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 behavior 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 subpixel line position, but also the width of the line for each line point, also with subpixel accuracy.

Index Terms—Feature extraction, curvilinear structures, lines, scale-space, contour linking, low-level processing, aerial images, medical images.

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]. In addition, it is useful in medical imaging for the extraction of anatomical features, e.g., blood vessels from an X-ray angiogram [2] or the bones in the skull from a CT or MR image [3].

Previous work on line detection can be classified into three categories. The first approach detects lines by considering the gray values of the image only [4], [5] 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 [6], [7], [5]. Furthermore, lines cannot be extracted with subpixel accuracy.

The second approach regards lines as objects having parallel edges [8], [9]. First, the local line direction is determined for each pixel. Then two specially tuned edge detection filters are applied perpendicularly to the line, where each filter detects either the left or right edge of the line. The responses of each filter are combined nonlinearly [8]. The advantage of this approach is that, since derivatives of Gaussian kernels are used, the procedure can be iterated in scale-space to detect lines of arbitrary widths. However, because the directional edge detection filters are not separable, 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 differential geometric properties. The basic idea behind these algorithms is to locate the positions of ridges and ravines in the image function. They can be further divided according to which property they use.

The first subcategory defines ridges as points on a contour line of the image where the curvature is maximum [3], [10], [11]. One way to do this is to extract the contour lines explicitly, to find the points of maximum curvature, and to link the extracted points into ridges [10]. This scheme suffers from two main drawbacks. First, since no contour lines will be found for horizontal ridges, such ridges will be labeled as an extended peak. Furthermore, for ridges that have a small gradient the contour lines 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 [3], [11]. This procedure will also fail for horizontal ridges. Furthermore, the ridge positions found by this operator will often be in wrong positions [3], [12].

In the second subcategory, ridges are found at points where one of the principal curvatures of the image assumes a local maximum [13], [11]. For lines with a flat profile, it has the problem that two separate points of maximum curvature symmetrical to the true line position will be found [11].

In the third subcategory, 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 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 [14], [15], [16], [17]. The direction of the line is determined from the Hessian matrix of the Taylor polynomial. Line points are 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 subpixel accuracy [14]. 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

• *The author is with the Forschungsgruppe Bildverstehen (FG BV), Informatik IX, Technische Universität München, Orleansstraße 34, 81667 München, Germany. E-mail: stegerc@informatik.tu-muenchen.de.*

Manuscript received 13 Aug. 1996; revised 15 Dec. 1997. Recommended for acceptance by K. Boyer.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 106144.

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 [18], [19]. 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 [18]. However, since the surroundings of the line are not modeled, the extracted line position becomes progressively less accurate as σ increases.

Few approaches to line detection consider 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 of a scale-normalized response as the line width [8], [18]. 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 coarse estimate of the line width, since the scale-space is quantized in rather coarse intervals. A different approach is given in [14], 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 subpixel accuracy. However, since a third-order facet model is used, the same problems 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 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 subpixel 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.

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

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

The flatness of this profile is characteristic for many interesting objects, e.g., roads in aerial images or printed characters. However, the assumption that lines have the same contrast on both sides is rarely true for real images. Therefore, asymmetrical bar-shaped lines

$$f_a(x) = \begin{cases} 0, & x < -w \\ 1, & |x| \le w \\ a, & x > w \end{cases} \tag{2}$$

($a \in [0, 1]$) are considered as the most common line profile in this paper. General lines of height h can be obtained by considering a scaled asymmetrical profile, i.e., $hf_a(x)$.

2.2 Detection of Lines in 1D

To derive the line detection algorithm, let us for the moment assume the lines $z(x)$ do not have a flat profile as in (1) and (2), but a "rounded" profile, e.g., a parabolic profile as in [20]. Then 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 have $z''(x) \ll 0$, while dark lines on a bright background have $z''(x) \gg 0$. Because of noise, this scheme will not yield good results for real images. 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 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 [21]. The Gaussian kernels are given by:

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

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

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

Thus, a scale-space description of the line profile can be obtained [20]. The desirable properties of this approach are that for the parabolic line profile the extracted position is always the true line position, and that the magnitude of the second derivative is always maximum at the line position. Thus, salient lines can be selected based their second derivative for all σ [20].

After the analysis of the parabolic profile has given us the desirable properties a line extraction algorithm should possess, let us now consider the more common case of a bar-shaped profile for the rest of this paper. 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:

$$\begin{aligned} r_b(x, \sigma, w, h) &= g_\sigma(x) * f_b(x) \\ &= h(\phi_\sigma(x + w) - \phi_\sigma(x - w)) \end{aligned} \tag{6}$$

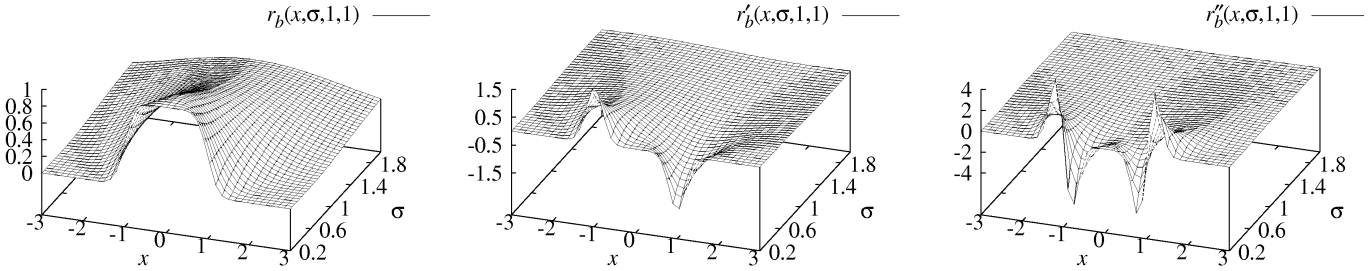


Fig. 1. Scale-space behavior of the bar-shaped line f_b when convolved with the derivatives of Gaussian kernels for $x \in [-3, 3]$ and $\sigma \in [0.2, 2]$.

$$r'_b(x, \sigma, w, h) = g'_\sigma(x) * f_b(x) = h(g_\sigma(x + w) - g_\sigma(x - w)) \quad (7)$$

$$r''_b(x, \sigma, w, h) = g''_\sigma(x) * f_b(x) = h\,(g'_\sigma(x + w) - g'_\sigma(x - w)), \quad (8)$$

where

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

Fig. 1 shows the scale-space behavior of a bar profile with $w = 1$ and $h = 1$. It can be seen that the bar profile gradually becomes "round" at its corners. The first derivative vanishes only at $x = 0$ for all $\sigma > 0$ because of the infinite support of $g_\sigma(x)$. However, the second derivative $r''_b(x, \sigma, w, h)$ does not take on its maximum negative value for small σ. In fact, for $\sigma \le 0.2\,w$ it is very close to zero. Furthermore, there are 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 selected by this value. This value of σ is given by the solution of the equation $\partial/\partial\sigma\,(r''_b(0, \sigma, w, h)) = 0$. It can be shown that

$$\sigma \ge \frac{w}{\sqrt{3}} \quad (10)$$

has to hold. Furthermore, from this it is obvious that $r''_b(x, \sigma, w, h)$ has 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 should be observed to ensure that salient lines can be selected based of the magnitude of their second derivative.

In addition to this, (7) and (8) can be used to derive how the edges of a bar-shaped line behave in scale-space. The positions of the edges are given by the maxima of $|r'_b(x, \sigma, w, h)|$ or the zero crossings of $r''_b(x, \sigma, w, h)$. In the one-dimensional case, these definitions are equivalent. As will be discussed in Section 4, the implementation in 2D uses the maxima of the gradient in the direction perpendicular to the line. This will give the correct edge locations unless the curvature of the line is very high compared to σ [22], in which case the width will be extracted too small. Since this scale-space analysis involves equations which cannot be solved analytically, the calculations must be done using a root finding algorithm [23]. Fig. 2 shows the location of the line and its corresponding edges for $w \in [0, 4]$ and $\sigma = 1$. From (8) 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. This effect was also studied qualitatively in the context of edges in [24] and [25], where a solution was proposed based on edge focusing by successively applying the filters with smaller values of σ. However, for the 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.

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 (2). Their responses are

$$r_a(x, \sigma, w, a) = \phi_\sigma(x + w) + (a - 1)\phi_\sigma(x - w) \quad (11)$$

$$r'_a(x, \sigma, w, a) = g_\sigma(x + w) + (a - 1)g_\sigma(x - w) \quad (12)$$

$$r''_a(x, \sigma, w, a) = g'_\sigma(x + w) + (a - 1)\,g'_\sigma(x - w). \quad (13)$$

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

$$l = -\frac{\sigma^2}{2w}\ln(1 - a)\,. \quad (14)$$

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

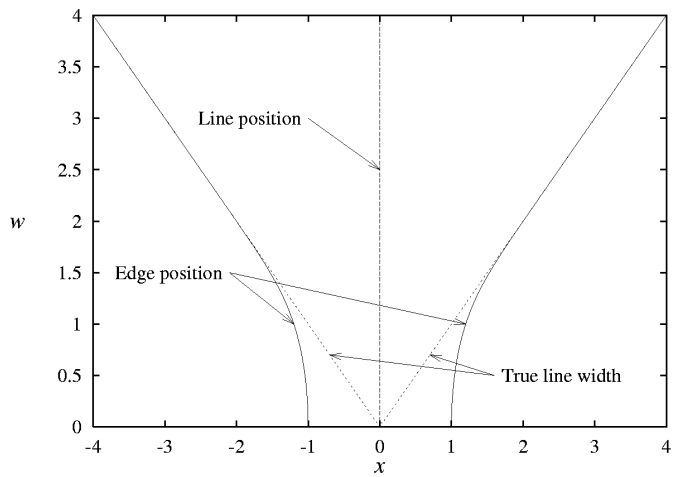


Fig. 2. Location of a bar-shaped line and its edges with width $w \in [0, 4]$ for $\sigma = 1$.

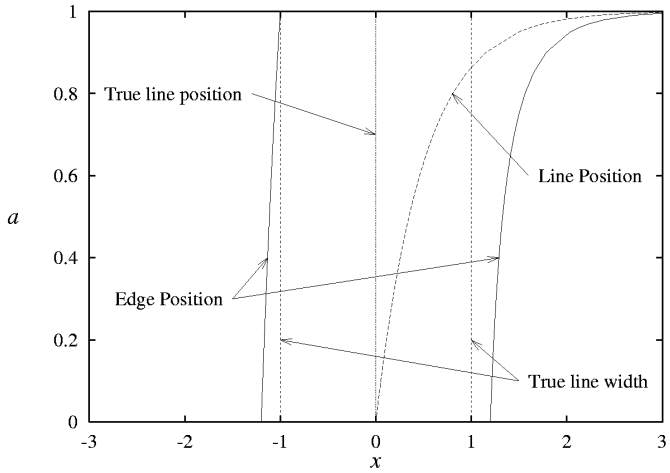


Fig. 3. Location of an asymmetrical bar-shaped line and its corresponding edges with width $w = 1$, $\sigma = 1$, and $a \in [0, 1]$.

$$a \leq 1 - e^{-\frac{2w^2}{\sigma^2}}. \tag{15}$$

The location of the corresponding edges can again only be computed numerically. Fig. 3 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 positions of the line and the edges are greatly influenced by line asymmetry. As a gets larger, the line and edge positions are pushed to the weak side, i.e., the side with the smaller edge gradient.

Note that (14) 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.

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. In case a different line profile might be more appropriate [2], e.g., an appropriately modified parabolic profile, the methodology presented here can be used to model the behavior of the profile in scale-space, and thus to increase precision.

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 [26] 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 $x \in \left(n - \frac{1}{2}, n + \frac{1}{2}\right]$. In this case, the convolution masks are given by:

$$g_{n,\sigma} = \phi_\sigma\left(n + \frac{1}{2}\right) - \phi_\sigma\left(n - \frac{1}{2}\right) \tag{16}$$

$$g'_{n,\sigma} = g_\sigma\left(n + \frac{1}{2}\right) - g_\sigma\left(n - \frac{1}{2}\right) \tag{17}$$

$$g''_{n,\sigma} = g'_\sigma\left(n + \frac{1}{2}\right) - g'_\sigma\left(n - \frac{1}{2}\right). \tag{18}$$

For the implementation, the approximation error ϵ 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 [26] or a recursive computation [27], are suitable as well.

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

$$p(x) = r + r'x + \frac{1}{2}r''x^2. \tag{19}$$

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

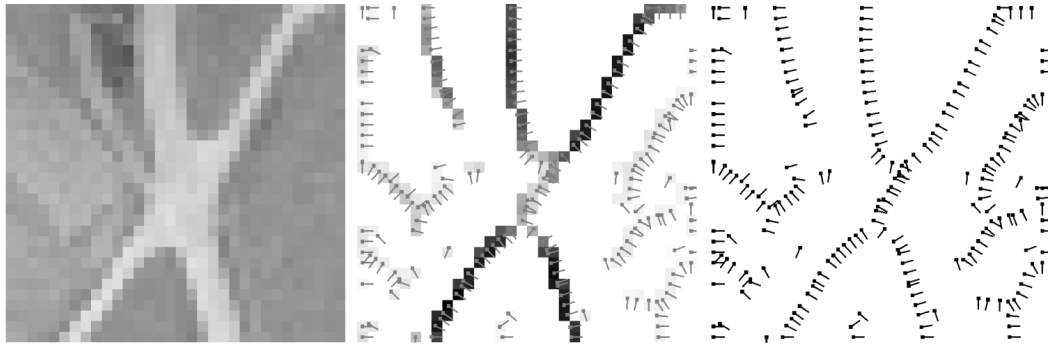
$$x = -\frac{r'}{r''}. \tag{20}$$

The point n is declared a line point if this position falls within the pixel's boundaries, i.e., if $x \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ 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 the characteristic 1D line profile 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 is made about the derivatives in the direction of $s'(t)$.

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 by convolving the image with the discrete two-dimensional Gaussian partial derivative kernels corresponding to (16)–(18). 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



(a) (b) (c)

Fig. 4. 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.

$$H(x,y) = \begin{pmatrix} r_{xx} & r_{xy} \\ r_{xy} & r_{yy} \end{pmatrix}. \tag{21}$$

The calculation can be done in a numerically stable and efficient way by using one Jacobi rotation to annihilate the r_{xy} term [23]. 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, and setting its derivative along t to zero. Hence, the point is given by

$$(p_x, p_y) = (tn_x, tn_y) \tag{22}$$

where

$$t = -\frac{r_x n_x + r_y n_y}{r_{xx} n_x^2 + 2 r_{xy} n_x n_y + r_{yy} n_y^2}. \tag{23}$$

Again, $(p_x, p_y) \in \left[-\frac{1}{2}, \frac{1}{2}\right] \times \left[-\frac{1}{2}, \frac{1}{2}\right]$ 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

Figs. 4b and 4c give an example of the results obtainable with this approach. Here, bright line points were extracted from the input image given in Fig. 4a with $\sigma = 1.1$. This image is part of an aerial image with a ground resolution of 2 m. The subpixel 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 Fig. 4 it might appear, if an eight-neighborhood is used, that the proposed approach returns multiple responses to each line. However, when the subpixel 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 perfectly. Therefore, linking will be considerably easier than in approaches that yield multiple responses, e.g., [19], [14], [16].

3 Linking Line Points Into Lines

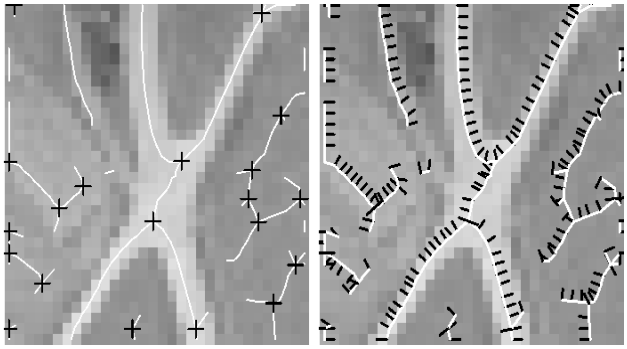
After individual line pixels have been extracted, they need to be linked into lines. It is necessary to do this right after the extraction of the line points because the later stages of determining line width and removing the bias require a data structure that uses the notion of a left and right side of an entire line. Therefore, the normals to the line have to be oriented to the same side of the line. As is evident from Fig. 4, 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 [14] or [15]. 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 is 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 subpixel location of the line (p_x, p_y).

Starting from the pixel with maximum second derivative, lines are constructed by adding the appropriate neighbor to the current line. Since 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



(a) (b)

Fig. 5. (a) Linked lines. (b) Oriented normals. Lines are drawn in white while junctions are displayed as black crosses and normals as black lines.

line point detection algorithm yields 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 subpixel 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 $\beta = |\alpha_2 - \alpha_1|$, $\beta \in [0, \pi/2]$, 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 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, unless it is the first point of the currently processed line, in which case a line describing a closed loop is found.

New lines are 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 [28], [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 points 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 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

Fig. 5a shows the result of linking the line points in Fig. 4 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 subpixel location of the line is quite precise. Fig. 5b displays the way the normals to the line were oriented for this example.

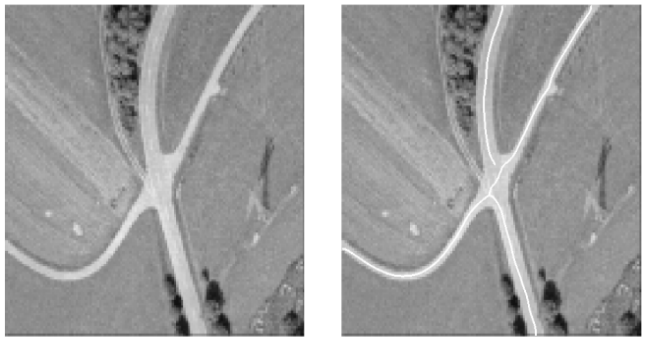
3.3 Parameter Selection

The selection of thresholds is very important to make an operator generally 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 σ should be chosen according to (10). Then, σ, w, and h can be plugged into (8) to yield an upper threshold for the operator.

Fig. 6 exemplifies this procedure and shows that the presented line detector can be scaled arbitrarily. In Fig. 6a, a larger part of the aerial image in Fig. 5 is displayed, but this time with a ground resolution of 1 m, i.e., twice the resolution. If seven-pixel-wide lines are to be detected, i.e., if $w = 3.5$, according to (10), 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, (8) 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 five. The lower threshold was set to 0.8 to follow the roads as far as possible into the road intersection. Fig. 6b displays the lines that were detected with these parameters.

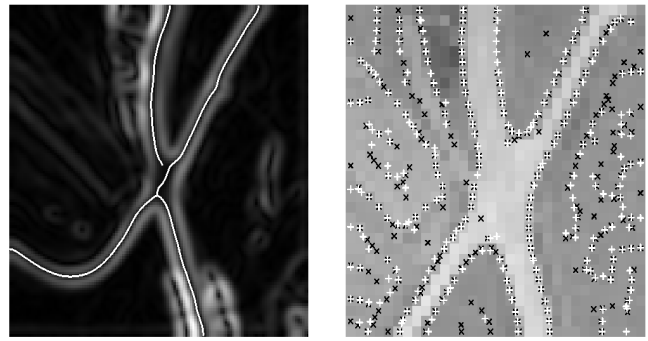
4 Determination of the Line Width

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.



(a) (b)

Fig. 6. Lines detected (b) in an aerial image (a) of ground resolution 1 m.



(a) (b)

Fig. 7. (a) Lines and their corresponding edges in an image of the absolute value of the gradient. (b) Comparison between the locations of edge points extracted using the exact formula (black crosses) and the 3×3 facet model (white crosses).

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 of the Bresenham line drawing algorithm [30] is necessary to yield all pixels that this line intersects. The analysis in Section 2.2 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 2.5σ. This covers most of the asymmetrical lines as well, for which the width can grow beyond $\sqrt{3}\sigma$. Only the extremely asymmetrical cases in the upper left corner of Fig. 11 are not covered.

In an image of the absolute value of the gradient of the image, the desired edges appear as bright lines. Fig. 7a exemplifies this for the aerial image of Fig. 6a. In order to extract the lines from the gradient image

$$e(x,y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2} = \sqrt{f_x^2 + f_y^2} \tag{24}$$

where

$$f(x,y) = g_\sigma(x,y) * z(x,y) \tag{25}$$

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

$$e_x = \frac{f_x f_{xx} + f_y f_{xy}}{e} \tag{26}$$

$$e_y = \frac{f_x f_{xy} + f_y f_{yy}}{e} \tag{27}$$

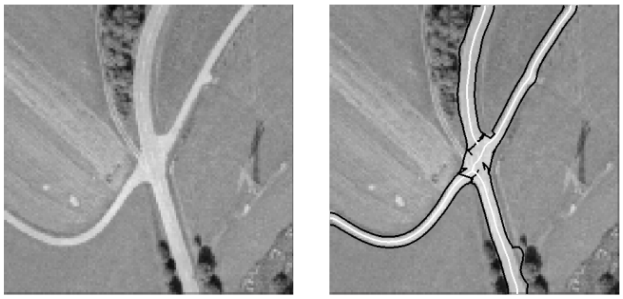
$$e_{xx} = \frac{f_x f_{xxx} + f_y f_{xxy} + f_{xx}^2 + f_{xy}^2 - e_x^2}{e} \tag{28}$$

$$e_{xy} = \frac{f_x f_{xxy} + f_y f_{xyy} + f_{xx} f_{xy} + f_{xy} f_{yy} - e_x e_y}{e} \tag{29}$$

$$e_{yy} = \frac{f_x f_{xyy} + f_y f_{yyy} + f_{xy}^2 + f_{yy}^2 - e_y^2}{e}. \tag{30}$$

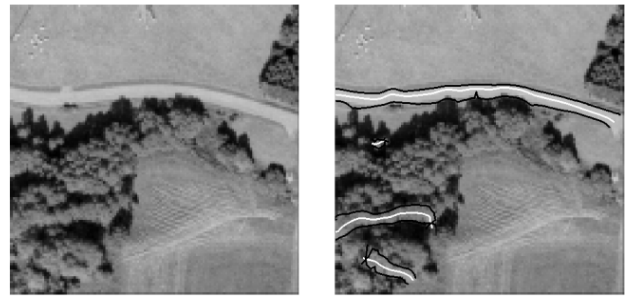
This has two main disadvantages. First, the computational load increases by almost a factor of two, since four additional partial derivatives with slightly larger mask sizes have to be computed. Second, the expressions above are undefined whenever $e(x, y) = 0$.

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 subpixel accuracy. However, this would mean that some additional smoothing would be applied to the gradient image. 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 same principles as described in Section 2, but uses different convolution masks to determine the partial derivatives of the image [14], [20]. The smallest possible mask size (3×3) is used, since this results in the most accurate localization of the edge points. 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, are only affected very slightly. Fig. 7b illustrates this on the image of Fig. 4a. 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.



(a) (b)

Fig. 8. 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) (b)

Fig. 9. Lines and their width detected (b) in an aerial image (a).

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 2.5 σ. 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: The width of the line is extracted for each line point, where possible. 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

$$w_k = \frac{b-a}{b} w_i + \frac{a}{b} w_j. \tag{31}$$

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 is extrapolated to the last defined line width.

4.3 Examples

Fig. 8b displays the results of the line width extraction algorithm for the example image of Fig. 6. This image is fairly good-natured in the sense that the lines it contains are rather symmetrical. From Fig. 8a it can be seen that the algorithm is able to locate the edges of the wider line with 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 the algorithm to estimate the line width too large. 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. 2, 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.

Fig. 9b exhibits the results of the proposed approach on another aerial image of the same ground resolution, given in Fig. 9a. The line in the upper part of the image contains a very asymmetrical stretch 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. 3, the line position is shifted toward the edge of the line that possesses the weaker gradient, i.e., the upper edge in this case. The line and edge positions are very accurate in the rest of the image.

5 Removing the Bias From Asymmetrical 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 (11)–(13) 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 (14). 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 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 $l_{1,2} = -\left(\sigma_{1,2}^2 / 2w_{1,2}\right)\ln(1-a)$. Hence, we have $l_2 = -\left(\sigma_2^2/2w_2\right)\ln(1-a) = -\left(c^2\sigma_1^2/(2cw_1)\right)\ln(1-a) = cl_1$.

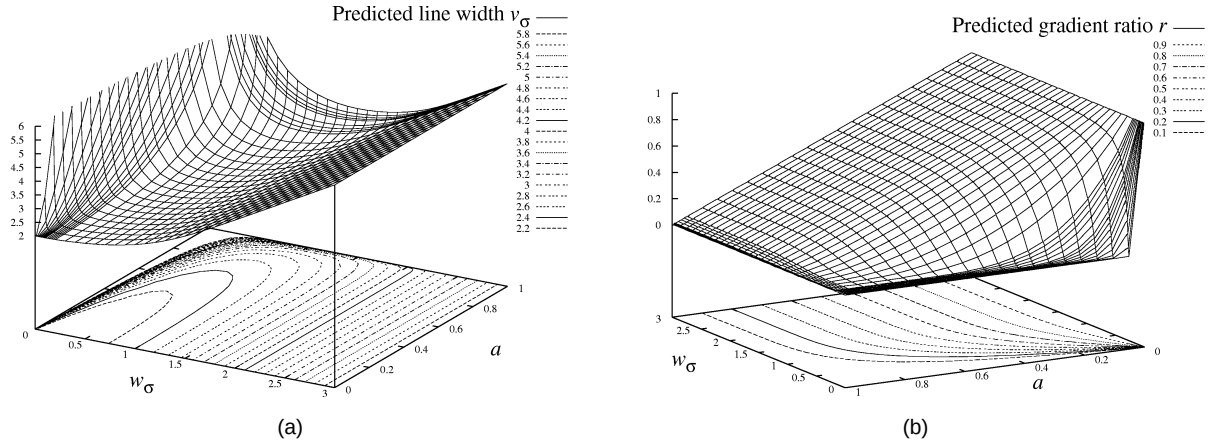


Fig. 10. Predicted behavior of the asymmetrical line f_a for $w_\sigma \in [0, 3]$ and $a \in [0, 1]$. (a) Predicted line width v_σ. (b) Predicted gradient ratio r.

Now, let e_1 be one of the two solutions 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 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. This property also holds for the derived quantities v_l, v_r, and v. □

The meaning of Proposition 1 is that w and σ are not independent. 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. Hence 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 σ.

With all this being established, the predicted total line width v_σ can be calculated for all w_σ and $a \in [0, 1]$. Fig. 10 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, since $v_\sigma \in [2, \infty]$, 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. It 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, 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. Fig. 10 displays the predicted r for $w_\sigma \in [0, 3]$. Since $r \in [0, 1]$, the contour lines for r in this range are displayed in Fig. 10. For large w_σ, r is very close to $1 - a$. For small w_σ, it drops 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 2.5 σ 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. Fig. 11 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 is only one intersection point. Hence, f is invertible.

To calculate f^{-1}, a multidimensional root finding algorithm has to be used [23]. 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. First, 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

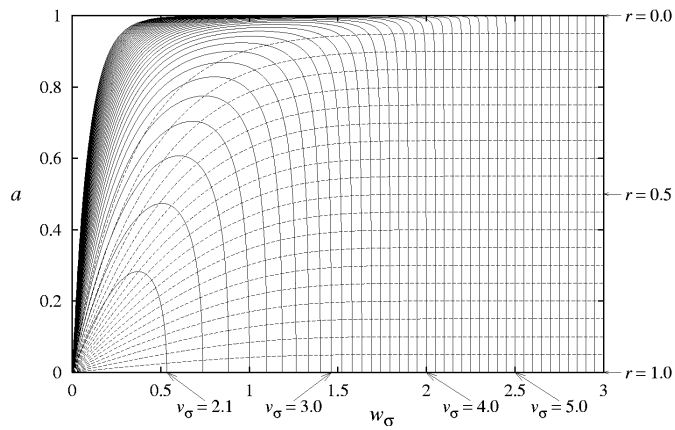


Fig. 11. Contour lines of the predicted line width $v_\sigma \in [2, 6]$ and the predicted gradient ratio $r \in [0, 1]$.

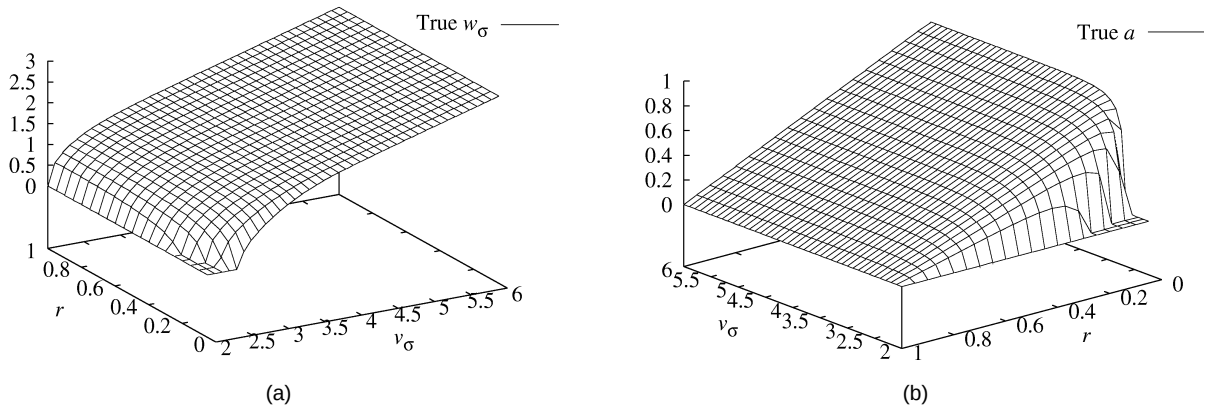


Fig. 12. The inverted bias function: true values of the line width w_σ (a) and the asymmetry a (b), both for $v_\sigma \in [2, 6]$ and $r \in [0, 1]$.

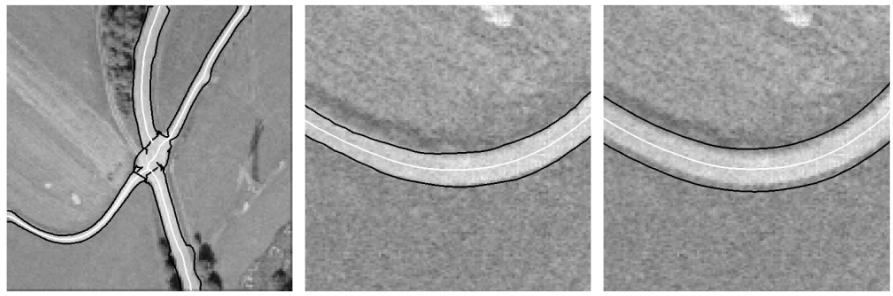


Fig. 13. Lines and their width detected (a) in an aerial image of resolution 1 m with the bias removed. A four times enlarged detail (b) superimposed onto the original image of resolution 0.25 m. (c) Comparison to the line extraction without bias removal.

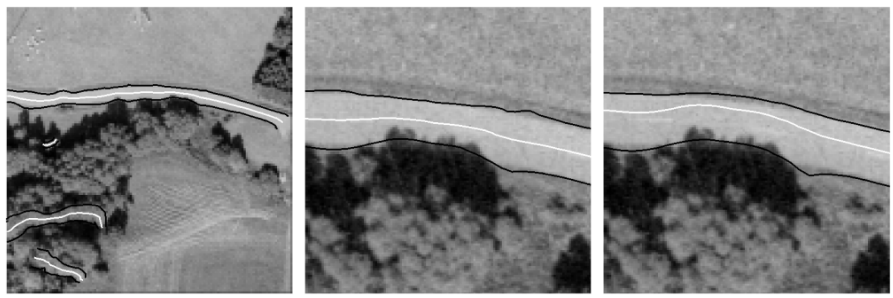


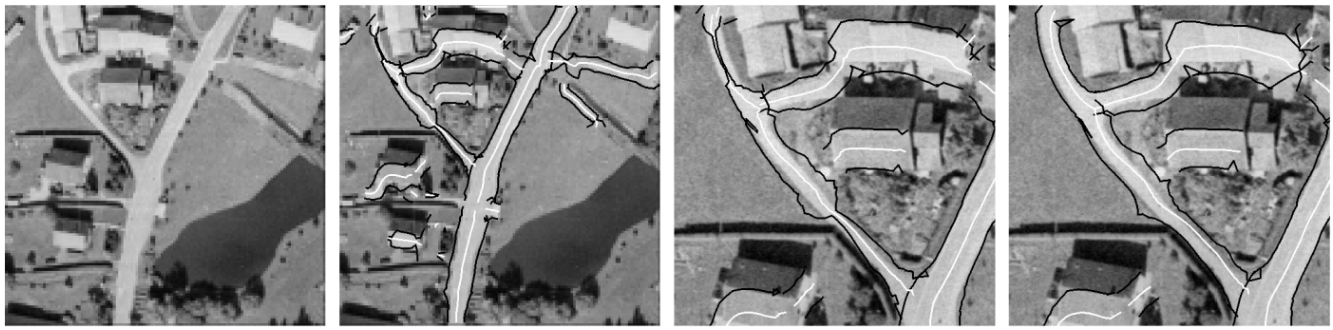
Fig. 14. Lines and their width detected (a) in an aerial image of resolution 1 m with the bias removed. A four times enlarged detail (b) superimposed onto the original image of resolution 0.25 m. (c) Comparison to the line extraction without bias removal.

converge, especially for small v_σ. Therefore, the inverse f^{-1} is precomputed 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. 11 are the entries in the table of f^{-1}. Fig. 12 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 ill-behaved for small w_σ f^{-1} is quite well-behaved. This leads to the conclusion that bilinear 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.

5.3 Examples

Fig. 13 shows how the bias removal algorithm is able to successfully adjust the line widths in the aerial image of Fig. 8. Fig. 13a shows 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. Fig. 13b shows a four times enlarged part of the results



(a) (b) (c) (d)

Fig. 15. 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.

(a) (b) (c) (d)

Fig. 16. Lines and their width detected (b) in an 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.

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. Fig. 13c shows the same detail without the removal of the bias. In this case, the extracted edges are about two to four pixels from their true locations.

Fig. 14 shows the results of removing the bias from the test image of Fig. 9. In the areas of the image where the line is highly asymmetrical, especially in the center part of the road, 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. 14b. If this is compared to the detail in Fig. 14c, 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. Fig. 15a shows an aerial image, again of ground resolution 1 m. This image contains a large area where the model of the line does not hold. There is a very narrow line starting in the upper-left corner and continuing to the center 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 here can be gained from [31], where line models of up to three parallel lines are examined. Fig. 15b 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 two, as can be seen from Fig. 10a. However, the algorithm is still able to move the line position to within the true line in its asymmetrical part. This is displayed in Figs. 15c and d. The extraction results are enlarged by a factor of two and superimposed onto the original image of ground resolution 0.25 m. 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 from the domain of medical imaging, and was taken from [11]. Fig. 16a shows a magnetic resonance (MR) image of a human head. The results of extracting bright lines with bias removal are displayed in Fig. 16b, while a three times enlarged detail from the image is given in Fig. 16c. 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

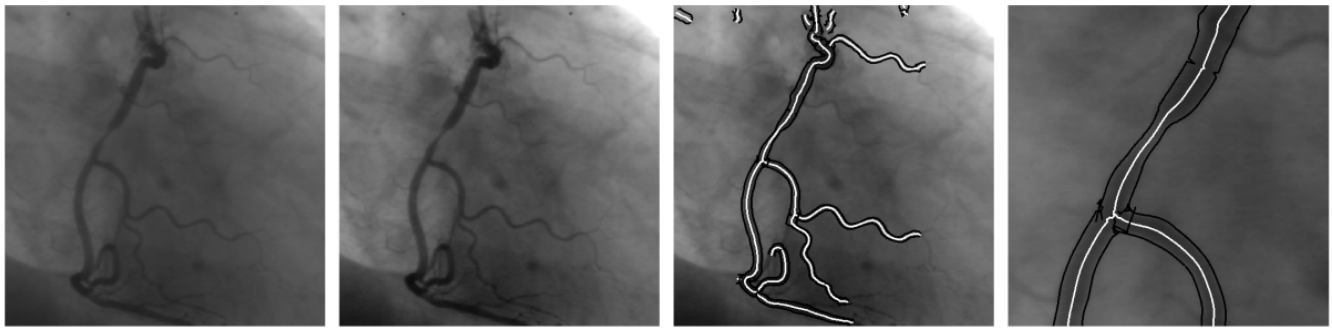


Fig. 17. Lines detected in the coronary angiogram (a). Since this image has very low contrast, the results (c) extracted from (a) are superimposed onto a version of the image with better contrast (b). A three times enlarged detail of (c) is displayed in (d).

this to Fig. 16d, where the line extraction was done without bias removal. The line positions are much worse for the gyri of the brain since they are highly asymmetrical lines. A comparison to the results in [11] clearly shows the superiority of this approach to all ridge detectors presented there.

The final example is again from the domain of medical imaging, but this time the input is an X-ray image. Fig. 17 shows the results of applying the proposed approach to a coronary angiogram. Since the image in Fig. 17a has very low contrast, Fig. 17b shows the same image with higher contrast. Fig. 17c displays the results of extracting dark lines from Fig. 17a, the low-contrast image, superimposed onto the high-contrast image. A three times enlarged detail is displayed in Fig. 17d. The algorithm is very successful in delineating the vascular stenosis in the central part of the image, while it was also 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 [2] 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.

6 Conclusions

This paper has presented an approach to extract lines and their widths with high precision. A model for the most common type of lines, the asymmetrical bar-shaped line, was introduced. A scale-space analysis carried out for this profile 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 occurring in an image, while asymmetry influences 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. Furthermore, tests have been carried out on synthetically generated images in order to evaluate the accuracy of the extracted lines. Subpixel shifts were modeled by adjusting the gray values at the sides of the line. In the experiments, the extracted points were never more than 0.04 pixel away from the true line location. A thorough evaluation of the algorithm in the presence of noise, however, still needs to be done. 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 [15]. 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 small masks. This makes the approach computationally efficient. For example, the time to process the MR image of Fig. 16 of size 256×256 is about 1.7 seconds on an 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 2.5 σ. 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 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 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 (8) to obtain the

desired thresholds. Again, this is not a severe restriction, 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 [12], [32]. 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.

7 Code Availability

An implementation of the algorithm described in this paper is available from
ftp://ftp9.informatik.tu-muenchen.de/pub/detect-lines/

References

[1] A. Baumgartner, C. Steger, H. Mayer, and W. Eckstein, "Multi-resolution, Semantic Objects, and Context for Road Extraction," *Semantic Modeling for the Acquisition of Topographic Information From Images and Maps*, W. Förstner and L. Plümer, eds., pp. 140-156. Basel, Switzerland: Birkhäuser Verlag, 1997.

[2] C. Coppini, M. Demi, R. Poli, and G. Valli, "An Artificial Vision System for X-Ray Images of Human Coronary Trees," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 2, pp. 156-162, Feb. 1993.

[3] J.B.A. Maintz, P.A. van den Elsen, and M.A. Viergever, "Evaluation of Ridge Seeking Operators for Multimodality Medical Image Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 353-365, Apr. 1996.

[4] 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*, vol. 15, pp. 201-223, 1981.

[5] D. Geman and B. Jedynak, "An Active Testing Model for Tracking Roads in Satellite Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, pp. 1-14, Jan. 1996.

[6] M.A. Fischler, "The Perception of Linear Structure: A Generic Linker," *Image Understanding Workshop*, pp. 1,565-1,579. San Francisco: Morgan Kaufmann Publishers, 1994.

[7] M.A. Fischler and H.C. Wolf, "Linear Delineation," *Computer Vision and Pattern Recognition*, pp. 351-356. Los Alamitos, Calif: IEEE CS Press, 1983.

[8] T.M. Koller, G. Gerig, G. Székely, and D. Dettwiler, "Multiscale Detection of Curvilinear Structures in 2-D and 3-D Image Data," *Fifth Int'l Conf. Computer Vision*, pp. 864-869. Los Alamitos, Calif: IEEE CS Press, 1995.

[9] J.B. Subirana-Vilanova and K.K. Sung, "Multi-Scale Vector-Ridge-Detection for Perceptual Organization Without Edges," A.I. Memo 1318, MIT Artificial Intelligence Lab., Cambridge, Mass., Dec. 1992.

[10] I.S. Kweon and T. Kanade, "Extracting Topographic Terrain Features From Elevation Maps," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 59, no. 2, pp. 171-182, Mar. 1994.

[11] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach, "Ridges for Image Analysis," Technical Report TR93–055, Dept. of Computer Science, Univ. of North Carolina, Chapel Hill, 1993.

[12] J.J. Koenderink and A.J. van Doorn, "Two-Plus-One-Dimensional Differential Geometry," *Pattern Recognition Letters*, vol. 15, no. 5, pp. 439-443, May 1994.

[13] 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, France, Feb. 1995.

[14] A. Busch, "A Common Framework for the Extraction of Lines and Edges," *Int'l Archives of Photogrammetry and Remote Sensing*, vol. 31, part B3, pp. 88-93, 1996.

[15] L. Wang and T. Pavlidis, "Detection of Curved and Straight Segments From Gray Scale Topography," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 58, no. 3, pp. 352-365, Nov. 1993.

[16] L. Wang and T. Pavlidis, "Direct Gray-Scale Extraction of Features for Character Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1,053-1,067, Oct. 1993.

[17] R.M. Haralick, L.T. Watson, and T.J. Laffey, "The Topographic Primal Sketch," *Int'l J. Robotics Research*, vol. 2, no. 1, pp. 50-72, 1983.

[18] T. Lindeberg, "Edge Detection and Ridge Detection With Automatic Scale Selection," *Computer Vision and Pattern Recognition*, pp. 465-470. Los Alamitos, Calif: IEEE Computer Society Press, 1996.

[19] L.A. Iverson and S.W. Zucker, "Logical/Linear Operators for Image Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 10, pp. 982-996, Oct. 1995.

[20] C. Steger, "Extracting Curvilinear Structures: A Differential Geometric Approach," *Fourth European Conf. Computer Vision*, *Lecture Notes in Computer Science*, vol. 1,064, B. Buxton and R. Cipolla, eds. pp. 630-641. Springer-Verlag, 1996.

[21] L.M.J. Florack, B.M. ter Haar Romney, J.J. Koenderink, and M.A. Viergever, "Scale and the Differential Structure of Images," *Image and Vision Computing*, vol. 10, no. 6, pp. 376-388, July 1992.

[22] R. Deriche and G. Giraudon, "A Computational Approach for Corner and Vertex Detection," *Int'l J. Computer Vision*, vol. 10, no. 2, pp. 101-124, Apr. 1993.

[23] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, England: Cambridge Univ. Press, 1992.

[24] M. Shah, A. Sood, and R. Jain, "Pulse and Staircase Edge Models," *Computer Vision, Graphics, and Image Processing*, vol. 34, pp. 321-343, 1986.

[25] J.S. Chen and G. Medioni, "Detection, Localization, and Estimation of Edges," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 191-198, Feb. 1989.

[26] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1994.

[27] R. Deriche, "Recursively Implementing the Gaussian and Its Derivatives," Rapport de Recherche 1893, INRIA, Sophia Antipolis, France, Apr. 1993.

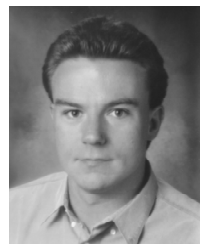
[28] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, Nov. 1986.

[29] A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, 2nd ed., vol. 2. San Diego, Calif.: Academic Press, Inc., 1982.

[30] D.F. Rogers, *Procedural Elements for Computer Graphics*. New York: McGraw-Hill Book Company, 1985.

[31] H. Mayer and C. Steger, "A New Approach for Line Extraction and Its Integration in a Multi-Scale, Multi-Abstraction-Level Road Extraction System," *Mapping Buildings, Roads and Other Man-Made Structures From Images*, F. Leberl, R. Kalliany, and M. Gruber, eds., pp. 331-348. Vienna: R. Oldenbourg Verlag, 1996.

[32] A.M. López and J. Serrat, "Tracing Crease Curves by Solving a System of Differential Equations," *Fourth European Conf. Computer Vision*, *Lecture Notes in Computer Science*, vol. 1,064, B. Buxton and R. Cipolla, eds., pp. 241-250. Springer-Verlag, 1996.



Carsten Steger received his master's degree in computer science from the Technische Universität München in 1993. He is currently a PhD student at the Forschungsgruppe Bildverstehen FG BV (computer vision research group) at the same institution. His main research interests are semantic models for the interpretation of aerial images, low-level vision, perceptual grouping, and knowledge-based graphical user interfaces for computer vision.