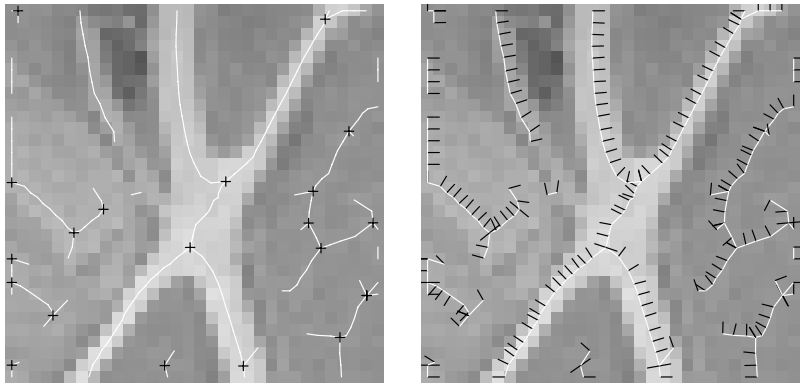
In order to facilitate later mid-level vision processes, e.g., perceptual grouping, the data structure that results from the linking process should contain explicit information about the lines as well as the junctions between them. This data structure should be topologically sound in the sense that junctions are represented by points and not by extended areas as in [21] or [23]. Furthermore, since the presented approach yields only single responses to each line, 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 2 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 α), 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. 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. 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. 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.

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 [35].

The problem of orienting the normals $n(t)$ of the line is solved by the following procedure. Firstly, 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 difference between the angle of the normal of the previous point and the current point is chosen



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

Figure 7: 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.

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 no 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 13×13. 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). 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.

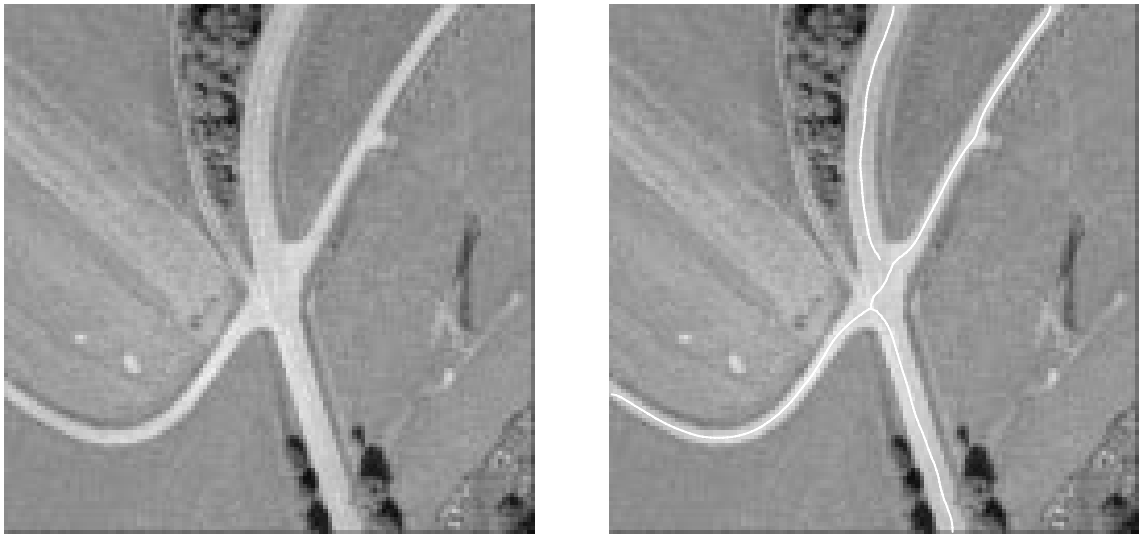
3.2 Example

Figure 7(a) shows the result of linking the line points in Fig. 6 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. It is apparent that the lines obtained with the proposed approach are very smooth and the sub-pixel location of the line is quite precise. Figure 7(b) displays the way the normals to the line were oriented for this example.

3.3 Parameter Selection

The selection of thresholds is very important to make an operator generally useable. 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 σ should be chosen according to (13). Then, σ, w, and h can be plugged into (12) to yield an upper threshold for the operator.

Figure 8 exemplifies this procedure and shows that the presented line detector can be scaled



(a) Aerial image (b) Detected lines

Figure 8: Lines detected (b) in an aerial image (a) of ground resolution 1 m.

arbitrarily. In Fig. 8(a) a larger part of the aerial image in Fig. 7 is displyed, 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 (13), 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, (12) shows that these lines will have a second derivative of ≈ -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. Figure 8(b) displays the lines that were detected with these parameters. As can be seen, all of the roads were detected.

4 Determination of the Line Width

The width of a line is an important feature in its own right. Many applications, especially in remote sensing tasks, are interested in obtaining the width of an object, e.g., a road or a river, 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 that is introduced by asymmetrical lines can be removed.

4.1 Extraction of Edge Points

From the discussion in Section 2.2 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. Only a trivial modification

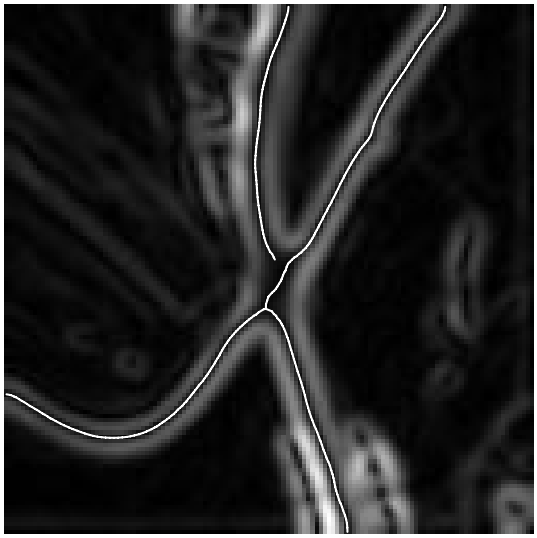


Figure 9: Lines and their corresponding edges in an image of the absolute value of the gradient.

of the Bresenham line drawing algorithm [36] is necessary to yield all pixels that this line will intersect. The analysis in Section 2.2 shows that it is only reasonable to search for edges in a restricted neighborhood of the line. Ideally, 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 current implementation uses a slightly larger search line length of 2.5σ.

In an image of the absolute value of the gradient of the image, the desired edges will appear as bright lines. Figure 9 exemplifies this for the aerial image of Fig. 8(a). In order to extract the lines from the gradient image

$$e(x,y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2} = \sqrt{f_x^2 + f_y^2} \tag{33}$$

where

$$f(x,y) = g_\sigma(x,y) * z(x,y) \tag{34}$$

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

$$e_x = \frac{f_x f_{xx} + f_y f_{xy}}{e} \tag{35}$$

$$e_y = \frac{f_x f_{xy} + f_y f_{yy}}{e} \tag{36}$$

$$e_{xx} = \frac{f_x f_{xxx} + f_y f_{xxy} + f_{xx}^2 + f_{xy}^2 - e_x^2}{e} \tag{37}$$

$$e_{xy} = \frac{f_x f_{xxy} + f_y f_{xyy} + f_{xx} f_{xy} + f_{xy} f_{yy} - e_x e_y}{e} \tag{38}$$

$$e_{yy} = \frac{f_x f_{xyy} + f_y f_{yyy} + f_{xy}^2 + f_{yy}^2 - e_y^2}{e} \; . \tag{39}$$

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

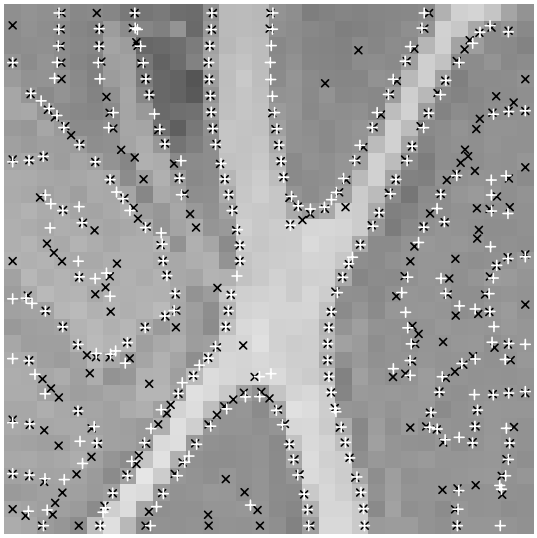


Figure 10: Comparison between the locations of edge points extracted using the exact formula (black crosses) and the 3×3 facet model (white crosses).

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 2 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. This is undesireable 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 same principles as described in Section 2, but uses different convolution masks to determine the partial derivatives of the image [21, 20, 34]. The smallest possible mask size (3×3) 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 1 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 10 illustrates this on the image of Fig. 6(a). Edge points extracted with the correct formulas are displayed as black crosses, while those extracted with the 3×3 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.

4.2 Handling of Missing Edge Points

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 2.5σ. 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.

The algorithm can be described as follows. First of all, the width of the line is extracted for each line point. 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 in front and behind 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

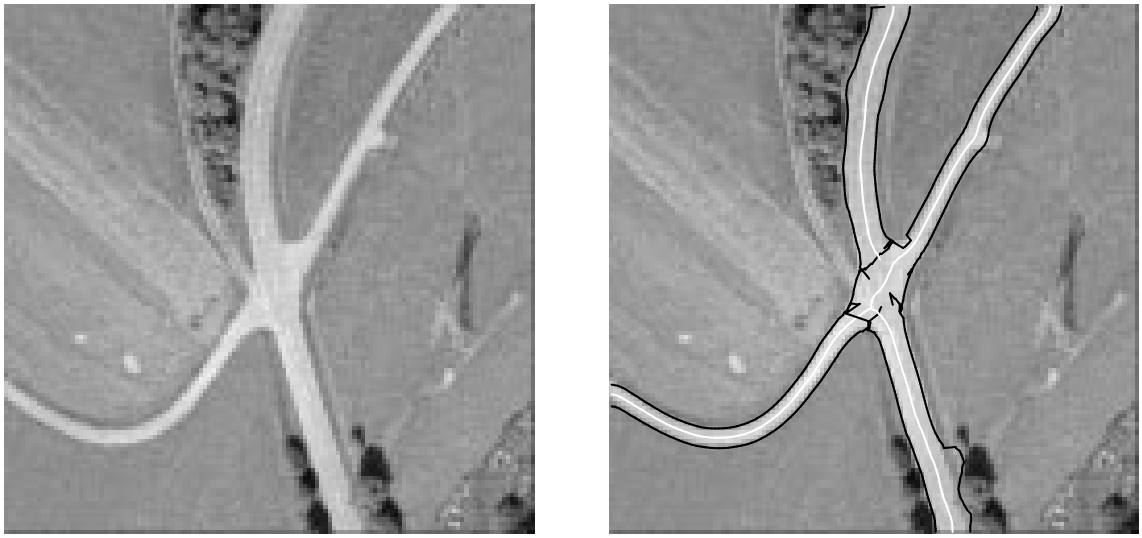
$$w_k = \frac{b-a}{b} w_i + \frac{a}{b} w_j \tag{40}$$

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.

4.3 Examples

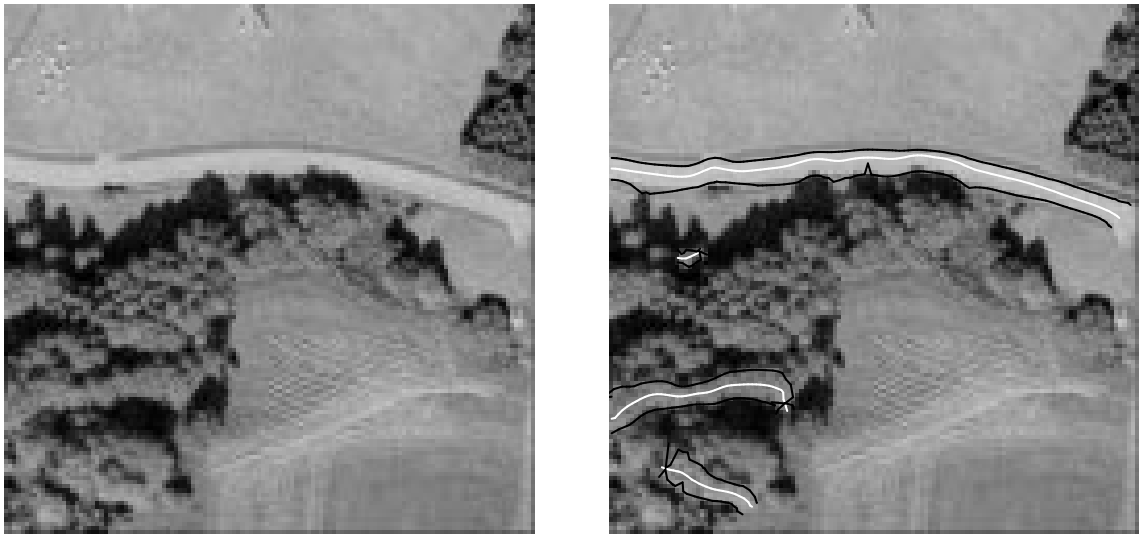
Figure 11(b) displays the results of the line width extraction algorithm for the example image of Fig. 8. This image is fairly good-natured in the sense that the lines it contains are rather symmetrical. From Fig. 11(a) it can be seen that the algorithm is able to locate the edges of the wider line with 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 when the discussion in Section 2.2 is taken into account. Revisiting Fig. 4 again, it is clear that an effect like this is to be expected. How to remove this effect is the topic of Section 5. 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 in Section 4.2. This explains the seemingly unjustified edge points in this area.

Figure 12(b) exhibits the results of the proposed approach on another aerial image of the same ground resolution, given in Fig. 12(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 2.2, especially Fig. 5, the line position is shifted towards the edge of the line that posesses the weaker gradient, i.e., the



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

Figure 11: 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 12: Lines and their width detected (b) in an aerial image (a).

upper edge in this case. Please note also that the line and edge positions are very accurate in the rest of the image.

5 Removing the Bias from Asymmetric Lines

5.1 Detailed Analysis of Asymmetrical Line Profiles

Recall from the discussion at the end of Section 2.2 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 (15)–(17) give an explict 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 (18). 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. Let us 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 σ 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 σ_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 we have $l_2 = -(\sigma_2^2/2w_2) \ln(1-a) = -(c^2\sigma_1^2/(2cw_1)) \ln(1-a) = cl_1$.

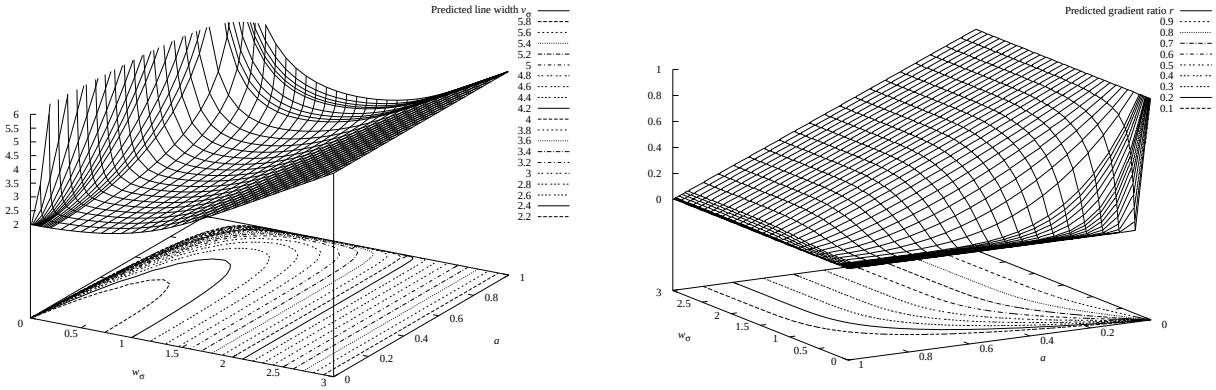
Now let e_1 be one of the two solution of $r''_a(x, \sigma, w_1, a) = 0$, either e_l or e_r, and likewise for e_2, with $\sigma_{1,2}$ and $w_{1,2}$ as above. This expression can be transformed to $(a-1)(e_2 - w_2)/(e_2 + w_2) = \exp(-2e_2w_2/\sigma_2^2)$. If we plug in $\sigma_2 = c\sigma_1$ and $w_2 = cw_1$, we see 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 will also hold for the derived quantities v_l, v_r, and v.

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

With all this being established, the predicted total line width v_σ can be calculated for all w_σ and $a \in [0, 1]$. Figure 13 displays the predicted v_σ for $w_\sigma \in [0, 3]$. It be seen that v_σ 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 Fig. 13 the contour lines for $v_\sigma \in [2, 6]$ are also displayed.

Section 4 gave a procedure to extract the quantity v_σ 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 σ and w. The quantity r has the advantage that it is easy to extract from the image. Figure 13 displays the predicted r for $w_\sigma \in [0, 3]$. It is



(a) Predicted line width

(b) Predicted gradient ratio

Figure 13: Predicted behaviour of the asymmetrical line f_a for $w_\sigma \in [0, 3]$ and $a \in [0, 1]$. (a) Predicted line width v_σ. (b) Predicted gradient ratio r.

obvious that $r \in [0, 1]$. Therefore, the contour lines for r in this range are displayed in Figure 13 as well. It can be seen that for large w_σ, r is very close to $1 - a$. For small w_σ it will drop to near-zero for all a.

5.2 Inversion of the Bias Function

The discussion above can be summarized as follows: The true values of w_σ and a are mapped to the quantities v_σ 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 Section 4 it follows that it is only useful to consider $v_\sigma \in [0, 5]$. However, for very small σ it is possible that an edge point will be found within a pixel in which the center of the pixel is less than 2.5σ 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_σ. Since the algorithm needs to determine the true values (w_σ, a) from the observed (v_σ, r), the inverse f^{-1} of the map f has to be determined. Figure 14 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_σ 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_σ-axis. It can be seen that, for any pair of contour lines from v_σ 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 [31]. To obtain maximum precision for w_σ 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. 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_σ. Therefore, the inverse f^{-1} is computed for selected values of v_σ and r and the true values are obtained by interpolation. The step size of v_σ was chosen as 0.1, while r was sampled at 0.05 intervals. Hence, the intersection points of the contour lines in Fig. 14 are the entries in the table of f^{-1}. Figure 15 shows the true values of w_σ and a for any given v_σ and r. It can be seen that despite the fact that f is very

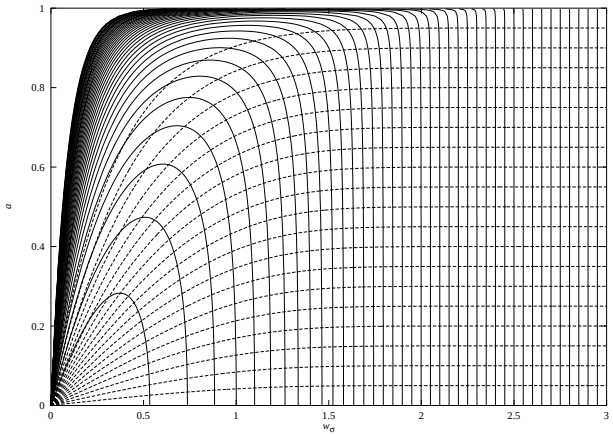
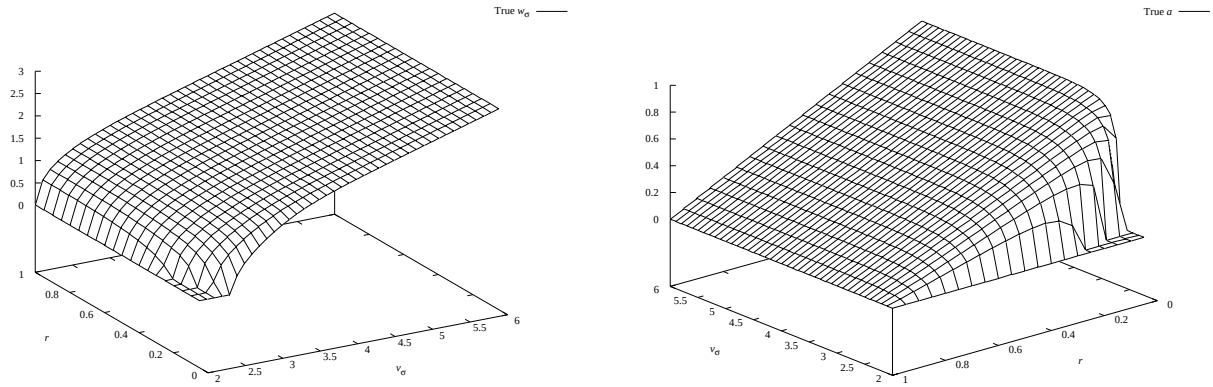


Figure 14: Contour lines of $v_\sigma \in [2, 6]$ and $r \in [0, 1]$.



(a) True w_σ (b) True a

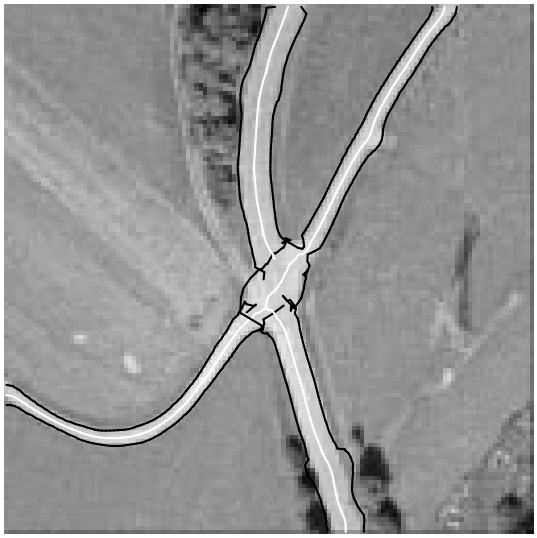
Figure 15: True values of the line width w_σ (a) and the asymmetry a (b).

ill-behaved for small w_σ, f^{-1} is quite well-behaved. This behaviour leads to the conclusion that linear interpolation can be used to obtain good values for w_σ and a.

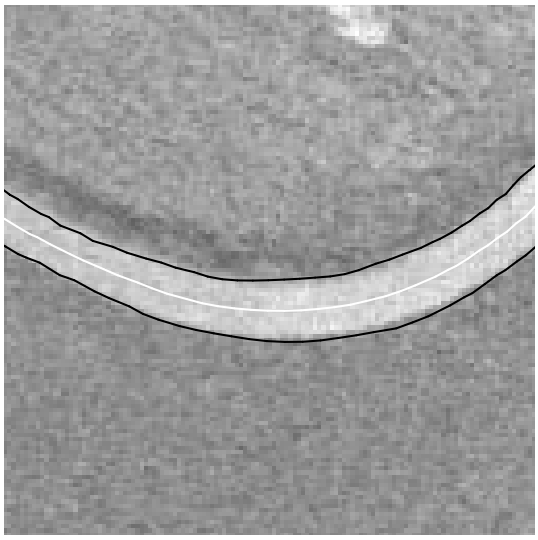
One final important detail is how the algorithm should handle line points where $v_\sigma < 2$, i.e., where f^{-1} is undefined. This can happen, for example, because the facet model sometimes gives a multiple response for an edge point, or because there are two lines very close to each other. In this 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 procedure given in Section 4.2 that fills such gaps.

5.3 Examples

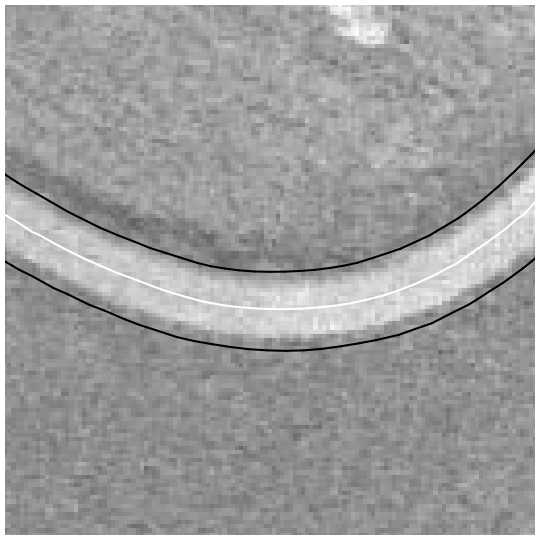
Figure 16 shows how the bias removal algorithm is able to succesfully adjust the line widths in the aerial image of Fig. 11. Please note from Fig. 16(a) that because the lines in this image are fairly symmetrical, the line positions have been adjusted only minimally. Furthermore, it can be seen that the line widths correspond much better to the true line widths. Figure 16(b) shows



(a) Lines detected with bias removal



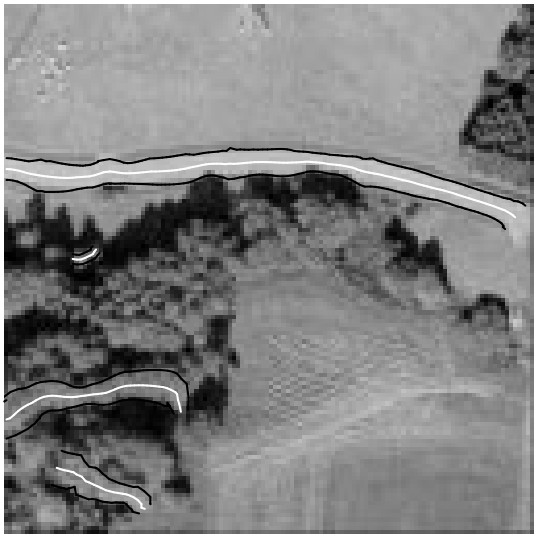
(b) Detail of (a)



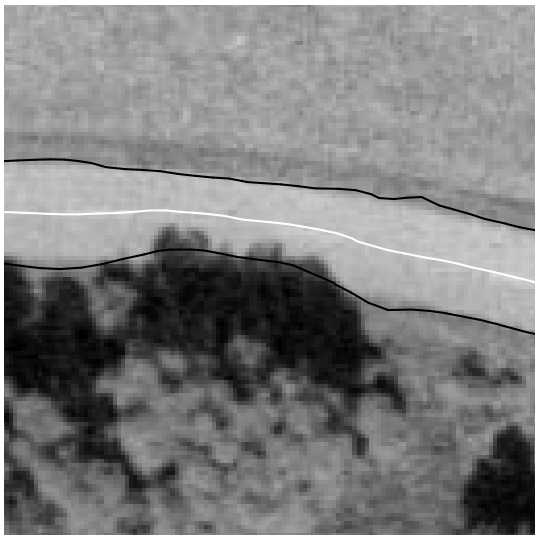
(c) Detail of (a) without bias removal

Figure 16: 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.

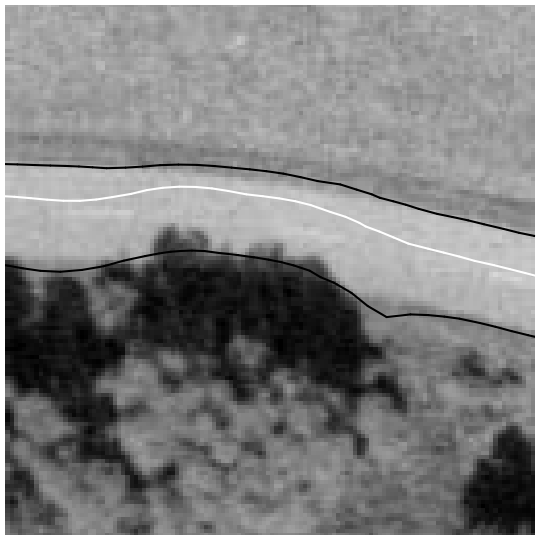
a four times enlarged part of the results superimposed onto the image in its original ground resolution of 0.25 m, i.e., four times the resolution in which the line extraction was carried out. For most of the line the edges are well within one pixel of the edge in the larger resolution. Figure 16(c) 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 Fig. 16(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.



(a) Lines detected with bias removal



(b) Detail of (a)



(c) Detail of (a) without bias removal

Figure 17: 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.

Figure 17 shows the results of removing the bias from the test image of Fig. 12. Please note that in the areas of the image where the line is highly asymmetrical the line and 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 Fig. 17(b). If this is compared to the detail in Fig. 17(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 18(a) shows an aerial image, again of ground resolution 1 m. This image is very tough to process correctly because it contains a large area where the model of the line 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. Unfortunately, due to space limitations this property cannot be elaborated here. Figure 18(b) shows the result of the line extraction algorithm with bias removal. Since in the upper part the line edges 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_σ is close to 2, as can be seen from Fig. 13(a). 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 18(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 are not affected by this, i.e., they correspond very closely to the true line positions in the whole image.

The next example is taken from the domain of medical imaging. Figure 19(a) shows a magnetic resonance (MR) image of a human head. The results of extracting bright lines with bias removal are displayed in Fig. 19(b), while a three times enlarged detail from the left center of the image is given in Fig. 19(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 skull bone and several other features are extracted with high precision. Compare this to Fig. 19(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 20 shows the results of applying the proposed approach to a coronary angiogram. Since the image in Fig. 20(a) has very low contrast, Fig. 20(b) shows the same image with higher contrast. Figure 20(c) displays the results of extracting dark lines from Fig. 20(a), the low contrast image, superimposed onto the high contrast image. A three times enlarged detail is displayed in Fig. 20(d). In particular, it can be seen that the algorithm is very succesful 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 [3] 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.

(a) Input image

(b) Lines detected with bias removal

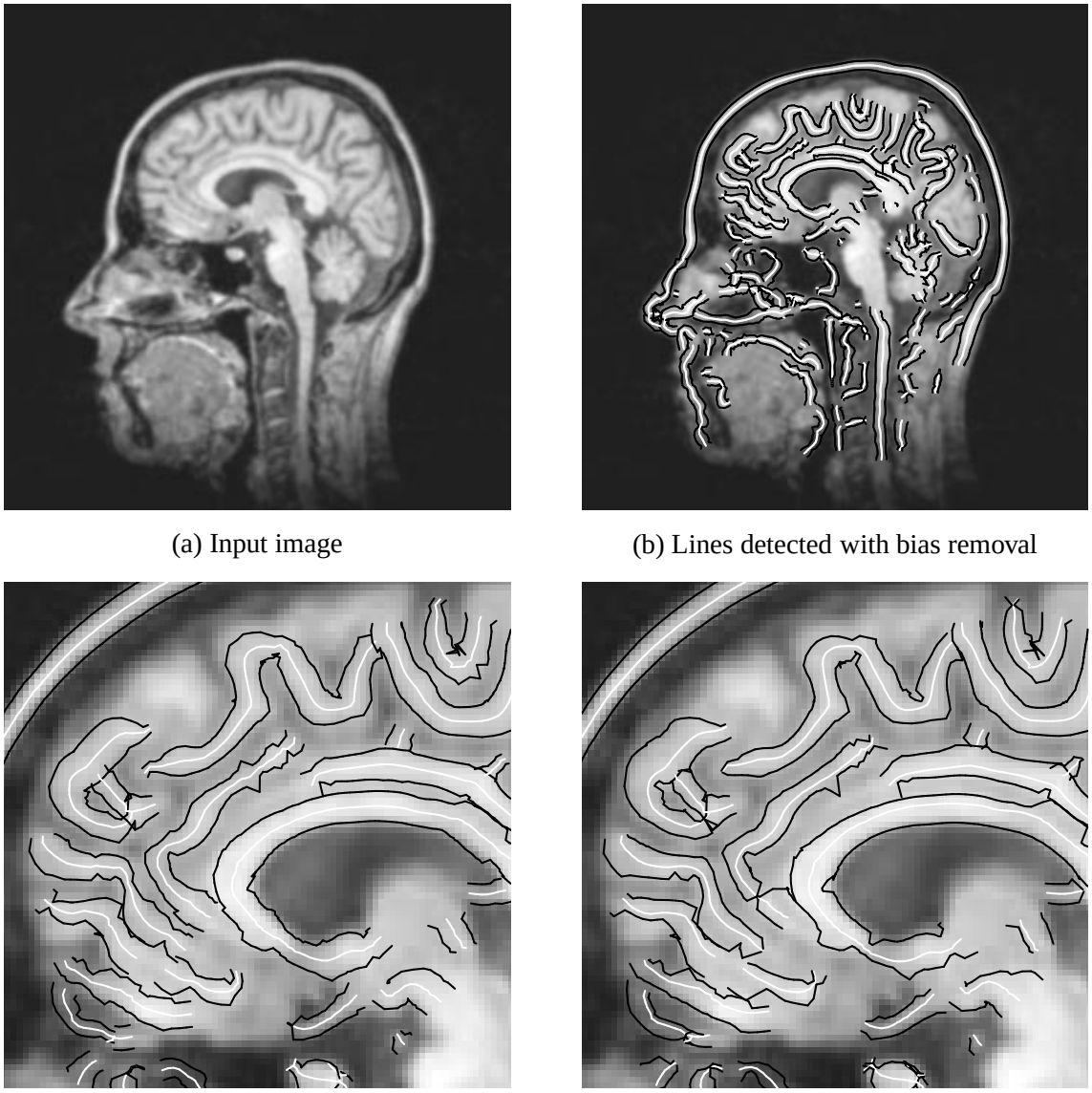
(c) Detail of (b)

(d) Detail of (b) without bias removal

Figure 18: 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.

6 Conclusions

This paper has presented an approach to extract lines and their widths with very high precision. A model for the most common type of lines, the asymmetrical bar-shaped line, was developed from simpler types of lines, namely the parabolic and symmetrical bar-shaped line. A scale-space analysis was carried out for each of these model profiles. This analysis shows that there is a strong interaction between a line and its two corresponding edges which cannot be ignored. The true line width influences the line width occuring in an image, while asymmetry influ-



(a) Input image

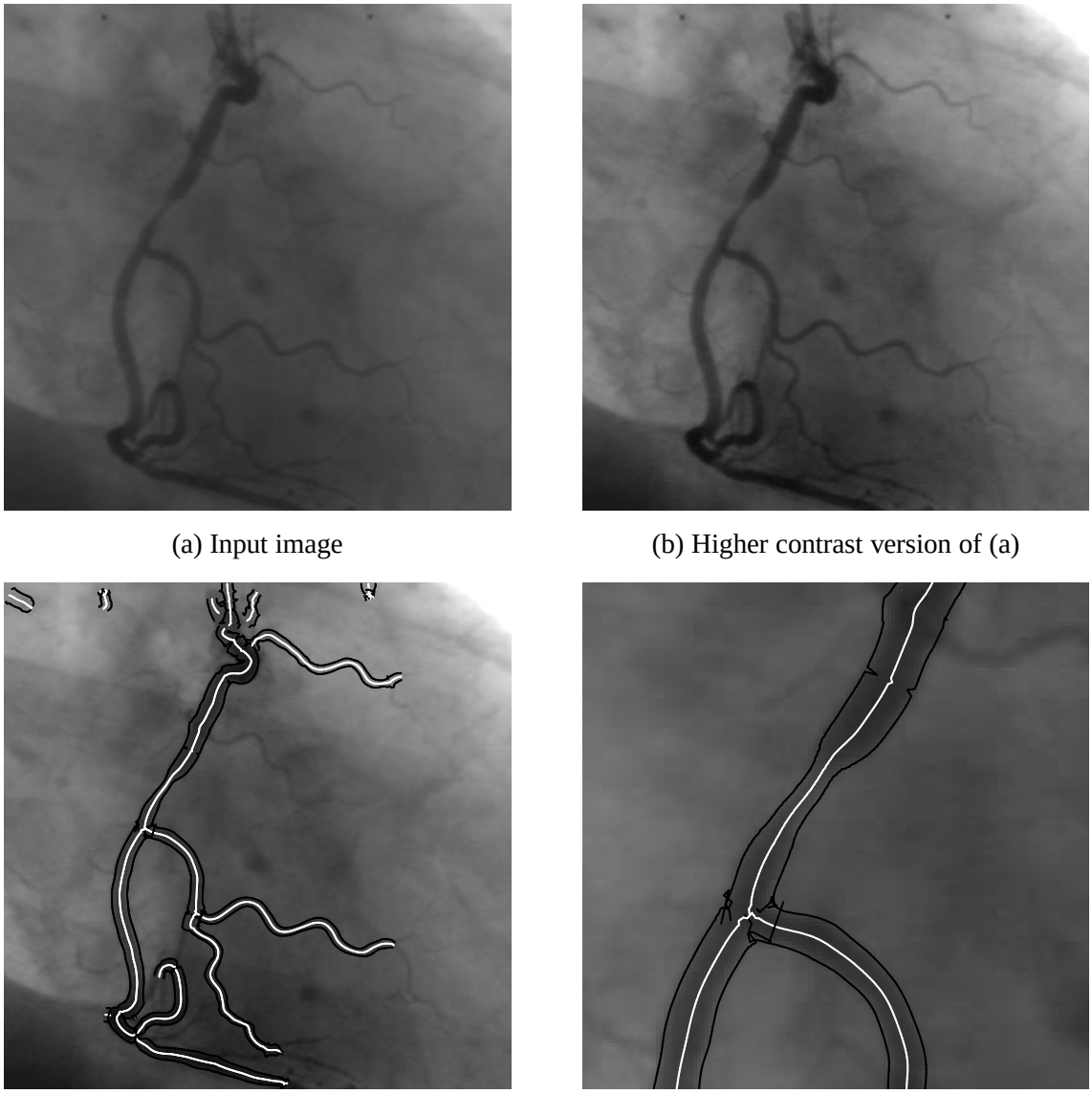
(b) Lines detected with bias removal

(c) Detail of (b)

(d) Detail of (b) without bias removal

Figure 19: Lines and their width detected (b) in a MR image (a) with the bias removed. A three times enlarged detail (c) superimposed onto the original image. (d) Comparison to the line extraction without bias removal.

ences both the line width and its position. From this analysis an algorithm to extract the line position and its width was derived. This algorithm exhibits the bias that is predicted by the model for the asymmetrical line. Therefore, a method to remove this bias was proposed. The resulting algorithm works very well for a range of images containing lines of different widths and asymmetries, as was demonstrated by a number of test images. High resolution versions of the test images were used to check the validity of the obtained results. They show that the proposed approach is able to extract lines with very high precision from low resolution images. The extracted line positions and edges correspond to semantically meaningful entities in the im-



(a) Input image

(b) Higher contrast version of (a)

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

(d) Detail of (c)

Figure 20: 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).

age, e.g., road center lines and roadsides or blood vessels. Although the test images used were mainly aerial and medical images, the algorithm can be applied in many other domains as well, e.g., optical character recognition [23]. The approach only uses the first and second directional derivatives of an image for the extraction of the line points. No specialized directional filters are needed. The edge point extraction is done by a localized search around the line points already found using five very small masks. This makes the approach computationally very efficient. For example, the time to process the MR image of Fig. 19 of size 256×256 is about 1.7 seconds

on a HP 735 workstation.

The presented approach shows two fundamental limitations. First of all, it can only be used to detect lines with a certain range of widths, i.e., between 0 and 2.5σ. This is a problem if the width of the important lines varies greatly in the image. However, since the bias is removed by the algorithm, one can in principle select σ large enough to cover all desired line widths and the algorithm will still yield valid results. This will work if the narrow lines are relatively salient. Otherwise they will be smoothed away in scale-space. Of course, once σ is selected so large that neighboring lines will start to influence each other the line model will fail and the results will deteriorate. Hence, in reality there is a limited range in which σ can be chosen to yield good results. In most applications this is not a very significant restriction since one is usually only interested in lines in a certain range of widths. Furthermore, the algorithm could be iterated through scale-space to extract lines of very different widths. The second problem is that the definition of salient lines is done via the second directional derivatives. However, one can plug semantically meaningful values, i.e., the width and height of the line, as well as σ, into (12) to obtain the desired thresholds. Again, this is not a severe restriction of the algorithm, but only a matter of convenience.

Finally, it should be stressed that the lines extracted are not ridges in the topographic sense, i.e., they do not define the way water runs downhill or accumulates [17, 37]. In fact, they are much more than a ridge in the sense that a ridge can be regarded in isolation, while a line needs to model its surroundings. If a ridge detection algorithm is used to extract lines, the asymmetry of the lines will invariably cause it to return biased results.

References

[1] Carsten Steger, Clemens Glock, Wolfgang Eckstein, Helmut Mayer, and Bernd Radig. Model-based road extraction from images. In Gruen et al. [38], pages 275–284.

[2] Albert Baumgartner, Carsten Steger, Christian Wiedemann, Helmut Mayer, Wolfgang Eckstein, and Heinrich 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.

[3] 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.

[4] J. B. Antoine Maintz, Petra A. van den Elsen, and Max 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.

[5] 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.

[6] Bruno Jedynak and Jean-Philippe Rozé. Tracking roads in satellite images by playing twenty questions. In Gruen et al. [38], pages 243–253.

[7] Donald Geman and Bruno 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.

[8] Martin A. Fischler. The perception of linear structure: A generic linker. In *Image Understanding Workshop*, pages 1565–1579, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers.

[9] Martin A. Fischler and Helen C. Wolf. Linear delineation. In *Computer Vision and Pattern Recognition*, pages 351–356. IEEE Computer Society Press, 1983.

[10] Donald Geman and Bruno Jedynak. Shape recognition and twenty questions. Rapport de Recherche 2155, INRIA, Rocquencourt, France, November 1993.

[11] 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. IEEE Computer Society Press, 1995.

[12] Alain Filbois and Didier Gemmerlé. From step edge to line edge: Combining geometric and photometric information. In *MVA '94 IAPR Workshop on Machine Vision Applications*, pages 87–90, 1994.

[13] J. Brian Subirana-Vilanova and Kah Kay Sung. Multi-scale vector-ridge-detection for perceptual organization without edges. A.I. Memo 1318, MIT Artificial Intelligence Laboratory, Cambridge, MA, USA, December 1992.

[14] In So Kweon and Takeo Kanade. Extracting topographic terrain features from elevation maps. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 59(2):171–182, March 1994.

[15] 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, USA, 1993.

[16] J. W. Bruce and P. J. Giblin. *Curves and singularities: A geometrical introduction to singularity theory*. Cambridge University Press, Cambridge, England, 2nd edition, 1992.

[17] Jan J. Koenderink and Andrea J. van Doorn. Two-plus-one-dimensional differential geometry. *Pattern Recognition Letters*, 15(5):439–443, May 1994.

[18] Olivier Monga, Nasser Armande, and Philippe Montesinos. Thin nets and crest lines: Application to satellite data and medical images. Rapport de Recherche 2480, INRIA, Rocquencourt, France, February 1995.

[19] Ian R. Porteous. *Geometric differentiation for the intelligence of curves and surfaces*. Cambridge University Press, Cambridge, England, 1994.

[20] Frank Glazer. Curve finding by ridge detection and grouping. In W. Kropatsch and H. Bischof, editors, *Mustererkennung*, Informatik Xpress 5, pages 109–116. Deutsche Arbeitsgemeinschaft für Mustererkennung, 1994.

[21] Andreas 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.

[22] Li Wang and Theo Pavlidis. Direct gray-scale extraction of features for character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1053–1067, October 1993.

[23] Li Wang and Theo 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.

[24] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume I. Addison-Wesley Publishing Company, Reading, MA, USA, 1992.

[25] Robert M. Haralick, Layne T. Watson, and Thomas J. Laffey. The topographic primal sketch. *International Journal of Robotics Research*, 2(1):50–72, 1983.

[26] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Computer Vision and Pattern Recognition*, pages 465–470. IEEE Computer Society Press, 1996.

[27] Lee A. Iverson and Steven W. Zucker. Logical/linear operators for image curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10):982–996, October 1995.

[28] Andreas 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.

[29] Luc M. J. Florack, Bart M. ter Haar Romney, Jan J. Koenderink, and Max A. Viergever. Scale and the differential structure of images. *Image and Vision Computing*, 10(6):376–388, July 1992.

[30] Bart M. ter Haar Romney, Luc M. Florack, Alfons H. Salden, and Max A. Viergever. Higher order differential structure of images. *Image and Vision Computing*, 12(6):317–325, July/August 1994.

[31] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 2nd edition, 1992.

[32] Tony 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.

[33] Rachid Deriche. Recursively implementing the Gaussian and its derivatives. Rapport de Recherche 1893, INRIA, Sophia Antipolis, France, April 1993.

[34] Carsten Steger. Extracting curvilinear structures: A differential geometric approach. In Bernard Buxton and Roberto Cipolla, editors, *Fourth European Conference on Computer Vision*, volume 1064 of *Lecture Notes in Computer Science*, pages 630–641. Springer-Verlag, 1996.

[35] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[36] David F. Rogers. *Procedural Elements for Computer Graphics*. McGraw-Hill Book Company, New York, NY, USA, 1985.

[37] Antonio M. López and Joan Serrat. Tracing crease curves by solving a system of differential equations. In Bernard Buxton and Roberto Cipolla, editors, *Fourth European Conference on Computer Vision*, volume 1064 of *Lecture Notes in Computer Science*, pages 241–250. Springer-Verlag, 1996.

[38] A. Gruen, O. Kuebler, and P. Agouris, editors. *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, Basel, Switzerland, 1995. Birkhäuser Verlag.