High Speed, Multi-Scale Tracing of Curvilinear Features with Automated Scale Selection and Enhanced Orientation Computation

R. D. Wedowski, A. R. Farooq, L. N. Smith, M. L. Smith
Machine Vision Laboratory, University of the West of England, Bristol, UK
raphael2.wedowski@uwe.ac.uk

ABSTRACT

We propose a new high speed line tracing algorithm based on a well known differential geometric line extraction algorithm. The previously separate steps of line detection and line tracing are performed simultaneously. This allows the exclusion of non-candidates from processing. Exploiting the inherent continuity of lines and using extracted line characteristics in subsequent detection/tracing also solves the problem of multiple, computationally expensive scale space iterations. Consequently, processing time is shown to be reduced by up to a factor of fifty. Furthermore, the extraction is very sensitive as hard to set global thresholds are no longer required. In the context of these proposals, we also review methods to identify the pixel-wise line orientation. The previously used orientation of maximum second derivative proved to suffer from systematic errors, whereas, our two novel methods proved more reliable. Our algorithm is designed for images containing only a single line but can be applied to images with multiple lines, especially if the global image structure is known.

KEYWORDS: Curve Tracing, Line Detection, Gaussian Scale Space, High Speed, Sub-Pixel

1. INTRODUCTION

The extraction of curvilinear features, or simply (curved) lines, is a very important operation in low level vision. Applications include the extraction of roads from aerial images [1], the extraction of vessels from MRI-images [2] and the processing of solar images [3]. Our application requires the rapid extraction and tracing of specular signatures. Our study concerns the on-line inspection and reverse engineering of complex specular surfaces, an industrial problem that has so far only been solved for Lambertian surfaces [4]. Outcomes of the project will bring benefits to many different manufacturing industries, including those of ceramic, metal and lacquered or polished goods or pharmaceutical packaging. We are exploiting what we coined as the specular signature, in other words the reflection of a laser line of the specular surface on to the back of a translucent screen. We have developed an apparatus where a continuous stream of images of the front of this screen is taken while the specular specimen travels beneath it on top of a conveyor belt. Often the surfaces contain topographic textures and the resulting projection is therefore a highly complex curvilinear structure. As these signatures can take on arbitrary shapes and can vary in scale, no direct information is apparent to facilitate the extraction/tracing. This therefore is very challenging. What sets our problem apart from general line or edge detection is that, as long as the observed surface is continuous, only one single signature line will be present. We can also set the geometry of our apparatus in such a way that the signatures are not self intersecting. It is crucial for us to know the path of the signature and we furthermore require the precise extraction of the line centre as well as line width. To make the device suitable for high speed production lines, we also depend on high speed processing. To bridge large gaps that can appear in the signature and flag the presence of surface defects, it is also crucial to precisely know the orientation of the line at every point.

So far, we have not found evidence of line extraction methods that address our problem, with our required combination of attributes (processing speed, sub-pixel accuracy, line width determination, ability to bridge gaps). We close this gap by a straight forward interweavement of the extraction and tracing steps of a well known differential geometric line extraction algorithm [5], that we carry out simultaneously. By exploiting a line's inherent continuity, we can extract every line point using knowledge of its characteristics, most importantly its position, orientation, scale and degree of bulging. This brings a large reduction in processing time. Moreover, the method makes redundant any global thresholds as required in previous work. As these are notoriously difficult to set, it is virtually impossible to satisfy all regions of an image, especially if

different scales are present. This results in a vastly improved sensitivity of our algorithm.

The remainder of this paper is organized as follows. Section two looks at related work and discusses the underlying algorithm. Section three introduces our novel algorithm and discusses line orientation computation; section four shows and discusses exemplary experiments. The conclusion follows in section five.

2. LINE EXTRACTION

Line detection, tracing and extraction are well known problems and have been investigated for several decades. Popular methods include active contours [6] and Hough transform based approaches. The former is a notoriously slow technique whilst some effort has gone into creating a real time Hough transform [7, 8]. However, the Hough transform, which normally relies on a binary input (and hence a pre-processed, thresholded image), does not compute line orientation or width and is limited to detecting predefined analytical functions such as straight lines or circles. Often multiple tuned filters [1] or filters in combination with subsequent region growing [9] are used. The result is a rough estimate of the line orientation; the line width cannot be determined. Sargin et al [10] chose an initial line point and grew the line along a tree with a branching factor of eight where each branch represents a neighbouring pixel. As criterion they use a second directional derivative threshold. A real time line extraction algorithm was proposed in [11] where the Laplacian edge detector is applied to the area of pixels that are most likely to represent the following course of the line. Again, no sub-pixel accurate line point detection is possible and no accurate information on orientation and width are extracted.

More sophisticated methods employ differential geometry. The image is seen as a height map and line centre positions are defined as regions of high directional curvature. They return the line orientation in addition to reliable and sub pixel accurate line centre positions and also extract the width of lines. A particularly well made algorithm was published by Carsten Steger in 1998 [5] and has since then been the basis of a much research work. Like all differential line detection algorithms it relies on robust and meaningful derivatives which can be obtained by convolution with the Gaussian kernel. However, the kernel parameters have to be precisely matched to the feature's scale to obtain an optimum balance between noise suppression and accuracy. Therefore a time intensive scale space iteration needs to be employed, a technique that greatly limits the speed of application. Here we begin to identify a requirement for a method that omits the need for iteration and guarantees best scale parameter selection at the same time.

The classical approach consists of treating the scale space scale by scale, i.e. by combining the responses and searching for maxima in scale space. Some publications deal with the optimum selection of the parameters. Lindeberg [12] proposed an automatic selection based on the local degree of diffuseness of the image while [13] aim to select a set of best scales based on maximizing the correlation between the smoothed and the noise free image which is estimated using robust statistics. However, all these methods iterate through scale space first and select the best scales by comparison afterwards.

We propose a straight forward approach for scale selection. This is made possible as we trace and detect lines simultaneously in a purely local approach. As foundation for our algorithm, the method of Steger is used. Steger detects line centre points as points where the directional gradient is zero and the second directional derivative has a high absolute value (negative for ridges and vice versa). Line edges are defined as the points of maximum directional gradient on a sweep perpendicular to the line. In a first step, line pixels are identified within the whole image matrix. In a second step, the detected line pixels are linked together. Because of their close proximity and the known line orientation, this is a fairly easy procedure.

Since for bar shaped lines the derivatives vanish in its interval, the original image is convolved with a Gaussian smoothing kernel

$$I(x, y, \sigma) = I(x, y) * G(x, y, \sigma) \tag{1}$$

where

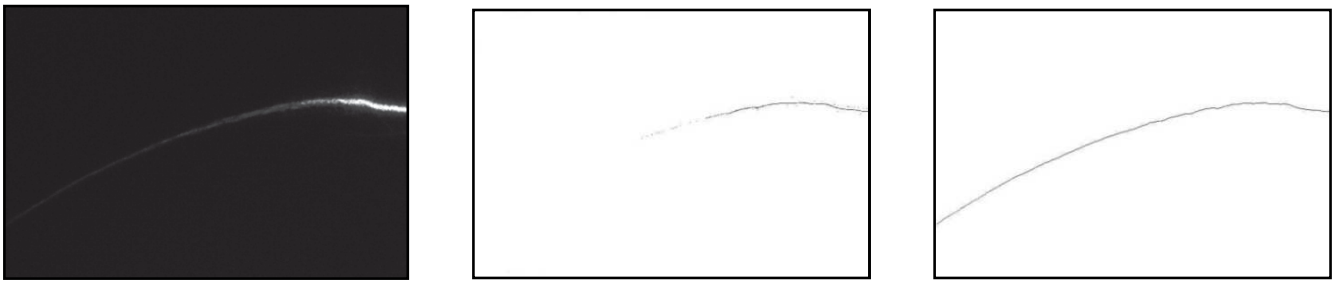
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{2}$$

is the Gaussian kernel. The image derivatives can then be calculated using the central difference scheme:

$$I_x(x, y, \sigma) = \frac{I(x + h, y, \sigma) - I(x - h, y, \sigma)}{2h} \tag{3}$$

where h is the spacing, in our case one pixel. I_y, I_{xx}, I_{yy} are calculated respectively. Alternatively convolution with the Gaussian derivative kernels is suitable as well.

It was shown by [14] and [15] that the convolution with well chosen Gaussian kernels produces the desired second derivative maxima at the centre position for different line shapes, i.e. bar-shaped, parabolic and roof-like. The centre line point (the position of the zero crossing of the directional curvature) is found by setting the derivative of



(a) **(b)** **(c)**

Figure 1. Fading Out Signature (Detail); (a)Input; (b)Limitations Of Global Threshold; (c)Result Of Novel Method

the second order Taylor polynomial to zero. In two dimensions it follows that the centre point location is:

$$[p_x, p_y]^T = [tn_x, tn_y]^T \quad (4)$$

where

$$t = \frac{I_x n_x + I_y n_y}{I_{xx} n_x^2 + 2 I_{xy} n_x n_y + I_{yy} n_y^2} \quad (5)$$

and $[n_x, n_y]^T$ is the orientation of the maximum second derivative computed as the eigenvector of maximum eigenvalue of the Hessian matrix

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix} \quad (6)$$

and hence the orientation perpendicular to the line. Note that the line orientation computation is ultimately based on a very limited neighbourhood.

A pixel is declared a line point if p_x, p_y, the interpolated line centre position, lies within its boundaries

$$[p_x, p_y]^T \epsilon \left[-\frac{1}{2}, \frac{1}{2}\right] \mathrm{X} \left[-\frac{1}{2}, \frac{1}{2}\right] \quad (7)$$

This alone creates manifold false positive responses. Therefore, as final criterion, thresholding of the magnitude of the curvature (the degree of bulging) is applied. This reliably excludes pixels in the image background but has to be individually fine tuned to detect only line centre pixels. Figure 1 demonstrates the difficulty of defining this global threshold. However, once a line centre is defined, the line edges are found along a sweep perpendicular to the line orientation. In a second step the identified line pixels are linked together. Therefore the found line orientations have to be harmonised as initially they can point in either one of the two opposing directions of line travel.

However, the standard deviation σ of the kernel has to be accurately matched to the scale of the object to be extracted. In the case of lines, scale denotes line width. If σ is chosen too small, the resulting convolution profile will be flat around the line centre which hence cannot be detected. Steger as well as Canny [16] have shown that, in order for the second directional derivative to have its maximum at x=0, the line centre position, $\sigma \geq \frac{\sqrt{3}}{6} W$ has to hold, where W is the full width of the line. On the other hand, if σ is chosen too large, oversmoothing will occur and relevant information can be lost. Steger, in the context of accurate width determination also observed that the edges of the line can never move closer to the line centre than σ. Hence

$$\frac{\sqrt{3}}{6} W \leq \sigma \leq \frac{1}{2} W \quad (8)$$

represents the lower and upper bounds of sigma. The range of scales of lines that can be detected by a given fixed sigma is therefore very limited. In the work of Steger, multi-scale processing is not included. The algorithm is for this reason limited to certain scales. Multi-scale processing could be implemented if the line bits identified at different scales were added in the end. However, in addition to the vast time requirement, note that with this approach a different global threshold would have to be defined for every single scale.

Table 1. Required Optimal Sigmas σ For Line Widths W Between 4 And 45 Pixels

W	< 8	9-14	15-18	19-22	23-27	28-32	33-38	39-45
σ	2.0	4	6.4	7.6	9.2	11.2	13.2	15.6

3. REAL TIME CONVERSION

3.1. Principal Algorithm

Through a simple change in the algorithm, all these issues can be resolved and scale space computation can be included without the need for time-consuming iterations. A full scale space analysis is only performed along one column or row that the sought after line is guaranteed to cross. In our case, that is the centre column of the image. Every successive step is then only performed for those pixels that lie in the immediate extension of the latest previously detected point; non candidates are strictly excluded. All characteristics of a line, be it width, profile shape or orientation, will only change slowly along its path. Therefore the previously employed global threshold

can be omitted and instead a local threshold of the curvature's magnitude can be introduced that for every pixel depends on the curvature of the previously detected line pixel. Additionally scale space iteration can be avoided as the optimal Gaussian kernel scale parameter can be determined through the width of the previous line point.

Using local curvature thresholding is sufficient to trace curves with high precision and over various levels of width. Fading out lines can be traced well (see Figure 1). However if curves are fading out, the second directional curvature is slowly reduced until the resulting threshold is close to zero. Consequently, in these conditions the algorithm tends to detect false positives in the background of the image. However, these bogus line pixels have no order; their width and orientation change randomly. Hence two more variable local thresholds, defining the line's change in orientation and width, are introduced. They can be set loosely in order to guarantee not to stop the detection/tracing prematurely and will still guarantee to stop the algorithm after very few false positives. Without further investigation, we found that 10 pixels as a width threshold and around 70 degrees as orientation threshold proved successful.

Due to the local approach, it is especially important that the algorithm is able to bridge gaps and intersections to guarantee continued processing. Otherwise whole line parts can be missed. Steger's linkage algorithm was enhanced by [17]. They compute a first order Radon transform and search for the line segment that yields the maximum inner product with the image. In cases where no direct neighbour can be identified as line member, they search within a triangle centred on the average orientation of the last M line points and with internal angle θ. We use the same triangle and apply our tracing algorithm on an outwards sweep within it (see Figure 2). When no line pixels can be identified the search distance d is increased. If d gets larger than a predefined maximum gap size, the last detected pixel is declared a line end and the algorithm stops. As M we used 10, θ was set to 40 deg; the maximum gap size depends on the image and the occurring gaps; we used 35 pixels

The algorithm is initialised by detecting seed pixels through application of Steger's algorithm to a single centred column. No thresholding is applied in order not to exclude any lines. As a result, multiple false seed points will be created. These can however be easily excluded as real lines are usually several hundred pixels long while the algorithm breaks down after only a few pixels for bogus lines. In our case, however, we limit the seed points to the one with the highest directional curvature as we are after one line only. Every seed point is created twice with opposing directions so that detected lines are traced in both directions. Should one line cross the initialisation column more than once, the line is tackled from various points simultaneously. Hence several single line parts will be created and have to be connected. This however is straight forward as two connected line parts will inevitably detect each other's ends. The complete algorithm is presented in the shape of a flow chart in Figure 4.

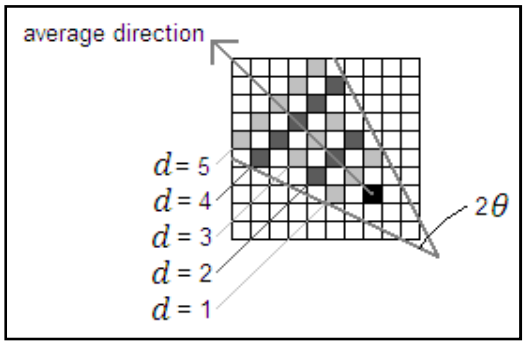


Figure 2. Search Sequence At Gaps And Intersections With Increasing Distance *d*

3.2. Accurate Line Orientation Computation

If directly neighbouring pixels are to be connected, a crude orientation accuracy of 0.75 π is sufficient to limit the search to three pixels within the eight-pixel-neighbourhood. However, a precise measure is needed whenever the distance between two conjoint pixels is larger as caused by gaps or intersections.

In the original work of Steger, the orientation of the maximum second derivative at the line centre location is used. While this is a valid and reasonably robust approach, the second derivate has a tendency towards the horizontal, vertical and diagonal orientations as it is only dependent on a limited neighbourhood (the 4-pixel neighbourhoods of its 4-pixel neighbourhood). The discrete nature of an image does not allow for accurate interpolations.

To include a wider neighbourhood, a weighted average of the orientations of the maximum gradients along a sweep perpendicular to the line can be employed. To detect the line borders and hence the line width, this sweep is necessary anyway. As weighting factor, the gradient magnitude is employed. This is based on the idea that gradients with a high magnitude, as they appear at the edges of the line, are predominantly defined by the line and are less vulnerable to surrounding noise. Noise sources could e.g. be differences in the line intensity. Hence

$$n_x(p_x, p_y) = \frac{\sum_{a=-w}^{w} n_x(P_x, P_y) \left\| \nabla(P_x, P_y) \right\|_2}{\sum_{a=-w}^{w} \left\| \nabla(P_x, P_y) \right\|_2} \qquad (9)$$

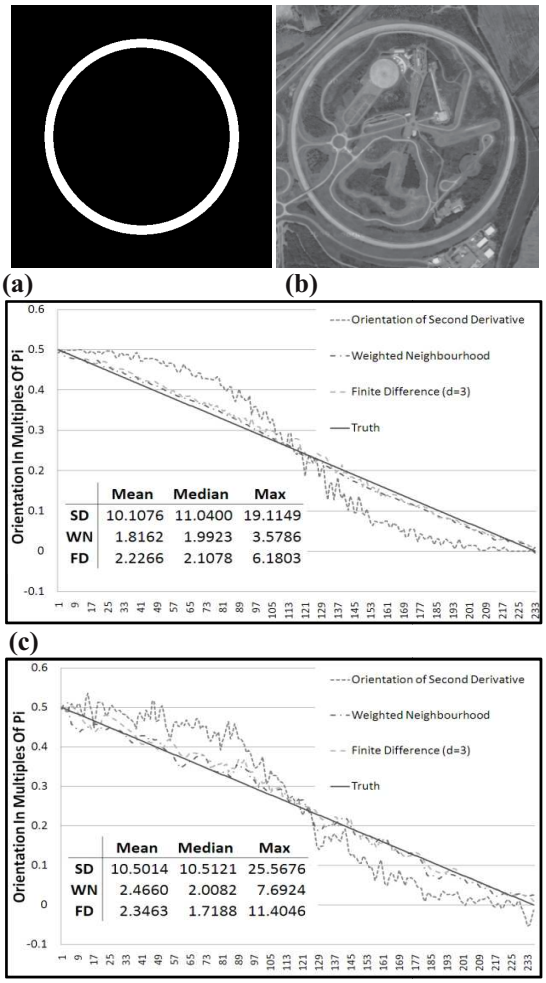


Figure 3. Comparison Of Orientation Computation Of Existing And Our Two Novel Algorithms. Input (a) And Results (c) For Artificial Image; Input (b) And Results (d) For Real Image

where

$$P_x = p_x + a\, n_{x\,Prior} \quad for\ (-w < a < w) \tag{10}$$

$$P_y = p_y + a\, n_{y\,Prior} \quad for\ (-w < a < w) \tag{11}$$

$n_{x\,Prior}$, as a first approximation, is the orientation of maximum curvature and

$$n_x(P_x, P_y) = \begin{cases} -\ n_x(P_x, P_y) & , a < 0 \\ \quad n_x(P_x, P_y) & , a > 0 \end{cases} \tag{12}$$

$n_y(p_x, p_y)$ is calculated accordingly. The latter step takes into account that gradients on opposing sides have opposing orientations. Furthermore it harmonises the defined line direction. As the preceding pixel position will always be known, once the sub pixel accurate centre position of the actual pixel is computed, the orientation can also be calculated using a backwards difference scheme.

$$n_x = \frac{p_x - p_{x\,Prior}}{\|n_x, n_y\|} \tag{13}$$

n_y accordingly.

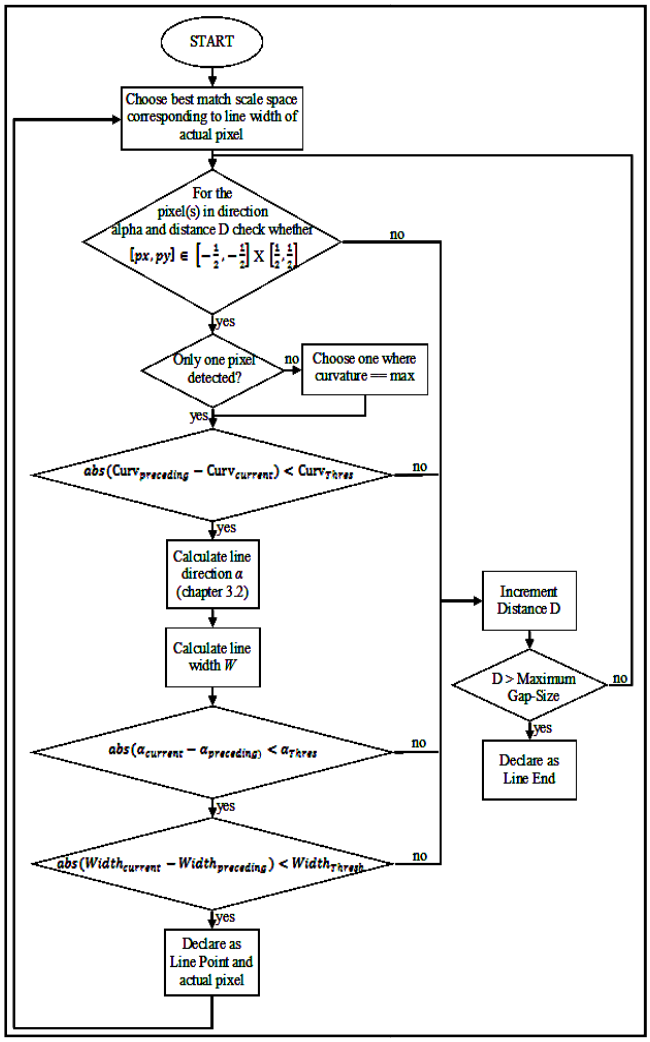


Figure 4. Flow Chart Of The Algorithm

Here, the weighted gradient neighbourhood orientation is used to define the subpixel centre position. The two methods show a very similar accuracy, giving confirmation of the high accuracy of the subpixel centre position computation. In our algorithm the orientation is chosen that minimises the difference to the orientation of the previous pixel.

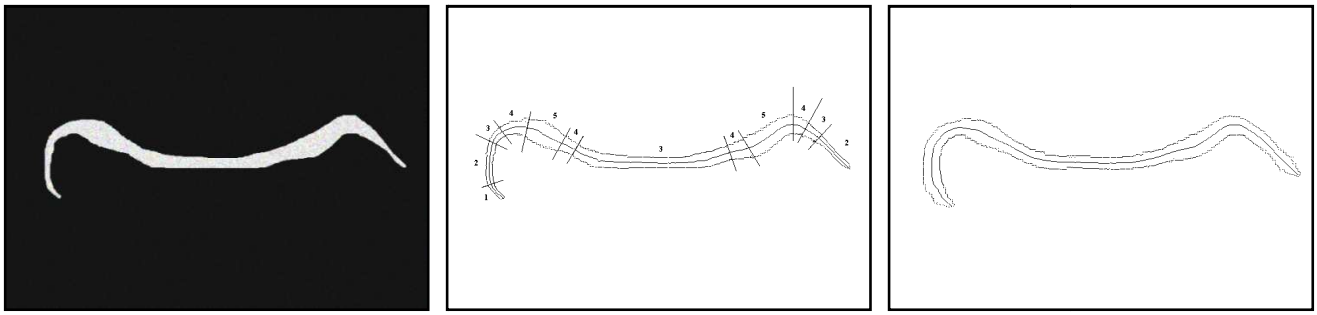
Figure 3 compares the performance of the three methods, i.e. maximum second derivative (SD), weighted gradient neighbourhood (WN) and finite difference (FD). We computed the pixel-wise orientations of the upper right quarter of a synthetic as well as of a real image of a circle so that the ground truth is known. The real world data is represented by the aerial image of a circular automotive testing ground of roughly one kilometre in diameter. Sigma was carefully matched to the line widths of seventeen (artificial) and seven (real) pixels respectively.

Table 2. Summary Of Results, Comparison Of Processing Times To Original Algorithm And Breakdown Thereof

Image Name	# Of Pixels (Size)	# Of Line Pixels	# Of Implemented Scales	Processing Time Of Our Algorithm (s)	Processing Time of Steger's Algorithm (s)	Ratio	Time Required For Convolutions (Both Algorithms) (s)	Time Required Per Line Pixel (Our Algorithm) (s)	Time Required Per Image Pixel And Scale (Steger's Algorithm) (s)
Signature 1	1,028,000 (500*2056)	1920	1	1.35	24.75	**18.4**	0.1	6.50E-04	2.40E-05
Signature 1	1,028,000 (500*2056)	1951	8	3.90	205.09	**52.6**	2.78	5.75E-04	2.46E-05
Signature 2	1,028,000 (500*2056)	1885	1	1.29	24.40	**18.9**	0.16	6.00E-04	2.36E-05
Signature 2	1,028,000 (500*2056)	1890	8	4.38	208.78	**47.7**	3.22	6.14E-04	2.50E-05
River 1	564,160 (688*820)	1112	1	0.83	13.70	**16.5**	0.08	6.74E-04	2.41E-05
River 1	564,160 (688*820)	1112	8	2.97	109.05	**36.7**	2.27	6.29E-04	2.37E-05
River Elbe	91,840 (287*320)	260	8	0.52	17.60	**34.0**	0.34	6.83E-04	2.35E-05
Multi Scale Snake	307,200 (480*640)	619	8	1.53	62.00	**40.6**	1.13	6.40E-04	2.48E-05

(a) **(b)** **(c)**

Figure 5. River 1 Traced; Input (a), Result Of Our Algorithm (b), Result Of Canny Edge Detector (c)



(a) **(b)** **(c)**

Figure 6. Artificial Multi-Scale Image "Snake": Input (a), Result Using Our Multi-Scale Algorithm (b), Poor Result Using Only One Scale (c)

As the curvature is constant, the true orientation is a straight line running from 0.5 π to 0. Figure 3 a) and b) show the input images, c) and d) plot the calculated orientations over line pixels for the artificial and the real image. The latter two also include key statistical error measures. Note that in c) and d) the second derivative orientation tends towards the horizontal orientation until it declines steeply, cuts the ground truth at the diagonal orientation and quickly converges towards the vertical orientation. Both the gradient neighbourhood method and the backwards difference scheme prove much more accurate than the orientation of maximum curvature. For this, the critical maximum error reaches 19 deg and 25 deg for the artificial and real image respectively which makes the reliable bridging of gaps or intersections nearly impossible. The weighted gradient neighbourhood method's error reaches a maximum of only 3.5 and 7.6 deg respectively, while the values for the finite difference method are 6.2 and 11.4 deg.

4. EXPERIMENTS

In this section the benefits of the algorithm are further demonstrated. Although the described high speed algorithm was primarily designed to rapidly extract and trace specular signatures, in the following also examples of its performance on two aerial images and one artificial image will be presented and discussed. The curvilinear structures on one of the aerial images and on the artificial image strongly vary in scale and require multi scale processing. The results and the processing times in comparison to the underlying algorithm are summarised in Table 2.

Figure 1 shows details of a fading out line, where a) shows the input image, c) shows the result after tracing with our approach and b) the result of the extraction step

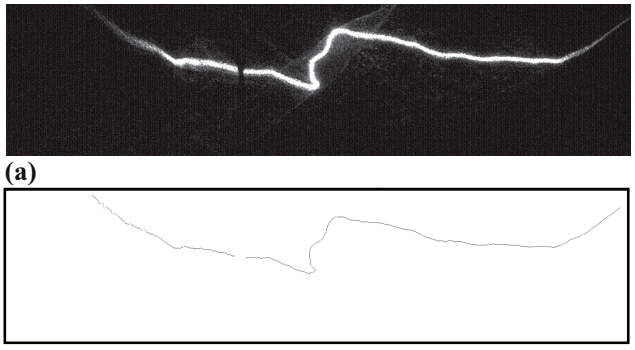
of the original algorithm. It shows in b) that, because of the global threshold, line pixels at the left extreme of the image are excluded while false positives exist above and beneath the line where it was detected. In c), the signature is well traced. Both algorithms used the same fixed sigma of 4 at a detected line width between 3 and 12 pixels; processing times for the uniscalar implementation were 24.75 seconds for a), 1.35 seconds for b); image size was 2056*500 pixels and the detected line had a length of 1920 pixels.

Another example of the algorithm's results in its designated area is given in Figure 7 with input image in a) and traced line in b). The gap in the line was well bridged and the background noise around the image centre did not influence the tracing. The image size was 2056 * 500 pixels, 1885 line pixels were identified and processing took 1.29 seconds. As in Figure 1, similar results can be obtained by Steger's original code, with processing taking 24.4 seconds.

Figure 5 shows the result of our algorithm applied to the aerial image of a river and compares it to the result achieved with the Canny edge detector. Where a) is the input image, b) is the line centre and edges computed by our algorithm and c) shows the result using Canny edge detection. Note that b) and c) were both applied to the hue plane of the image (not shown). Processing time for our algorithm was 0.83 seconds while that of the Matlab® standard implementation of Canny was 0.66 seconds and hence of the same order of magnitude. However, within the given time, our algorithm computed sub-pixel centre position, line edges and line-connectivity while c) only contains the edges. Image size was 688*820 pixels; 1112 line-pixels were detected. For the three images discussed so far, a single scale implementation was sufficient as the line's widths were constant. However, Table 1 also lists the respective processing times for multi scale implementations using 8 scales. Here, a few extra line pixels could be detected where the signatures were fading out.

The benefits of multi scale processing are demonstrated in Figure 8. It shows an aerial image of the river Elbe and the multi scale tracing result. The river's width on the image varies between 5 and 49 pixels. The algorithm was implemented using the eight different scales given in Table 1. These were in ascending order automatically applied to the eight different areas marked in Figure 8 b). As can be seen, the river is well traced and its borders are reasonably accurately detected; only the small aits towards the lower end were lost in the smoothing process. Also the algorithm did not trace accurately where the river became wider than the maximum sigma allowed. However, a similar accuracy or length of the tracing can not be established using a single scale. The eight-scale run time of our algorithm was 0.52 seconds for the image of size 287*320 pixels and 260 detected line pixels Eight runs of the original algorithm with the respective scales would take 17.6 seconds with the problem of optimal merging of the output images and the need for five different global thresholds.

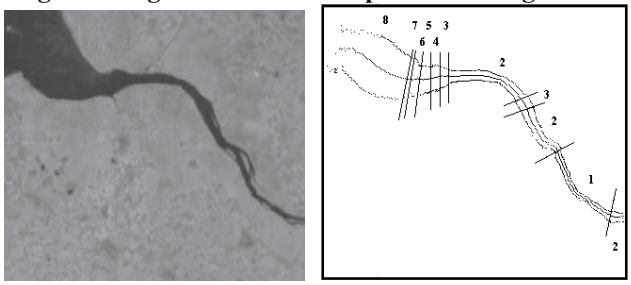
Figure 6 shows an artificial image of size 480*640 pixels, showing a line with varying width. The image was corrupted with uniformly distributed random noise with mean 0 and variance 0.04. Again the algorithm was implemented with eight-scales. However, only the five largest were actually employed. Our algorithm detected 619 line-pixels. The areas where different scales were applied are marked in Figure 6 b) in ascending order. Multi-Scale line tracing using eight different scales took 1.53 seconds. The theoretical run time of Steger's algorithm, including eight convolutions and eight successive iterations in scale space over the whole image matrix, is 62.0 seconds. Figure 6 c) shows the result using only one scale, in this case sigma = 12. Because of the prominence of the line in the image, it is traced despite the lack of scale-variation but differences in the line width are largely levelled out and centre as well as edge detection are far less accurate.



(a)

(b)

Figure 7. Signature 2 With Gap And Tracing Result



(a) **(b)**

Figure 8. The River Elbe And Its Delta: Input (a) And Tracing Result Using Five Different Scales (b)

5. CONCLUSION

In this paper an existing and widely acknowledged differential geometrical line tracing algorithm was

substantially advanced in that the required processing time was dramatically reduced while at the same time multi-scale processing was implemented. Our algorithm carries out line point identification and line tracing simultaneously and hence allows the two steps to fructify each other. It was shown that time requirement can be reduced by a factor of more than 50 but can be reduced even more as the speed advantage rises with bigger images, shorter lines and more required scales. Our novel algorithm delivers high level tracing results with a speed similar to that of basic low level edge-detectors. It has the added benefit of improved accuracy as the previously required global threshold could be replaced by a local one that does not have to be set manually. Although applied here to single line images, the algorithm is fully able to trace several independent lines once seed points are identified.

Furthermore in this paper we presented novel and more accurate methods to define a line's pixel-wise orientation. This largely simplifies the bridging of large gaps and intersections. Through accurate direction computation using a backwards difference scheme the supreme accuracy of the sub-pixel line centre position computation could once more be proven. The algorithm was tested using images of specular signatures as well as artificial and aerial images showing lines of varying width and containing gaps.

ACKNOWLEDGEMENTS

This work was jointly funded by Great Western Research and FIMA SI Ltd under project code 280.

REFERENCES

[1] F.M. Porikli, "Road extraction by point-wise Gaussian models", Proceedings of SPIE, pp. 758-764, 2003.

[2] C. Renault, M. Desvignes and M. Revenu, "3D curves tracking and its application to cortical sulci detection", ICIP 2000, Vancouver, pp. 491-494, 2000.

[3] J.K. Lee, T.S. Newman and G.A.Gary, "Automated detection of solar loops by the oriented connectivity method", 17th International Conference on Pattern Recognition, Cambridge, UK, pp. 315-318, August 2004.

[4] A. R. Farooq, M. L. Smith, L. N. Smith and P. S. Midha, "Dynamic photometric stereo for on line quality control of ceramic tiles", *Computers in Industry*, 56(8), pp. 918-934, 2005.

[5] C. Steger, "An unbiased detector of curvilinear structures", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2), pp. 113-125, 1998.

[6] X. Bresson, P. Vandergheynst and J.P. Thiran, "Multiscale active contours", *International Journal of Computer Vision,* 70(3), pp. 197-211, 2006.

[7] L.A.F. Fernandes and M.M. Oliveira, "Real-time line detection through an improved Hough transform voting scheme", *Pattern Recognition,* **41**(1), pp. 299-314. 2008.

[8] M. Nakanishi and T. Ogura, "Real-time CAM-based Hough transform algorithm and its performance evaluation" *Machine Vision and Applications*, **12**(2), pp. 59-68, 2000.

[9] A. Plaza, E. Cernadas, M.L. Duran, P.G. Rodriguez and M.J. Petron, "Multi-Scale Detection of Curvilinear Structures with High Contour Accuracy", 5th Iberoamerican Symposium on Pattern Recognition, Lisbon, pp. 405-412, 2000.

[10] M.E. Sargin, A. Altinok, K. Rose and B.S. Manjunath, "Tracing curvilinear structures in live cell images", IEEE International Conference on Image Processing, San Antonio, pp. 285–288, 2007.

[11] J.Gates, M. Haseyama and H. Kitajima, "A real-time line extraction algorithm", Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, Orlando, pp. IV-68 – IV-71, 1999.

[12] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection" *International Journal of Computer Vision*, **30**(2), pp. 117-154, 1998.

[13] I. Vanhamel, C. Mihai, H. Sahli, A. Katartzis and I. Pratikakis, "Scale Selection for Compact Scale-Space Representation of Vector-Valued Images", *International Journal of Computer Vision*, 84(2), pp. 194-204, 2009.

[14] T.M Koller, G. Gerig, G. Szekely and D. Dettwiler, "Multiscale detection of curvilinear structures in 2-D and 3-D image data", Proceedings of the Fifth International Conference on Computer Vision, Washington, pp. 864-870, 1995.

[15] G.J. Streekstra, R. van den Boomgaard and A.W.M. Smeulders, "Scale dependency of image derivatives for feature measurement in curvilinear structures", *International Journal of Computer Vision*, 42(3), pp. 177-189. 2001.

[16] J. Canny, "A computational approach to edge detection" in Readings in Computer Vision: Issues, Problems, Principles and Paradigms, vol. 1, M. A. Fischler and O. Firschlein, Eds, Los Altos: Morgan Kaufmann Publishers, 1987, pp. 184-203.

[17] K. Raghupathy and T.W. Parks, "Improved curve tracing in images", International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Montreal, pp. 581-584, 2004.