CURVE TRACING AND CURVE DETECTION IN IMAGES

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Karthik Raghupathy

August 2004

ABSTRACT

This thesis studies the problems of curve tracing and curve detection in images. We begin with a description of the curve tracing problem and compare it with curve detection/extraction, a fundamentally different problem. Issues concerning curve tracing are addressed. We describe a good curve tracing algorithm proposed by Carsten Steger and improve this method to better suit the goals of curve tracing. Results show that the improved algorithm emulates human perception by using both global and local features.

Next, we consider the problem of curve detection. The use of the Generalized Radon Transform (GRT) for detection of curves in images is well known. We discuss several issues that arise when applying the GRT for curve detection. We then describe improvements to the GRT to address these issues. Modifications are in the form of incorporating local information in the Generalized Radon Transform and this results in improved curve detection.

BIOGRAPHICAL SKETCH

Karthik Raghupathy received his Bachelor of Technology degree in Electrical Engineering from the Indian Institute of Technology, Madras in 2002. Since 2002, he has been a research and teaching assistant in the Department of Electrical and Computer Engineering at Cornell University. He was with IBM's T.J.Watson Research Center as a technical intern in the summer of 2003. His research interests include image modeling, image analysis, speech recognition, speech processing and other areas of signal processing. He was awarded the 2002 Olin Fellowship from the Graduate School of Cornell University. He was also awarded the Phillips India Prize and the Siemens Prize for the student with the best academic record in Electrical Engineering during the period 1998-2002 at the Indian Institute of Technology, Madras.

To my parents

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Prof. Thomas Parks, for his guidance and encouragement. I thank Prof. Sheila Hemami for her help and advice. Her course in digital image processing gave me good insight into human perception of images. I thank Keigo Hirakawa for many suggestions, advice and insightful discussions.

I thank Dr. Thomas J. Barnard from Lockheed Martin for introducing me to this problem and for his helpful suggestions towards solving this problem. I thank Lockheed Martin Naval Electronics and Surveillance Systems, Syracuse and the Center for Electronic Imaging Systems, Rochester for supporting this work.

I thank all my friends for their support and good cheer. Finally, I thank my parents and relatives for their support and love.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Curve tracing and curve detection in images are important operations in image processing and computer vision. Their applications include road following, object tracking, fracture detection in borehole images and extraction of features from medical images. However, curve tracing and curve detection are fundamentally different problems.

Curve tracing is the operation of extracting **and** automatically classifying curves in images. In curve tracing, we want to identify an individual curve, label it, and follow it across the image, even if it crosses other curves, fades out, and then reappears. The result that we expect from a curve tracing algorithm is one that matches the identification done by a person. In curve detection, the emphasis is on the extraction and not on the identification or labeling of a curve.

Chapter 2 describes the problem of curve tracing in detail. Issues concerning curve tracing are addressed. We describe a good curve tracing algorithm proposed by Carsten Steger and improve this method to better suit the goals of curve tracing.

In Chapter 3, we consider the problem of curve detection. The use of the Generalized Radon Transform (GRT) for detection of curves in images is well known. We discuss several issues that arise when applying the GRT for curve detection. We then describe improvements to the GRT to address these issues. Modifications are in the form of incorporating local information in the Generalized Radon Transform and this results in improved curve detection.

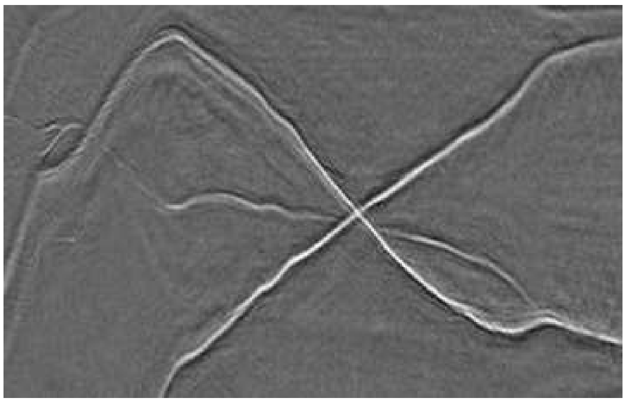
The central idea in this work is the use of both local and global information for detecting and tracing curves in images. Steger's method for curve tracing (described in Chapter 2) is a local method. Our algorithm achieves improvements by incorporating global information in Steger's curve tracing algorithm. Similarly, the use of the GRT is a global approach for curve detection and our improvements are achieved by incorporating local information in this method. A human observer has both an overall perception (a global view) as well as a local perception of an image. Thus, in order to emulate human perception, curve tracing and curve detection algorithms should incorporate both local and global information.

Chapter 2

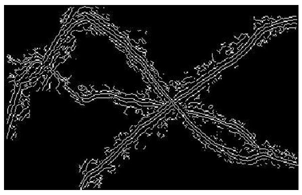
Curve tracing

2.1 Introduction

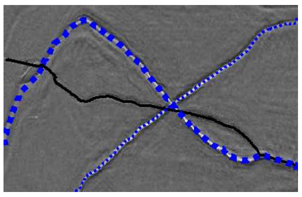
Curve tracing is the operation of extracting **and** automatically classifying curves in images. A person can trace several curves simply by looking carefully at the image. For example, it is easy to visually trace three curves shown in Figure 2.1(a). Curve tracing has many applications in computer vision and image processing. In aerial images, curve tracing is used to extract curvilinear features such as roads, rivers and railroads [2]. Curve tracing is also used in medical imaging for the extraction of anatomical features such as blood vessels from an X-ray angiogram [3].



(a) Original image



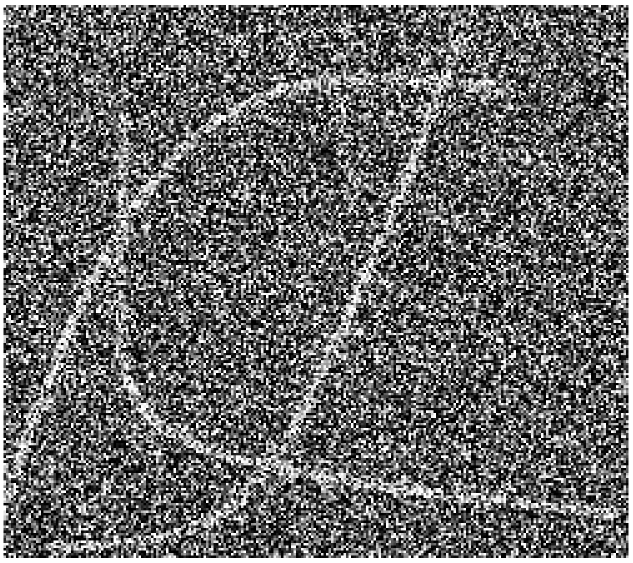
(b) Result of curve extraction



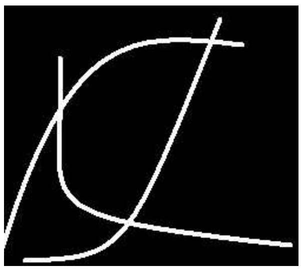
(c) Desired result

Figure 2.1: Curve tracing

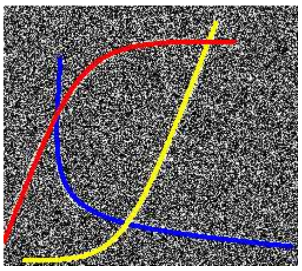
Curve tracing, however, is different from traditional curve extraction/detection where curves are extracted but not necessarily classified. In curve extraction [4, 5, 6], the emphasis is on the extraction and not on the identification or labeling of a curve. Note that the terms curve extraction and curve detection will be used interchangeably in this thesis. A Canny edge detector [6] is a simple curve extraction algorithm. Figure 2.1(b) shows the result of curve extraction using MATLAB's Canny edge detector. Many multi-resolution algorithms [4, 5] developed recently, such as curvelets and beamlets are very effective curve extraction algorithms and are used for compressing images with curves because they preserve the curves in a compressed image. Beamlets [4] are a collection of line segments at a range of locations, orientations, and scales, that give a multi-scale approximation to the collection of all line segments in an image. Polygonal curves in the $x - y$ plane are then built up by chaining beamlets together. Similarly, Curvelets [5] are a multiscale approach used to represent objects with edges and curves. These methods work better than edge detection algorithms when dealing with noisy images. However, all these methods would result in curves extracted but not labelled. The difference between curve extraction and curve tracing can be better understood using Figure 2.2. Figure 2.2(a) is a noisy image on which an edge detection algorithm (such as the Canny edge detector) would give poor results. However, curve extraction algorithms such as Beamlets and Curvelets work very well even with noisy images. Figure 2.2(b) shows the desired result of curve extraction on Figure 2.2(a).



(a) Original noisy image



(b) Result of curve extraction



(c) Result of curve tracing

Figure 2.2: Difference between curve extraction and curve tracing

As can be observed from Figure 2.2, curve extraction methods cannot be applied to the problem of curve tracing as they only extract or detect curves in an image. For curve tracing (see Figure 2.2(c)), we want to identify an individual curve, label it, and follow it across the image, even if it crosses other curves, fades out, and then reappears. The result that we expect from a curve tracing algorithm is one that matches the identification done by a person. The desired result of curve tracing on Figure 2.1(a) is shown in Figure 2.1(c).

A good curve tracing algorithm, proposed by Steger [1], uses a local approach to curve tracing. It is local in the sense that the algorithm classifies pixels as belonging to a curve or not, and then links curve points together. Section 2.2 describes this algorithm briefly. The result of curve tracing using this approach is shown in Figure 2.3. As can be observed, some curves in the image have been traced but have not been identified as a person would identify them. This is illustrated in Figure 2.3 where the algorithm follows the big dotted and the small dotted curves erroneously at the junction. That is, the algorithm follows a curve at a junction contrary to how the human eye would do so. We shall refer to this as incorrect curve following at junctions. We also observe that the two solid curve segments running horizontally through the image are perceptually just one curve that fades out and emerges again. However, the algorithm traces this single curve as two separate curves. Another problem is that the algorithm does not trace out the faint curves in the upper left portion of the image. These problems arise from the local nature of the algorithm. As a result, the algorithm cannot track faint curves that fade out or disappear and emerge again. In addition, at junctions the algorithm does not use any global information to perform curve tracing.

There are several issues that need to be taken into account when developing a curve tracing algorithm. Some of them include (i) Noisy images (ii) Multiple and possibly intersecting curves (iii) Disappearance and re-emergence of a curve (which is perceptually the same curve) and (iv) Tracking curves that fade out. In this chapter, I present modifications of the local algorithm [1] by incorporating global features to address the above mentioned issues.

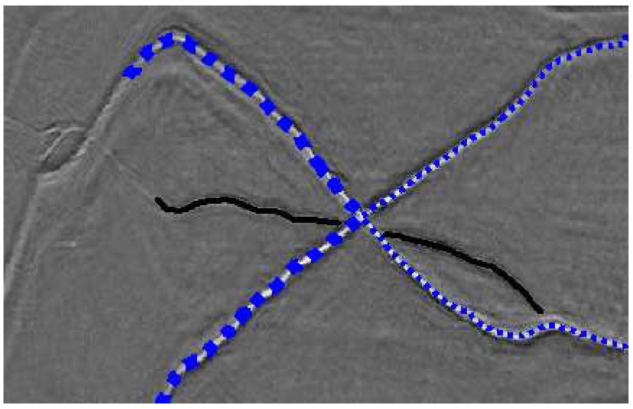


Figure 2.3: Result of curve tracing using [1]

A note about Pre-processing

It is conceivable that pre-processing techniques such as edge detection, curve extraction (using beamlets, curvelets etc.), and contrast stretching can improve the performance of a curve tracing algorithm. However, after testing this idea, we observed that pre-processing techniques do not, in general, improve the performance of a curve tracing algorithm. This arises out of the fact that pre-processed images do not retain the same image structure as the original image. That is, it is tougher for human observers to trace out curves from a pre-processed image as compared to the original image. In general, pre-processing techniques (such as edge detection for example) cause a loss of important curve information, especially when dealing with faint curves. As a result, since it is tougher even for humans to trace out curves from a pre-processed image, one cannot expect an algorithm to do better with pre-processed images.

Outline of the chapter

Steger's algorithm for curve tracing is described in Section 2.2. Section 2.3 describes our improvements to Steger's algorithm to tackle the issues mentioned earlier in this section namely (i) Curve following at junctions (See Section 2.3.1 and (ii) Tracing faint curves that fade out or disappear and re-emerge again (See Section 2.3.2). Some results are shown and discussed in Section 2.4. The selection of parameter values is discussed in Section 2.5. The performance of the improved curve tracing algorithm on noisy images is dealt with in Section 2.7. The chapter concludes with Section 2.8.

2.2 Local method

We begin with a method described by Steger [1] and modify the algorithm suitably. In accordance with Stegers method, the first step involves classifying pixels as being curve points or not depending on whether they lie on some curve in the image. The next step is the linking of curve points found in the first step to form curves.

2.2.1 Classification of curve points

Curvilinear structures in a 2D image can be modelled as curves $s(t)$ that exhibit a characteristic 1D line profile in the direction perpendicular to the line (refer Figure 2.4). The 1D line profile is characterized by a vanishing first derivative (i.e. a local maximum is achieved). Let the direction perpendicular to the curve be $n(t)$. Thus, at a curve point (i.e. a point lying on a curve), the first directional derivative in the direction $n(t)$ should vanish and the second directional derivative should be of large absolute value. When dealing with images, pixels are considered to be on the integer grid $Z_N \times Z_N$ (where Z_N is the ring of integers). A pixel (x, y) has a boundary defined by the unit square $[x - \frac{1}{2}, x + \frac{1}{2}] \times [y - \frac{1}{2}, y + \frac{1}{2}]$. Hence, a pixel in an image is classified as a curve point if the first derivative along $n(t)$ vanishes within a unit square centered around the pixel (i.e. within the pixel's boundaries). The problem of computing the direction perpendicular to a curve is tackled by

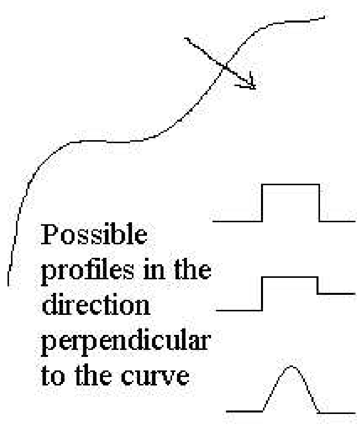

Possible profiles in the direction perpendicular to the curve

Figure 2.4: Classification of curve points

calculating the eigenvalues and eigenvectors of the Hessian matrix

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix} \tag{2.1}$$

The partial derivatives I_{xx}, I_{xy}, I_{yx} and I_{yy} are computed numerically using partial differences after convolving the image with a Gaussian smoothing kernel. Gaussian smoothing is essential since the problem of estimating derivatives of a noisy image is ill-posed and it is the only kernel that, under certain, very general, assumptions makes the problem well posed. This also leads to a scale-space description of the image [7, 8].

The direction in which the second directional derivative takes on its maximum absolute value is used as the direction perpendicular to the curve $n(t)$. It is a standard result in Differential Geometry [9] that this direction can be computed by finding the eigenvector corresponding to the maximum absolute eigenvalue of the Hessian matrix. To see this, note that the second derivative I_{vv} of the image I along a direction $\bar{v}$ is given by

$$I_{vv} = \bar{v}^T H \bar{v} \tag{2.2}$$

For example, the unit vector in the x direction is $[1 \quad 0]^T$. We get the second derivative in this direction by substituting $\bar{v} = [1 \quad 0]^T$ in 2.2 to obtain

$$I_{vv} = [1 \quad 0] \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = I_{xx} \tag{2.3}$$

We can do this with any unit vector, $\bar{v}$, in any direction. Now, the quantity $\bar{v}^T H \bar{v}$ (see Equation 2.2) is maximized by choosing $\bar{v}$ to be the eigenvector corresponding to the maximum absolute eigenvalue of H. This eigenvector is the direction of greatest curvature (second derivative). As mentioned earlier, this eigenvector is used as the direction perpendicular to the curve. Let this eigenvector be given by (n_x, n_y). A quadratic polynomial is then used to determine whether the first directional derivative along (n_x, n_y) vanishes within the current pixel. This point can be obtained by inserting (tn_x, tn_y) into the Taylor polynomial

$$z(x,y) = I(x,y) + \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} I_x \\ I_y \end{pmatrix} + \frac{1}{2} \begin{pmatrix} x & y \end{pmatrix} H \begin{pmatrix} x \\ y \end{pmatrix}$$

where $z(x,y)$ represents the curve.

By setting the derivative of $z(x,y)$ along (tn_x, tn_y) to zero and solving for t, we obtain

$$t = -\frac{n_x I_x + n_y I_y}{n_x^2 I_{xx} + 2n_x n_y I_{xy} + n_y^2 I_{yy}} \tag{2.4}$$

and hence the maximum or minimum occurs at

$$(p_x, p_y) = (tn_x, tn_y) \tag{2.5}$$

Thus the point (x, y) is declared a curve point if (p_x, p_y) falls within the pixel boundaries. That is, if $(p_x, p_y) \in [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$, then the point (x, y) is declared a curve point. The value of the second directional derivative along (n_x, n_y), i.e., the maximum eigenvalue, is used as a measure of the strength of the curve. That is, if λ_1 and λ_2 are the eigenvalues of H, then the strength $\mu(x, y)$ is defined as

$$\mu(x, y) = \max(\lambda_1, \lambda_2) \tag{2.6}$$

Also, (p_x, p_y) is used to calculate the sub-pixel location of the curve point. That is, the subpixel location (s_x, s_y) of a curve point (x, y) is given by:

$$(s_x, s_y) = (x + p_x, y + p_y) \tag{2.7}$$

Note that the orientation, $\theta(x, y)$, of the curve at the point (x, y) can be obtained using the normal (n_x, n_y)

$$\theta(x, y) = \arctan\left(\frac{-n_x}{n_y}\right) \tag{2.8}$$

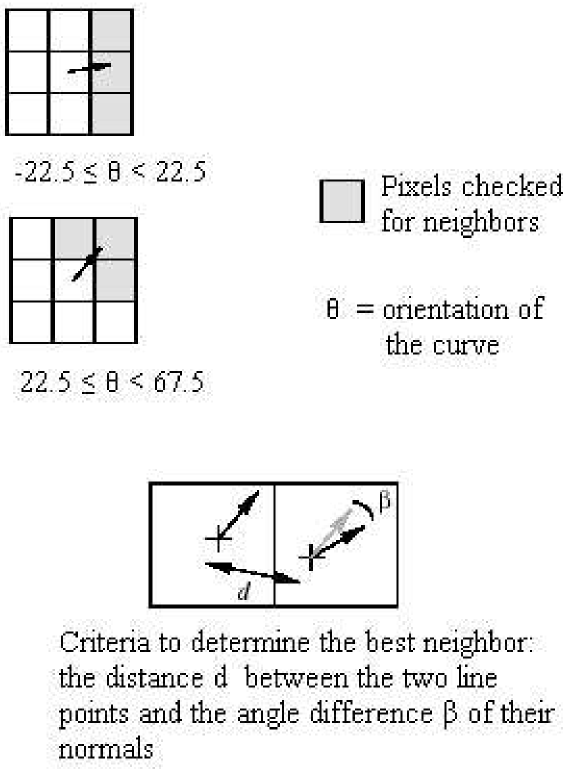
Note that both $\theta(x, y)$ and $\pi + \theta(x, y)$ are candidates for orientation of the curve at (x, y).

2.2.2 Linking of curve points

Linking proceeds by using the orientation information obtained in the classification step and joins curve points appropriately to form curves. Starting from the pixel with the maximum second derivative (i.e. maximum strength), curves are constructed by adding the appropriate neighbor to the current curve. Given a curve point, three neighboring pixels compatible with the current orientation of the curve are examined. The choice of the next curve point to be added is based on the distance between sub-pixel locations and the angle difference of the orientation of the curve at the two points. The linking terminates when there are no more curve points in the neighborhood of the current pixel. Figure 2.5 illustrates the linking process.

As mentioned in connection with Equation 2.8, there are two candidates for the orientation of the curve at a point. In accordance with [1], the problem of orienting the normals $n(t)$ of the curve is solved by the following procedure. First, at the starting point of the curve the normal is oriented such that it is turned $-90^{\cdot}$ to the direction the curve is traversed, i.e., it points to the right of the starting point. Then at each curve point there are two possible normals whose angles differ by $180^{\cdot}$. 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 curve as it is traversed from start to end.

Figure 2.3 shows the result of Steger's algorithm on Figure 2.1(a).



Criteria to determine the best neighbor: the distance d between the two line points and the angle difference β of their normals

Figure 2.5: Linking of curve points

2.3 Improvements - Incorporating global features

As mentioned earlier, Steger's algorithm suffers from two main problems (i) Curve following at junctions and (ii) Tracing faint curves that fade out or disappear and re-emerge again. The ideal curve tracing algorithm should emulate human perception. To do so, we have to remember that a human observer has an overall perception (a global view) as well as a local perception of an image. Thus the ideal curve tracing algorithm has to incorporate features from both the local as well as the global structure.

Sections 2.3.1 and 2.3.2 describe improvements to the linking algorithm of [1] to make it more global in nature. We first provide some definitions and notation that will be used in the following sections.

Let the set of pixels in the image be denoted by the set I. Let the set of curve points (as defined in Section 2.2.1) be denoted by the set C. Given $(x_1, y_1) \in C$ and $(x_2, y_2) \in C$, the difference in orientation of a curve at these curve points is denoted by $\Delta\theta(x_1, y_1, x_2, y_2)$ and is given by

$$\Delta\theta(x_1, y_1, x_2, y_2) \quad = \quad |\theta(x_2, y_2) - \theta(x_1, y_1)| \tag{2.9}$$

where $\theta(x_2, y_2)$ and $\theta(x_1, y_1)$ are as defined in 2.2.1. Owing to the fact that there are two possible candidates for the orientation of the curve at each point, the orientations are chosen such that the angle difference, $\Delta\theta(x_1, y_1, x_2, y_2)$, is minimized. Also, denote the distance $d(x_1, y_1, x_2, y_2)$ as the distance between two pixels $(x_1, y_1) \in I$ and $(x_2, y_2) \in I$. That is,

$$d(x_1, y_1, x_2, y_2) \quad = \quad \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.10}$$

2.3.1 Curve following at junctions

The linking algorithm proposed by Steger produces results that lead to incorrect curve following at junctions. To correct this problem, we modify the algorithm to disallow large bending of curves at junctions. Incorrect curve following at a junction is characterized by a large change in the curve orientation. Thus, when a large change in orientation of the curve is encountered while linking, the modified algorithm prevents this by looking ahead in the current direction and finding a more suitable curve point to be added to the existing curve.

Formally, let $(x_1, y_1) \in C$ be the current curve point and let $(x_2, y_2) \in C$ be the next curve point according to Steger's linking process. Incorrect curve following at a junction is hence characterized by the equation

$$\Delta\theta(x_1, y_1, x_2, y_2) \quad > \quad \beta_0 \tag{2.11}$$

where β_0 is a threshold controlling the allowable angle change.

Once a large change in orientation is detected (by Equation 2.11), incorrect curve following at a junction is prevented by looking ahead in the current direction. The predicted next curve point is given by

$$(x_2^{pred}, y_2^{pred}) \quad = \quad (x_1 + L\cos(\theta(x_1, y_1)), y_1 + L\sin(\theta(x_1, y_1))) \tag{2.12}$$

where L determines the extent to which we look ahead for the next curve point. We then search for a suitable curve point in the vicinity of (x_2^{pred}, y_2^{pred}). To do this, form the set S as

$$S \quad = \quad \Big\{(x, y) \in C| \; d(x, y, x_2^{pred}, y_2^{pred}) < R \;\wedge\; \Delta\theta(x, y, x_1, y_1) < \beta_0\Big\} \tag{2.13}$$

where R is the radius of the disc of pixels centered at (x_2^{pred}, y_2^{pred}) over which we perform our search for a suitable curve point. The most suitable next curve point (x_2^*, y_2^*) is then found using

$$(x_2^*, y_2^*) \quad = \quad \arg \min_{(x,y) \in S} \Delta\theta(x, y, x_1, y_1) \tag{2.14}$$

If such a suitable curve point is not found (i.e. the set S is empty), then the most suitable next curve point is assigned to be (x_2, y_2), the next curve point according to Steger's linking algorithm. This is the case when the curve actually does undergo a large change in orientation (such as a V shaped curve).

The above process emulates human perception of continuity at curve intersections and leads to accurate curve following at junctions.

2.3.2 Tracing faint curves

Steger's linking algorithm terminates when there are no more curve points in the immediate vicinity of the current curve point. This is clearly too local and hence the algorithm cannot track curves that (i) fade out or (ii) disappear and re-emerge but are perceptually the same curve.

Curves that fade

In order to tackle the problem of tracing curves that fade out (e.g. the faint curves in the upper left portion of the image), we use ideas from the Radon transform [10, 11]. The Radon transform is a global approach to curve extraction. Curves that are to be extracted are modeled and this is followed by calculation of the inner product of the image with all possible curves represented by the model. A particular example is the first order Radon Transform [12] that extracts lines out of an image. A line can be represented by two parameters (such as slope and intercept) and by varying these parameters, various lines can be obtained. Thus, the Radon Transform of a continuous two-dimensional function $g(x, y)$ is found by stacking or integrating values of $g(x, y)$ along slanted lines. The location of the line is determined from the line parameters; slope p and line offset τ.

$$\breve{g}(p, \tau) \quad = \quad \int_{-\infty}^{\infty} g(x, px + \tau)dx \tag{2.15}$$

Salient curves are extracted from the image by locating inner product maxima in the parameter space. In the case of a first order Radon Transform, the inner product of each line with the image is calculated to give rise to a two dimensional parameter representation (in the (p, τ) domain instead of the (x, y) domain) of the image. Peaks in the parameter domain correspond to the salient lines present in the image. The Radon transform can be used effectively to extract faint curves from noisy images. Radon's original paper [11] deals with the extension of the Radon Transform to higher order polynomials and other general functions. In related work [13], we show how to use the Radon transform for curve tracing.

To trace curves that fade, modify the linking algorithm as follows; Once a curve has been declared to have ended (i.e. there are no more curve points in the vicinity), we look for the possibility of a faint curve by calculating the inner product of the image with various curves (we chose straight lines) starting from the last known curve point. Let θ be the average orientation of the curve just prior to it being declared as terminated. We use this average orientation information to limit our search over inner products. That is, we only compute the inner product of the image with lines whose orientations belong to the set $[\theta - \phi, \theta + \phi]$ where

2ϕ defines the size of the interval over which we search for the maximum inner product (Refer Figure 2.6). If the maximum inner product is higher than a certain threshold, then this would indicate the presence of a curve that is fading out. As a result, we can trace a curve even if it fades. Formally, let $(x_1, y_1) \in C$ be the

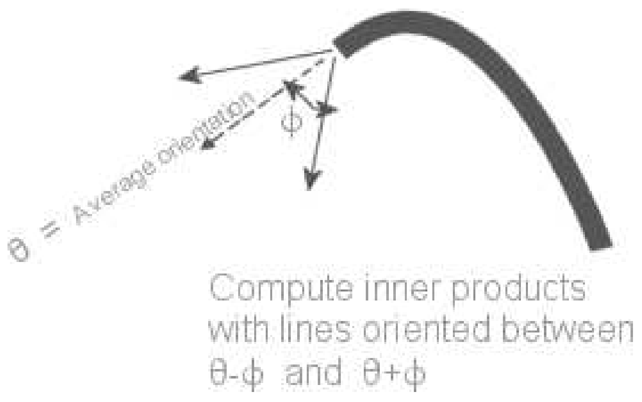


Figure 2.6: Tracing curves that fade

curve point when the curve is declared as ended. We wish to find out if the curve fades out and trace the faint curve to the extent a human observer would be able to. As mentioned earlier, let θ be the average orientation of the curve over the last M curve points, where M controls the extent to which we look back. We then find the line segment which yields the maximum inner product with the image I, using the following:

$$\theta_0 = \arg \max_{\psi \in [\theta-\phi, \theta+\phi]} \sum_{k=0}^{\ell-1} I(x_1 + k\cos\psi, y_1 + k\sin\psi) \tag{2.16}$$

That is, we perform a search over all line segments starting from the point (x_1, y_1) with length ℓ having slope (orientation) in the range $[\theta - \phi, \theta + \phi]$ and find the line segment (defined by its slope θ_0) that yields the maximum inner product. If the curve actually fades out, this line segment (θ_0) will result in a high inner product. Thus, by comparing the value of this maximum inner product with a threshold, we can trace a curve even if it fades out before ending. Clearly, by performing a search over higher order curves (rather than lines), we can trace a fiat curve more accurately.

Disappearance and re-emergence of curves

We modify the linking algorithm by predicting the possible path of a curve once it has been declared to have ended. This is done again my making use of the curve extracted prior to it being declared as terminated. Thus, when the algorithm is not able to find any curve points in the vicinity of the current curve point, three possibilities arise: (i) The curve actually terminates or (ii) The curve fades (this case has been tackled in the previous section) or (iii) The curve disappears and re-emerges later. To account for these possibilities, we first determine the possible path of the curve by using the average orientation of the curve just prior to it being declared as terminated. Figure 2.7 illustrates this process. Having found the

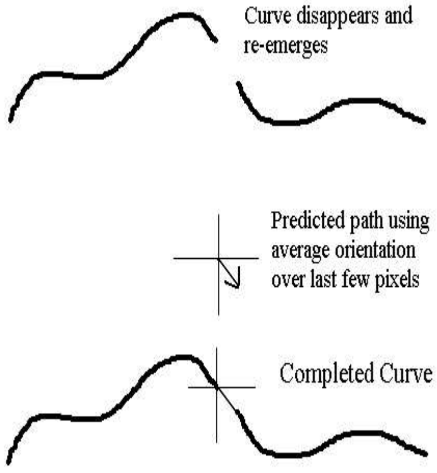

Curve disappears and re-emerges
Predicted path using average orientation over last few pixels
Completed Curve

Figure 2.7: Tracing re-emergent curves

average orientation of the curve over the last few curve points, it is extrapolated in this average direction to give the possible path of the curve. If the curve actually fades out and re-emerges, then the end of the extrapolated curve should be near the re-emergence of the curve. Thus, a search for a suitable curve point is done in the neighborhood of the end of the predicted curve. A suitable curve point is defined using two metrics (i) The difference in curve orientation at that point and the average orientation extracted as above and (ii) The difference between the strength of the curve at that point and the average strength of the curve nearby. In doing so, we are trying to find a curve whose characteristics are similar to the one extracted hitherto. If no suitable curve point is found, then the curve is declared as terminated.

Formally, let $(x_1, y_1) \in C$ be the curve point when the curve is declared as ended. We wish to find out if this curve disappears and re-emerges again. Let θ be the average orientation of the curve over the last M curve points, where M controls the extent to which we look back. Similar to Section 2.3.1, we find the predicted next curve point (x_2^{pred}, y_2^{pred}) as

$$(x_2^{pred}, y_2^{pred}) \quad = \quad (x_1 + K\cos(\theta), y_1 + K\sin(\theta)) \tag{2.17}$$

where K determines the extent to which we look ahead for the next curve point. Again, similar to steps described in Section 2.3.1, we search for a suitable curve point in the vicinity of (x_2^{pred}, y_2^{pred}). To do this, we form the set S as

$$S \quad = \quad \Big\{(x, y) \in C|\ d(x, y, x_2^{pred}, y_2^{pred}) < R\ \wedge\ |\theta(x, y) - \theta| < \beta_o\Big\} \tag{2.18}$$

Note that, by substituting $\theta = \theta(x_1, y_1)$ we obtain the same equations as those mentioned in Section 2.3.1. However, the definition of a suitable curve point in this case is different from that of Section 2.3.1 as we also use curve strength as a metric in addition to curve orientation. Let μ be the average strength of the curve over the last M points. The next suitable curve point (x_2^*, y_2^*) is then found using

$$(x_2^*, y_2^*) \quad = \quad \arg \min_{(x,y)\in S} (w|\theta(x,y) - \theta| + (1-w)|\mu(x,y) - \mu|) \tag{2.19}$$

where w is a weight that controls the significance of orientation similarity and strength similarity to finding a suitable curve point. If such a suitable curve point is not found (i.e. the set S is empty), then the curve does not re-emerge and the curve is declared as terminated.

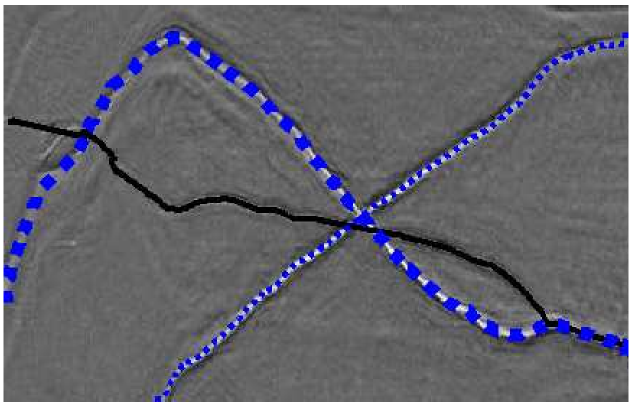


Figure 2.8: Results of improved curve tracing algorithm

The results of the above mentioned improvements are shown in Figure 2.8. Note that the problem of incorrect curve following at junctions and the problem of tracing faint curves have been solved for this example.

2.4 More results

Figures 2.10 and 2.13 show the results of using the improved curve tracing algorithm on Figures 2.9 and 2.12 respectively. Figure 2.12 is a medical image obtained from [1].

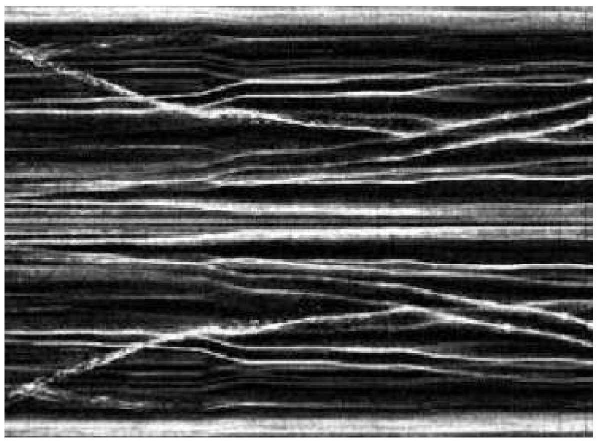


Figure 2.9: Original

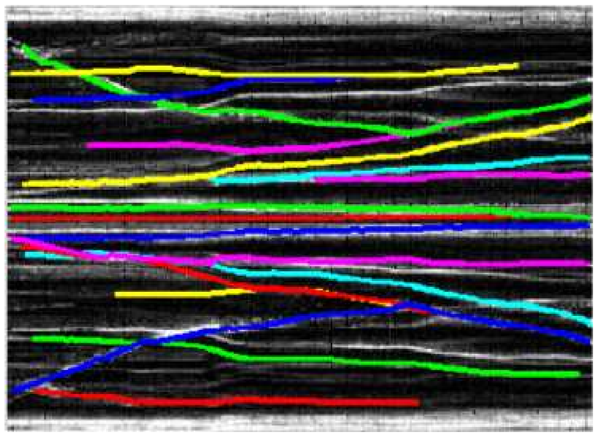


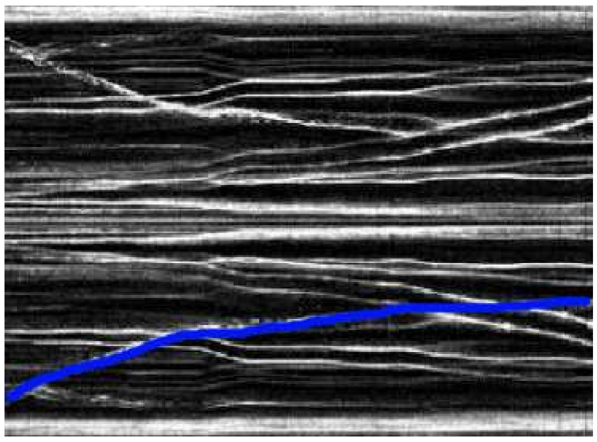
Figure 2.10: Result of curve tracing on Figure 2.9

Figure 2.9 is similar to Figure 2.1(a) in the sense that it has the problems of intersecting curves, curves that fade out as well as curves that disappear and re-emerge. However, the nature of curves in Figure 2.9 makes it a more difficult image for curve tracing. The two main reasons that account for the fact that Figure 2.9 is a difficult problem for curve tracing are:

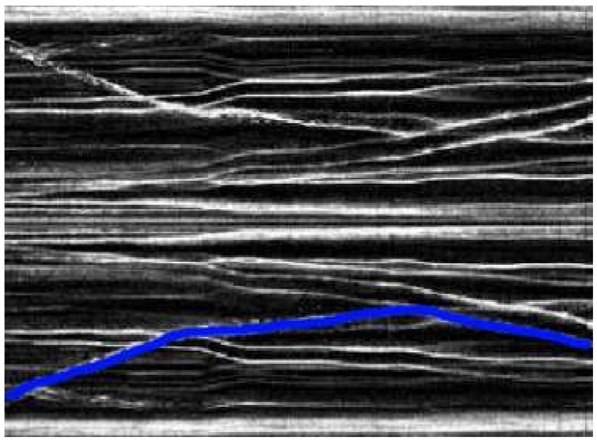
- Several distinct curves: There are several distinct curves that are near each other as well as intersect each other. This affects the calculation of curve orientation as well as classification of curve points. As a result, there is a higher chance that the algorithm can make a mistake.

- Desired result not clearly defined: In curve tracing, we want to identify an individual curve, label it, and follow it across the image, even if it crosses other curves, fades out, and then reappears. The result that we expect from a curve tracing algorithm is one that matches the identification done by a person. However, in Figure 2.9, the desired result of curve tracing is ambiguous. That is, different people would trace out curves in this image differently. For example, consider the blue curve in the bottom half of Figure 2.10. Even though this could match the tracing performed by a human observer, a different human observer could trace the same curve differently. To further clarify this issue, consider Figures 2.11(a) and 2.11(b). Both the figures represent possible curve tracing performed by humans. These figures bring out the fact that the desired result is not well defined for this image.

Figures 2.10 and 2.13 were obtained using the same set of parameter values as those used to obtain Figure 2.8. That is, the parameters in the improved curve tracing algorithm are image independent to a certain extent. A more detailed discussion of selection of parameter values is in Section 2.5.



(a) Curve tracing by Human 1



(b) Curve tracing by Human 2

Figure 2.11: Desired result is not well defined

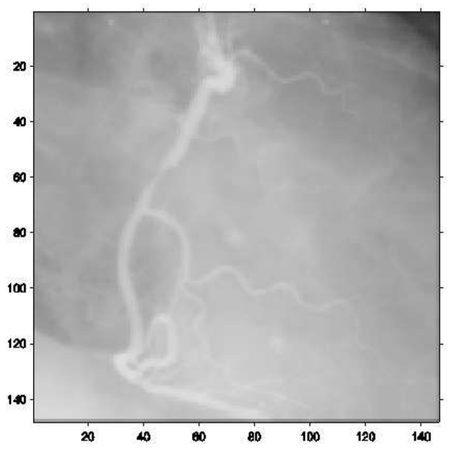


Figure 2.12: A medical image [1]

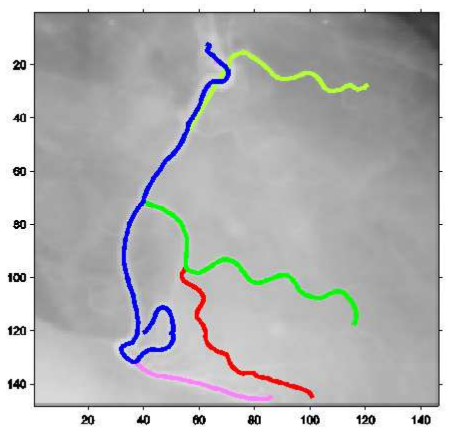


Figure 2.13: Result of curve tracing on Figure 2.12

2.5 Selection of parameter values

This improved curve tracing algorithm requires appropriate selection of several parameters such as β_0 (see Equation 2.11), L (see Equation 2.12) and ϕ (see Equation 2.16). We tested our improved curve tracing algorithm on several images with a fixed set of parameter values and observed that the algorithm performed well on all the images indicating that the parameters are image independent to a certain extent. The following are the values of various parameters used to produce the results shown in Figures 2.8, 2.10 and 2.13:

$\beta_0 = 70^\circ$	(see Equation 2.11)
$L = 5$	(see Equation 2.12)
$R = 5\sqrt{2}$	(see Equation 2.13)
$\phi = 7^\circ$	(see Equation 2.16)
$\ell = 10$	(see Equation 2.16)
$K = 35$	(see Equation 2.17)
$M = 15$	(see Section 2.3.2)
$w = 1/2$	(see Equation 2.19)

As mentioned earlier, the same set of parameter values were used to produce the results shown in Figure 2.8 (Original Image: Figure 2.1(a), Dimensions: $359x228$), Figure 2.10 (Original Image: Figure 2.9, Dimensions: $293x214$) and Figure 2.13 (Original Image: Figure 2.12, Dimensions: $146x148$).

In general, the selection of parameter values should be based on the following factors:

- Image dimensions: Clearly, image dimensions would affect parameters such as L (see Equation 2.12), R (see Equation 2.13) and ℓ (see Equation 2.16). These parameters are dependent on the size of the image as they define properties such as the extent to which we look ahead and the radius of the disc over which we search for a suitable next point.

- Nature of curves: The above mentioned parameters also depend on the nature of curves such as the degree of bending (curvature) and the distance between curves. If the original image has curves with high curvature, then β (see Equation 2.11) would have to be high as well.

- Strength of curves: Parameters such as K (see Equation 2.17) and ℓ (see Equation 2.16) depend on the strength of curves when they fade out or disappear and re-emerge. If the original image has fading curves that reduce rapidly in strength, these parameters would have to be set to a higher value.

- Noise: This algorithm is not robust to noise as noise affects the accuracy of the curve point classification step as well as the calculation of curve orientation at curve points. Section 2.7 describes the performance of this algorithm on noisy images in detail.

The parameter values mentioned above were chosen manually based on experimentation. There is scope for future work in developing an algorithm for automatic selection of parameter values.

2.6 Choice of starting point

In Steger's algorithm [1], the starting point is the pixel with the maximum second derivative (i.e. maximum strength). That is

$$(x_{start}, y_{start}) = \arg \max_{(x,y)\in I} \mu(x, y) \tag{2.20}$$

where $\mu(x, y)$ is obtained using Equation 2.6. Curves are constructed by starting with (x_{start}, y_{start}) and adding the appropriate neighbor to the current curve.

The choice of the starting point is a critical step as it affects the curve that is traced by the algorithm. The results shown in this chapter were obtained using a manual starting point. That is, the user chooses the starting point by clicking on a point in the image. This method works better than the above mentioned automatic procedure for choosing a starting point. There is scope for future work in developing an algorithm for automatic selection of good starting points.

2.7 Performance with noisy images

As explained in Section 2.2, this algorithm involves curve point classification as well as calculation of orientation at curve points. Steger's method for curve point classification and curve orientation calculation involves convolving the image with a Gaussian smoothing kernel (see Section 2.2.1). This smoothing is necessary since the problem of estimating derivatives of a noisy image is ill-posed and smoothing with a Gaussian kernel makes the problem well posed. However, smoothing with a Gaussian kernel is not sufficient for accurate classification of curve points as well as calculation of curve orientation at curve points when dealing with noisy images. As a result, the performance of this algorithm is affected by the presence of noise.

Figure 2.14 shows Figure 2.1(a) with additive white Gaussian noise (standard deviation $\sigma = 15.8$). Figure 2.15 shows the result of applying the improved curve tracing algorithm on Figure 2.14. Figure 2.16 shows Figure 2.1(a) with additive white Gaussian noise (standard deviation $\sigma = 20$). Figure 2.17 shows the result of applying the improved curve tracing algorithm on Figure 2.16. As can be observed, the classification of curve points as well as curve orientation calculation is not robust to noise and this leads to poor curve tracing. There is scope for future work in developing robust algorithms for classification of curve points and calculation of curve orientation.

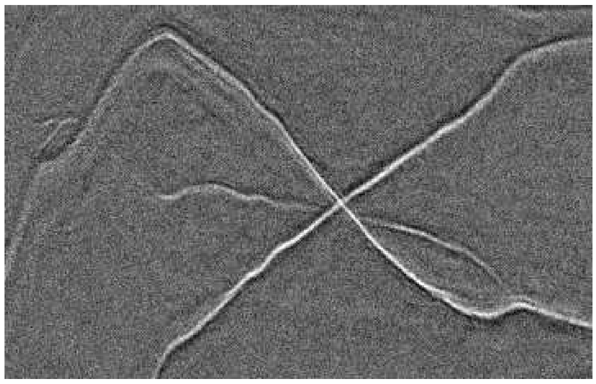


Figure 2.14: Figure 2.1(a) with AWGN ($\mu = 0, \sigma = 15.8$)

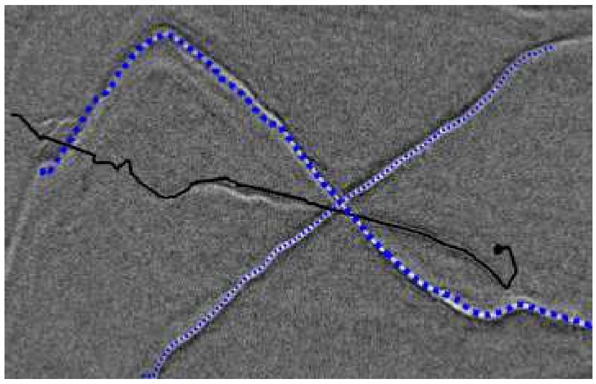


Figure 2.15: Result of curve tracing on Figure 2.14

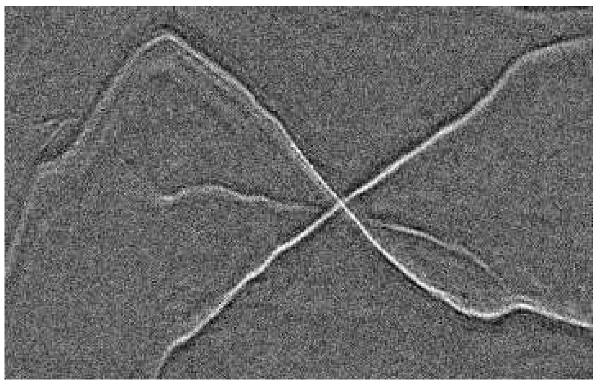


Figure 2.16: Figure 2.1(a) with AWGN ($\mu = 0, \sigma = 20$)

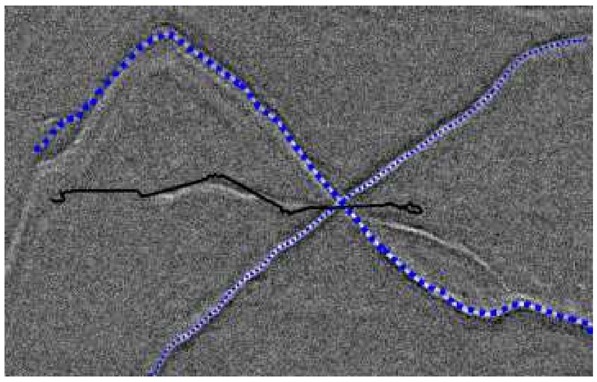


Figure 2.17: Result of curve tracing on Figure 2.14

2.8 Conclusion

An improved curve tracing algorithm that incorporates global features in a local algorithm has been proposed. The problem of incorrect curve following at junctions has been solved by disallowing large bending of curves. To trace a curve that disappears and re-emerges, we look ahead and predict the possible path of the curve when it disappears. The problem of tracing curves that fade out has been solved by using ideas from the Radon Transform. Comparing Figure 2.8 and Figure 2.3, we observe that the proposed algorithm results in improved curve tracing.

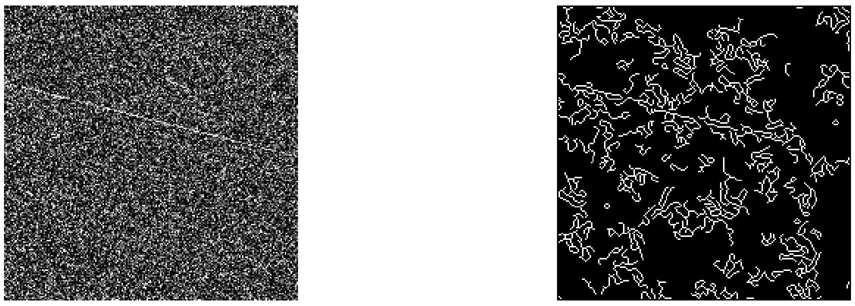
This algorithm requires appropriate selection of many parameters such as β_0 (see Equation 2.11), L (see Equation 2.12) and ϕ (see Equation 2.16). We tested our improved curve tracing algorithm on several images with a fixed set of parameter values and observed that the algorithm performed well on all the images indicating that the parameters are image independent to a certain extent. In general, the parameter values depend on the image dimensions, nature of curves (such as degree of bending and distance between curves) and the amount of noise present in the image. In addition, as mentioned in Section 2.7, this algorithm is not robust to noise. Curve point classification and calculation of curve orientation are vital to the curve tracing algorithm and there is scope for future work in developing robust algorithms for the same. Hence, developing a robust algorithm for automatic selection of parameters would result in a complete improved curve tracing algorithm.

Chapter 3

Curve detection

3.1 Introduction

Curve detection in images is an important operation in image processing and computer vision that has many applications. There is a large body of literature dealing with curve detection methods. For example, edge detection algorithms also fall in the category of curve detection algorithms but these methods do not work well in the presence of noise. Figure 3.1(a) shows a noisy image containing a single line. Figure 3.1(b) shows the result of applying the Canny edge detection algorithm [6] on Figure 3.1(a).



(a) Line in noise (b) Canny edge detection

Figure 3.1: Edge detection

As mentioned in Section 2.1 curve detection (or curve extraction), is different from curve tracing. In curve extraction [4, 5, 6], the emphasis is on the extraction and not on the identification or labeling of a curve. Many multi-resolution algorithms [4, 5] developed recently, such as curvelets and beamlets are very effective curve extraction algorithms and are used for compressing images with curves because they preserve the curves in a compressed image. Beamlets [4] are a collection of line segments at a range of locations, orientations, and scales, that give a multi-scale approximation to the collection of all line segments in an image. Polygonal curves in the $x - y$ plane are then built up by chaining beamlets together. Similarly, Curvelets [5] are a multiscale approach used to represent objects with edges and curves. These methods work better than edge detection algorithms when dealing with noisy images.

The Hough transform and it's more general version, the Radon Transform [10, 11] are well known approaches for detection of lines even in extremely noisy images. These methods transform images with lines into a domain of possible line parameters, where each line in the image will give a peak positioned at the corresponding line parameters. Thus the Radon Transform converts a line detection problem in the image domain to a simpler peak detection problem in the parameter domain. Figure 3.2 is an image representation of the discrete Radon Transform of the image in Figure 3.1(a) (the black dot in the image represents the peak in the parameter domain). As can be observed, the line parameters can be detected by thresholding the Radon Transform.

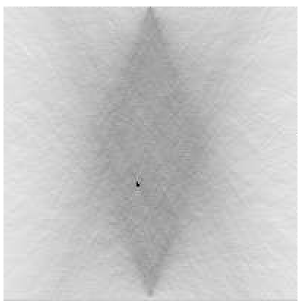


Figure 3.2: Radon Transform of Figure 3.1(a)

A natural generalization of the Radon Transform is the Generalized Radon Transform (GRT) [10]. The GRT maps an image into a parameter domain. Thus, a curve in an image following a specific parameterized curve form will correspond to a peak in the parameter domain. Hence, a curve detection problem is reduced to a peak detection problem in the parameter domain.

3.1.1 Motivation - A real world application

This chapter is concerned with issues that arise when applying the Generalized Radon Transform to the problem of curve detection/extraction (note that the terms curve extraction and curve detection have been used interchangeably in this thesis). These issues arose when I used the Generalized Radon Transform for curve detection on the images presented in Chapter 2 (see Figures 2.1(a), 2.9, 2.12). More specifically, the issues that arose are of three kinds:

- Multiple similar curves: Simple thresholding of the GRT results in the extraction of multiple similar curves that are very close to each other. That is, a single curve in the original image is detected multiple times.
- Spurious curves: The GRT method for curve detection is global in nature and does not use curve orientation information. As a result, some of the detected curves are a result of connecting up segments that belong to different curves in the original image.
- Model insufficiency: Cases may arise where the parameterized model assumed for the curves are insufficient to extract the curves in an image. For example, a second order curve will not be extracted when we search only for lines in an image.

These issues (and how they have been tackled) are described in detail in Sections 3.4 and 3.5.

Fracture detection in borehole images

The issues described in 3.1.1 are practical issues that occur when using the Radon Transform for fracture detection in borehole images [17]. The following description of borehole images is adapted from [17]. Borehole images are acquired by lowering a tool and measuring a physical quantity during the ascent back up. Images are obtained via many small buttons (0.1 inch), placed on four pads, that are pressed to the borehole wall. The amount of current that flows into the formation via each button is a function of the formation resistivity. These tools are fitted with either 64 or 192 buttons. The pads do not cover the entire wall. As a result, the recorded image contains large vertical stripes for which no data is available. This kind of imaging (Micro-resistivity imaging) yields a high resolution image of the subsurface. If the borehole is intersected with a planar event, the unwrapping of the cylindrical image to a plane will show a sinusoid. This can be easily shown. Let the cylinder be represented (in cylindrical coordinates r, ϕ and z) by the equation $r = R$, where R is the radius of the cylinder. Thus any point on the cylinder can be represented (in cartesian coordinates) by $(R\cos\phi, R\sin\phi, z)$ where ϕ is the azimuth (angle around the cylinder) and z is the distance of the point along the cylinder axis. Now, to obtain the intersection of a plane with the cylinder, we need to solve the equation of a plane with the cylinder. A plane can be defined using the equation

$$\bar{n}.\bar{p} \;=\; 0 \tag{3.1}$$

where $\bar{n} = (n_x, n_y, n_z)$ is the unit vector in the direction perpendicular to the plane and $\bar{p}$ is any point in space. Thus, by substituting $\bar{p} = (R\cos\phi, R\sin\phi, z)$ in Equation 3.1 and solving for z, we get

$$z = \frac{-Rn_x \cos\phi - Rn_y \sin\phi}{n_z} \tag{3.2}$$

Now, let the normal (unit vector in the direction perpendicular to the plane) $\bar{n} = (n_x, n_y, n_z)$ be represented in spherical coordinates as

$$\bar{n} = (\sin\phi_n \cos\theta_n, \sin\phi_n \sin\theta_n, \cos\phi_n)$$

where ϕ_n is the angle made by the normal $\bar{n}$ with the z-axis and θ_n is the angle between the positive x-axis and the projection of the normal onto the XY-plane. Thus, by substituting $(n_x, n_y, n_z) = (\sin\phi_n \cos\theta_n, \sin\phi_n \sin\theta_n, \cos\phi_n)$ in Equation 3.2, we obtain

$$\begin{aligned} z &= \frac{-R\sin\phi_n \cos\theta_n \cos\phi - R\sin\phi_n \sin\theta_n \sin\phi}{\cos\phi_n} && (3.3) \\ &= -R\tan\phi_n \cos\theta_n \cos\phi - R\tan\phi_n \sin\theta_n \sin\phi && (3.4) \\ &= -R\tan\phi_n \cos(\phi - \theta_n) && (3.5) \end{aligned}$$

Thus, z has a sinusoidal curve dependence on the azimuth ϕ for an intersecting bedding plane.

Fractures in the formation, therefore, can be detected as sinusoids in the image. A complication is that a fracture is often not an isolated event. Typically, fractures are hidden in a prominent background of bedding planes. The bedding generally manifests itself as stacked sinusoids with a common amplitude (resulting from the bedding dip) and phase (resulting from the bedding azimuth). The automatic estimation of bedding dip and azimuth can be performed fairly well as robust estimates of the dip and azimuth can be obtained by spatial averaging. The detection of fractures, however, is much more difficult as they are relatively rare, they have a small coverage in the image and their orientation deviates significantly from that of the background. In other words: fractures are the exceptions where the bedding is the rule. Also, multiple (intersecting) fractures may be present. Figure 3.3 shows a schematic representation of a borehole image containing bedding and a single fracture. Figure 3.4(a) shows a typical borehole image and Figure 3.4(b) shows the desired result (sinusoidal curves picked by a geologist).

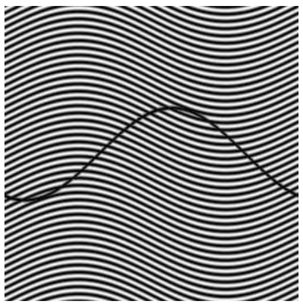
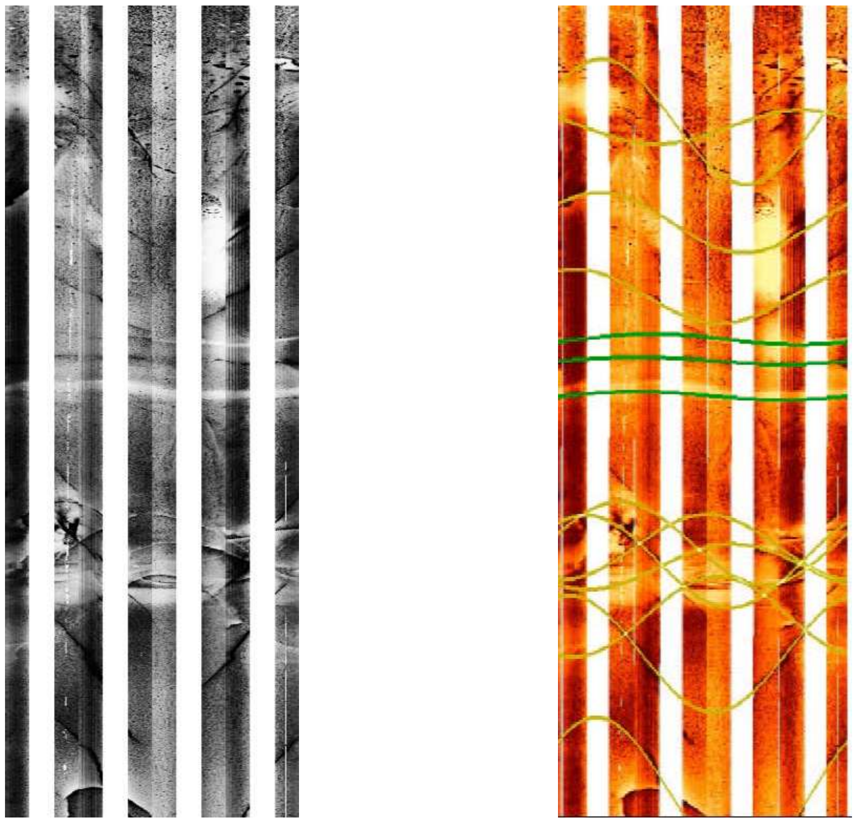


Figure 3.3: Schematic of a borehole image

The standard approach for fracture detection, i.e. computing an edge map and applying a Radon transform to it, has been widely applied. These existing methods commonly suffer from two problems. First, the edge maps used are obtained by crude segmentation approaches. Second, these methods do not take a number of important properties of fractures in the images into account. The results of the existing methods typically show just a few, isolated, detected fractures, indicating that these methods indeed fail to perform well.



(a) A typical borehole image (b) Curves picked by a geologist (desired result)

Figure 3.4: Actual borehole image

There are two problems with the detection of these sinusoids. First, the borehole images suffer from a number of distortions, most of which can be identified in Figure 3.4(a):

- Imperfect sinusoids due to:
  - Variations in the speed of the tool
  - The borehole is not perfectly cylindrical
  - The fracture is not perfectly planar
- Partial sinusoids due to:
  - Missing data due to the partial coverage
  - True partial fractures (either through cementation or drilling induced fractures)
- Structures other than fractures introduced by some of the effects above, especially the partial coverage and the speed variations

Second, the presence of multiple, intersecting, curves is a problem for the standard Radon technique. Even though the Radon transform itself is capable of dealing with intersecting curves, we must first obtain an edge map. This edge map must be able to represent intersecting curves, because if we loose information at this stage we cannot retrieve at a later stage. In [17], this problem is tackled by adopting an orientation space approach. However, the parameter space computed by the Radon transform contains a large number of local maxima which do not correspond to actual sinusoidal events in the image (despite the orientation constraints). [17] distinguishes between three types of bad picks (detected curves):

- Picks due to background noise: Integration along a curve through a noisy region results in a significant value in parameter space.
- Picks due to connecting up segments that belong to different fractures.
- Picks due to detecting a single fracture multiple times. Given a correctly picked fracture with amplitude A, curves with a slight larger or small amplitude will still give a good fit on the flanks of the sinusoid.

To reduce the number of incorrect picks, [17] uses a post processing pruning algorithm. The above mentioned issues are identical to the issues of multiple similar curves and spurious curves tackled in this thesis.

Outline of the chapter

The Radon Transform and the Generalized Radon Transform are briefly discussed in Section 3.2. In Section 3.3, I briefly describe some of the issues that have been tackled in previous literature. In Section 3.4, I mention the issues that are the focus of this chapter and some approaches (along with results) to tackle these issues are described in Section 3.5. The chapter concludes with Section 3.6.

3.2 The Radon Transform

The Radon Transform $\breve{g}(p, \tau)$ of a continuous two-dimensional function $g(x, y)$ is an integral transform found by stacking or integrating values of $g(x, y)$ along slanted lines. The location of the line is determined from the line parameters; slope p and line offset τ.

$$\breve{g}(p, \tau) \quad = \quad \int_{-\infty}^{\infty} g(x, px + \tau) dx \tag{3.6}$$

Thus the Radon Transform maps a two-dimensional function $g(x, y)$ to another function $\breve{g}(p, \tau)$ in the (p, τ) space (parameter domain).

A discrete version of the Radon Transform is obtained by sampling the four variables (namely x,y,p and τ in Equation 3.6). Let

$$x_m = \quad x_{min} + m\Delta x, \quad m = 0, 1, \ldots, M - 1 \tag{3.7}$$

$$y_n = \quad y_{min} + n\Delta y, \quad n = 0, 1, \ldots, N - 1 \tag{3.8}$$

$$p_k = \quad p_{min} + k\Delta p, \quad k = 0, 1, \ldots, K - 1 \tag{3.9}$$

$$\tau_h = \quad \tau_{min} + h\Delta \tau, \quad h = 0, 1, \ldots, H - 1 \tag{3.10}$$

The discrete Radon Transform can hence be written as

$$\breve{g}(k, h) \quad = \breve{g}(p_k, \tau_h) = \quad \Delta x \sum_{m=0}^{M-1} g(x_m, p_k x_m + \tau_h) \tag{3.11}$$

When dealing with images, we have a two-dimensional function $g(m, n)$ defined on the integer grid. Equation 3.11 cannot be directly used on an image as the value $p_k x_m + \tau_h$ is not an integer in general and hence we would need an interpolation technique. A simple modification such as rounding can correct this problem. Other techniques such as linear interpolation and sinc interpolation have been studied [10] and it has been observed that, if sampling of the parameter domain is sufficiently dense, then a nearest neighbor approximation (i.e. rounding) is adequate. That is, the Radon Transform, $\breve{g}(k,h)$, of an image $g(m,n)$ is given by

$$\breve{g}(k,h) \;=\; \Delta x \sum_{m=0}^{M-1} g(m, n(m;k,h)) \quad , n(m;k,h) = \left[\frac{p_k x_m + \tau_h - y_{min}}{\Delta y}\right] \tag{3.12}$$

where $[\cdot]$ means rounding the argument to the nearest integer.

3.2.1 The Generalized Radon Transform

A natural generalization of the Radon Transform is the Generalized Radon Transform (GRT). The GRT also maps an image into a parameter domain. A curve in an image following a specific parameterized curve form will correspond to a peak in the parameter domain. The Radon Transform is hence a first order Generalized Radon Transform.

The GRT of a two dimensional function $g(x,y)$ is the integral of the function along a transformation curve $y = \Phi(x;\epsilon)$ where ϵ denotes an η dimensional parameter vector. In the case of the linear Radon Transform $\epsilon = (p,\tau)$ and $y = \Phi(x;p,\tau) = px + \tau$. The discrete GRT $g(\mathbf{j})$ of an image $g(m,n)$ is given by

$$\breve{g}(\mathbf{j}) \;=\; \Delta x \sum_{m=0}^{M-1} g(m, \Phi(m;\mathbf{j})) \tag{3.13}$$

where

$$\Phi(m;\mathbf{j}) \;=\; \left[\frac{\Phi(x_{min} + m\Delta x; \mathbf{j}) - y_{min}}{\Delta y}\right] \tag{3.14}$$

where $\mathbf{j}$ is the η dimensional discrete sampled version of the parameter vector ϵ.

An image containing curves that are modelled by the function Φ would result in peaks in the parameter domain corresponding to the curve parameters. Hence it is easy to detect curves in an image by a simple thresholding method. Thus, if we wish to detect C curves in an image, the following algorithm can be used:

```
for i=1:C
    $\mathbf{j_i} = \arg\max_{\mathbf{j}} \breve{g}(\mathbf{j})$
    $y_i(x) = \Phi(x; \mathbf{j_i})$
    $\breve{g}(\mathbf{j_i}) = 0$
next
```

Hence the C extracted curves namely $y_1(x), \ldots, y_C(x)$ represent the C peaks in the parameter domain.

3.3 Curve detection issues

Peter Tofts PhD thesis [10] is an excellent reference for the Radon Transform and its application to curve detection. Many of the following issues are mentioned and tackled in his thesis.

3.3.1 Discretization

Computing the Radon Transform (essentially the integral of the image along a particular curve) of a digital image requires discretization of curve points on an integer grid. Methods such as nearest neighbor, linear and sinc interpolation have been examined for this purpose [10].

3.3.2 Choice of parameter domain

A curve can have different parameterized representations. For example, one can represent a line using the slope-intercept form or the normal form. In addition, once the type of parameterization is fixed, we still have to choose the range of parameters over which the Radon transform has to be computed. For example, for detection of steep lines (high slope) in an image, the normal form of a line is more useful as the range of slopes (in the slope intercept form) has to be restricted. This issue has also been dealt with in detail in [10].

3.3.3 Noisy images

Curves in the image correspond to peaks in the parameter domain. Thus, a simple thresholding algorithm is enough for detection of curve parameters. Estimating the threshold level for noisy images along with detector performance analysis has been tackled in [14].

3.3.4 Fuzzy Radon Transform

This is a concept introduced in [10] for detection of lines with wiggles. Assuming that the lines have wiggles, this prior knowledge is incorporated by allowing the discrete Radon Transform to search for the maximum image value within a fixed window.

3.3.5 Implementation

Finding the GRT of an image is computationally intensive and there have been many efforts to design efficient algorithms to compute the GRT [15].

3.3.6 Localized Radon Transform

This is used to detect line segments in an image. The length of a line is used as an additional parameter to model a line. Thus, inner products are computed over all possible line segments (rather than lines running entirely through the image) [16].

3.3.7 Inverse Radon Transform

This is an important tool in medical imaging, astronomy and microscopy and is used for reconstructing images of the interior of a function from a set of external measurements. This is also dealt with in detail in [10].

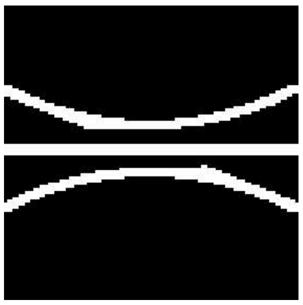
3.4 Issues tackled in this thesis

In this thesis work, I tackle three main issues that arise when the Radon Transform is used for curve detection namely (i) multiple similar curves (ii) spurious curves and (iii) model insufficiency. These problems are described in the following sections.

3.4.1 Multiple similar curves (Detecting thick curves)

The Radon Transform does not take into account curve width and thus simple thresholding methods would result in multiple similar curves that are very close to each other. This is because the Radon Transform computes inner products of single pixel wide curves with the image. A thick curve consists of many single pixel wide curves and each of these single pixel wide curves would result in a peak in the parameter domain. As a result, a thick curve would result in multiple similar curves being extracted. Consider the extraction of second order curves in Figure

3.5(a). As can be observed in Figure 3.5(b), there are multiple curves (three of them) that result in peaks in the parameter domain.


(a) Original image

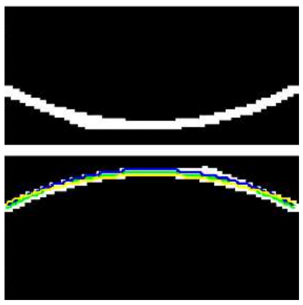
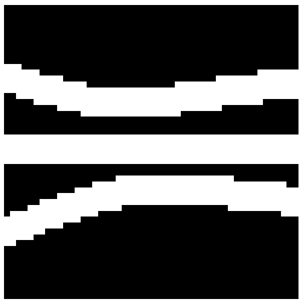

(b) Second order curves extracted

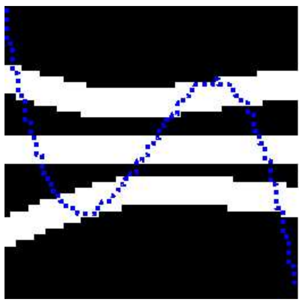
Figure 3.5: Multiple similar curves

3.4.2 Spurious curves

The Radon Transform does not use any local information. That is, the method is global in nature and does not use curve orientation information. As a result, the following problem could arise: Consider the extraction of third order curves in Figure 3.6(a). As can be observed in Figure 3.6(b), since the inner product of the dotted curve with the image is high (as it goes through many white pixels), simple thresholding would result in the dotted curve being extracted. Clearly this is not a curve present in the original image and such a problem should be tackled.



(a) Original image

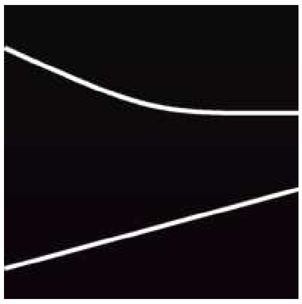


(b) A third order curve extracted

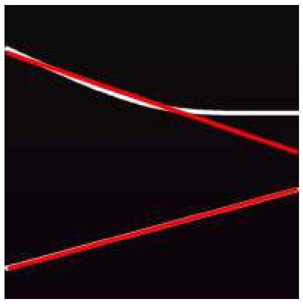
Figure 3.6: Spurious curve

3.4.3 Insufficiency of model

Cases may arise where the parameterized model assumed for the curves are insufficient to extract the curves in an image. For example, a second order curve will not be extracted when we search only for lines in an image. This problem is illustrated by Figure 3.7(a) (original image) and Figure 3.7(b) (lines extracted from the original image).



(a) Original image



(b) Lines extracted

Figure 3.7: Insufficiency of model

3.5 Improvements

The issues mentioned in the previous sections are important problems that arise when the Radon Transform is applied for curve detection. [17] discusses the problem of multiple similar curves as well as the problem of spurious curves. In [17], curves in the two-dimensional image space are transformed into other curves in a three-dimensional orientation space. In this space each point not only encodes a spatial position, but also the local orientation: the normal of the curve. These new curves have the same parameters as the original curves, since they are obtained by a deterministic mapping. A Radon transform is then used to compute a parameter space. The local maxima in parameter space correspond to the curves in orientation space and correspond to the curves in the original image. However, the parameter space contains a large number of local maxima which do not correspond to actual curves in the image, despite the orientation constraints. To reduce the number of incorrect curves, [17] uses a post processing pruning algorithm to reject curves.

In this thesis, separate methods are developed to tackle each of the problems mentioned in Section 3.4 and these methods are described in the following sections.

3.5.1 Multiple similar curves

As mentioned in Section 3.4.1, the Radon Transform does not take into account curve width. The Radon Transform is calculated as the running sum of the image along a particular curve. However, these curves are unit-pixel wide. Thus, an image that consists of a thick curve will result in multiple similar curves being extracted.

To correct this problem, the Radon Transform is modified to compute inner products of the image with thick curves rather than single pixel wide curves. To define this operation formally, we first define a two-dimensional function (or equivalently, an image),$h_{\mathbf{j}}$, which is essentially an image of the curve represented by the parameter vector $\mathbf{j}$. That is

$$h_{\mathbf{j}}(m,n) = \begin{cases} 1, n = \Phi(m;\mathbf{j}) \\ 0, \quad \text{otherwise} \end{cases} \tag{3.15}$$

Note that $m \in \{0, 1, \ldots, M-1\}$ and $n \in \{0, 1, \ldots, N-1\}$ where the image g has $M \times N$ pixels. We can re-write the Radon Transform as

$$\breve{g}(\mathbf{j}) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(m,n) h_{\mathbf{j}}(m,n) \tag{3.16}$$

Now, as mentioned earlier, we want to compute the inner product of the image with thick curves. Similar to $h_{\mathbf{j}}$, the image of a single-pixel wide curve, define $\tilde{h_{\mathbf{j}}}$, the image of a thick curve as

$$\tilde{h_{\mathbf{j}}} = h_{\mathbf{j}} \star G_{\sigma,H,V} \tag{3.17}$$

where $\star$ represents convolution and $G_{\sigma,H,V}$ is a Gaussian filter of size $H \times V$ pixels with standard deviation σ. The parameters, H and V, determine the thickness of the curve.

We now compute the Radon Transform using the following equation

$$\breve{g}(\mathbf{j}) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(m,n) \tilde{h_{\mathbf{j}}}(m,n) \tag{3.18}$$

By calculating inner products using thick curves, we reduce the number of multiple maxima in the parameter space as thick curves that are near the actual thick curve do not result in complete overlap.

In addition to the above, we can also eliminate multiple similar curves with some modifications to the peak detection process. This is done by computing spatial distances between curves. We modify the detection algorithm (see Section 3.2.1) to reject curves that are too 'close' to curves that have already been extracted. Note that, computing Euclidean distance in the parameter domain is not useful as distances in the parameter domain do not correspond to distances in the spatial domain (i.e. in the image). Define the distance $d(y_1, y_2)$ between two curves $y_1(x)$ and $y_2(x)$ as

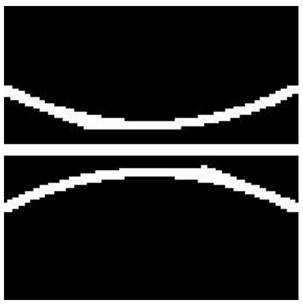
$$d(y_1, y_2) \quad = \quad \sum_{x=0}^{M-1} (y_1(x) - y_2(x))^2 \tag{3.19}$$

Hence, if y_1 is an extracted curve, then the curve y_2 is eliminated if

$$d(y_1, y_2) \quad < \quad D \tag{3.20}$$

where D is a distance threshold.

The above modifications eliminate the possibility of multiple similar curves. The result of applying the above procedure on Figure 3.8(a) is shown in Figure 3.8(b). Comparing Figures 3.8(b) and 3.5(b), we observe that the problem of multiple similar curves has been solved for this example.


(a) Original image

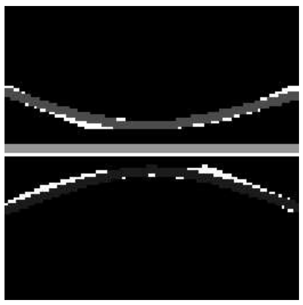

(b) Result of improvements

Figure 3.8: Eliminating multiple similar curves

3.5.2 Spurious curves

To tackle the problem of spurious curves, we need to incorporate orientation information. To do so, the method described in [1] is used to obtain orientation information. According to [1] we can classify pixels as curve points or not depending on if they lie on a curve or not. In addition, for curve points one can also calculate the orientation of the curve at that point.

Obtaining local information

Curvilinear structures in a 2D image can be modelled as curves $s(t)$ that exhibit a characteristic 1D line profile in the direction perpendicular to the curve(refer Figure 3.9). The 1D line profile is characterized by a vanishing first derivative (i.e. a local maximum is achieved). Let the direction perpendicular to the curve be $n(t)$. Thus, at a curve point (i.e. a point lying on a curve), the first directional derivative in the direction $n(t)$ should vanish and the second directional derivative should be of large absolute value. When dealing with images, pixels are considered to be on the integer grid $Z_N \times Z_N$ (where Z_N is the ring of integers). A pixel (x, y) has a boundary defined by the unit square $[x - \frac{1}{2}, x + \frac{1}{2}] \times [y - \frac{1}{2}, y + \frac{1}{2}]$. Hence, a pixel in an image is classified as a curve point if the first derivative along $n(t)$ vanishes within a unit square centered around the pixel (i.e. within the pixel's boundaries). The problem of computing the direction perpendicular to a curve is

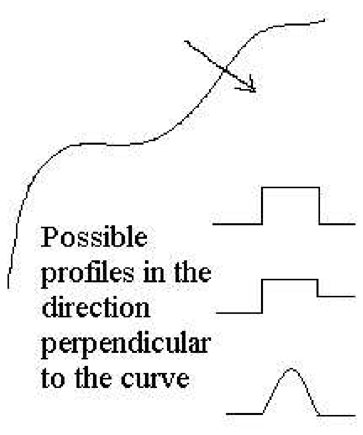

Possible profiles in the direction perpendicular to the curve

Figure 3.9: Classification of curve points

tackled by calculating the eigenvalues and eigenvectors of the Hessian matrix

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix} \tag{3.21}$$

The partial derivatives I_{xx}, I_{xy}, I_{yx} and I_{yy} are computed numerically using partial differences after convolving the image with a Gaussian smoothing kernel. Gaussian smoothing is essential since the problem of estimating derivatives of a noisy image is ill-posed and it is the only kernel that, under certain, very general, assumptions makes the problem well posed. This also leads to a scale-space description of the image [7].

The direction in which the second directional derivative takes on its maximum absolute value is used as the direction perpendicular to the curve $n(t)$. It is a standard result in Differential Geometry [9] that this direction can be computed by finding the eigenvector corresponding to the maximum absolute eigenvalue of the Hessian matrix. Let this eigenvector be given by (n_x, n_y). A quadratic polynomial is then used to determine whether the first directional derivative along (n_x, n_y) vanishes within the current pixel. This point can be obtained by inserting (tn_x, tn_y) into the Taylor polynomial

$$z(x,y) \quad = \quad I(x,y) + \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} I_x \\ I_y \end{pmatrix} + \frac{1}{2} \begin{pmatrix} x & y \end{pmatrix} H \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.22}$$

where $z(x,y)$ represents the curve.

By setting the derivative of $z(x,y)$ along (tn_x, tn_y) to zero and solving for t, we obtain

$$t \quad = \quad -\frac{n_x I_x + n_y I_y}{n_x^2 I_{xx} + 2n_x n_y I_{xy} + n_y^2 I_{yy}} \tag{3.23}$$

and hence the maximum or minimum occurs at

$$(p_x, p_y) \quad = \quad (tn_x, tn_y) \tag{3.24}$$

Thus the point (x,y) is declared a curve point if (p_x, p_y) falls within the pixel boundaries. That is, if $(p_x, p_y) \in [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$, then the point (x,y) is declared a curve point. The orientation, $\theta(x,y)$, of the curve at the point (x,y) can be obtained using the normal (n_x, n_y)

$$\theta(x,y) \quad = \quad \arctan\left(\frac{-n_x}{n_y}\right) \tag{3.25}$$

Note that both $\theta(x,y)$ and $\pi + \theta(x,y)$ are candidates for orientation of the curve at (x,y).

Curve point and orientation information can be used to eliminate spurious curves. To do so, we define some useful quantities. Let $\mathcal{C}$ be the set of all curve points in the image I. That is

$$\mathcal{C} \quad = \quad \{(x,y) \in I | (x,y) \text{ is a curve point}\} \tag{3.26}$$

We also define

$$1(x,y) \quad = \quad \begin{cases} 1, & \text{if } (x,y) \in \mathcal{C} \\ 0, & \text{otherwise} \end{cases} \tag{3.27}$$

$$\tag{3.28}$$

For every curve (any curve is specified by its corresponding parameter vector $\mathbf{j}$), we define

$$\mathcal{L}(\mathbf{j}) \quad = \quad \sum_{x=0}^{M-1} 1(x, \Phi(x;j)) \tag{3.29}$$

Thus, $\mathcal{L}(\mathbf{j})$ is the number of curve points on the curve $y = \Phi(x;\mathbf{j})$. A spurious curve would not pass through many curve points and thus a small value of $\mathcal{L}(\mathbf{j})$ would indicate that $y = \Phi(x;\mathbf{j})$ is a spurious curve.

We also define the quantity $\mathcal{A}(\mathbf{j})$ as follows:

$$\mathcal{A}(\mathbf{j}) \quad = \quad \frac{1}{\mathcal{L}(\mathbf{j})} \sum_{x=0}^{M-1} \left| \theta(x, \Phi(x;\mathbf{j})) - \frac{\partial(\Phi(x;\mathbf{j}))}{\partial x} \right| 1(x, \Phi(x;\mathbf{j})) \tag{3.30}$$

$\mathcal{A}(\mathbf{j})$ is a measure of the deviation of the actual curve orientation (refer Equation 3.25) from the curve orientation obtained analytically from the curve $y = \Phi(x;\mathbf{j})$. Thus, a large value of $\mathcal{A}(\mathbf{j})$ would mean that the curve $y = \Phi(x;\mathbf{j})$ may be a spurious curve. As a result, if we do obtain a high Radon Transform value for a particular curve $y = \Phi(x;\mathbf{j})$, one can check if it is a spurious curve by limiting the value of $\mathcal{A}(\mathbf{j})$.

Using local information

As mentioned in Section 3.5.2, the quantities $\mathcal{L}(\mathbf{j})$ and $\mathcal{A}(\mathbf{j})$ can be used to eliminate spurious curves. One way of doing so is to threshold these quantities. Thus, if a curve $y = \Phi(x; \mathbf{j})$ satisfies either of the following equations, it is termed a spurious curves.

$$\mathcal{L}(\mathbf{j}) < L \tag{3.31}$$

$$\mathcal{A}(\mathbf{j}) > A \tag{3.32}$$

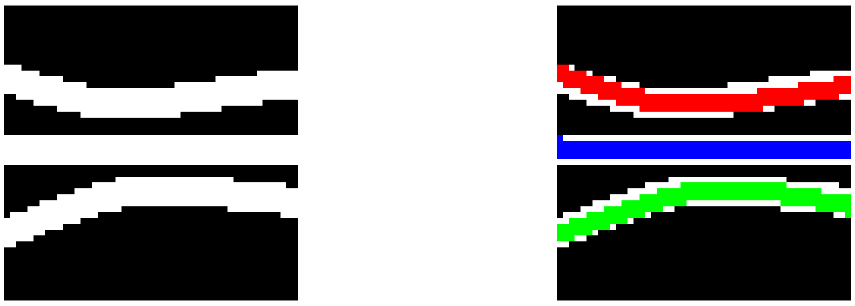
where L and A are thresholds for the number of line points and average angle deviation respectively.

An alternative method to eliminate spurious curves would be to search for maxima in the parameter space using a metric that combines the Radon Transform and the above mentioned quantities. For example, one could form the combined metric as follows:

$$M(\mathbf{j}) = \alpha \breve{g}(\mathbf{j}) + \beta \mathcal{L}(\mathbf{j}) \tag{3.33}$$

where α and β are scale factors that control the relative contribution of $\breve{g}(\mathbf{j})$ and $\mathcal{L}(\mathbf{j})$ in the metric. One can follow this by a peak detection algorithm as described in 3.2.1. In addition, multiple curves can be rejected using the procedure described in Section 3.4.1. Note that $\breve{g}(\mathbf{j})$ is obtained as in Equation 3.18 and is hence useful to eliminate multiple curves. In addition we can also eliminate multiple similar curves by using Equation 3.20.

The result of applying the above procedure on Figure 3.10(a) is shown in Figure 3.10(b). Comparing Figures 3.10(b) and 3.6(b), we observe that the problem of spurious curves has been solved for this example.



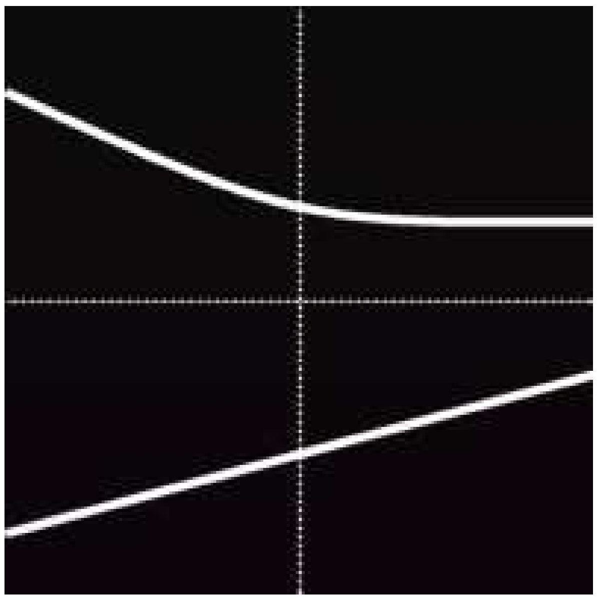
(a) Original image (b) Result of improvements

Figure 3.10: Eliminating spurious curves

3.5.3 Insufficiency of model

As mentioned in Section 3.4.3, cases may arise where the parameterized model assumed for the curves in an image are insufficient and do not represent all the curves in an image. In a sense, this is again a problem that arises out of the global nature of the Radon Transform and has to be solved by 'localizing' the Radon Transform.

The problem of model insufficiency can be solved by computing the Radon Transform over smaller blocks of the original image. As mentioned in Section 3.4.3, if we only compute the first order Radon Transform (i.e. lines) on Figure 3.7(a), the curves are not extracted correctly. We observe that, given a suitable block size (see Figure 3.11(a)), a piecewise linear fit is sufficient to extract the curves correctly (see Figure 3.11(b)).


(a) Original image

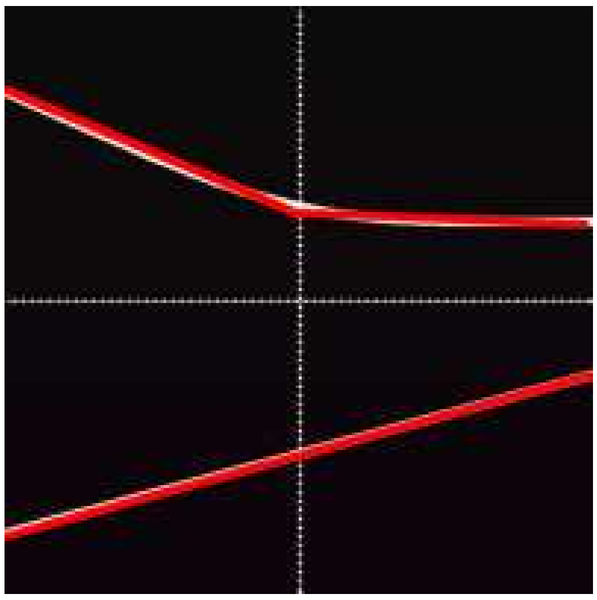

(b) Lines extracted

Figure 3.11: Piecewise linear fit

3.6 Conclusion

The improvements described in Section 3.5 address certain issues that arise when applying the Radon Transform for curve detection. The Radon Transform is global in nature and the improvements have been achieved by incorporating local information in the Radon Transform. These improvements also bring out issues that are yet to be addressed. The main issue concerns automatic selection of parameters. Some of the parameters include estimation of curve width (for using the thick Radon Transform as in Section 3.5.1), selection of window size and shape (see Section 3.5.3) and choice of thresholds such as L, A (refer Equation 3.31) etc. Developing an algorithm for automatic selection of parameters would result in a complete improved curve detection algorithm.

Chapter 4

Conclusion - future work

The central idea in this thesis is the use of both local and global information for detecting and tracing curves in images. The improvements to Steger's algorithm (a local algorithm for curve tracing) are achieved by using global features. For example, to trace a curve that disappears and re-emerges, we look ahead and predict the possible path of the curve when it disappears. In addition, the problem of tracing curves that fade out has been solved by using ideas from the Radon Transform (a global method). On a similar note, the improvements to the Generalized Radon Transform for curve detection are achieved by incorporating local features.

However, both the improved curve tracing algorithm (Chapter 2) and the improved curve detection algorithm (Chapter 3) share a common issue, viz. automatic selection of parameters. Both the algorithms require appropriate selection of many parameters such as β_0 (see Equation 2.11), L (see Equation 2.12), curve width (for using the thick Radon Transform as in Section 3.5.1) and window size and shape (see Section 3.5.3). There is scope for future work in developing algorithms for automatic selection of parameters used in these algorithms.

BIBLIOGRAPHY

[1] Carsten Steger, “An unbiased detector of curvilinear structures,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, February 1998.

[2] A. Baumgartner, C. Steger, C. Wiedemann, H. Mayer, W. Eckstein, and H. Ebner, “Update of roads in gis from aerial imagery: Verification and multi-resolution extraction,” *International Archives of Photogrammetry and Remote Sensing*, 1996.

[3] C. Coppini, M. Demi, R. Poli, and G. Valli, “An artificial vision system for x-ray images of human coronary arteries,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, February 1993.

[4] D. Donoho and Xiaoming Huo, “Applications of beamlets to detection and extraction of lines, curves and objects in very noisy images,” in *Proceedings of NSIP*, 2001.

[5] E. J. Candes and D. L. Donoho, *Curvelets - a suprisingly effective nonadaptive representation for objects with edges*, Vanderbilt University Press, 1999.

[6] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, November 1998.

[7] A. P. Witkin, “Scale-space filtering,” in *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, 1983.

[8] L.M.J. Florack, B.M. ter Haar Romeny, J.J. Koenderink, and M.A. Viergever, “Scale and the differential structure of images,” *Image and Vision Computing*, vol. 10, no. 6, pp. 376–388, 1992.

[9] Manfredo do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice Hall, 1976.

[10] Peter Toft, *The Radon Transform - Theory and Implementation*, Ph.D. thesis, Informatics and Mathematical Modelling, Technical University of Denmark, 1996.

[11] S. R. Deans, *The Radon Transform and some of its applications*, John Wiley and Sons, 1983.

[12] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1998.

[13] Karthik Raghupathy and Thomas W. Parks, “Curve tracing in images,” in *Proceedings of IEEE Western New York Image Processing Workshop*, October 2003.

[14] Peter Toft, “Using the generalized radon transform for detection of curves in noisy images,” in *Proceedings of IEEE ICASSP*, 1996, pp. 2221–2225.

[15] K. V. Hansen and P. A. Toft, "Fast curve estimation using pre-conditioned generalized radon transform," *IEEE Transactions on Image Processing*, pp. 1651–1661, December 1996.

[16] A. C. Copeland, G. Ravichandran, and M. M. Trivedi, "Localized radon transform-based detection of linear features in noisy images," in *Proceedings of Computer Vision and Pattern Recognition*, June 1994.

[17] M. van Ginkel, L.J. van Vliet, P.W. Verbeek, M.A. Kraaijveld, E.P. Reding, and H.J. Lammers, "Robust curve detection using a radon transform in orientation space applied to fracture detection in borehole images," in *Proceedings of the 7th Annual Conf. of the Advanced School for Computing and Imaging*, Heijen, Netherlands, 2001, pp. 84–91.