On the Use of Level Curves in Image Analysis

M. J. Donahue and S. I. Rokhlin

The Ohio State University
Nondestructive Evaluation Program
190 West 19th Avenue
Columbus, OH 43210

May 22, 1992

Abstract

A digitized image is viewed as a surface over the xy-plane. The level curves of this surface provide information about edge directions and feature locations. This paper presents algorithms for the extraction of tangent directions and curvatures of these level curves. The tangent direction is determined by a least-squares minimization over the surface normals (calculated for each 2×2 pixel neighborhood) in an averaging window. The curvature calculation, unlike most previous work on this topic, does not require a parameterized curve, but works instead directly on the tangents across adjacent level curves. The curvature is found by fitting concentric circles to the tangent directions via least-squares minimization. The stability of these algorithms with respect to noise is studied via controlled tests on computer generated data corrupted by simulated noise. Examples on real images are given which show application of these algorithms for directional enhancement, segmentation, and feature point detection.

1 Introduction

An image may be viewed as a surface $h(x, y)$ over the xy-plane, where peaks in the surface correspond to light areas of the image and valleys correspond to dark areas. The topological properties of this surface are useful in many applications [1, 2]. In particular, the level curves contain information which can be used in adaptive processing of the image for enhancement and pattern recognition. Tangents to the level curves run parallel to ridges in the image, so the tangent directions can be used for edge detection and enhancement. For example, obtaining this directional information is a necessary step in the processing of fingerprints for automated identification [3, 4] and for orientation specific filtering [5].

There are different approaches to extracting directional information. One common technique is to use an ideal edge as a template to detect the presence of an edge and to determine its orientation. Hueckel [6] and O'Gorman [7] used a modified form of this approach with least-squares minimization to match against an edge of arbitrary orientation. Kawagoe and Tojo [3] used a different method in their work with digitized fingerprints. On each 2×2 pixel neighborhood they made a straight comparison against 4 edge templates to extract a crude directional estimate, which was then arithmetically averaged over a larger region to obtain a more accurate estimate. Our approach uses a gradient-type operator to extract a directional estimate from each 2×2 pixel neighborhood, which is then averaged over a larger region by least-squares minimization to control noise.

Curvature of level curves is sensitive to broader topological properties of the image that can be used for feature detection and identification. For example, curvature is large near corners and peaks in the image, which are typical feature points used for pattern recognition [8]. Most previous work on curvature determination ([9], for instance) has centered on finding the curvature of a single parameterized curve that has been previously extracted from the image by a separate algorithm (a chain code algorithm, for example). Adjacent level curves, on the other hand, locally form a family of parallel curves. Our algorithm extracts curvature using data determined by these curves. This is accomplished without curve parameterization by fitting a family of concentric circles to the extracted tangent directions via least-squares minimization.

The work [10] of Parent and Zucker is along similar lines. (See also [11].) They also extract tangent directions and curvature directly from the image without curve parameterization. In their work the tangent directions are quantitized to represent finite ranges. At each point in the image and for each of the allowed directions a probability is assigned. This probability is determined from two components. The first component is the result of convolutions with local "line detector" operators. The second component is a curvature compatibility factor, which measures how closely the local tangent directions model a smooth flow. The tangent directions are adjusted using an iterative relaxation approach. This algorithm goes further by using the tangent and curvature information to essentially extract curve traces. It is actually an integrated curve extraction algorithm.

The algorithm of Parent and Zucker assumes the existence of curves in the image and it is optimized towards the extraction of these curves. The algorithm we present does not assume the existence of curves and does not aim to extract them. It extacts only the tangent directions and curvature for further processing by other algorithms. It will work on any image possessing smooth level curves. Moreover, our algorithm directly extracts the directional information using least-squares minimization—no iterative processing is performed. Thus our algorithm is considerably less complicated and should be much faster than the algorithm of Parent and Zucker. On the other hand, the algorithm of Parent and Zucker explicitly handles curve intersections, an event our algorithm cannot interpret since it does not deal with curve traces. However, our algorithm will find curve intersections to have large curvature, so such events can be managed during subsequent processing.

In addition to the presentation of algorithms for the extraction of tangents and curvatures from level curves in digitized images, we show results of controlled tests of our implementation of these algorithms on computer generated data corrupted by simulated noise. Example applications are also provided showing the use of the tangent and curvature information in the image processing of fingerprints, materials microstructures, printed circuit boards, and radiographs.

2 Problem statement

Let us first introduce some terminology. Fig. 1 shows three level curves with tangent and curvature vectors marked. Let P be a point in the xy-plane as indicated and consider the level curve passing through that point. Let $\vec{r}(s)$ be a parameterization by arc length of this level curve, with $\vec{r}(0) = P$. The derivative $\vec{r}\,'(0)$ is the **tangent vector** to the level curve at point P. We are interested only in the unoriented direction of this vector, since both its magnitude and sign (+ or -) depend upon the parameterization. Choosing arc length parameterization forces the magnitude of $\vec{r}\,'(t)$ to 1, which makes the second derivative, $\vec{r}\,''(0)$, perpendicular to the tangent vector with magnitude proportional to the rate of change in direction of the tangent vector. The vector $\vec{r}\,''(0)$ is called the **curvature vector** for the level curve at the point P. The direction of $\vec{r}\,''(0)$ is towards the center of the curvature of the level curve $\vec{r}(s)$, and the magnitude of $\vec{r}\,''(0)$, called the **curvature** of the level curve, is equal to the reciprocal of the curvature radius. This is evident in Fig. 1 where the magnitudes of the illustrated curvature vectors are largest on the inner level curve and smallest on the outer level curve.

We have so far assumed that the level set at the point P is one dimensional. However, if P corresponds to a local extrema or plateau, then the level set will be 0 or 2 dimensional, respectively. In this case it makes no sense to talk of a "level curve", and accordingly the tangent and curvature are not defined. The algorithms described in this paper detect such occurrences.

In a digitized image the image values are only known on a discrete set of points (the sample nodes), so the curve $\vec{r}(s)$ and its derivatives can only be approximated. Rather than attempting to approximate $\vec{r}\,'(s)$ and $\vec{r}\,''(s)$ at the point P, we find an averaged value over a rectangular window around P using least-squares minimization. The effects of random noise are minimized in the average, so this method is noise tolerant. The choice of window size is application specific, depending on the size of the features of interest (with respect to the pixel size) and the noise level. There is generally a tradeoff between noise tolerance (large windows) and precision (small windows).

3 Tangent vector calculation

3.1 Method outline

At each point in the averaging window we calculate a vector $\vec{n}$ that is normal to the surface $z = h(x, y)$. The tangent vector to the level curve at this point (if defined) lies in the xy-plane and is perpendicular to the normal. The averaged tangent vector for an averaging window is that (unit) vector in the xy-plane which comes closest to being perpendicular to all the surface normals in the window. We could average the tangents directly, but there are advantages to working with the surface normals. Consider, for example, the edge function defined by

$$g(x, y) = \begin{cases} 1 & \text{if } y \geq 0 \\ 0 & \text{if } y < 0 \end{cases}$$

which is a unit step with edge along the x-axis. Let $h(x, y)$ be a smooth approximation to $g(x, y)$. Fig. 2 is an illustration of this surface and several of the surface normals. Away from the x-axis the normals to $h(x, y)$ are parallel to the z-axis, and accordingly the tangents are undefined. As one moves near the x-axis, the normal vectors tilt away from the z-axis. If the approximation of $h(x, y)$ to the step function $g(x, y)$ is very tight, then along the x-axis the normal vectors to the surface $h(x, y)$ will be nearly perpendicular to the z-axis. In particular, the steeper the surface the smaller the angle between the surface normal and the xy-plane. Since noise corrupts shallow edges more easily than steep edges, we want to weight normals from steep edges more heavily in the average than normals from shallow edges. Thus there are two advantages in working with the normal vectors: the normal vectors are always defined, and they carry information which is used to weight the average.

Of course, in a digitized image the function $h(x, y)$ is not known at every point in the xy-plane. Rather the value is known (with limited precision) on a finite grid of points, called the **sample nodes**. To simplify our discussion we shall assume that these nodes are evenly spaced on a square mesh, although the algorithm is easily adapted to other geometries. We call a set of 4 nodes that are corners of a grid-minimal square a **2x2 neighborhood**.

Our approach to calculating the tangent vector in a digitized image consists of two steps. First we estimate the surface normal at the center of each 2x2 neighborhood with the normal to a plane fitted through the image values at the 4 sample nodes. We refer to this as **point normal determination**. These normals are grouped into (possibly overlapping) regions called **tangent windows**. The second step consists of finding, for each tangent window, a vector $\vec{u}$ lying in the xy-plane that is nearly perpendicular to all the normals in that window. This vector is called the **averaged tangent vector**.

3.2 Point normal determination

Let (x_1, y_1), (x_2, y_2), $\ldots$, (x_m, y_m) be a collection of m sample nodes in the xy-plane, and let a_1, a_2, $\ldots$, a_m be the image values at these nodes. We fit to these values a plane $p(x, y)$ of the form $p(x, y) = -n_1 x - n_2 y + c$, which has normal vector $\vec{n} = (n_1, n_2, 1)$. (The digitized values are restricted to a finite range, insuring that $\vec{n}$ will not be perpendicular to the z-axis.) We want to select values for n_1, n_2, and c to minimize the sum of the squared difference of the image values with the plane values over the m sample nodes. That is,

$$\begin{aligned}
&\min_{n_1,n_2,c} \sum_{\substack{\text{sample}\\ \text{nodes}}} |h(x,y) - p(x,y)|^2 \\
&= \min_{n_1,n_2,c} \left\| \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} - \begin{pmatrix} -x_1 & -y_1 & 1 \\ -x_2 & -y_2 & 1 \\ \vdots & \vdots & \vdots \\ -x_m & -y_m & 1 \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ c \end{pmatrix} \right\|^2, \qquad (1)
\end{aligned}$$

which is a standard least-squares minimization problem (see [12]). Let us set $\vec{w}^1 = (-x_1, -x_2, \ldots, -x_m)^T$, $\vec{w}^2 = (-y_1, -y_2, \ldots, -y_m)^T$, and $\vec{w}^3 = (1, 1, \ldots, 1)^T$. Then this minimization problem can be viewed as finding that (m-dimensional) vector in the span of $\{\vec{w}^1, \vec{w}^2, \vec{w}^3\}$ which is closest to the vector $\vec{a} = (a_1, a_2, \ldots, a_m)^T$. This is given simply by the orthogonal projection of $\vec{a}$ onto the subspace spanned by $\{\vec{w}^1, \vec{w}^2, \vec{w}^3\}$.

For example, consider the 2x2 neighborhood depicted in Fig. 3. Here $m = 4$, and $\vec{w}^1 = (-1, 1, 1, -1)^T$, $\vec{w}^2 = (-1, -1, 1, 1)^T$, and $\vec{w}^3 = (1, 1, 1, 1)^T$. Since the $\vec{w}^i$'s are orthogonal and $\|\vec{w}^i\|^2 = 4$ for each $i = 1, 2, 3$, it follows that the minimizing vector is

$$\frac{\langle \vec{a}, \vec{w}^1 \rangle \vec{w}^1}{4} + \frac{\langle \vec{a}, \vec{w}^2 \rangle \vec{w}^2}{4} + \frac{\langle \vec{a}, \vec{w}^3 \rangle \vec{w}^3}{4} =$$

$$\frac{-a_1 + a_2 + a_3 - a_4}{4} \vec{w}^1 + \frac{-a_1 - a_2 + a_3 + a_4}{4} \vec{w}^2 + \frac{a_1 + a_2 + a_3 + a_4}{4} \vec{w}^3$$

where $\langle \cdot\,, \cdot \rangle$ denotes the usual scalar product.

Comparing to Eq. 1, we see that the minimizing values are

$$\begin{aligned} n_1 &= \frac{-a_1 + a_2 + a_3 - a_4}{4} \\ n_2 &= \frac{-a_1 - a_2 + a_3 + a_4}{4} \\ c &= \frac{a_1 + a_2 + a_3 + a_4}{4} \end{aligned} \qquad (2)$$

Therefore we approximate the surface normal at the center of the 2x2 neighborhood by $\vec{n} = ((-a_1 + a_2 + a_3 - a_4)/4, (-a_1 - a_2 + a_3 + a_4)/4, 1))^T$ (the value of c is irrelevant). Note that the magnitude of this vector depends on the angle it makes with the xy-plane. If the normal is parallel to the z-axis then the magnitude is minimized to 1, and the magnitude grows as the angle with the xy-plane is decreased. This variation is used to weight the average discussed in the next section.

Of course, one can choose a neighborhood larger than 2x2 (see [13]). The advantages of a larger neighborhood are increased directional accuracy and noise tolerance. Noise tolerance is not an issue here, since we do an independent averaging step to control noise (Section 3.3). Directional accuracy may be a factor, however, depending on the type of edges examined. It is well known that gradient type operators such as this one have orientation biases due to discretization. A thorough study of this effect for step edges was done by Kitchen and Malin [14]. (Refer also to O'Gorman [7], which proposes the use of Walsh functions for edge detection.) Nonetheless, we found this effect to be insignificant in our work, and for the smooth edges studied in Section 3.5 the orientation bias was found to be less that 3°. Moreover, larger neighborhoods increase the computational requirement and can have problems relating to edge curvature [11].

3.3 Determining the averaged tangent direction

We want next to find a vector (the unit tangent vector) that is nearly perpendicular to the collection of normal vectors $\{\vec{n}^k\}$, where the index k runs over all 2x2 neighborhoods in the tangent window. One approach is to calculate the tangent direction to each normal, express that direction as an angle between say 0° and 180°, and then calculate the arithmetic average these directions. This is essentially the method used by Kawagoe and Tojo [3]. There are some difficulties with this procedure, however. For example, notice that the average of the directions 0° and 178° should be 179° (not 89°). (See [15] for background on managing directional data.) We take instead a least-squares minimization approach. First we discard the z-components (which are identically one) to get the collection $\{\vec{v}^k\}$, where $\vec{v}^k = (n_1^k, n_2^k)^T$. Note that the magnitude of the vector $\vec{v}^k$ is given by the cotangent of the angle between the normal vector $\vec{n}^k$ and the xy-plane. Let $\vec{u} = (u_1, u_2)^T$ be the unit tangent vector that we want to determine. Formally,

$$\begin{aligned} \text{minimize} \qquad & \sum_k |\langle \vec{v}^k, \vec{u} \rangle|^2 \\ \text{subject to} \qquad & \|\vec{u}\| = 1 \end{aligned}$$

(See [12].)

Let $F(u_1, u_2)$ be the minimization function, i.e., $F(u_1, u_2) = \sum_k |\langle \vec{v}^k, \vec{u} \rangle|^2$. Let $A = \sum_k (v_1^k)^2$, $B = \sum_k (v_2^k)^2$, and $C = \sum_k v_1^k v_2^k$. Then

$$F(u_1, u_2) = (u_1, u_2) \begin{pmatrix} A & C \\ C & B \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{3}$$

Thus F is the quadratic form associated with the (real) symmetric matrix

$$S = \begin{pmatrix} A & C \\ C & B \end{pmatrix}$$

The eigenvalues for this matrix are the extremal values of the function F for $\|\vec{u}\| = 1$. Let us order the eigenvalues so that $\lambda_1 \geq \lambda_2$. Then the minimum value for F is λ_2, and the corresponding eigenvector is the solution to the minimization problem, i.e., yields the averaged tangent direction. (If $\lambda_1 = \lambda_2$ then we have the degenerate case where there is no preferred direction. This indicates a peak or plateau in the image, or else a region that is overcome by random noise.)

The eigenvalues for S are

$$\begin{aligned} \lambda_1 &= \frac{1}{2}\left((A+B) + \sqrt{(A-B)^2 + 4C^2}\right) \\ \lambda_2 &= \frac{1}{2}\left((A+B) - \sqrt{(A-B)^2 + 4C^2}\right) \end{aligned} \tag{4}$$

If $C = 0$ then the original matrix S is diagonal and the eigenvector corresponding to λ_2 is either $(1,0)^T$ or $(0,1)^T$, depending on whether $A < B$ or $B < A$. Otherwise one has the relation

$$\begin{aligned} u_2 &= u_1(\lambda_2 - A)/C \\ &= u_1\left(\frac{B-A}{2C} - \sqrt{\left(\frac{B-A}{2C}\right)^2 + 1}\right) \end{aligned}$$

which defines the averaged tangent direction.

3.4 Region contrast and consistency

In addition to determining the tangent direction, the calculations in the preceding sections also give measures of region contrast and tangent consistency. For example, if the contrast in a region is large, then many of the normal vectors calculated in Section 3.2 will lie close to the xy-plane, and the projected vectors $\vec{v}^k$ (from Section 3.3) will be large. Conversely, if the contrast is small, then most of the normal vectors will be nearly parallel to the z-axis, and the vectors $\vec{v}^k$ will be small. Thus the region contrast is gauged by

$$\begin{aligned} C_R &= \sum_k \left\|\vec{v}^k\right\|^2 \\ &= \sum_k (n_1^k)^2 + (n_2^k)^2 = A + B \end{aligned}$$

where A and B are defined in Section 3.3. Normalization of C_R requires knowledge of the maximum possible magnitude of the normal vectors (which depends on digitization restrictions) in addition to the size of the averaging window (i.e., the number of vectors in the summation). For example, in our system the digitized values lie between 0 and 255, and suppose we choose a 9×9 averaging window. The maximum value for any $\|\vec{v}^k\|$ is $255/2$ (refer to Eq. 2), and there are $8 \times 8 = 64$ 2x2 neighborhoods in the averaging region, so the maximum possible value for C_R is $C_{\max} = (255/2)^2 \times 64$. The normalized contrast score is defined by $C_N = C_R/C_{\max}$.

Tangent consistency is measured by the extent to which the averaged tangent direction $\vec{u}$ is perpendicular to the collection $\vec{n}^k$. This provides an estimate of the noise level in the averaging window. Let E_R denote the sum of the squared error,

$$E_R = \sum_k |\langle \vec{n}^k, \vec{u}\rangle|^2 = \sum_k |\langle \vec{v}^k, \vec{u}\rangle|^2$$

which is of course just the function F of Section 3.3 evaluated in the averaged tangent direction. This minimal value of F is given in Eq. 4, i.e., $E_R = \lambda_2$. This value can be normalized by dividing by C_R. A value of zero indicates a perfect fit (all the vectors $\vec{n}^k$ are perpendicular to the averaged tangent direction), whereas $E_R/C_R = 1$ indicates that all the vectors $\vec{n}^k$ are parallel to the averaged tangent direction—the worst case. Of course the latter does not occur since in this case selecting the perpendicular direction as the tangent direction gives a perfect fit. In fact, the minimization insures that $E_R/C_R \leq 1/2$. Therefore the normalized consistency error is defined to be $E_N = 2E_R/C_R$.

Combined, the two parameters C_N and E_N give a measure of the reliability of the calculated tangent direction. Ideally the region will be high contrast (large C_N) with high consistency (small E_N).

3.5 Controlled tests

To test the stability of the tangent calculation and the usefulness of the consistency score E_N, we performed experiments with two idealized surfaces (a smooth edge and a sinusoid) corrupted by simulated noise. The results are displayed in Tables 1 and 2. Part (a) in each table presents the results using a small (9×9) tangent window, while part (b) shows the results using a larger (19×19) tangent window. The data in Table 1 (a) and Table 2 (b) are represented graphically in Fig. 5 and Fig. 6, respectively. Graphs of the data in Table 1 (b) and Table 2 (a) are very similar to Fig. 5, and so are not reproduced here. The reader may examine the tabular data for details.

The first surface for which we performed tests was a smooth edge. The profile for this edge was defined by

$$h(t) = \begin{cases} 128 + 48(1 - e^{-2.3t}) & \text{if } t \geq 0 \\ 128 - 48(1 - e^{2.3t}) & \text{if } t < 0 \end{cases}$$

where t is the distance (in pixels) from the center of the edge. This profile has an edge transition range, measured from the 10% down points, of two pixels. We used this profile to generate a smooth step (similar to that depicted in Fig. 2) through the center of a tangent window at an angle of 40°. The resulting pixel values, after the required rounding to integer, are shown in Fig. 4. Noise was added to this in the form of a sequence of computer generated uncorrelated zero-mean Gaussian random variables.

Table 1 (a) shows statistics for the tangent calculation using a 9×9 tangent window. (The window dimensions are chosen odd so that the center of the window will correspond to a pixel sampling point.) The first column lists the signal-to-noise ratio (SNR), defined here as the ratio of the variance of the original (no noise) image to the variance of the added noise. For each SNR level, 10^4 trials were performed, and the mean and standard deviation of the tangent angle, consistency score, and contrast score were collected. Notice that even in the noise-free case, the angle calculation is 0.4° off from the true value of 40°. This is due to the discretization of the edge and because the point normal calculation works under the assumption that a plane is a good local approximation to the surface. The magnitude of the error depends upon the angle and the shape of the edge. As shown in [14], this error can be significant across a step edge. The effect is less noticeable across a smooth edge. In fact, for the smooth edge described above, the calculated angle differs from the true angle by less than 3° regardless of the edge orientation. Nonetheless, one could improve the directional accuracy by improving the point normal calculations (Section 3.2), either by increasing the size of the local neighborhood or by using a higher order surface (instead of a plane), albeit at the cost of increased computation time. The contrast scores are not affected by the orientation of the edge, and the effect of edge orientation on the consistency (error) scores is less than 10%.

The data from Table 1 (a) are displayed graphically in Fig. 5, where the error bars denote one standard deviation from the mean. (The distributions about the mean are not quite Gaussian, but are close enough so that roughly 68% of the trials fell within one standard deviation of the mean, and approximately 95% fell within two standard deviations.) We see from the graph that the mean direction value is nearly independent of the noise level, but the variation of the direction from the mean increases with increasing noise, as one expects. Let us now increase the tangent window size to 19×19, so that the window contains (roughly) 4 times as many pixels as before (Table 1 (b)). Offhand, one expects the standard deviation of the angle calculation to drop by 1/2. Comparing the results of part (a) with part (b) from Table 1, one sees that this is not the case. Refer back to Fig. 4. In this tangent window, directional information can only be obtained along the edge, which is confined to a narrow strip passing through the center of the window. Any directional information obtained in the larger flat regions on either side of the edge can only be due to noise. Increasing the dimensions of this window to 19×19 will quadruple the flat area, but the area of the strip containing the edge only doubles. So in the larger window we have twice as much true directional information and four times as much noise. The two effectively cancel, as we see from the results in Table 1, although for small noise levels the larger window is somewhat preferable.

This effect disappears if directional information is present throughout the tangent window. Table 2 gathers the results for the same calculations as Table 1, but with a sinusoid replacing the edge. (The results for the larger window are shown graphically in Fig. 6.) Specifically, we used

$$h(x,y) = 128 + 48 \sin\left(\frac{\pi}{5}(y\cos\theta - x\sin\theta)\right) \tag{5}$$

where the coordinate origin corresponds to the window center and $\theta = 40°$. (This sinusoid has a period of 10 pixels, which roughly approximates the ridge period in the fingerprint image used in Section 5.) With the sinusoid, we see that increasing the window size from 9×9 to 19×19 cuts the standard deviation of the angle calculation by at least one half (and by as much as two thirds for low noise levels).

Let us next consider the tangent consistency score E_N. To be a good predictor of angle reliability, the value of E_N should correlate closely to the standard deviation of the angle calculation. If we compare values of E_N in Table 1 (a) against those in (b), we see that for any fixed angle standard deviation level the corresponding mean E_N values in (a) are smaller than those in (b). This is expected and simply means that the larger window can tolerate larger errors and still keep the same angle calculation variance. It is more meaningful to compare the results from the smoothed edge with the sinusoidal at the same window size (i.e., compare Table 1 (a) with Table 2 (a), and Table 1 (b) with Table 2 (b)). Doing this we see that for any given angle standard deviation value, the corresponding mean E_N values are within 15% across the tables. For example, suppose we want to consider the calculated tangent angle reliable if the standard deviation is less than 5°. Then for the 9×9 window, the angle is reliable if the value of E_N is less than about 0.47, regardless of whether we look at the single smoothed edge (Table 1 (a)) or the sinusoid (Table 2 (a)). Thus we can use the value of E_N as a measure of angle reliability, independent of the image structure. Similarly, for the 19×19 window, the 5° standard deviation value corresponds to a E_N score of about 0.68.

Of course, for any one trial the value E_N is a random variable, so its variance is important as well. For example, for the 19×19 window, the standard deviation of E_N with mean value 0.68 is 0.04. Thus if the observed value for E_N is less than 0.6, then with better than 97% probability the noise is in the acceptable range. For smaller windows the variance of E_N is larger, so the estimate is less tight. For E_N with mean value of 0.47 in the 9×9 window case, the standard deviation is 0.07, so to reach the 97% assurance level we need $E_N < 0.34$.

Consider now the contrast score C_N. In the smoothed edge case, comparing the results from the small window to the large window shows that for any given noise level the C_N score drops. This is correct since the larger window contains a larger percentage of flat (zero contrast) area. (Recall that the C_N score is normalized by the window size.) For the sinusoid we see no difference between the C_N scores for the small versus large window, also a correct result. It is somewhat surprising that the C_N score for the small window, smoothed edge (Table 1 (a)) is larger than the C_N score for the sinusoids. However, the edge is much steeper than the sinusoid slope, and for the small window the edge fills a significant amount of the total area. Of more importance is the relative insensitivity of the C_N score with respect to the noise level, which is evident from the graphs of C_N provided in Figs. 5 and 6. Also the standard deviation of C_N is less than 10%, so the C_N score is a reasonable predictor of the underlying (noise-free) image contrast.

As a final comment, note that an edge is not required for the tangent calculation to return meaningful information. Indeed, the surface may be any for which the level curves are defined (i.e., any inclined surface).

4 Curvature vector calculation

4.1 Problem formulation

Let us now consider the problem of calculating curvature from level curves. Most previous work on image curvature has been restricted to calculating curvature magnitude from a given boundary curve [8, 9]. In this approach one first uses an edge detection algorithm to locate the image boundary, then pieces the boundary together (using, for example, a chain code algorithm), and then approximates the second derivative of the resulting (1-dimensional) curve. Conversely, our approach uses the results of the tangent calculation only, without any intermediate processing. Moreover, although our procedure works along single edges (especially if one incorporates the extensions suggested in Section 4.6), it works best on striated images such as fingerprints, where it can use in the calculations data from several adjacent level curves.

Refer back to Fig. 1, and consider the curvature at the point P. We have at our disposal not only the level curve through P, but also the adjacent level curves, which we want to include in our calculation. Notice that the magnitude of the curvature changes as one moves between level curves, increasing as one moves closer to the curvature center, and decreasing as one moves farther away. Similarly, as one moves along a level curve, the magnitude of the curvature is (or is nearly) constant, but the direction of the curvature vector (towards the center of the curvature) varies. Thus we see that the curvature vector is not constant in any neighborhood of P. The curvature center, however, is (or is nearly) constant. So instead of trying to find an average value for the curvature vector directly, we calculate first an average value for the curvature center. Then the curvature vector can be calculated from the relation $\vec{r}''(0) = \overrightarrow{PP_c}/\|\overrightarrow{PP_c}\|^2$, where P_c denotes the curvature center.

The curvature center lies on the line $\mathbf{L}$ through P that is perpendicular to the level curve tangent vector at P. Since we can calculate the tangent vector using the technique discussed in Section 3, we need only calculate the position of P_c on the line $\mathbf{L}$. To this end choose another point near P which is not on $\mathbf{L}$. If this new point is close enough to P then we expect the two points to share the same curvature center. Construct a line through the second point perpendicular to its tangent. The curvature center P_c is on both lines and is therefore given by the intersection of the two lines. (If the lines are parallel then the curvature is zero and we consider the point P_c to be at infinity.)

Let us expand this idea to a larger region. In Fig. 7 there are nine points (P_1, P_2, …, P_9) with associated tangent vectors ($\vec{u}^1$, $\vec{u}^2$, …, $\vec{u}^9$) inside a dashed rectangle called the **curvature window**. (The size of the curvature window depends upon the application, but is generally several times larger than the tangent window.) Now we want to find a point P_c that can be regarded as the curvature center for all points P_k inside the curvature window. In particular, each radius vector $\overrightarrow{P_kP_c}$ should be perpendicular to the associated tangent vector $\vec{u}^k$, i.e.,

$$\left\langle \overrightarrow{P_kP_c}, \vec{u}^k \right\rangle = 0 \quad \text{for } k = 1, 2, \ldots, 9 \tag{6}$$

Then the curvature vector at the center of the window would be $\overrightarrow{P_5P_c}/\|\overrightarrow{P_5P_c}\|^2$.

Of course, it is unlikely that the nine lines will intersect in a point, so we cannot expect there to be a point P_c satisfying the orthogonality conditions of Eq. 6. Let us find instead a point P_c such that each radius vector $\overrightarrow{P_kP_c}$ in the curvature window is nearly orthogonal to the corresponding tangent vector $\vec{u}^k$. But how to quantify the concept of "nearly orthogonal"? We consider two different formulations. The first formulation is simple to implement, but is unstable if the tangent vectors are nearly parallel. The second formulation does not suffer from this instability, but is more difficult to implement.

It is interesting to note that similar problems arise in the application of computer vision to mobile robots [16, 17, 18, 19]. Objects in successive image frames appear to move outward as the robot moves towards them (pure translation). The point from which the motion seems to originate is called the **focus of expansion** (FOE). Lines drawn through the objects in their direction of apparent motion should all intersect at the FOE. This doesn't occur, of course, due to noise and quantization errors. So given the lines of motion, the problem is to find a point which is as close as possible to all the lines. This is equivalent to the problem that results from using the first orthogonality formulation described below.

4.2 First formulation of orthogonality condition

Each pair $(P_k, \vec{u}^k)$ defines a line on which we would like the center to lie—the line $\mathbf{L}^k$ that runs through P_k and is perpendicular to $\vec{u}^k$. Let P_c be a candidate point for the curvature center, and let $d(P_c, \mathbf{L}^k)$ be the perpendicular distance from P_c to the line $\mathbf{L}^k$, as illustrated in Fig. 8. The point P_c that we select as the actual curvature center is the point that minimizes the sum of the square of these distances, i.e.,

$$\min_{P_c} \sum_{k=1}^{n} \langle \overrightarrow{P_kP_c}, \vec{u}^k \rangle^2 \tag{7}$$

where n is the number of points in the curvature window. Here we have assumed that the $\vec{u}^k$ are unit vectors, although we will later relax this restriction. In this formulation (Eq. 7), points P_k that are far from P_c have a greater weight in the sum than those close to P_c. The advantage in this is that the tangent vector calculation is more reliable in regions of shallow curvature, so it is natural to weight points away from the curvature center more heavily. Unfortunately, moving the point P_c in towards the curvature window will generally reduce the sum as a whole (since this reduces the size of the vectors $\overrightarrow{P_kP_c}$). Therefore this formulation tends to pull the curvature center inward, increasing the calculated curvature. This is related to the instability problem discussed below.

But let us first derive an explicit solution to Eq. 7. Impose a coordinate system on the plane and consider points in the plane as position vectors. Let $\vec{u}^k = (a_k, b_k)^T$, $\vec{P}_c = (x, y)^T$, and $r_k = \langle \vec{P}_k, \vec{u}^k \rangle$. Furthermore, let $G(\vec{P}_c) = G(x, y)$ be the function being minimized in Eq. 7. Then

$$\begin{aligned} G(x,y) &= \sum_{k=1}^{n} (r_k - a_k x - b_k y)^2 \\ &= Ax^2 + By^2 + 2Cxy - 2Dx - 2Ey + M \end{aligned} \tag{8}$$

where $A = \sum_{k=1}^{n} a_k^2$, $B = \sum_{k=1}^{n} b_k^2$, $C = \sum_{k=1}^{n} a_k b_k$, $D = \sum_{k=1}^{n} a_k r_k$, $E = \sum_{k=1}^{n} b_k r_k$, and $M = \sum_{k=1}^{n} r_k^2$.

The minimization of $G(x, y)$ requires the partial derivatives $G_x = G_y = 0$, so

$$\begin{aligned} G_x(x,y) &= 2Ax + 2Cy - 2D = 0 \\ G_y(x,y) &= 2By + 2Cx - 2E = 0 \end{aligned} \tag{9}$$

In matrix form this becomes

$$\begin{pmatrix} A & C \\ C & B \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} D \\ E \end{pmatrix} \tag{10}$$

Solving explicitly gives

$$\vec{P}_c = \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{AB - C^2} \begin{pmatrix} B & -C \\ -C & A \end{pmatrix} \begin{pmatrix} D \\ E \end{pmatrix}, \tag{11}$$

provided that the discriminant $AB - C^2 \neq 0$, in which case this is the unique solution to the minimization problem. On the other hand, the discriminant is zero only if all the tangent vectors $\vec{u}^k$ are parallel, in which case the curvature is zero and we would like the center point $\vec{P}_c$ to be placed at infinity.

Unfortunately, this method is unstable if the tangent vectors are nearly parallel. To see this, consider first the case where the tangent vectors are all exactly parallel. Then Eq. 7 attains its minimum value along the entire line through the center of the curvature window that is perpendicular to the tangent vectors' direction. We would like to select the minimization point to be infinity, but from Eq. 7 any point along this line works as well. Now rotate any of the tangent vectors by a small amount, so that the tangent vectors are not all parallel. Then the discriminant $AB - C^2 \neq 0$, so there is a unique solution, but the discriminant is small so the solution is unstable. In particular, even though the solution will lie close to the aforementioned line, it will not necessarily lie near infinity. Moreover, since the formulation penalizes large distances, it is likely that the solution will lie near the center of the curvature window. This would imply a very large curvature, even though the tangent vectors are only slightly perturbed from the parallel condition indicative of zero curvature. Experimental results confirm that this effect is a serious weakness of this formulation.

4.3 Second formulation of orthogonality condition

So we need a different formulation for the orthogonality condition. Let $\vec{v}^k$ be a unit vector perpendicular to $\overrightarrow{P_kP_c}$, and define $\vec{e}^k = \vec{u}^k - \langle \vec{u}^k, \vec{v}^k \rangle \vec{v}^k$ as illustrated in Fig. 9. Now pick P_c to minimize

$$\min_{P_c} \sum_{k=1}^{n} \|\vec{e}^k\|^2 = \min_{P_c} \sum_{k=1}^{n} \left\langle \vec{u}^k, \frac{\overrightarrow{P_kP_c}}{\|\overrightarrow{P_kP_c}\|} \right\rangle^2 \tag{12}$$

This formulation weights each point equally, but is more complicated than the formulation of Eq. 7. In particular, notice that this new formulation suffers from discontinuities at $P_c = P_k$ for each $k = 1, 2, \ldots, n$. However, let us consider the situation where the center point P_c is far from the curvature window, in which case the terms $\|\overrightarrow{P_kP_c}\|$ are nearly identical. Let us place our coordinate system origin at the center of the curvature window and approximate each term $\|\overrightarrow{P_kP_c}\|$ by $\|\vec{P}_c\|$. We then have the simplified formulation

$$\min_{P_c} \sum_{k=1}^{n} \frac{\left\langle \vec{u}^k, \overrightarrow{P_kP_c} \right\rangle^2}{\|\vec{P}_c\|^2}. \tag{13}$$

Introducing A, B, C, D, E, and M as in Eq. 8 yields the minimization function

$$\tilde{G}(x, y) = \frac{Ax^2 + By^2 + 2Cxy - 2Dx - 2Ey + M}{x^2 + y^2}. \tag{14}$$

We want to find the global minimum of this function on the xy-plane. A necessary condition for a local minimum is that the directional derivatives in the radial and angular directions be equal to zero. So let us write $\tilde{G}$ in polar coordinates ($x = r\cos\theta$, $y = r\sin\theta$):

$$\begin{aligned}\tilde{G}(r,\theta) &= A(\cos\theta)^2 + B(\sin\theta)^2 + 2C\cos\theta\sin\theta \\ &\quad + \frac{-2Dr\cos\theta - 2Er\sin\theta + M}{r^2}.\end{aligned} \tag{15}$$

The partial derivatives are given by

$$\tilde{G}_r(r,\theta) = \frac{2Dr^2\cos\theta + 2Er^2\sin\theta - 2Mr}{r^4} \tag{16}$$

and

$$\begin{aligned}\tilde{G}_\theta(r,\theta) &= -2A\cos\theta\sin\theta + 2B\cos\theta\sin\theta + 2C\left((\cos\theta)^2 - (\sin\theta)^2\right) \\ &\quad + \frac{2D\sin\theta - 2E\cos\theta}{r}.\end{aligned} \tag{17}$$

Setting $\tilde{G}_r = 0$ and $\tilde{G}_\theta = 0$, simplifying, and converting back to rectangular coordinates produces the requirements

$$Dx + Ey = M, \tag{18}$$

$$(B-A)xy + C(x^2 - y^2) + Dy - Ex = 0. \tag{19}$$

If M is not zero then at least one of D or E will be non-zero, so these two equations can be combined to yield a quadratic equation in one variable. The (at most) two roots of this quadratic combine with Eq. 18 to produce (at most) two candidate points for local minima of $\tilde{G}$. It is also possible that the global minimum of $\tilde{G}$ is attained at infinity. Note that

$$\lim_{r\leftarrow\infty} \tilde{G}(r,\theta) = A(\cos\theta)^2 + B(\sin\theta)^2 + 2C\cos\theta\sin\theta, \tag{20}$$

which can be written

$$\lim_{r\leftarrow\infty} \tilde{G}(r,\theta) = (\cos\theta, \sin\theta)\begin{pmatrix} A & C \\ C & B \end{pmatrix}\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}. \tag{21}$$

Comparing to Eq. 3 and Eq. 4 shows that the (boundary) minimum value at infinity is given by the smaller matrix eigenvalue, namely

$$\lambda = \frac{1}{2}\left((A+B) - \sqrt{(A-B)^2 + 4C^2}\right). \tag{22}$$

(Notice that Eq. 13 implies that $\lambda \geq 0$.) One now compares this value with the value of $\tilde{G}$ evaluated at the (at most) two candidate points found previously. The global minimum is the smallest of these three values, and the curvature center is at the corresponding point.

If M is zero, then the curvature center should be placed at the center of the curvature window (infinite curvature). However, $M = 0$ implies that $D = 0$ and $E = 0$ as well. Introducing this into Eq. 14 shows that the minimizing set consists of an entire line through the center of the curvature window. We can therefore expect the same type of instability in this case as we had for the first orthogonality formulation when the curvature was near zero. Moreover, this second formulation will tend to force the calculated curvature center away from the origin because if $M \neq 0$ then $\tilde{G}$ has a pole of order 2 at the origin. Thus the results using the simplified formulation of Eq. 13 are unsatisfactory for high curvature situations. (This is expected, of course, since the simplified formulation is based on the assumption that the curvature center P_c is far from the curvature window.)

4.4 Combining the two orthogonality formulations

Fig. 10 shows the results of applying the two orthogonality formulations to noisy images. For this example we used a curvature window consisting of 9 tangent vector nodes, laid out in a 3×3 grid as in Fig. 7. Both the row and column spacing was set to 10 pixels. We fixed a curvature center point P_c and calculated the ideal tangent direction at each of the 9 nodes. The distance of the point P_c from the center of the curvature window determines the magnitude of the curvature vector $\overrightarrow{P_5P_c}/\|\overrightarrow{P_5P_c}\|^2$ (where P_5 is the window center), and the orientation of P_c relative to the fixed 3×3 tangent node grid determines the angular component of the curvature vector.

Both orthogonality formulations produced perfect results on ideal data, so we introduced noise by rotating the 9 tangent directions separately by random amounts. The rotations were determined from a sequence of computer generated zero-mean uncorrelated Gaussian random variables. The noise level was controlled by adjusting the standard deviation of the random variables to $1°$, $3°$, or $5°$. The curvature calculations were performed using the perturbed tangent directions. The random variables were then resampled to generate new tangent perturbations, and the process was repeated. Each point in Fig. 10 represents the mean curvature value from 10^5 trials. The abscissa in Fig. 10 gauges the original (before noise) curvature, while the ordinate provides the calculated curvature. (The results were found to be independent of the angular component of the curvature vector.) The points marked with $+$'s are the results using the first orthogonality formulation, while $\triangle$'s mark the results using the second orthogonality formulation. There is one curve at each noise level for each formulation.

Ideally the data points in this graph should lie on the line of slope 1 through the origin. However, in the low curvature region (10^{-3} to 10^{-2} pixels^{-1}) the first orthogonality formulation produces exaggerated curvatures from noisy data, as expected. The second formulation performs better across most of the curvature range, although for curvatures near 1 pixel^{-1} the calculated values fall below the ideal values, as also anticipated. Since the first formulation works well for high curvature situations, and the second formulation works well for low curvature situations, it is natural to combine the two formulations. Fig. 10 suggests that the first formulation be used for curvatures above about 0.1 pixels^{-1}, and the second formulation below. (The actual curvature values are relative to the size of the curvature window. Here the radius of the curvature window is roughly 10 pixels, so a curvature of 0.1 pixels^{-1} corresponds to the curvature center $\vec{P}_c$ being positioned on the edge of the curvature window.) Therefore we need an estimator for the curvature which is reliable (with respect to noise) in some range around 0.1 pixels^{-1}. Our experiments suggest that the value of λ from Eq. 22 provides such an estimate. This value ranges between 0 if the tangent vectors are parallel to $n/2$ if the tangent vectors are concentric about the center of the curvature window (recall that n is the number of tangent vectors in the curvature window). The examples presented in Section 5 use $\lambda/n = 0.1$ as the method crossover point, i.e., if $\lambda/n > 0.1$ the first curvature formulation was used, otherwise the second.

Fig. 11 graphs λ/n under the conditions used to generate Fig. 10 (here $n = 9$). We see that $\lambda/n = 0.1$ corresponds to a curvature of about 0.04 pixels^{-1}, a little lower than ideal, but still an acceptable method crossover point. Moreover, if one examines the portion of the graph in Fig. 11 near curvature 0.1, one finds a small indentation in the knee of the curve. This is due to discontinuities in Eq. 22 as the curvature center $\vec{P}_c$ approaches tangent vector nodes $\vec{P}_k$. Since the tangent vector node spacing is 10 pixels, the curvature center enters the curvature window at curvatures between 0.0707 and 0.1 pixels^{-1}, depending on the angular component of the curvature vector. Unlike the curvature calculations in Fig. 10, the values graphed in Fig. 11 are dependent on the angular component of the curvature vector, though only in the curvature range between about 0.05 and 0.12 pixels^{-1}. (In Fig. 11 the angular component of the curvature vectors is fixed at 57°.) Thus setting the method crossover point at $\lambda/n = 0.1$ is in fact a reasonable choice.

If the minimization error ($G(x, y)$ or $\tilde{G}(x, y)$) is not zero, then the curvature center determined using the first method will lie closer to the curvature window than the curvature center determined using the second method. There will generally be, therefore, some discontinuity in the calculated curvature at the method crossover point. If necessary, one can force continuity by using a weighted average of the two calculated curvature centers over a transition region. For example, let $\vec{P}_{c1}$ be the curvature center calculated by the first method, $\vec{P}_{c2}$ by the second. Then use

$$\vec{P}_c = \begin{cases} \vec{P}_{c1} & \text{if } \lambda > 0.12n \\ \frac{\lambda - 0.08n}{0.04n}\vec{P}_{c1} + \frac{0.12n - \lambda}{0.04n}\vec{P}_{c2} & \text{if } 0.08n \leq \lambda \leq 0.12n \\ \vec{P}_{c2} & \text{if } \lambda < 0.08n \end{cases}$$

As an alternative to combining the two curvature formulations, one can explore the possibility of using Eq. 12 directly. Minimizing this function requires numerical methods, which may be poorly behaved in high curvature situations (due to discontinuities at $P_c = P_k$). This is left for future study.

4.5 Controlled tests

In order to study the behavior of the combined curvature calculation method in a controlled fashion, we repeated the experiment of Section 4.4 using λ/n to automatically select the orthogonality formulation. The results are gathered in Table 3. The curvature window consisted of 9 tangent vector nodes, laid out in a 3×3 grid with both row and column spacing set to 10 pixels. As noted before, the dependence of the curvature calculation on the orientation of the curvature vector (directional component) was found to be minimal. Therefore only the dependence on the distance component ("Noise-free Curvature") is listed in Table 3.

The second column in Table 3 lists the noise level introduced into the data (standard deviations of 1°, 3°, or 5°). For most curvature/noise level combinations, 10^4 trials were performed, but in several instances 10^6 trials were necessary to provide the desired precision. The resulting statistics are presented in columns 3 through 5.

The results are divided into three columns: curvature mean error, curvature standard deviation, and direction standard deviation. Recall that the result of the curvature calculation is the curvature vector, with magnitude equal to the reciprocal of the curvature radius and direction towards the center of the curvature. The curvature mean error is the difference of the mean value of the calculated curvature magnitude from the curvature before noise, expressed as a percentage with respect to the before noise value. The curvature standard deviation is the standard deviation of the curvature magnitude about its mean value. The difference between the calculated curvature direction mean value and the noise-free direction was insignificant. The standard deviation of the direction component was significant, however, and is listed in the last column of Table 3.

The orthogonality formulation used for curvature calculation was selected automatically as detailed in Section 4.4, using $\lambda/n = 0.1$ as the method crossover point. In these tests the curvature at the method crossover point was close to 0.04 pixels^{-1}.

This means that the method of Section 4.2 was used in those cases where the curvature was more than 0.05 pixels^{-1}, and the method of Section 4.3 was used where the curvature was less than 0.03 pixels^{-1}. Near 0.04 pixels^{-1}, the algorithm chosen depended upon the noise sample. The trials at noise-free curvature of 0.04 pixels^{-1} had each method selected roughly half the time. The results in the range below 0.03 pixels^{-1} are most straightforward, so we shall discuss them first.

The calculation errors were nearly independent of curvature in the range between 0.001 and 0.03 pixels^{-1}. The curvature mean value is generally slightly below the true value, and the error increases with increased noise. Notice that in this range the standard deviation of both the curvature magnitude and direction increase linearly with standard deviation of the added noise. No results are listed for curvature less than 0.001 pixels^{-1} because extremely shallow curvature cannot be separated from noise. Even at a curvature of 0.001 pixels^{-1}, the tangent directions in the curvature window vary from the center direction by at most 0.58°. Rotate these directions by noise with standard deviation as small as even 1° and the original curvature is completely lost. The net result is that for extremely shallow curvature (say less than 0.001 pixels^{-1}), the curvature calculation reports a small curvature, but it is not possible to distinguish between two different but very small curvatures in the presence of noise. Along the same lines, the returned direction in the small curvature situation has meaning only modulo 180°. For example, consider Fig. 12, which illustrates two curves ((a) and (b)) with small curvature. Curve (a) has curvature direction near 90°, whereas curve (b) has curvature direction near 270°. In the presence of noise, these two curves cannot be distinguished—even though their curvature directions differ by 180°. To allow proper comparisons, it is necessary to restrict the curvature directions to a fixed 180° range, say between 0° and 180°, and to introduce negative curvatures. If the curvature direction is inside the restricted range, no change is made. However, if the curvature direction is outside this range, then we subtract 180° degrees from the curvature direction and multiply the curvature magnitude by -1. For example, with this modification the curvature vector of curve (a) is unchanged, but curve (b) would have curvature direction near 90° and a (small) negative curvature.

As the curvature increases from 0.03 through 0.04 pixels^{-1}, the curvature calculation method used shifts from the method of Section 4.3 to the method of Section 4.2, until for curvatures ≥ 0.05 pixels^{-1} the method of Section 4.2 is used almost exclusively. We see that the variance in the direction result increases as the curvature center point P_c moves in towards the curvature window. The curvature magnitude results, on the other hand, actually improve as the curvature increases, until the curvature reaches 0.1 pixels^{-1}. The dimensions of the curvature window are such that at this point the curvature center is just inside the curvature window. After this point the results degrade. In high curvature situations there are problems analogous to the problems at very low curvature. For example, suppose the true curvature is 1 pixel^{-1}. Then the curvature center is only 1 pixel away from the center of the curvature window. Since the curvature magnitude and direction is measured relative to the window center, small changes in the location of P_c have dramatic changes in the curvature values. For example, a location error of only 1 pixel can move P_c from its original location to the exact center of the curvature window, at which point the curvature magnitude is infinite and the curvature direction is indeterminate. However, this error cannot change the calculated curvature from a large value to a small value. We can say that the curvature is very large, but we can not say exactly how large. That is to say, the distribution of the curvature magnitude about the mean is not symmetric. Consider, for example, the test from the last line of Table 3, where the curvature mean value is 1.27 pixels^{-1} with standard deviation of 2.6 pixels^{-1}. In this test only 27% of the trials were above the mean value, indicating that the curvature values beneath 1.27 pixels^{-1} were mostly not far from that value. Indeed, the median value for this test was 0.92 pixels^{-1}, and fewer than 4% of the values were below 0.5 pixels^{-1}.

4.6 Extensions

The contrast (C_N) and consistency (E_N) measures of the tangent direction from Section 3.4 give a measure of the reliability of the directional information. It is natural, therefore, to weight the tangent directions in the minimization problems of Eq. 7, 12, and 13 according to their reliability. In particular, one can introduce weights w_k^2 into Eq. 7 by

$$\min_{P_c} \sum_{k=1}^{n} w_k^2 \langle \overrightarrow{P_k P_c}, \vec{u}^k \rangle^2. \tag{23}$$

For example, we can set $w_k = 0$ if the contrast score C_N for tangent vector u_k is too small, and otherwise set $w_k = 1 - E_N$.

Let us rewrite Eq. 23 as

$$\min_{P_c} \sum_{k=1}^{n} \langle \overrightarrow{P_k P_c}, w_k \vec{u}^k \rangle^2$$

where each weight w_k becomes the magnitude component of the tangent vector $\vec{u}^k$. Although we originally specified in Eq. 7 that the vectors $\vec{u}^k$ be unit vectors, this restriction is not used in the derivation of the solution. Working through the derivation with $\vec{u}^k$ replaced by $w_k\vec{u}^k$, we find that the solution to Eq. 23 is given by Eq. 11 but with $A = \sum_{k=1}^{n}(w_k a_k)^2$, $B = \sum_{k=1}^{n}(w_k b_k)^2$, $C = \sum_{k=1}^{n} w_k^2 a_k b_k$, $D = \sum_{k=1}^{n} w_k^2 a_k r_k$, $E = \sum_{k=1}^{n} w_k^2 b_k r_k$, and $M = \sum_{k=1}^{n}(w_k r_k)^2$. Weighting can be added to Eq. 12 and 13 in a similar fashion.

One can generally improve the curvature results by averaging the tangent vectors before the curvature calculation takes place. For example, consider images for which the tangent vectors form a smooth flow, with a handful of singular points. One wants to average the tangent vectors in the smooth regions without losing the singular points. Kawagoe and Tojo [3], working on fingerprint classification, used an effective relaxation technique along with singular region detection. Alternately, one may be able to adapt the convex projection technique of Simard and Mailloux [20] to this problem.

5 Examples

In our experimental system images are input from a standard video camera and digitized using a Data Translation IBM-AT compatible frame grabber. The digitized images have up to 256 grey levels and a full screen image is 512 pixel columns by 480 pixel rows. The pixel aspect ratio is such that a rectangle 4 columns wide by 5 rows high appears square. Since this makes the sample node grid non-square, a modification is required to the point normal calculation. The details are straightforward, however, and are left to the reader.

5.1 Tangent examples

One sample use of the tangent calculation is in fingerprint identification. Ridge flow directions must be established in order to locate both minutiae (ridge endings and bifurcations) and flow singularities (cores and deltas). An inked fingerprint is shown in Fig. 13. Notice the hole on the right side and the poor contrast due to over inking at the top and lower left side. We applied the algorithm of Section 3 to this image using tangent windows of 19 rows by 15 columns. (When the pixel aspect ratio of the frame grabber board is taken into account, this window is approximately square.) The window size is chosen large enough so that noise can be controlled, but small enough so that curvature inside the window is negligible. We have found this size window, which will generally contain 2 ridges, to work well in practice.

Fig. 14 shows the results of the tangent calculation for each point on a 10 row by 8 column grid. (Compare also to examples in [10, 11].) The corresponding tangent direction is indicated by a line segment unless the contrast or consistency is poor (refer to Section 3.4), in which case the point is left unmarked. This gives a rough indication of which regions of the print are of good quality (marked) and poor quality (unmarked). In our actual experimental system the good regions are further classified by encoding the line segments with color. Notice that the hole on the right side is left unmarked, as are sections of the top and bottom of the print. Also notice that there are no tangent directions marked outside the fingerprint (in the background on the extreme left and right). We see that in addition to giving the ridge flow direction, the tangent calculation can also be used for fingerprint segmentation.

A more general use of the tangent calculation is for adaptive filtering. For example, if one wishes to enhance edges in an image, then knowledge of the tangent direction allows one to select an appropriate orientation specific filter. In the following examples we used 8 convolution-type filters (one each for 0°, 22.5°, 45°, ..., 157.5°). The kernel of the filter for the direction 22.5° was

$$\begin{array}{rrrrrrr}
0 & 0 & -8 & -6 & 0 & 0 & 0 \\
-1 & -9 & -8 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 4 \\
0 & 0 & 0 & 2 & 11 & 18 & 7 \\
0 & 2 & 11 & 18 & 11 & 2 & 0 \\
7 & 18 & 11 & 2 & 0 & 0 & 0 \\
4 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -8 & -9 & -1 \\
0 & 0 & 0 & -6 & -8 & 0 & 0
\end{array}$$

with the resulting value scaled via division by 66. Each filter exhibits low-pass (smoothing) behavior in the specified direction and high-pass (sharpening) behavior in the perpendicular direction. We then used the following procedure. At each point in the image, calculate the tangent direction along with the contrast and consistency scores. If the contrast and consistency are sufficiently good (as determined by C_N and E_N), then select and apply the filter which has orientation corresponding to the tangent direction. If the contrast or consistency of the region is poor, then leave the image value at that point untouched.

The results of apply this procedure to the fingerprint of Fig 13 is shown in Fig. 15. The tangent averaging window was 19 rows by 15 columns. Regions were considered to be of poor quality (and hence left untouched) if the normalized contrast score C_N was less than 0.07 or if the consistency error E_N was larger than 0.7. Repetition of the filtering allows good regions to expand into poor regions. (This effect was also noticed by Peli [5], who obtained similar results using frequency domain analysis and filtering.) Fig. 16 shows the results after 8 iterations of the filter. To prevent over-filtering, the first 6 passes filtered only those points which were not modified (due to poor contrast or tangent consistency) in the preceding passes. After the sixth pass most noise had been eliminated from the image, so for the final two passes a smaller 9×7 tangent window was used for increased directional precision.

As another example of the use of this directional enhancement, consider Fig. 17 which is an image of a porous membrane obtained from a scanning electron microscope. For experimental work we needed to determine the pore volume fraction.

Thresholding the image is made difficult by the vagueness of the pore boundaries. Two passes of this directional filtering gives the image in Fig. 18. Thresholding this image yields pore fractions which agree to within 1% with values obtained by manual inspection.

5.2 Curvature examples

Regions of high curvature denote singular regions in images. Refer back to the fingerprint image of Fig. 13. The most notable regions are the core (near the center) and the two deltas (below and on either side of the core). Fig. 19 illustrates the results of the curvature calculation. (Compare to [10].) The curvature is displayed on a mesh of 20 rows by 16 columns. (This mesh size was chosen for presentation purposes; a denser grid makes the results more difficult to read.) Curvature is not calculated in poor regions (as defined by the tangent calculations), which explains the lack of markings in the hole on the right side and in the lower left corner. Each curvature window consisted of a 3×3 array of tangent vectors from Fig. 14. If the calculated curvature radius is less than 150 pixels (roughly 1/3 the image height), then the calculation window center is marked with a '$+$' and a radial line is drawn to the curvature center, marked with a '$\circ$'. If the curvature radius is larger than 150 pixels, then a short line segment is drawn at the calculation window center in the direction perpendicular to the calculated curvature direction. (Most of the curvature radii are quite large; if every radial line were drawn then the image would be a tangle of lines.)

Careful examination of this figure reveals that the regions with the largest curvature (shortest radial lines) are the deltas and the areas just above and below the core. This is borne out in the curvature magnitude contour plot of Fig. 20. Actually, there is an implicit assumption in the curvature calculation that the tangent flow is continuous. This assumption is violated across the fingerprint deltas, but nonetheless the algorithm returns the desired result: large curvature.

The curvature can be used to locate features in a wide array of images. Fig. 21 shows the curvature results for a section of a printed circuited board. The curvature accurately marks the circular wire pads and the resistor elements. In addition, the tangent calculation has distinguished the circuit trace boundaries from the featureless background board.

As a final example, Fig. 22 shows curvature for a low contrast radiograph of a steel part with 4 shallow, oval slots (simulating flaws). Due to the poor contrast, only 3 of the 4 slots are definitely detectable (the second slot from the left is detectable mainly due to the presence of the other three slots). High curvature locates the 3 detectable slots.

6 Summary

In this paper we have presented algorithms for the extraction of tangent directions and curvatures of level curves of images. The effects of noise on these algorithms were studied under controlled conditions using simulated data and the usefulness of the tangent and curvature information was shown by examples with several real images.

The tangent direction is extracted by a least-squares minimization over the surface normals (calculated for each 2×2 pixel neighborhood) in the averaging window. Even with a signal-to-noise ratio (defined as the ratio of the variance of the original image to the variance of the noise) as low as 10, a single edge passing through the center of a 9×9 window results in a tangent angle calculation having mean value accurate to within 3°. The standard deviation of this calculation is less than 3.5°, a value which drops to 1.3° when the edge is replaced with a sinusoid passing through a 19×19 window. The minimization error from the tangent calculation can be used to estimate the reliability of the calculated tangent direction. This error grows with the variance of the calculated tangent direction, and has a standard deviation $< 10\%$ for the larger 19×19 window.

The usefulness of these calculations were illustrated by using the tangent information to select orientation sensitive filters for edge enhancement on images of a fingerprint and of the microstructure of a porous membrane. The contrast and consistency scores from the tangent calculation also effectively segmented the fingerprint and a section from a printed circuit board.

Unlike most previous work on this topic, the curvature calculation does not require a (single) parameterized curve, but works instead directly on the tangent directions across adjacent level curves. The curvature is found by fitting concentric circles to the tangent directions via least-squares minimization. Accuracy and reliability were studied by controlled tests with simulated noise. The curvature information can be used for feature detection and identification, as illustrated by results showing high curvature locating cores and deltas on a fingerprint, circular pads and circuit elements of a printed circuit, and slots simulating flaws in a radiograph of a steel part.

References

[1] Stephen P. Morse. A mathematical model for the analysis of contour-line data. *Journal of the Association for Computing Machinery*, 15(2):205–220, 1968.

[2] Thomas K. Peucker and David H. Douglas. Detection of surface-specific points by local parallel processing of discrete terrain elevation data. *Computer Graphics and Image Processing*, 4:375–387, 1975.

[3] Masahiro Kawagoe and Akio Tojo. Fingerprint pattern classification. *Pattern Recognition*, 17(3):295–303, 1984.

[4] Art S. Rabinowitz. Fingerprint card search results with ridge-contour based classification. In *5th International Conference on Pattern Recognition*, pages 475–477. IEEE, 1980.

[5] Eli Peli. Adaptive enhancement based on a visual model. *Optical Engineering*, 26(7):655–660, 1987.

[6] M. Hueckel. An operator which locates edges in digital pictures. *Journal of the Association for Computing Machinery*, 18:113–125, 1971.

[7] Frank O'Gorman. Edge detection using Walsh functions. *Artificial Intelligence*, 10:215–223, 1978.

[8] Theodosios Pavlidis. A review of algorithms for shape analysis. *Computer Graphics and Image Processing*, 7:243–258, 1978.

[9] Azriel Rosenfeld and Emily Johnston. Angle detection on digital curves. *IEEE Transactions on Computers*, 22:875–878, 1973.

[10] Pierre Parent and Steven W. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):823–839, 1989.

[11] Steven W. Zucker. Early orientation selection: Tangent fields and the dimensionality of their support. *Computer Vision, Graphics, and Image Processing*, 32:74–103, 1985.

[12] David G. Luenberger. *Optimization by Vector Space Methods*. Wiley, New York, 1969.

[13] Robert M. Haralick and Layne Watson. A facet model for image data. *Computer Graphics and Image Processing*, 15:113–129, 1981.

[14] L. J. Kitchen and J. A. Malin. The effect of spatial discretization on the magnitude and directional response of simple differential edge operators on a step edge. *Computer Vision, Graphics and Image Processing*, 47:243–258, 1989.

[15] K. V. Mardia. *Statistics of Directional Data*. Academic Press, New York, 1972.

[16] Ramesh Jain. Direct computation of the focus of expansion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):58–64, 1983.

[17] K. Prazdny. On the information in optical flows. *Computer Vision, Graphics, and Image Processing*, 22:239–259, 1983.

[18] Shahriar Negahdaripour and Berthold K. P. Horn. A direct method for locating the focus of expansion. *Computer Vision, Graphics, and Image Processing*, 46:303–326, 1989.

[19] Wilhelm Burger and Bir Bhanu. Estimating 3-D egomotion from perspective image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1040–1058, 1990.

[20] Patrice Y. Simard and Guy E. Mailloux. Vector field restoration by the method of convex projections. *Computer Graphics and Image Processing*, 52:360–385, 1990.

FIGURE CAPTIONS

Fig. 1 Schematic showing three level curves with associated tangent and curvature vectors.

Fig. 2 Illustration of a smooth approximation to a unit step function with edge along the x-axis. Also illustrated are several normal vectors to this surface.

Fig. 3 Local coordinate system imposed onto each 2x2 neighborhood.

Fig. 4 Tangent window pixel values for a smooth edge passing through the center of the window at an angle of $40°$.

Fig. 5 Graph showing results of tangent calculations for an smoothed edge (see Fig. 4) passing through the center of a 9×9 tangent window at an angle of $40°$. Noise was introduced by adding a sequence of computer generated zero-mean uncorrelated Gaussian random variables. The tangent direction (in degrees), normalized error (E_N), and normalized contrast score (C_N) are displayed as functions of the signal to noise ratio. The error bars mark one standard deviation.

Fig. 6 Graph showing results of tangent calculations for a sinusoid (see Eq. 5) passing through the center of a 19×19 tangent window at an angle of $40°$. Noise was introduced by adding a sequence of computer generated zero-mean uncorrelated Gaussian random variables. The tangent direction (in degrees), normalized error (E_N), and normalized contrast score (C_N) are displayed as functions of the signal to noise ratio. The error bars mark one standard deviation.

Fig. 7 Illustration of placement of curvature center point P_c.

Fig. 8 Illustration of the first orthogonality formulation for curvature calculation.

Fig. 9 Illustration of the second orthogonality formulation for curvature calculation.

Fig. 10 Calculated mean curvature as a function of noise-free ("true") curvature and standard deviation (1°, 3°, or 5°) of added Gaussian noise. The results using the first curvature formulation (Section 4.2) are marked by pluses, and the results using the second curvature formulation (Section 4.3) are marked by triangles. A 3x3 tangent grid with a 10 pixel internal spacing was used for these calculations.

Fig. 11 Normalized infinity error, λ/n (see Eq. 22), as a function of noise-free ("true") curvature and standard deviation (1°, 3°, or 5°) of added Gaussian noise. A 3x3 tangent grid with a 10 pixel internal spacing was used for these calculation. The angle of the noise-free curvature vector with respect to this grid was fixed at 57°.

Fig. 12 Illustration of two shallow curves that are indistinguishable in the presence of noise. The curvature vector direction in such situations has meaning only modulo 180°.

Fig. 13 Image of an inked fingerprint. Notice the missing hole on the right hand side and the poor contrast at the top.

Fig. 14 Tangent directions calculated on a 10 row by 8 column grid using a 19 row by 15 column tangent averaging window. Unmarked regions indicate either the contrast or the consistency is poor.

Fig. 15 Result of one pass of the directional filter. The tangent window was 19 rows by 15 columns. Regions were left unprocessed if the normalized contrast score was less than 0.07 or if the normalized consistency error was larger than 0.7.

Fig. 16 Result after 8 passes of directional filter. Each of the first 6 iterations only modified pixels untouched by preceding passes. The last two passes modified low contrast regions using a 9 row by 7 column tangent window. The smaller window allows for the capture of tangents in high curvature regions.

Fig. 17 Scanning electron microscope image of a membrane. Dark areas are pores in the membrane.

Fig. 18 Membrane image after two passes of directional filter using 15 row by 11 column tangent window. Thresholding this image gives pore volume fraction in agreement with results from manual inspection.

Fig. 19 Fingerprint overlaid with results from curvature calculation. Points with curvature radius of less than 150 pixels are marked with a '+', and a radial line is drawn from the point to the calculated curvature center, marked with a '$\circ$'.

Fig. 20 Curvature magnitude contours (units: $0.01 \times (\text{pixels})^{-1}$) overlaid on the fingerprint image. The high curvature values correctly mark the fingerprint core in the center of the image and the deltas below on either side.

Fig. 21 Section of a printed circuit board overlaid with calculated curvature. High curvature locates circular pads and resistor elements.

Fig. 22 (a) Radiograph of a steel specimen with 4 oval slots (which simulate flaws). (b) Curvature overlay. High curvature locates 3 of the 4 slots.

TABLE CAPTIONS

Table 1 Experimental results of tangent calculation for an smoothed edge passing through the center of the tangent window at an angle of $40°$. Results in (a) are from a 9×9 window, (b) from a 19×19 window. Noise was introduced by adding a sequence of computer generated zero-mean uncorrelated Gaussian random variables.

Table 2 Experimental results of tangent calculation for a sinusoidal wave passing through the center of the tangent window at an angle of $40°$. Results in (a) are from a 9×9 window, (b) from a 19×19 window. Noise was introduced by adding a sequence of computer generated zero-mean uncorrelated Gaussian random variables.

Table 3 Curvature calculation statistics at various curvature levels and noise magnitudes. Noise was introduced by rotating the tangent directions using a sequence of computer generated zero-mean uncorrelated Gaussian random variables. The curvature units are pixels^{-1}, direction units are degrees.

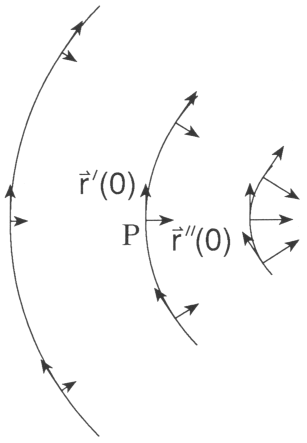


Figure 1: Schematic showing three level curves with associated tangent and curvature vectors.

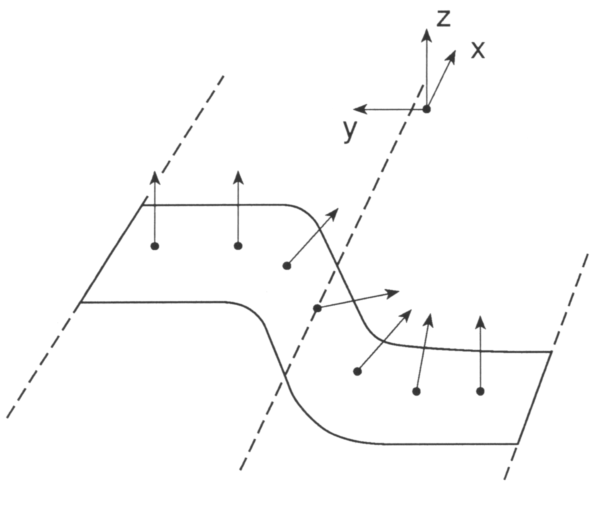


Figure 2: Illustration of a smooth approximation to a unit step function with edge along the x-axis. Also illustrated are several normal vectors to this surface.

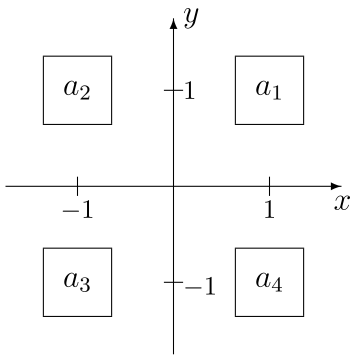


Figure 3: Local coordinate system imposed onto each 2x2 neighborhood.

80	80	80	80	80	80	81	84	95
80	80	80	80	80	81	85	100	150
80	80	80	80	81	86	107	157	172
80	80	80	82	88	116	161	173	175
80	81	82	91	128	165	174	175	176
81	83	95	140	168	174	176	176	176
84	99	149	170	175	176	176	176	176
106	156	171	175	176	176	176	176	176
161	172	175	176	176	176	176	176	176

Figure 4: Tangent window pixel values for a smooth edge passing through the center of the window at an angle of 40°.

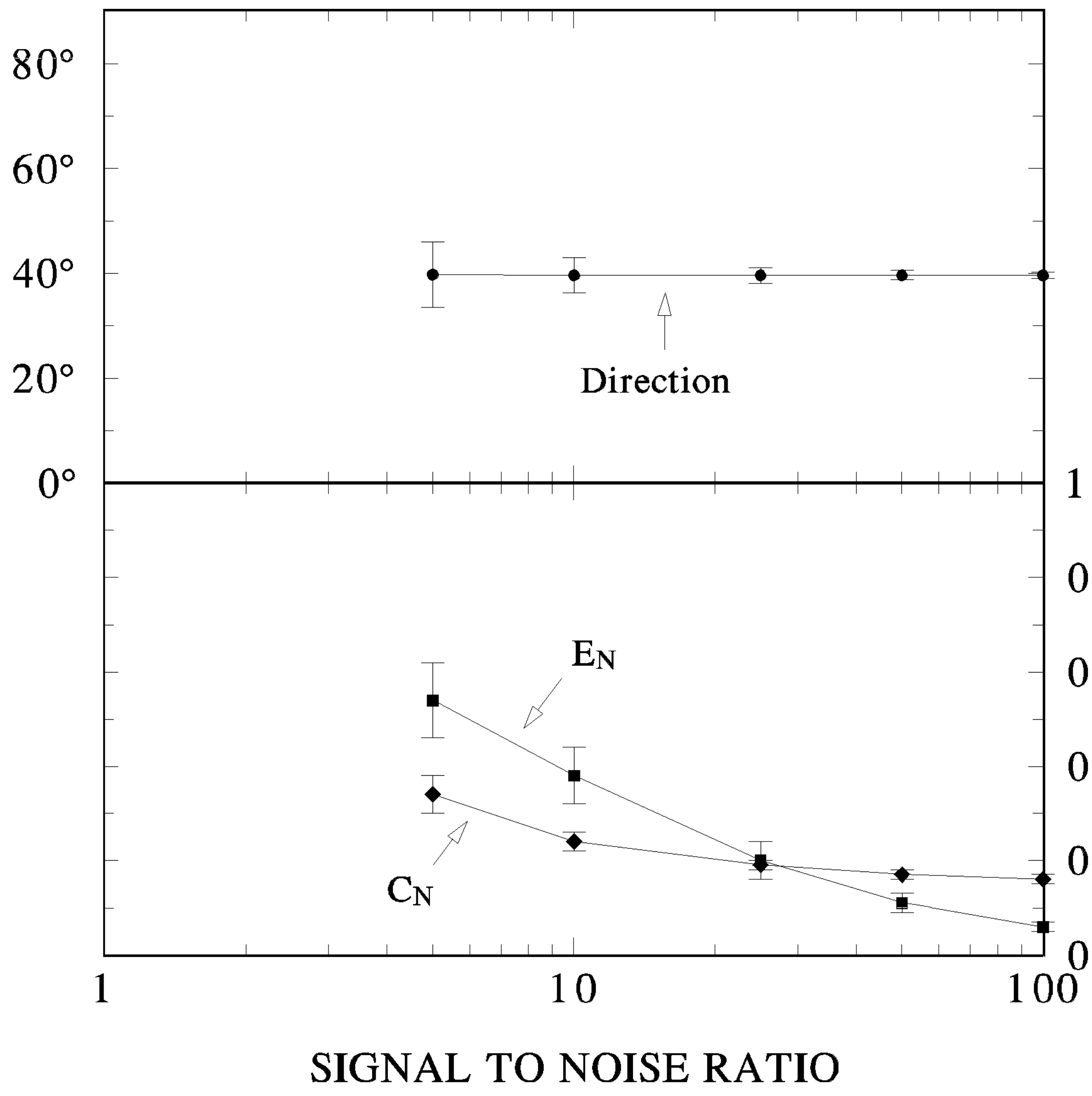


Figure 5: Graph showing results of tangent calculations for an smoothed edge (see Fig. 4) passing through the center of a 9×9 tangent window at an angle of 40°. Noise was introduced by adding a sequence of computer generated zero-mean uncorrelated Gaussian random variables. The tangent direction (in degrees), normalized error (E_N), and normalized contrast score (C_N) are displayed as functions of the signal to noise ratio. The error bars mark one standard deviation.

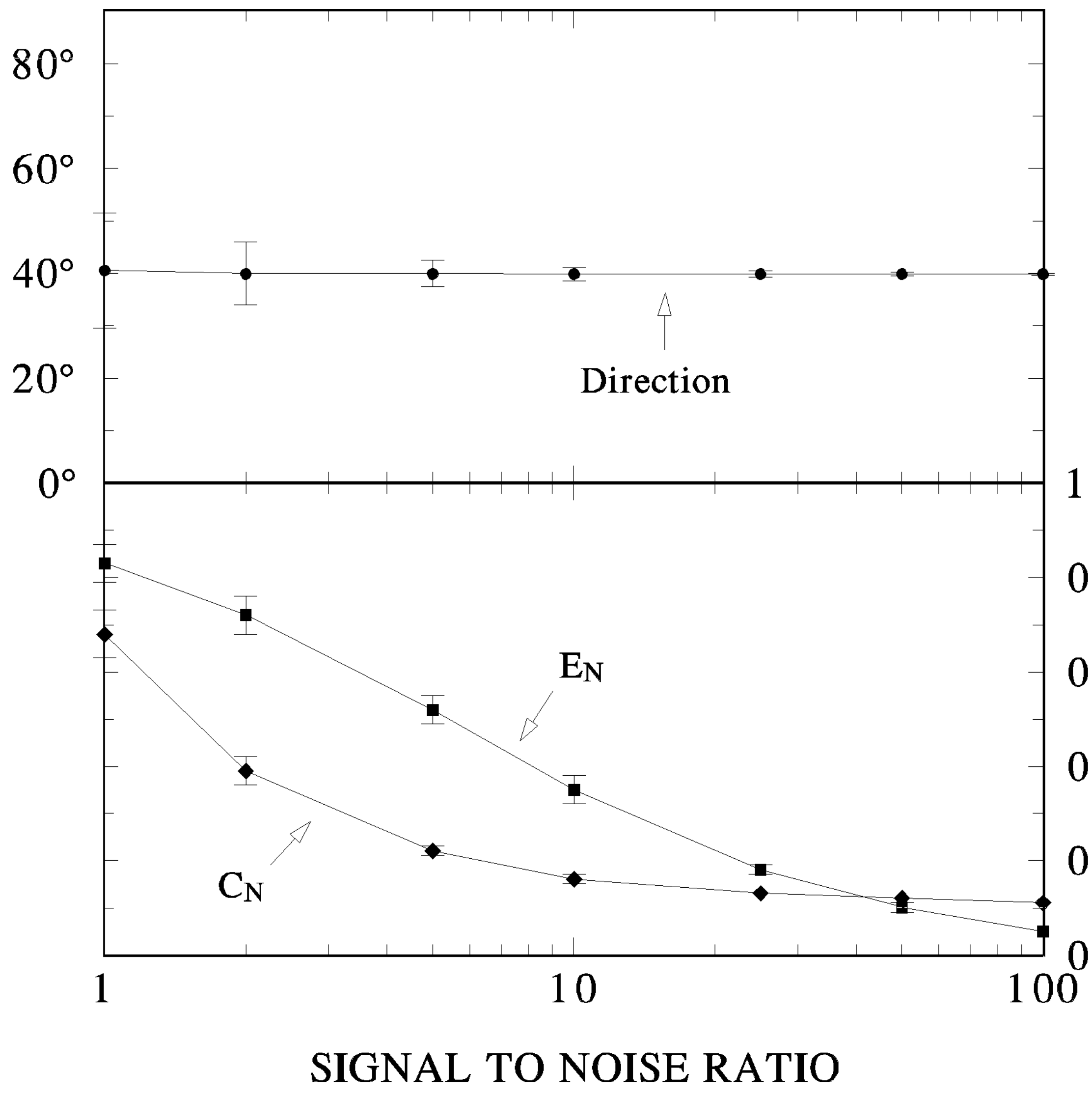


Figure 6: Graph showing results of tangent calculations for a sinusoid (see Eq. 5) passing through the center of a 19×19 tangent window at an angle of 40°. Noise was introduced by adding a sequence of computer generated zero-mean uncorrelated Gaussian random variables. The tangent direction (in degrees), normalized error (E_N), and normalized contrast score (C_N) are displayed as functions of the signal to noise ratio. The error bars mark one standard deviation.

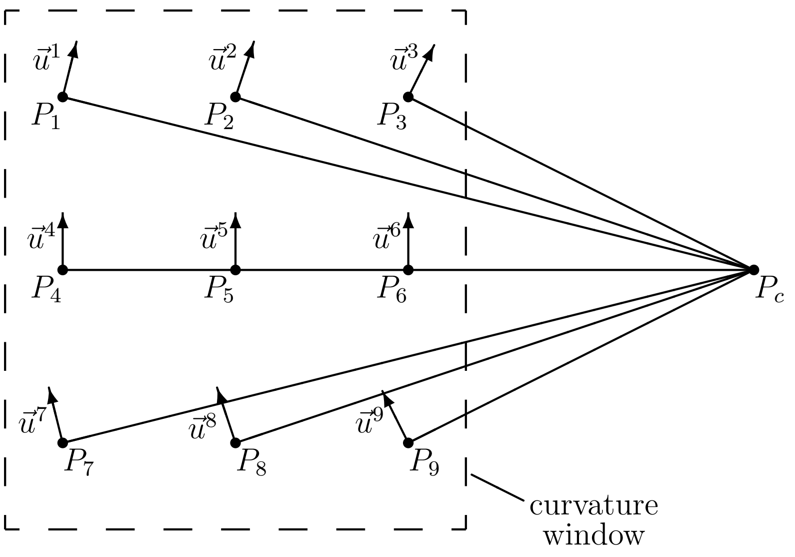


Figure 7: Illustration of placement of curvature center point P_c.

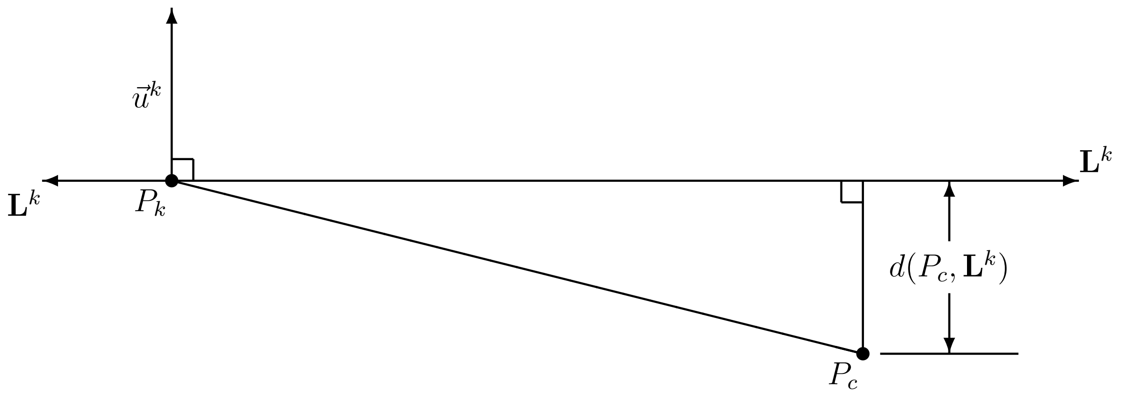


Figure 8: Illustration of the first orthogonality formulation for curvature calculation.

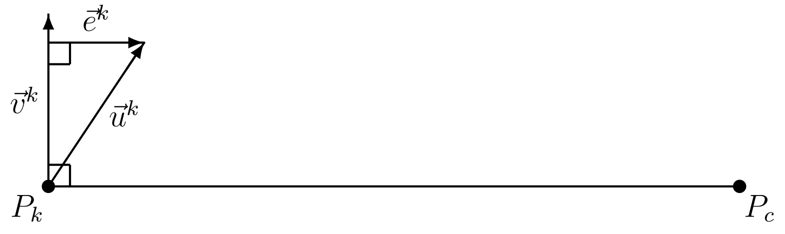


Figure 9: Illustration of the second orthogonality formulation for curvature calculation.

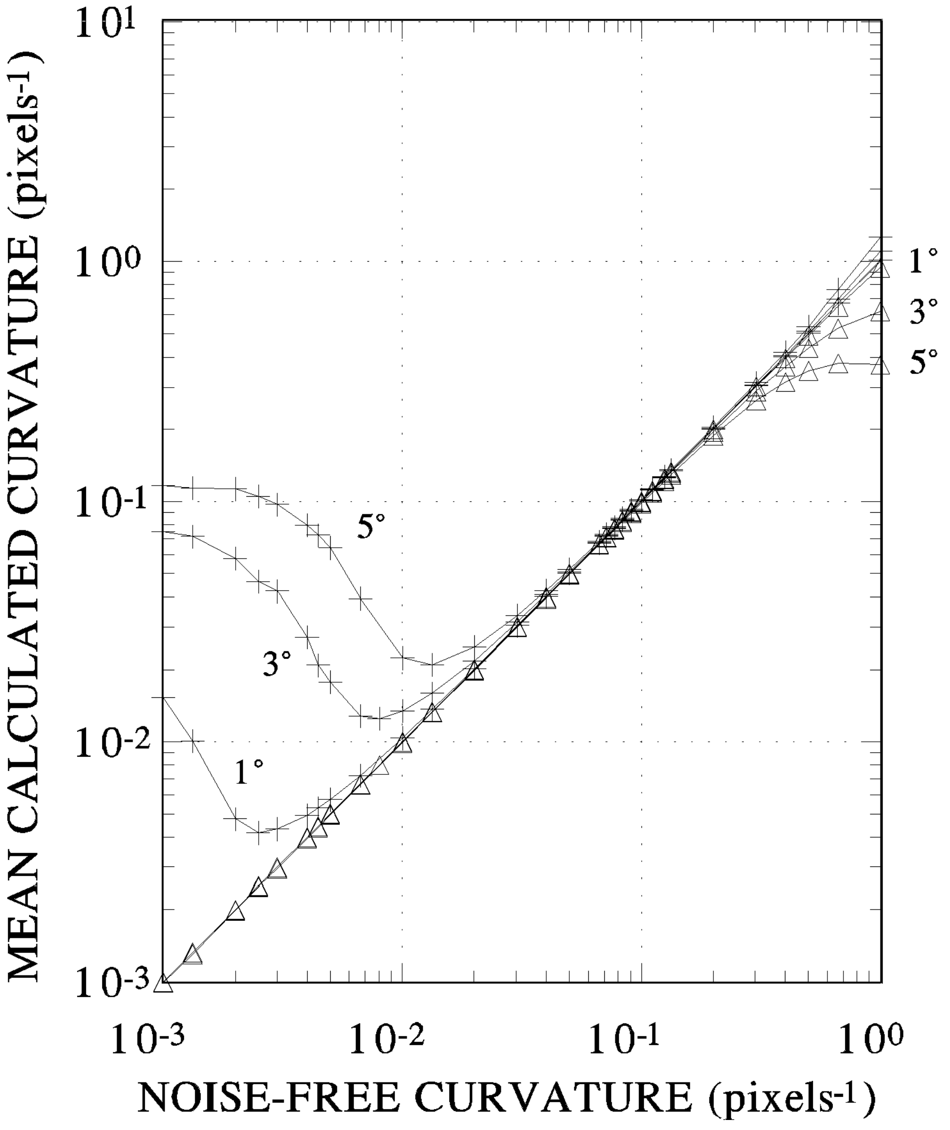


Figure 10: Calculated mean curvature as a function of noise-free ("true") curvature and standard deviation (1°, 3°, or 5°) of added Gaussian noise. The results using the first curvature formulation (Section 4.2) are marked by pluses, and the results using the second curvature formulation (Section 4.3) are marked by triangles. A 3x3 tangent grid with a 10 pixel internal spacing was used for these calculations.

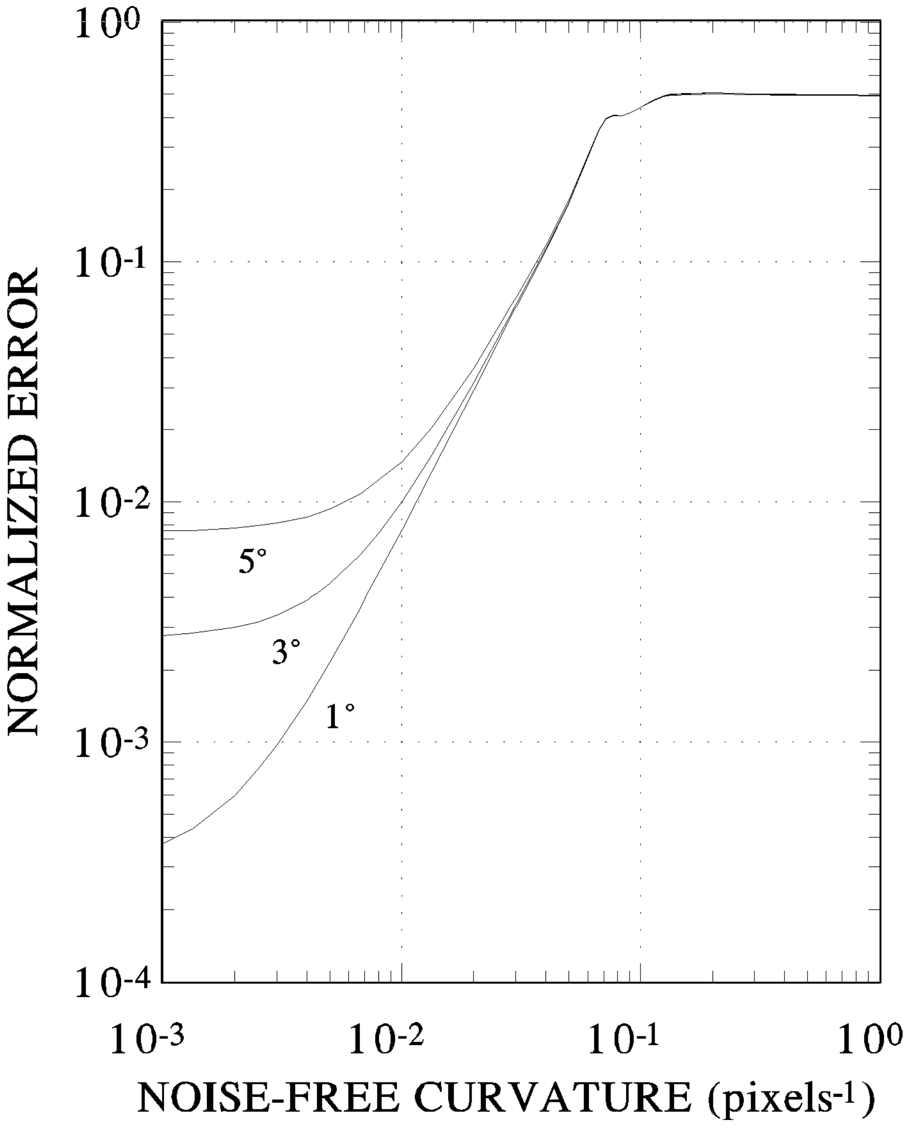


Figure 11: Normalized infinity error, λ/n (see Eq. 22), as a function of noise-free ("true") curvature and standard deviation (1°, 3°, or 5°) of added Gaussian noise. A 3x3 tangent grid with a 10 pixel internal spacing was used for these calculation. The angle of the noise-free curvature vector with respect to this grid was fixed at 57°.

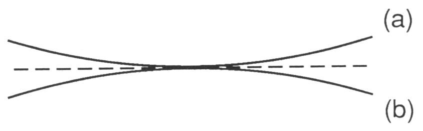


Figure 12: Illustration of two shallow curves that are indistinguishable in the presence of noise. The curvature vector direction in such situations has meaning only modulo 180°.

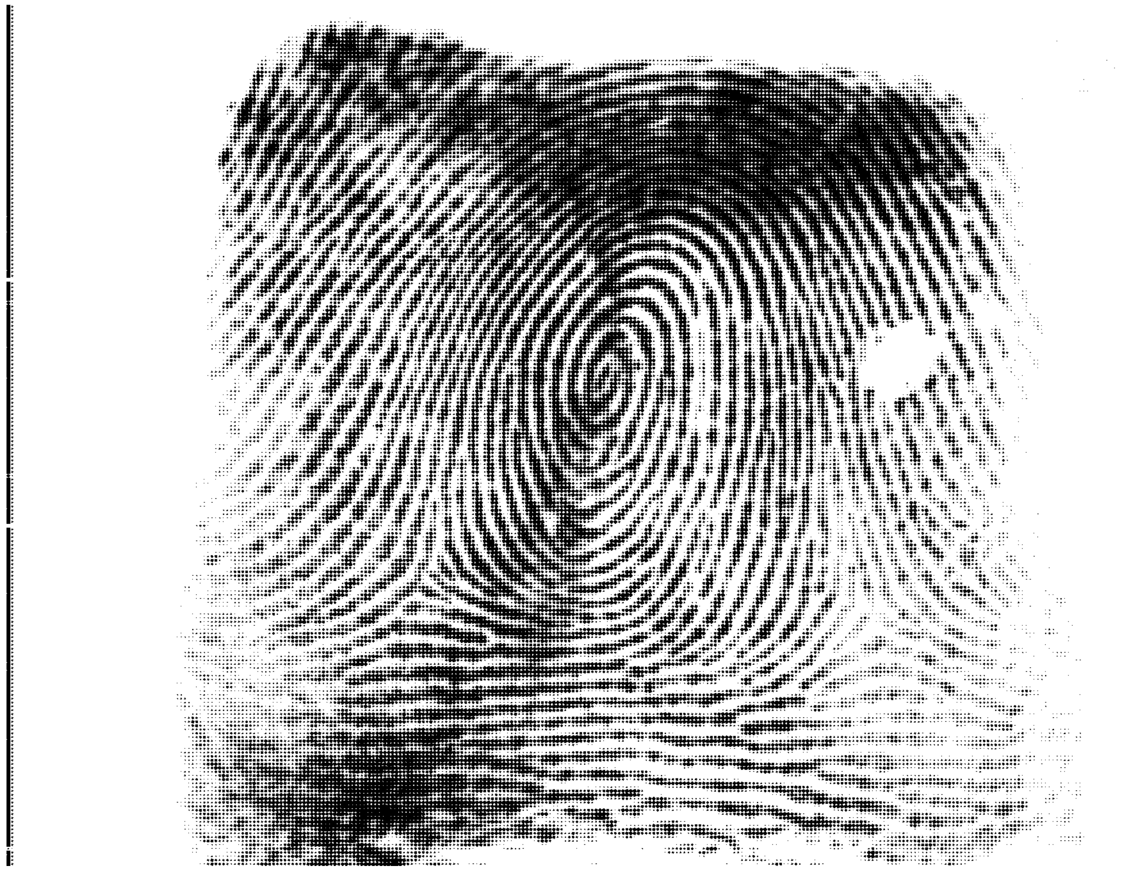


Figure 13: Image of an inked fingerprint. Notice the missing hole on the right hand side and the poor contrast at the top.

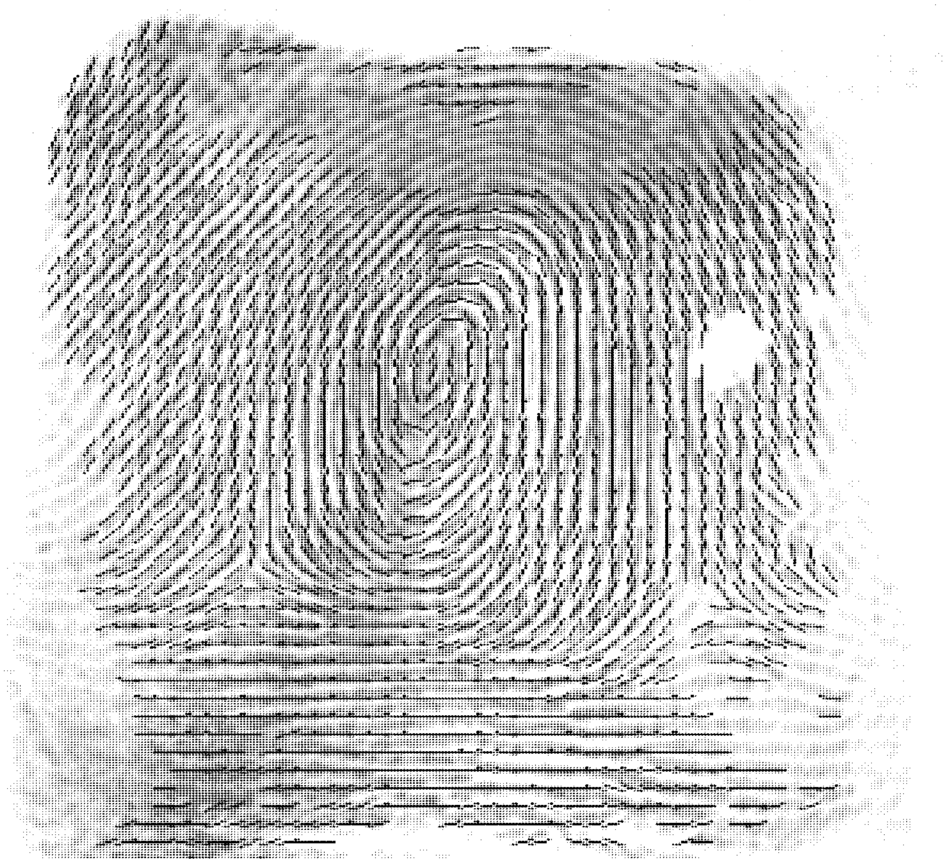


Figure 14: Tangent directions calculated on a 10 row by 8 column grid using a 19 row by 15 column tangent averaging window. Unmarked regions indicate either the contrast or the consistency is poor.

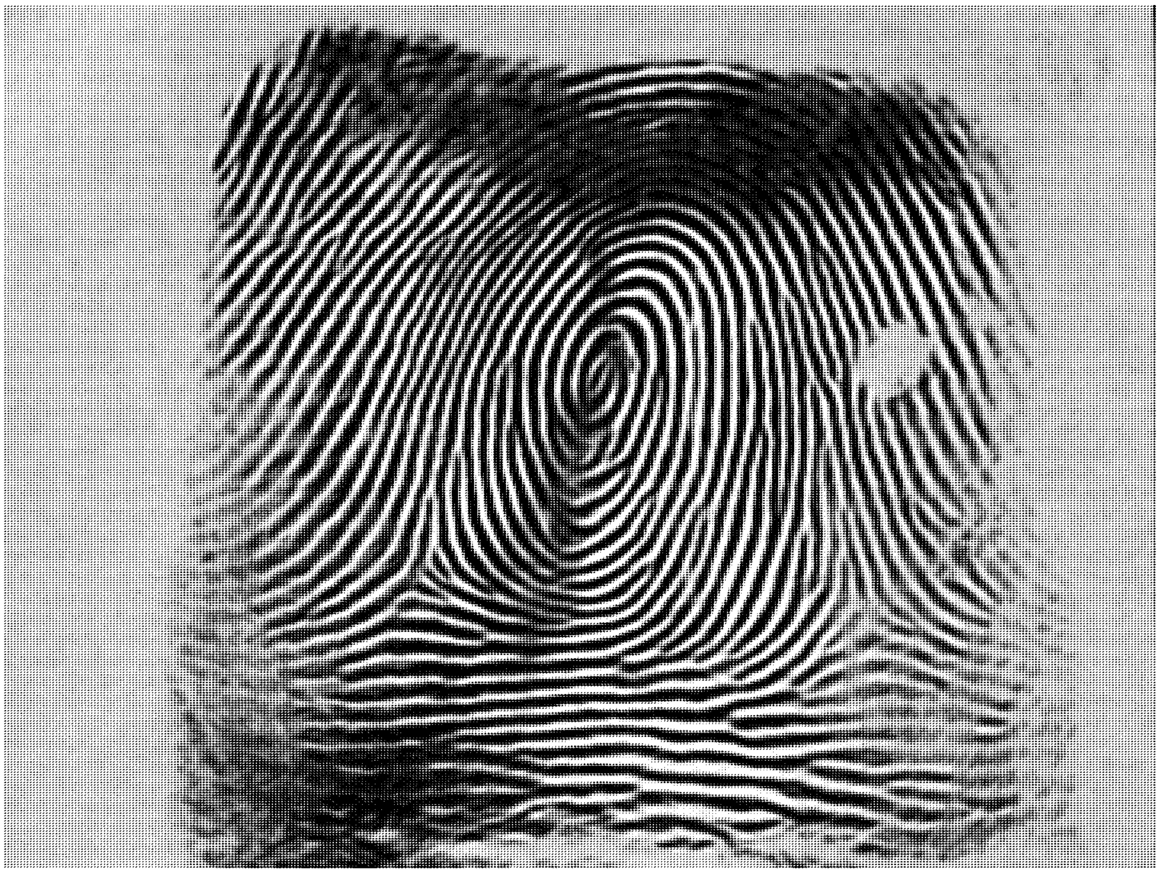


Figure 15: Result of one pass of the directional filter. The tangent window was 19 rows by 15 columns. Regions were left unprocessed if the normalized contrast score was less than 0.07 or if the normalized consistency error was larger than 0.7.

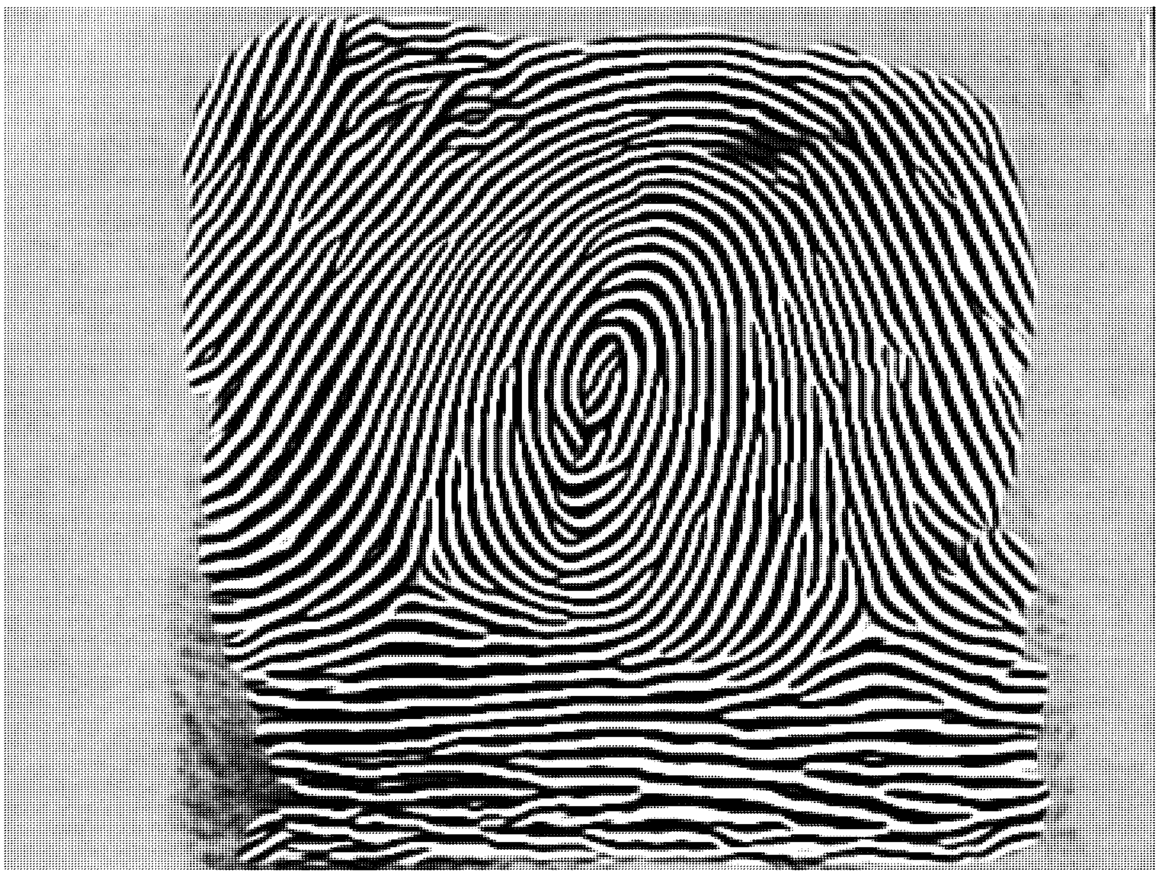


Figure 16: Result after 8 passes of directional filter. Each of the first 6 iterations only modified pixels untouched by preceding passes. The last two passes modified low contrast regions using a 9 row by 7 column tangent window. The smaller window allows for the capture of tangents in high curvature regions.

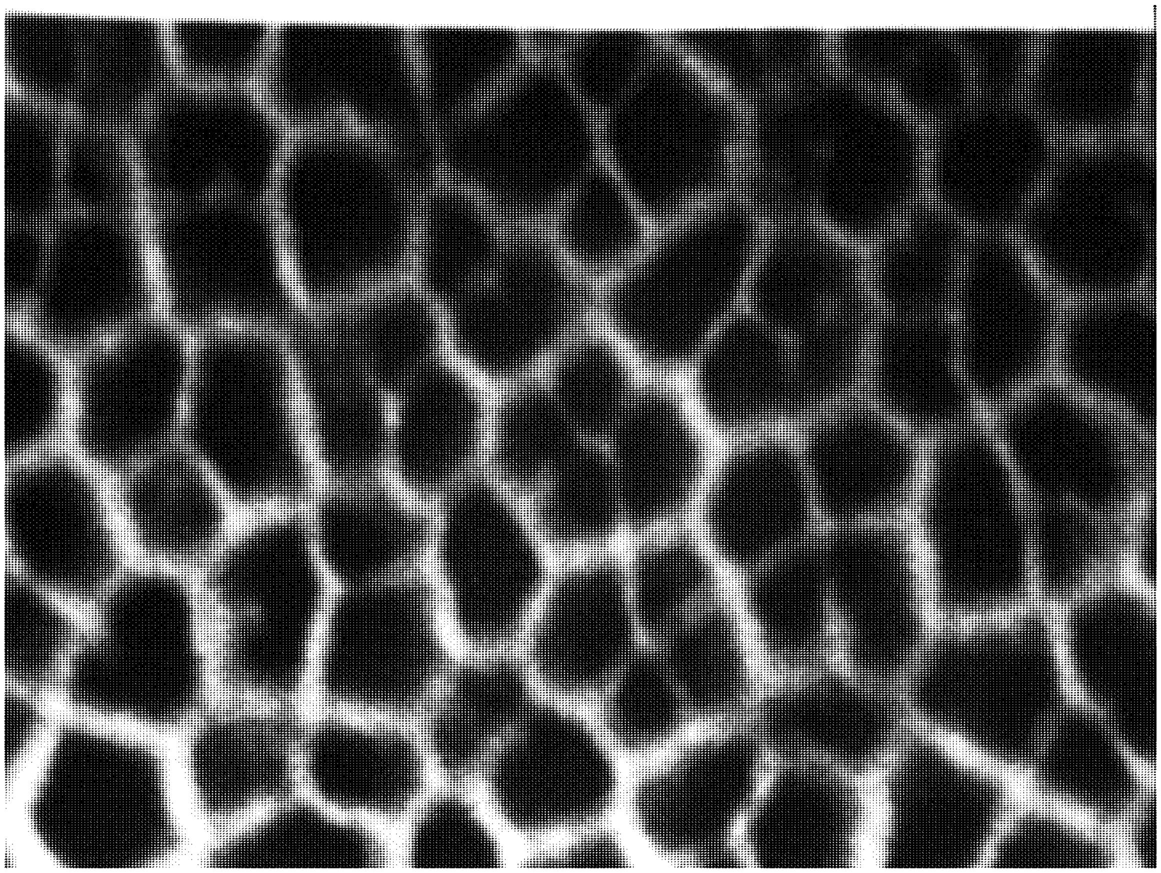


Figure 17: Scanning electron microscope image of a membrane. Dark areas are pores in the membrane.

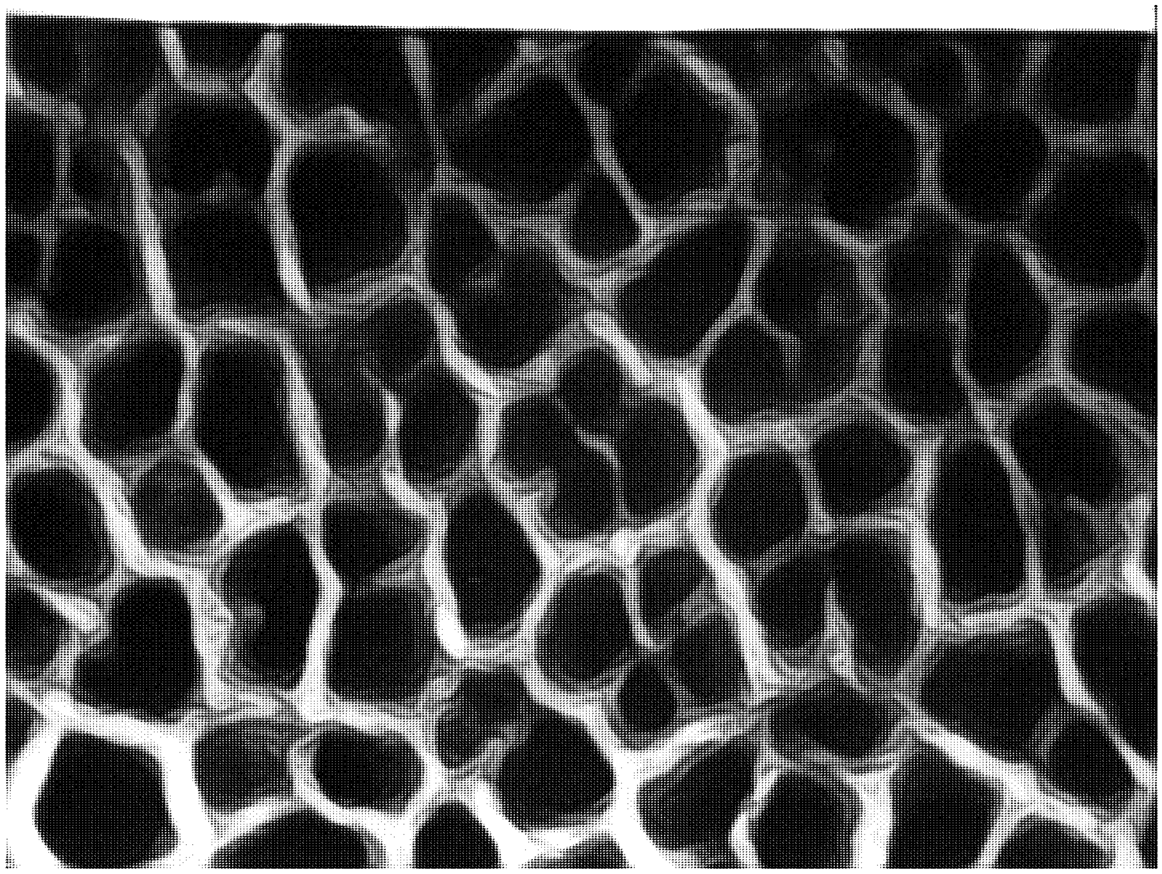


Figure 18: Membrane image after two passes of directional filter using 15 row by 11 column tangent window. Thresholding this image gives pore volume fraction in agreement with results from manual inspection.

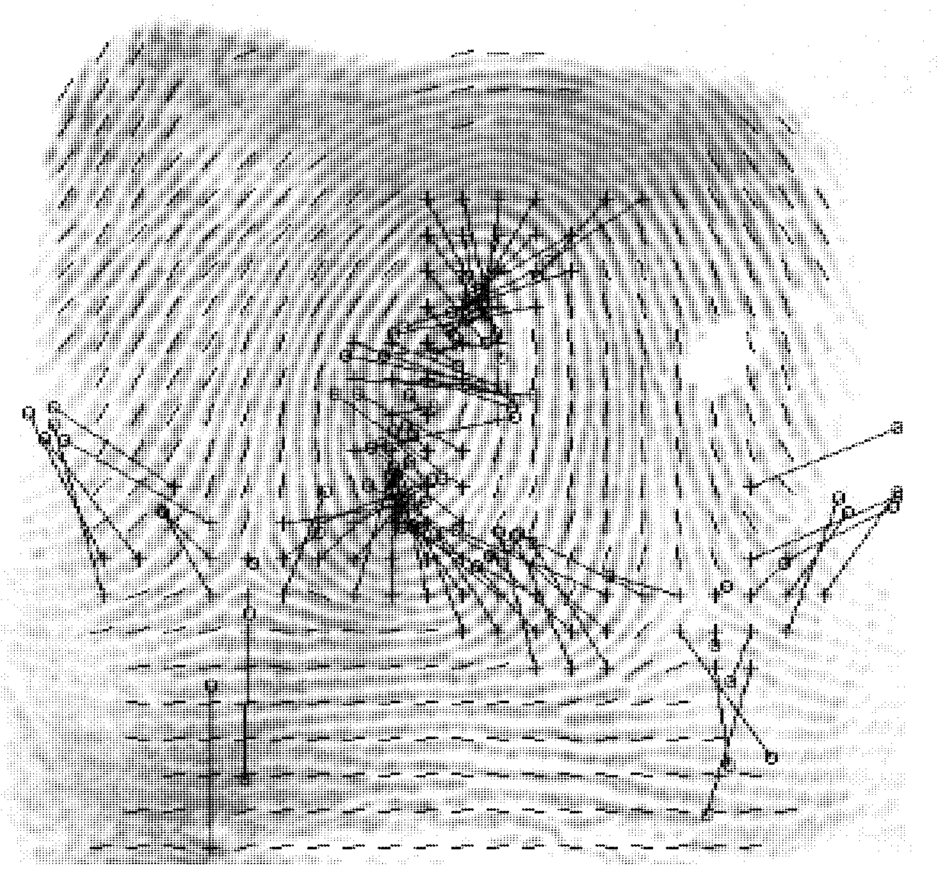


Figure 19: Fingerprint overlaid with results from curvature calculation. Points with curvature radius of less than 150 pixels are marked with a '+', and a radial line is drawn from the point to the calculated curvature center, marked with a '○'.

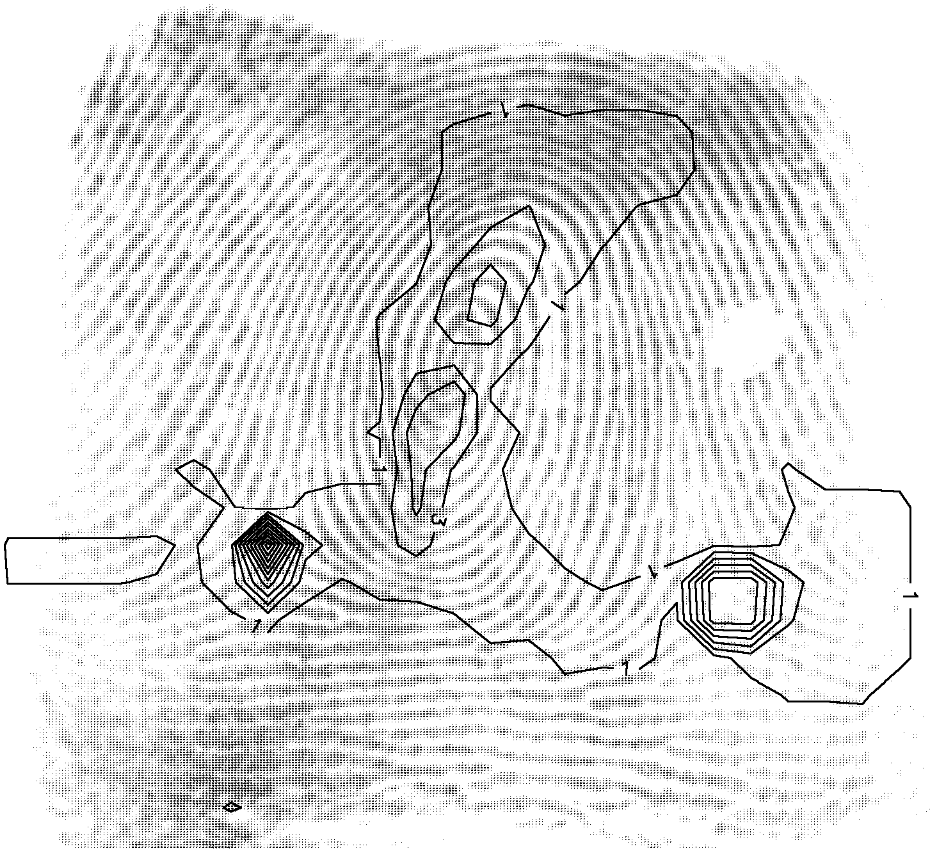


Figure 20: Curvature magnitude contours (units: $0.01 \times (\text{pixels})^{-1}$) overlaid on the fingerprint image. The high curvature values correctly mark the fingerprint core in the center of the image and the deltas below on either side.

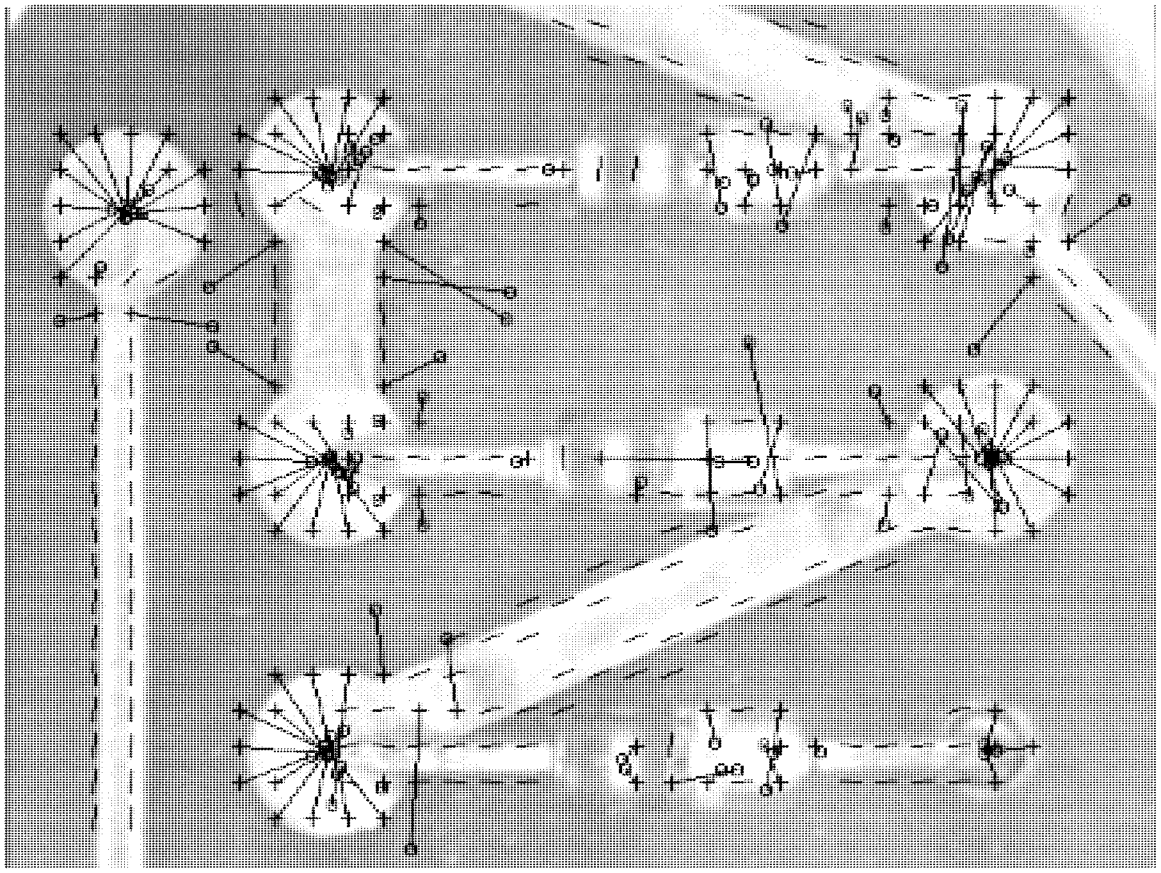
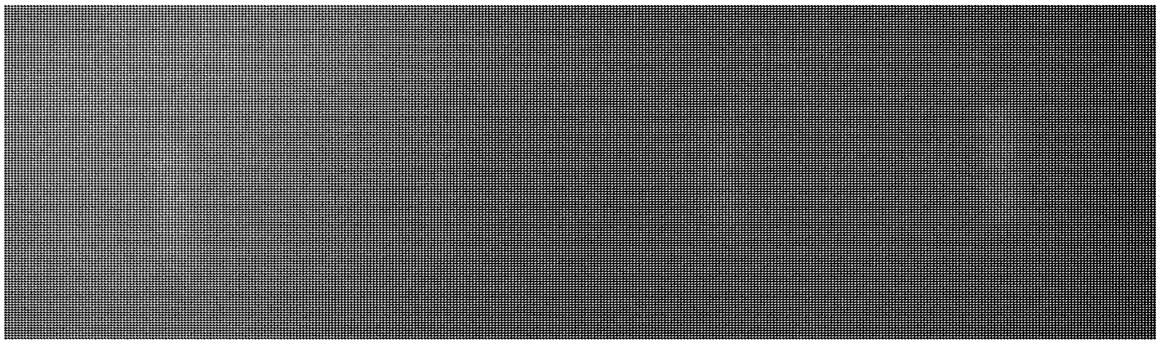
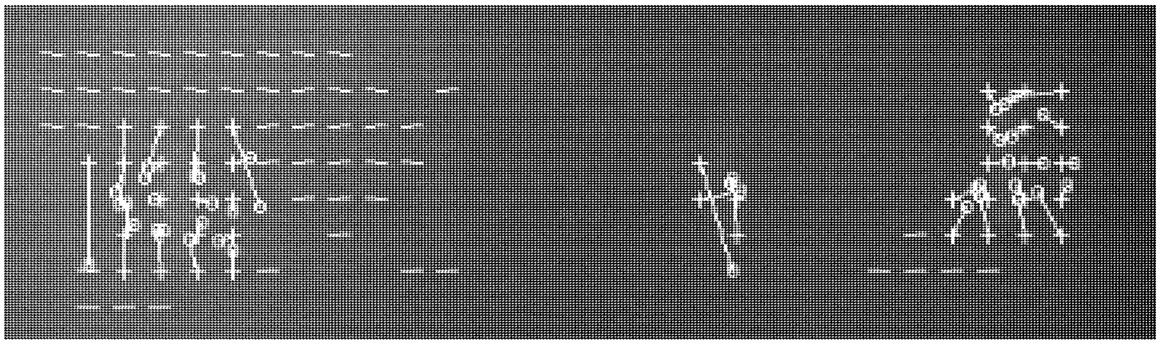


Figure 21: Section of a printed circuit board overlaid with calculated curvature. High curvature locates circular pads and resistor elements.



(a)



(b)

Figure 22: (a) Radiograph of a steel specimen with 4 oval slots (which simulate flaws). (b) Curvature overlay. High curvature locates 3 of the 4 slots.

	Angle		E_N		C_N	
SNR	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
No noise	39.6°	–	0.00	–	0.15	–
100	39.6°	0.57°	0.06	0.01	0.16	0.01
50	39.6°	0.90°	0.11	0.02	0.17	0.01
25	39.6°	1.5°	0.20	0.04	0.19	0.01
10	39.6°	3.4°	0.38	0.06	0.24	0.02
5	39.7°	6.3°	0.54	0.08	0.34	0.04

(a)

	Angle		E_N		C_N	
SNR	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
No noise	39.4°	–	0.00	–	0.07	–
100	39.4°	0.41°	0.14	0.01	0.08	0.00
50	39.4°	0.75°	0.23	0.02	0.09	0.00
25	39.4°	1.5°	0.38	0.03	0.11	0.01
10	39.4°	3.5°	0.61	0.04	0.17	0.01
5	39.6°	6.8°	0.75	0.04	0.28	0.02

(b)

Table 1: Experimental results of tangent calculation for an smoothed edge passing through the center of the tangent window at an angle of 40°. Results in (a) are from a 9×9 window, (b) from a 19×19 window. Noise was introduced by adding a sequence of computer generated zero-mean uncorrelated Gaussian random variables.

	Angle		E_N		C_N	
SNR	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
No noise	39.7°	–	0.00	–	0.11	–
100	39.8°	0.63°	0.05	0.01	0.11	0.00
50	39.8°	0.95°	0.09	0.02	0.12	0.00
25	39.8°	1.5°	0.17	0.03	0.13	0.01
10	39.8°	3.1°	0.34	0.06	0.16	0.01
5	39.9°	5.6°	0.50	0.07	0.22	0.02

(a)

	Angle		E_N		C_N	
SNR	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
No noise	39.9°	–	0.00	–	0.11	–
100	39.8°	0.21°	0.05	0.00	0.11	0.00
50	39.8°	0.34°	0.10	0.01	0.12	0.00
25	39.8°	0.59°	0.18	0.01	0.13	0.00
10	39.8°	1.3°	0.35	0.03	0.16	0.01
5	39.9°	2.5°	0.52	0.03	0.22	0.01
2	39.9°	6.0°	0.72	0.04	0.39	0.03
1	40.5°	11°	0.83	0.04	0.68	0.05

(b)

Table 2: Experimental results of tangent calculation for a sinusoidal wave passing through the center of the tangent window at an angle of 40°. Results in (a) are from a 9×9 window, (b) from a 19×19 window. Noise was introduced by adding a sequence of computer generated zero-mean uncorrelated Gaussian random variables.

		Calculation Results		
Noise-free Curvature	Noise Std. Dev.	Curvature Mean Relative Error (%)	Curvature Std. Dev.	Direction Std. Dev.
0.001	1°	0.2	0.0007	0.3
	3°	-1.8	0.0021	1.0
	5°	-1.8	0.0035	1.7
0.005	1°	-0.1	0.0007	0.3
	3°	-0.2	0.0021	1.0
	5°	-1.2	0.0035	1.7
0.01	1°	-0.1	0.0007	0.3
	3°	-0.3	0.0021	1.0
	5°	-1.2	0.0036	1.7
0.03	1°	-0.1	0.0007	0.4
	3°	-0.4	0.0022	1.1
	5°	-1.2	0.0038	1.8
0.04	1°	0.2	0.0008	0.4
	3°	2.0	0.0027	1.2
	5°	5.	0.0050	2.0
0.05	1°	0.2	0.0008	0.4
	3°	1.5	0.0023	1.3
	5°	4.	0.0039	2.2
0.1	1°	0.1	0.0012	0.8
	3°	0.6	0.0036	2.5
	5°	2.	0.0061	4.1
0.5	1°	0.3	0.027	2.5
	3°	2.	0.089	7.8
	5°	8.	0.22	14.
1.0	1°	0.9	0.11	5.0
	3°	11.	0.87	17.
	5°	27.	2.6	36.

Table 3: Curvature calculation statistics at various curvature levels and noise magnitudes. Noise was introduced by rotating the tangent directions from a sequence of computer generated zero-mean uncorrelated Gaussian random variables. The curvature units are pixels^{-1}, direction units are degrees.

Contents

List of Figures

List of Tables