Direct Gray-Scale Minutiae Detection In Fingerprints

Dario Maio, *Member, IEEE,* and Davide Maltoni

Abstract—Most automatic systems for fingerprint comparison are based on minutiae matching. Minutiae are essentially terminations and bifurcations of the *ridge lines* that constitute a fingerprint pattern. Automatic minutiae detection is an extremely critical process, especially in low-quality fingerprints where noise and contrast deficiency can originate pixel configurations similar to minutiae or hide real minutiae. Several approaches have been proposed in the literature; although rather different from each other, all these methods transform fingerprint images into binary images. In this work we propose an original technique, based on ridge line following, where the minutiae are extracted directly from gray scale images. The results achieved are compared with those obtained through some methods based on image binarization. In spite of a greater conceptual complexity, the method proposed performs better both in terms of efficiency and robustness.

Index Terms—Fingerprints, minutiae, feature extraction, gray scale images, directional image.

1 Introduction

Fingerprint-based identification has been known and used for a very long time [5], [9], [11] and [24]. Owing to their uniqueness and immutability [19], fingerprints are today the most widely used biometric features. Most automatic systems for fingerprint comparison are based on minutiae matching [12], [22], [26] and [36]. *Minutiae*, or Galton's characteristics [9], are local discontinuities in the fingerprint pattern. The American National Standards Institute has proposed a minutiae classification based on four classes: *terminations, bifurcations, trifurcations* (or crossovers) and *undetermined* [1]. In this work we adopt the identification model used by the Federal Bureau of Investigation [36]. This model, adopted in most automatic systems, is based on a two-class minutiae classification: termination and bifurcation. For each minutia we store the membership class, the coordinates and the angle that the tangent to the minutia forms with the horizontal direction (Fig. 1). The problem of automatic minutiae extraction has been thoroughly studied but never completely solved. The main difficulty is that fingerprint quality is often too low[1]; noise and contrast deficiency can produce false minutiae and hide valid minutiae.

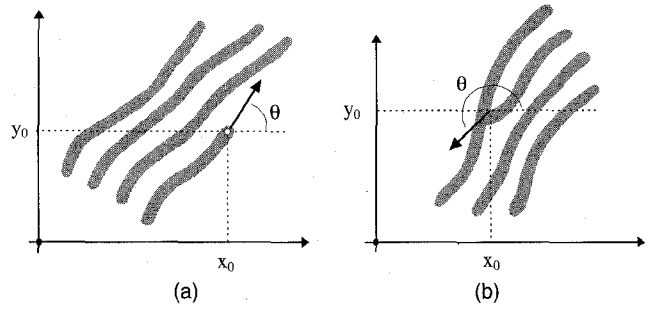


Fig. 1. Minutiae representation. Fig. 1a shows a termination minutia; (x_0, y_0) are the minutia coordinates; θ is the angle that the minutia tangent forms with the horizontal direction. Fig. 1b shows a bifurcation minutia; θ is now defined by means of the termination minutia existing in the complementary image in correspondence with the original bifurcation.

Several approaches to automatic minutiae extraction have been proposed: although rather different from one other, most of these methods transform fingerprint images into binary images through an ad hoc algorithm. The images obtained are submitted to a thinning process which allows for the ridge line thickness to be reduced to one pixel (Fig. 2). Finally, a simple image scan allows for locating the pixels that correspond to minutiae.

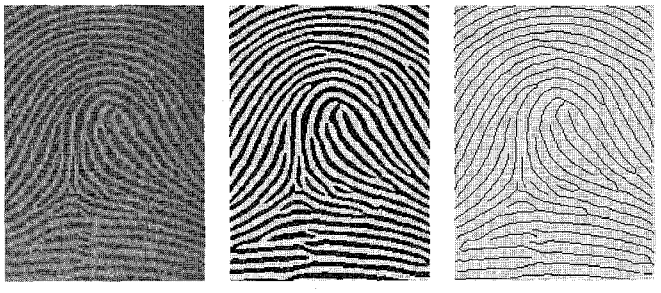
The FBI minutiae reader [29] binarizes the image through a composite approach based on a local thresholding and a "slit comparison" formula that compares pixel alignment along discrete directions. Moayer and Fu [23] proposed a binarization technique based on the iterative application of a laplacian operator and a dynamic threshold. A fuzzy approach to image enhancement and the use of an adaptive threshold, aimed to preserve the same number of 1 and 0 pixels for each neighborhood, form the basis of the binarization technique proposed by Verma, et al. in [31]. O'Gorman and Nickerson present, in [25], a technique for enhancement and binarization based on the convolution of the image with a filter oriented according to the *directional image*. The directional image may be conceived as a

- *D. Maio is with Corso di Laurea in Scienze dell'Informazione, Università di Bologna, via Sacchi 3, 47023 Cesena, Italy and DEIS, CSITE-CNR, Università di Bologna, viale Risorgimento 2, 40136 Bologna, Italy. E-mail: dmaio@deis.unibo.it.*
- *D. Maltoni is with DEIS, Università di Bologna, via Sacchi 3, 47023 Cesena, Italy. E-mail: dmaio@deis.unibo.it.*

Manuscript received June 8, 1995; revised Dec. 2, 1996. Recommended for acceptance by R. Kasturi.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96121.

1. The fingerprint acquisition process is rather critical. The most famous technique, known as the "ink technique", often produces images including regions which miss some information due to excessive inkiness or to ink deficiency. The techniques which use optical prisms [10] and holograms [15] require a high degree of accuracy during the acquisition process, that is, the finger pressure on the optical surface must be adequate. Furthermore, in some subjects, especially manual workers and elderly people, the prominence of the ridge lines can be considerably lower and the fingerprint pattern can be unreadable.

matrix whose elements represent the tangent direction to the ridge lines of the original image. E. Székely and V. Székely [30], starting from binary images, suggest a minutiae detection technique based on the computation of the directional image divergence. Sherlock, et al. [27][28] define a technique for fingerprint enhancement and binarization, which performs a frequency-domain filtering through position-dependent filters. Coetzee and Botha [6] propose a binarization technique based on the use of the edges in conjunction with the gray scale image. In their work, an interesting technique for improving the binary image obtained is introduced.



(a) (b) (c)

Fig. 2. Fig. 2a shows a fingerprint gray scale image; Fig. 2b shows the image obtained after a binarization process of the image 2a. Fig. 2c shows the image obtained after a thinning process of the image 2b.

Mehtre [20] describes a complete Automating Fingerprint Identification System (AFIS) which includes a useful enhancement/restoration step to control noise in the starting gray scale image, and implements the thinning step by means of a parallel algorithm. In the same work an exhaustive discussion about noise in fingerprint images can be found. In the work [35] by Weber, the gray scale fingerprints are enhanced by a bandpass filtering in the frequency domain and binarized via a local threshold; instead of using a conventional thinning the author proposes an algorithm which detects the minutiae starting from the thick-ridges in the binary image. M.T. Leung, et al. introduce in [17] a neural network based approach to the minutiae detection where a multilayer perceptron analyzes the output of a rank of Gabor's filters applied to the gray scale image. Another neural network schema is presented in [18]; in this work a tree layer perceptron is trained to extract the minutiae starting from skeletonized binary images. Among the techniques proposed, some provide good results when applied to high-quality fingerprints, but they are not robust enough in the presence of noise. Post-processing techniques, based on simple structural considerations, [13], [37] can be used to discard many false minutiae, thus giving better results.

In this work we present a direct gray scale minutiae detection approach (i.e. without binarization and thinning). In the field of image processing, some approaches to direct gray scale feature extraction have been proposed, see for instance [7], [16], [32] and [33]. We have chosen to extract the features directly from the gray scale image without binarization and thinning for the following reasons:

- A lot of information may be lost during the binarization process.
- Binarization and thinning are time-consuming.
- The binarization techniques which we experimented proved to be unsatisfactory when applied to low-quality images.

The basic idea of our method is to follow the ridge lines on the gray scale image, by "sailing" according to the fingerprint directional image. A set of starting points is determined by superimposing a square-meshed grid on the gray scale image. For each starting point, the algorithm keeps following the ridge lines until they terminate or intersect other ridge lines (minutiae detection). A labeling strategy is adopted to examine each ridge line only once and locate the intersections between ridge lines.

In Section 2 we present the basic definitions and sketch the ridge line following algorithm. In Section 3 we show how the ridge line following algorithm can be used for automatic minutiae detection. Section 4 presents the results obtained and compares our approach with the techniques described in [23], [25], [29] and [31]. Finally, in Section 5 some conclusions are drawn.

2 Ridge Line Following

Let $\mathbf{I}$ be an $a \times b$ gray scale image with g gray levels, and $gray(i, j)$ be the gray level of pixel (i, j) of $\mathbf{I}$, $i = 1, \ldots a$, $j = 1, \ldots b$. Let $z = S(i, j)$ be the discrete surface corresponding to the image $\mathbf{I}$: $S(i, j) = gray(i, j)$, $i = 1, \ldots a$, $j = 1, \ldots b$. By associating bright pixels with gray levels near zero and dark pixels with gray levels near $g - 1$, the fingerprint ridge lines (appearing dark in $\mathbf{I}$) correspond to surface ridges, and the spaces between the ridge lines (appearing bright in $\mathbf{I}$) correspond to surface ravines (Fig. 3).

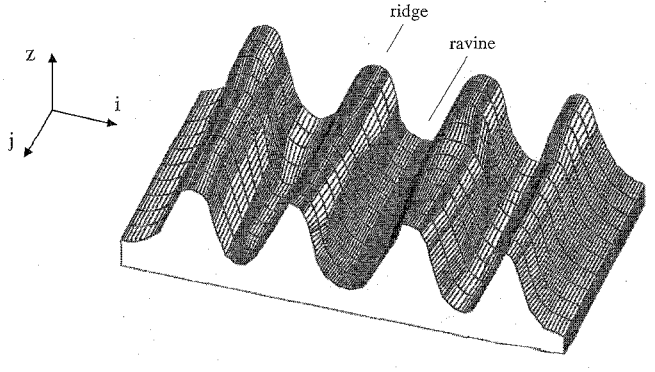


Fig. 3. A surface S, corresponding to a small area of a fingerprint is shown (the surface is depicted as continuous due to representation problems).

From a mathematical point of view, a ridge line is defined as a set of points which are local maxima along one direction. The ridge line extraction algorithm attempts to locate, at each step, a local maximum relative to a section orthogonal to the ridge direction. By connecting the consecutive maxima, a polygonal approximation of the ridge line can be obtained.

Let (i_s, j_s) be a local maximum of a ridge line of **I**, and φ_0 be the direction of the tangent to the ridge line in (i_s, j_s); a pseudo-code version of the ridge line following algorithm is:

```
ridge line following(i_s, j_s, φ_0)
  { end := false ;
    (i_c , j_c ) := (i_s , j_s );
    φ_c := φ_0 ;
    while (¬ end)
      { (i_t , j_t ) := (i_c , j_c ) + μ pixel
          along direction φ_c ;
        Ω := section set centered in
            (i_t, j_t) with direction φ_c+π/2
            and length 2σ+1;
        (i_n , j_n ) := local maximum over Ω ;
        store (i_n , j_n ) ;
        end := check stop criteria on
            (i_c , j_c ), (i_t , j_t ), (i_n , j_n ) ;
        ( i_c , j_c ) := ( i_n , j_n ) ;
        φ_c := tangent direction in (i_c , j_c
);
      }
  }
```

The algorithm runs until a stop criterion becomes true. At each step it computes a point (i_t, j_t), moving μ pixels from (i_c, j_c) along direction φ_c. Then, it computes the *section set* Ω as the set of points belonging to the section segment lying on the ij-plane and having median point (i_t, j_t), direction orthogonal to φ_c and length $2\sigma + 1$. A new point (i_n, j_n), belonging to the ridge line, is chosen among the local maxima of the set Ω. The point (i_n, j_n) becomes the current point (i_c, j_c) and a new direction φ_c is computed (Fig. 4). μ and σ are parameters whose optimal value can be determined according to the average thickness of the image ridge lines.

The main algorithm steps, namely, sectioning and maximum determination, computation of the direction φ_c and testing of the stop criteria, are discussed in detail in the following sub-sections.

2.1 Sectioning and Maximum Determination

The sectioning of the surface S corresponding to the image **I** can be achieved by intersecting S with a cutting plane parallel to the z direction. The section set $\Omega((i_t, j_t), \phi, \sigma)$ centered in (i_t, j_t), with direction $\phi = \varphi_c + \pi/2$, and length $2\sigma + 1$ pixels (in the following simply Ω), is defined as:

$$\Omega = \{ (i, j) \mid (i, j) \in \mathbf{I}, (i, j) \in segment((i_{start}, j_{start}), (i_{end}, j_{end})) \}$$

$$(i_{start}, j_{start}) = (round (i_t - \sigma \cdot cos\ \phi), round (j_t - \sigma \cdot sin\ \phi))$$

$$(i_{end}, j_{end}) = (round (i_t + \sigma \cdot cos\ \phi), round (j_t + \sigma \cdot sin\ \phi))$$

$$round(x) = \begin{cases} \lfloor x + 0.5 \rfloor & \text{if } x \geq 0 \\ \lfloor x + 0.5 \rfloor & \text{otherwise} \end{cases}$$

$segment((i_{start}, j_{start}), (i_{end}, j_{end}))$ is the set of points belonging to the discrete segment whose extremes are (i_{start}, j_{start}) and (i_{end}, j_{end}). By sorting the points of Ω from (i_{start}, j_{start}) to (i_{end}, j_{end}), we obtain: $(i_1, j_1) \equiv (i_{start}, j_{start})$, (i_2, j_2),... $(i_m, j_m) \equiv (i_{end}, j_{end})$, $m \approx 2\sigma+1$. A graphic representation of a section is depicted in Fig. 5.

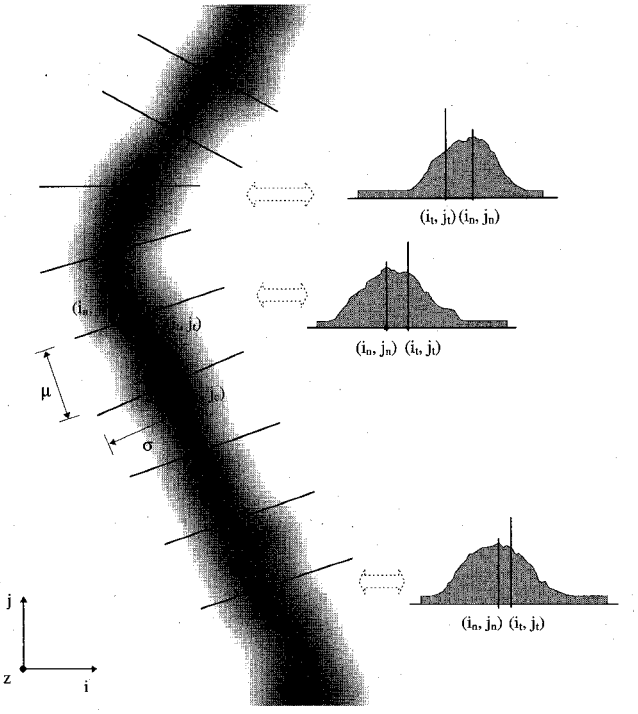


Fig. 4. Some ridge line following steps. On the right, some sections are shown.

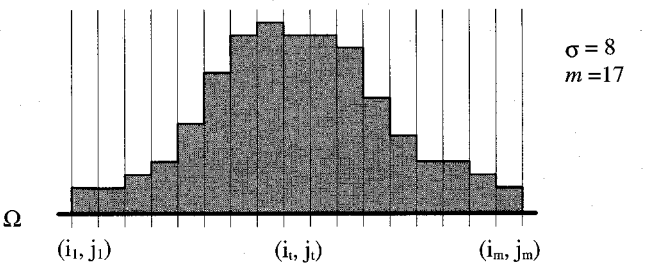


Fig. 5. A section is graphically represented by reporting the gray levels $gray(i_k, j_k)$ of the pixels (i_k, j_k), k=1,...*m* belonging to Ω.

Determining a local maximum of the section set Ω is a very important step. In principle, the maximum can be computed simply by comparing the gray levels of the points belonging to Ω. Noise and contrast deficiency make this technique unsuitable, except for excellent-quality images. Fig. 6 shows two sections which intersect five and six ridges, respectively.

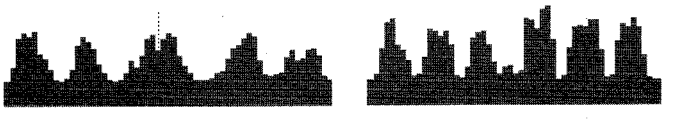


Fig. 6. The figure shows two sections, belonging to regions with different ridge line density. The dotted line denotes the point where a ridge center (hence, a local maximum) should exist. On the contrary, the image noise originates, in that point, a local minimum which produces a typical volcano silhouette.

In both sections we are able to easily locate the ridges, but detecting the corresponding maxima is not straightforward; sometimes, in the middle of a ridge (where a local maximum should exist) there is a local minimum which produces a typical volcano silhouette. Hereafter we describe an approach, aimed at regularizing the section silhouette, which makes the determination of the local maxima more reliable. During the ridge line following, each time a new section is determined we regularize its silhouette by means of two steps:

a1. The first step is based on a local average of the gray levels of the pixels belonging to a number of parallel adjacent sections. This can be obtained by sectioning S with $2h + 1$ parallel planes ($h \geq 0$), distant one pixel from each other, and by computing m local averages to determine the new gray levels. The section produced by the plane $h + 1$ (the central one) originates the section set Ω, but the gray level $\overline{gray}(i, j)$ of each point $(i, j) \in \Omega$ is computed as the average of the gray levels of corresponding points over the $2h + 1$ sections (Fig. 7).

a2. The second step is based on a convolution with a constant mask **d** resembling the gaussian silhouette (Fig. 8). Let $(i_1, j_1),...(i_m, j_m)$ be the points belonging to Ω and $\overline{gray}(i_1, j_1), \ldots \overline{gray}(i_m, j_m)$ be the gray levels computed at step (a1); let d_k, $k = 1, \ldots 2p + 1$, ($p \geq 0$, $d_k \geq 0$, $\Sigma d_k = 1$) be the elements of the mask **d**. The new gray levels $\overline{\overline{gray}}(i_{p+1}, j_{p+1}), \ldots \overline{\overline{gray}}(i_{m-p}, j_{m-p})$ are computed as:

$$\overline{\overline{gray}}(i_k, j_k) = \frac{1}{(2p+1)} \sum_{v=-(p)}^{(p)} d_{p+1+v} \cdot \overline{gray}(i_{k+v}, j_{k+v})$$

$$k = p+1, .., m-(p)$$

The local regularization process described has been divided into two parts for clarity; actually, this processing can be conceived as a convolution of a little portion of the image with a three-dimensional mask obtained by shifting the two-dimensional mask **d** by $2h+1$ pixels along the direction orthogonal to the ridge line direction. O'Gorman and Nickerson in [25] and Mehtre in [20] performed the image enhancement in a similar way, but they carried out the enhancement throughout the image whereas we regularize only a subset of points which are determined during the ridge line following. Fig. 9 compares the sections shown in Fig. 6 with the corresponding ones after they have been regularized.

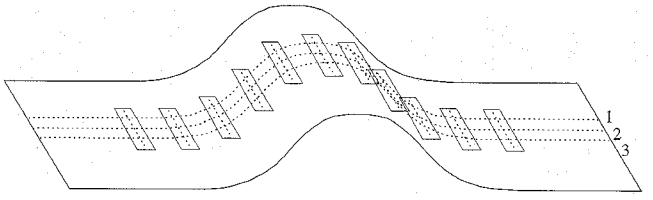


Fig. 7. The surface has been sectioned with three parallel planes ($h = 1$). The section set Ω is determined by plane 2. The gray level of each point belonging to Ω is computed as the average of the gray levels of the three corresponding points over the sections produced by planes 1, 2, and 3.

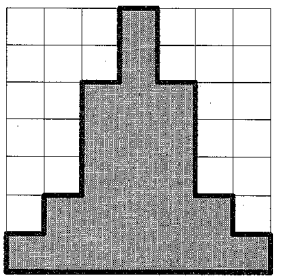


Fig. 8. The mask **d** has a symmetric Gaussian silhouette. The figure shows the mask adopted: $p = 3$, **d** = [1/23, 2/23, 5/23, 7/23, 5/23, 2/23, 1/23].

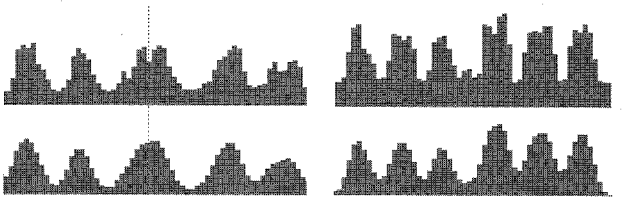


Fig. 9. The figure shows the comparison between the sections represented in Fig. 6 (top) and the same sections after regularization (bottom). The dotted line shows how, after regularization, a local maximum corresponding to the ridge center can be located.

After the regularization process, the local maximum required can be easily located by comparing the gray levels $\overline{\overline{gray}}(i_k, j_k)$, $k = p + 1, \ldots m - p$ and choosing the *weak* local maximum closest to the center (i_t, j_t). We recall that $\overline{\overline{gray}}(i_k, j_k)$, is a weak local maximum if and only if $\overline{\overline{gray}}(i_{k-1}, j_{k-1}) \leq \overline{\overline{gray}}(i_k, j_k) \leq \overline{\overline{gray}}(i_{k+1}, j_{k+1})$. By using weak maxima instead of strong maxima the ridge line following is guaranteed to work correctly even when a ridge line presents a flat profile.

We would like to point out the relevance of the regularization of the section silhouettes for correctly tracking the maxima points along the ridge lines. If regularization were not adopted the ridge following algorithm could be trapped by noise artifacts such as volcano profiles, small ridge breaks, ridge linkings, etc.

2.2 Tangent Direction Computation

At each step, the algorithm computes a new point (i_t, j_t) by moving μ pixels from the current point (i_c, j_c) along direction φ_c. The direction φ_c represents the ridge line local direction and can be computed as the tangent to the ridge in the point (i_c, j_c).

Several methods for estimating image directional information have been proposed in the literature. The simplest approach is based on gradient computation. It is well known that the gradient phase angle denotes the direction of the intensity maximum change. Therefore, the direction φ_c of a hypothetical edge which crosses the region centered in pixel (i_c, j_c) is orthogonal to the gradient phase angle in (i_c, j_c). This method, although simple and efficient, suffers from the non-linearity due to the computation of the gradient phase angle. Kawagoe and Tojo, in their work [14], use a different method. For each 2×2 pixel neighborhood, they make a straight comparison against four edge templates to

extract a rough directional estimate, which is then arithmetically averaged over a larger region to obtain a more accurate estimate. Stock and Swonger [29], Mehtre, et al. [21], following similar approaches, evaluate the tangent direction on the basis of pixel alignments relative to a fixed number of reference directions. The method used in this work, proposed by Donahue and Rokhlin [8], uses a gradient type operator to extract a directional estimate from each 2×2 pixel neighborhood, which is then averaged over a local window by least-squares minimization to control noise. In Appendix A, the basic steps of this method are described; more details can be found in [8]. This method allows for an *unoriented* direction to be computed. The computation of an *oriented* direction is subordinate to the choice of an orientation. For each step of the ridge line following, we choose the orientation in such a way that φ_c comes closest to the direction computed at the previous step. The technique used to compute the tangent directions, although rather efficient and robust, can become computationally expensive if the local windows used are large (if their side is 19 or more pixels) and the number of directions to be computed is very high. A more efficient implementation schema can be obtained by precomputing the directional image over a discrete grid (Fig. 10) and then determining the direction φ_c through lagrangian interpolation.

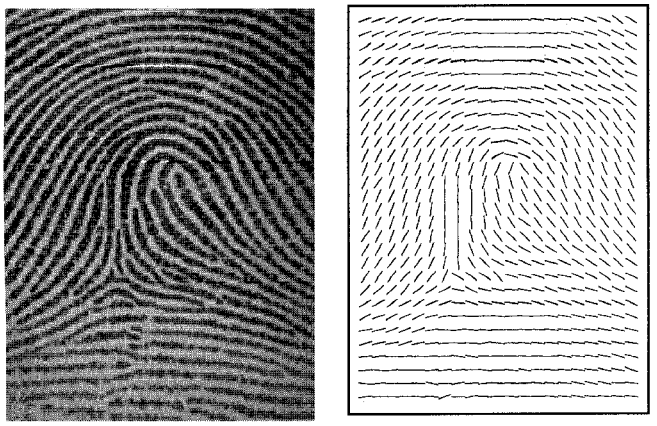


Fig. 10. A fingerprint and the corresponding directional image computed over a grid whose granularity is eight pixels.

2.3 Stop criteria

The stop criteria (i.e. the events which stop the ridge line following) are:

1) *Exit from interest area.* The new point (i_t, j_t) is external to a rectangular window **W** which represents the subimage whose minutiae are to be detected.
2) *Termination.* No local maxima, such that the segments having extremes (i_c, j_c) (i_n, j_n) form angles less than β (threshold value) with the direction φ_c, could be found in Ω. According to this criterion the ridge line following stops independently on the gray level of the current region, and the algorithm can work both on saturated regions and on contrast-deficient regions with no need for a particular tuning. In practice, by using the simple criterion stated above, the ridge line following could sometimes be trapped by small ridge breaks induced by the noise. Hence, we adopt a more sophisticated schema which tries to section the ridge line again through new planes adjacent to the initial one, before decreeing a stop by termination.
3) *Intersection.* The point (i_n, j_n) has been previously labeled as belonging to another ridge line. The point (i_n, j_n) is named *intersection point.*
4) *Excessive bending.* The segment delimited by (i_c, j_c) (i_n, j_n) forms with the *ridge line local direction* an angle greater than ψ (threshold value). The ridge line local direction is defined as the average of the directions of the segments (i_c, j_c) (i_n, j_n) relative to the last k steps ($k = 2,...4$). This criterion allows for the ridge line following to be stopped when the ridge line direction changes suddenly. In fact, due to the ridge line continuity, excessive bending always denotes an error in the ridge line following.

3 Minutiae Detection

In the previous section we have introduced an algorithm capable of extracting a ridge line given a starting point and an oriented direction. When a ridge line terminates or intersects another ridge line (originating a minutia) the algorithm stops and gives the characteristics (coordinates and direction) of the minutia found. It is now necessary to define a schema for extracting all the ridge lines in the image and, consequently, detecting all the minutiae.

The main problems arise from the difficulty of examining each ridge line only once and locating the intersections with the ridge lines already extracted. Our technique uses an auxiliary image **T** of the same dimension as **I**. **T** is initialized by setting its pixel values to 0. Every time a new ridge line is extracted from **I**, the pixels of **T** *corresponding* to the ridge line are labeled by assigning them an identifier.

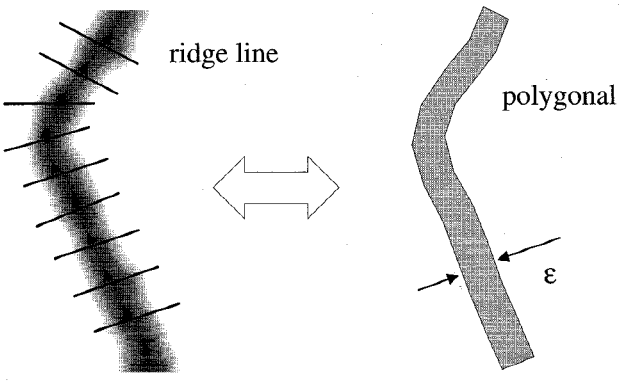


Fig. 11. A ridge line and the corresponding polygonal (ε-pixels thick).

The pixels of **T** corresponding to a ridge line are the pixels belonging to the polygonal, ε-pixels thick, which links the consecutive maximum points (i_n, j_n) located by the ridge line following algorithm on the ridge line (Fig. 11). The algorithm *find minutia* searches for a minutia by following the ridge line nearest to the starting point (i_s, j_s) in both directions:

```
find minutia ( i_s, j_s )
  { esc := false ;
    (i_c, j_c) := nearest ridge line maximum
      (i_s, j_s) ;
    if ((i_c, j_c) ∈ discovered ridge lines
      (T)) then esc := true ;
    if (¬ esc )
      {φ_c := tangent direction in (i_c, j_c) ;
        ridge line following (i_c, j_c, φ_c) ;
        if (termination ∨ excessive
          bending) then
          { // termination minutia has
            been found
            store termination minutia ;  }
        if  (intersection) then
          { // bifurcation minutia may
            exist
            if  (intersection point is
              valid) then
              store bifurcation minutia
            else delete false termination
              minutia ;
          }
        store polygonal ( T );
        // Perform similar operations in
          direction φ_c + π ;
        ridge line following (i_c, j_c, φ_c + π)
;
        ....
      }
  }
```

The algorithm starts by computing (procedure *nearest ridge line maximum*( )) a point (i_c, j_c) belonging to the ridge line nearest to the starting point (i_s, j_s). This operation can be carried out as follows:

1) Compute the tangent direction φ_s in (i_s, j_s).
2) Section S in (i_s, j_s), along direction $\varphi_s + \pi/2$ and with length $2\sigma + 1$.
3) Regularize the section determined in Step 2 and compute all the local maxima.
4) Choose, from the local maxima determined in Step 3, the local maximum (i_c, j_c) nearest to (i_s, j_s).

The computation of the tangent direction, the sectioning, the regularization and the determination of the maximum are performed as in the ridge line following algorithm. Fig. 12 shows an example.

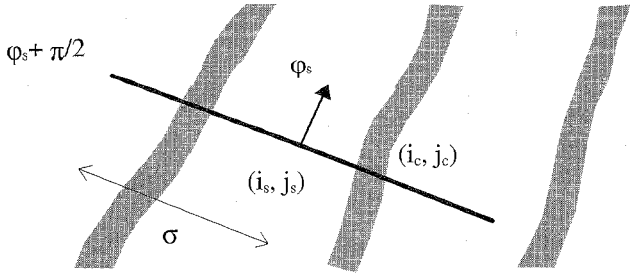


Fig. 12. The Fig. shows the point (i_c, j_c) belonging to the ridge line nearest to the starting point (i_s, j_s).

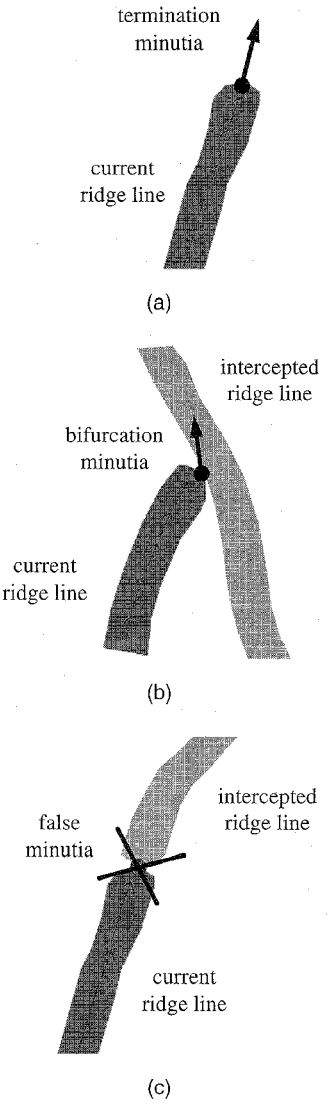


Fig. 13. Fig. 13a shows the detection of a termination minutia; Fig. 13b shows the detection of a bifurcation minutia; Fig. 13c shows the detection of a false intersection.

Once (i_c, j_c) has been determined, the algorithm verifies, on **T**, that the pixel (i_c, j_c) has not been labeled, or equivalently that the ridge line has not yet been examined (starting from a different starting point). If (i_c, j_c) is not labeled, the algorithm continues by computing the tangent direction φ_c, and following the ridge line through the algorithm *ridge line following* previously described. In this context the stop criteria can be interpreted as follows:

- *Exit from interest area*: no minutiae have been found.
- *Termination*: a termination minutia has been found (Fig. 13.a).
- *Intersection*: a ridge line already examined has been intersected. During the ridge line following, every time a new point (i_n, j_n) is computed, the algorithm checks whether the corresponding point in **T** has already been labeled. In this case the point (i_n, j_n) belongs to two ridge lines (i.e., it is an intersection point). This event can occur either when the current ridge line and the intersected ridge line form a bifur-

cation (Fig. 13.b), or when the intersected ridge line has been previously truncated due to the detection of a false termination minutia (Fig. 13.c). In the former case the intersection point originates a bifurcation minutia. In the latter case, both ridge lines belong to the same ridge line, and the intersection does not originate a minutia. Furthermore, in this case, it is necessary to remove the false termination minutia detected previously in correspondence with the ridge line termination.

- *Excessive bending*. The algorithm behaves in the same way as it does for a ridge line termination, and originates a termination minutia. Excessive bending termination can be due, although very rarely, to the loss of a ridge line being followed. This event does not constitute a serious problem because, probably, later on the algorithm will discover a situation similar to Fig. 13.c and the false termination minutia will be removed.

The polygonal trace (ε-pixels thick), corresponding to the portion of ridge line just extracted, is then stored in **T**.

The second part of the algorithm is analogous to the first one: in fact, the ridge line is now followed along the opposite direction ($\varphi_c + \pi$), starting from the same point (i_c, j_c).

The algorithm *find minutia* presented above enables all the fingerprint minutiae within a window **W** to be detected. Let G be a regular square-meshed grid, with granularity ν pixels, superimposed on the window **W**; by executing *find minutia* for each node of G we extract all the ridge lines inside **W** and consequently we detect all the minutiae. Fig. 14 shows the results obtained by applying our approach to a sample fingerprint.

Fig. 14. Minutiae detection on a sample fingerprint. The ridge lines are represented through the corresponding polylines of **T**. The termination minutiae are denoted by circles while the bifurcation minutiae are denoted by squares. The parameter values adopted are: $\mu = 3$, $\sigma = 7$, $h = 1$, $\beta = \psi = 30°$, $\varepsilon = 3$, $\nu = 2$.

4 Performance Evaluation and Comparison

The technique proposed in this paper has been adopted within a prototype of a biometric system for fingerprint identification, which has been widely tested and experimented in real environments. The aim of this section is to demonstrate that the proposed direct gray scale approach performs better than approaches which require binarization and thinning as intermediate steps. To this purpose we have implemented our technique (A) and four different schemes based on binarization and thinning, which have been derived from [29] (B), [23] (C), [31] (D), and [25] (E), respectively. It is worth remarking that the primary goal of papers [29] and [25] is the minutiae detection, while in [23] and [31] binarization and thinning are steps for the classification and/or recognition of fingerprint patterns. In order to compare our approach with other approaches known in the literature, we have assembled a sample set of fingerprints belonging to different sources and exhibiting a different degree of image quality.

The sample set has the following composition: seven fingerprints (no. 3, no. 4, no. 5, no. 11, no. 12, no. 13, and no. 14 in Fig. 15) taken from the NIST fingerprint database [34], four fingerprints (no. 1, no. 2, no. 9 and no. 10 in Fig. 15) from an FBI sample set, and three fingerprints (no. 6, no. 7 and no. 8 in Fig. 15) acquired through an opto-electronic device based on a prism. A remark is perhaps in order: conducting a reliable analysis of the results produced by the different approaches requires a human expert and a great deal of time. This justifies the small size of the sample set used in our comparison.

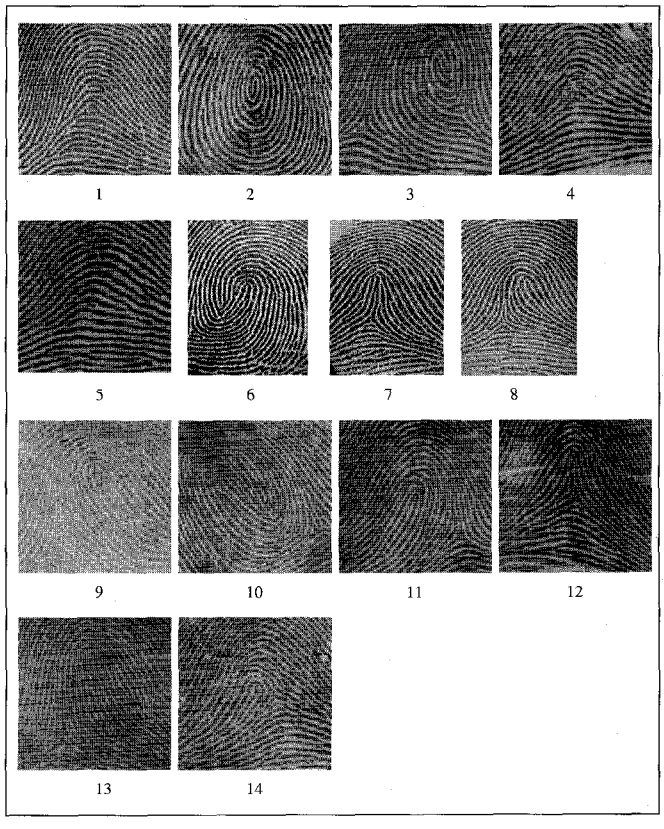


Fig. 15. The sample set of fingerprints used in our experiments. Fingerprints 1, ..., 8 belong to the class *good*, fingerprints 9, ... 14 belong to the class *poor*. All the fingerprints (except no. 6, no. 7 and no. 8) have dimensions 256 × 256, with 256 gray levels. The fingerprints no. 6, no. 7 and no. 8 have dimensions 190 × 250, 256 gray levels.

By using the contrast and consistency index proposed in [8], the fingerprints have been coarsely classified according to their quality into *good* and *poor*. On each fingerprint the expert has marked the *certain* minutiae, neglecting the minutiae located in regions with poor contrast, where minutiae detection cannot even be performed manually. Automatic minutiae detection has been achieved through our technique (A) and through the binarization-based techniques (B), (C), (D) and (E), respectively. The parameter values used in our approach are: $\sigma = 7$, $h = 1$, $\beta = \psi = 30°$, $\varepsilon = 3$, $\nu = 2$ for all the fingerprints in the sample set; $\mu = 3$ for fingerprints no. 6, no. 7 and no. 8; $\mu = 5$ for the other fingerprints in the sample set. Two different values for the parameter μ (the step of the ridge line following) are needed to process images taken at a different resolution.

In approaches B and C the binarization process has been preceded by a smoothing operation based on the convolution with a gaussian 5×5 pixels mask. This operation regularizes the starting image, so that the approaches B and C give better results. Approach E has been implemented without the post-processing noise reduction proposed in [25]; in that paper, the post-processing is applied on the binary image produced by the enhancement/binarization step, and it is useful to reduce artifacts in noisy background region outside the fingerprint or along the boundaries between the ridge region and the background. In our sample set we use only fingerprint images which exclusively contain ridges and for this reason that post processing step is not necessary. In approaches B, C, D and E the thinning process has been carried out through the algorithm presented in [3], which provides good results on fingerprints. In B, C, D and E, the minutiae detection on the binary skeleton has been performed by labelling as minutiae those pixels whose crossing number is different from 2. The crossing number [2] $cn(P)$ at a point P is defined as half of cumulative successive differences between pairs of adjacent pixels belonging to the 8-neighborhood of P:

$$\mathrm{cn}(P) = \frac{1}{2}\sum_{i=1}^{8}\left|\mathrm{val}\left(P_{i \bmod 8}\right) - \mathrm{val}\left(P_{i-1}\right)\right|$$

where P_0, P_1, ... P_7 are the pixels belonging to the 8-neighborhood of P and $val(P_j)$ is the value (0, 1) of the pixel P_j. In particular, the pixels having $cn(P) = 1$ correspond to termination minutiae, while the pixels which have $cn(P) \geq 3$ correspond to bifurcation minutiae. In all approaches, A, B, C, D and E, the minutiae detected have been filtered by removing:

- the minutiae belonging to regions where the image contrast (computed as in [8]) is less than half of the average image contrast.
- the pairs of termination minutiae which are less than k pixels (k = 6) distant from each other.
- the sets of bifurcation minutiae (except one minutia for each set) belonging to a neighborhood with diameter k pixels (k = 6).

Fig. 16 shows the *certain* minutiae manually detected on fingerprints no. 1 and no. 13 of the sample set. Figures 17, 18 and 19 show the automatic extraction through approaches A, B, C, D, and E on the same fingerprints.

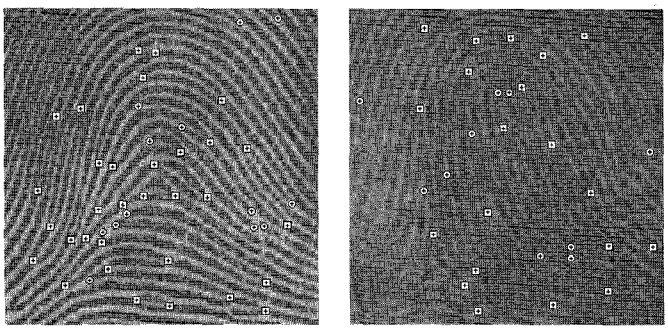


Fig. 16. The figure shows the *certain* minutiae detected manually by a human expert on fingerprints no. 1 (on the left) and no. 13 (on the right) of the sample set.

Table 1 reports the results in terms of undetected minutiae (*dropped*), non-existent minutiae (*false*) and type-exchanged minutiae (*exchanged*). Tables 2, 3, and 4 report the average error percentage relative to the classes *good*, *poor* and to the whole sample set, respectively.

Table 5 reports the average computational times spent in automatic minutiae extraction measured on a PC 80486-DX 50 Mhz. The graphics in Fig. 20 compare the average error percentage and the average computational times obtained with the different approaches.

The results achieved by the proposed technique on a real sample of 150 fingerprints, acquired through an opto-electronic device based on a prism, are very similar to those obtained for the class *good* of the sample set considered here (Table 2, Approach A).

The following conclusions can be drawn:

- the average error percentage, in terms of *dropped* and *exchanged* minutiae, as produced by our approach is comparable to the errors produced by the other approaches, although slightly larger.
- the average error percentage, in terms of *false* minutiae, as produced by our approach is considerably lower than the errors produced by the other approaches.
- the average computational time of our approach is considerably lower than the time of the other approaches.
- approach E, whose performance in terms of total error is comparable with that of our approach, is one order of magnitude slower than our approach.

The large number of false minutiae determined by approaches B, C and D (especially on class *poor*) is due to the irregularity of the binary traces produced by the binarization process. Regularization techniques, similar to that presented in [6], can substantially reduce the number of false minutiae. Structural considerations about minutiae position [13], [37] can be applied to all the approaches in order to decrease the number of false minutiae.

TABLE 1
AUTOMATIC MINUTIAE DETECTION.

		A			B			C			D			E		
fingerprint	minutiae	d	f	x	d	f	x	d	f	x	d	f	x	d	f	x
1	33	0	2	7	0	25	2	1	102	7	0	53	6	0	5	4
2	29	3	1	4	0	20	2	2	24	1	1	34	2	2	4	0
3	28	1	2	4	0	24	0	1	35	1	1	28	2	0	4	4
4	37	3	0	4	0	15	3	4	32	0	1	42	3	2	2	2
5	22	0	0	3	0	38	1	0	80	2	1	18	0	0	8	2
6	23	0	0	4	0	3	2	0	25	3	0	19	2	0	2	1
7	31	2	1	2	3	13	2	2	27	2	0	40	3	2	0	2
8	31	1	0	3	0	2	0	1	10	3	0	5	4	0	0	3
9	21	1	10	1	1	115	1	0	180	1	0	153	1	0	24	2
10	22	1	0	4	0	53	1	0	104	3	0	100	1	0	8	4
11	32	3	5	4	1	22	4	2	22	3	3	73	4	1	5	2
12	33	3	8	2	0	23	3	4	45	1	1	79	3	0	10	5
13	20	0	0	4	0	48	2	0	57	3	0	81	2	0	7	5
14	37	0	5	6	1	43	5	3	67	5	1	57	4	0	11	2

The second column indicates the number of certain *minutiae detected manually. d, f and x denote the number of* dropped *minutiae,* false *minutiae and* exchanged *minutiae, respectively.*

TABLE 2
AVERAGE ERROR PERCENTAGE RELATIVE TO THE FINGERPRINTS OF THE CLASS GOOD.

good					
	A	**B**	**C**	**D**	**E**
dropped minutiae	4.27%	1.28%	4.70%	1.71%	2.56%
false minutiae	2.56%	59.83%	143.16%	102.14%	10.68%
exchanged minutiae	13.25%	5.13%	8.12%	9.40%	7.69%
total error	20.09%	66.24%	155.98%	113.25%	20.94%

TABLE 3
AVERAGE ERROR PERCENTAGE RELATIVE TO THE FINGERPRINTS OF THE CLASS POOR.

poor					
	A	**B**	**C**	**D**	**E**
dropped minutiae	4.85%	1.82%	5.45%	3.03%	0.61%
false minutiae	16.97%	184.24%	287.88%	329.09%	39.39%
exchanged minutiae	12.73%	9.70%	9.70%	9.09%	12.12%
total error	34.55%	195.76%	303.03%	341.21%	52.12%

TABLE 4
AVERAGE ERROR PERCENTAGE RELATIVE TO THE FINGERPRINTS OF THE WHOLE SAMPLE SET.

Whole set					
	A	**B**	**C**	**D**	**E**
dropped minutiae	4.51%	1.50%	5.01%	2.26%	1.75%
false minutiae	8.52%	111.28%	203.01%	195.99%	22.56%
exchanged minutiae	13.03%	7.02%	8.77%	9.27%	9.52%
total error	26.07%	119.80%	216.79%	207.52%	33.83%

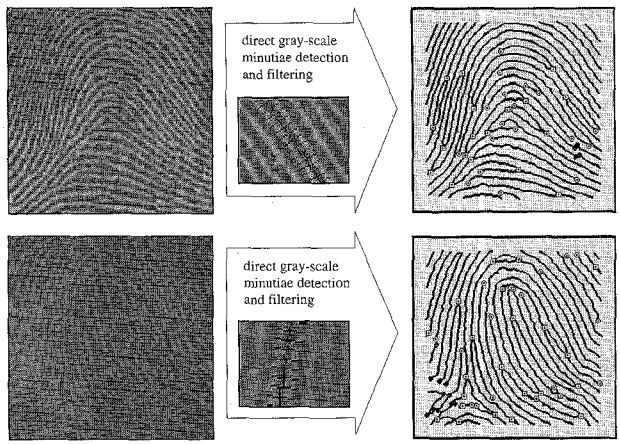


Fig. 17. Automatic minutiae detection in fingerprints no. 1 and no. 13 using our approach (A). Inside the arrow-box a snapshot of a direct gray scale ridge line following is shown. In the output image the ridge lines are represented through the corresponding polylines of **T**. The minutiae are denoted by small white circles (termination minutiae) and squares (bifurcation minutiae). Both black squares and black circles denote filtered minutiae.

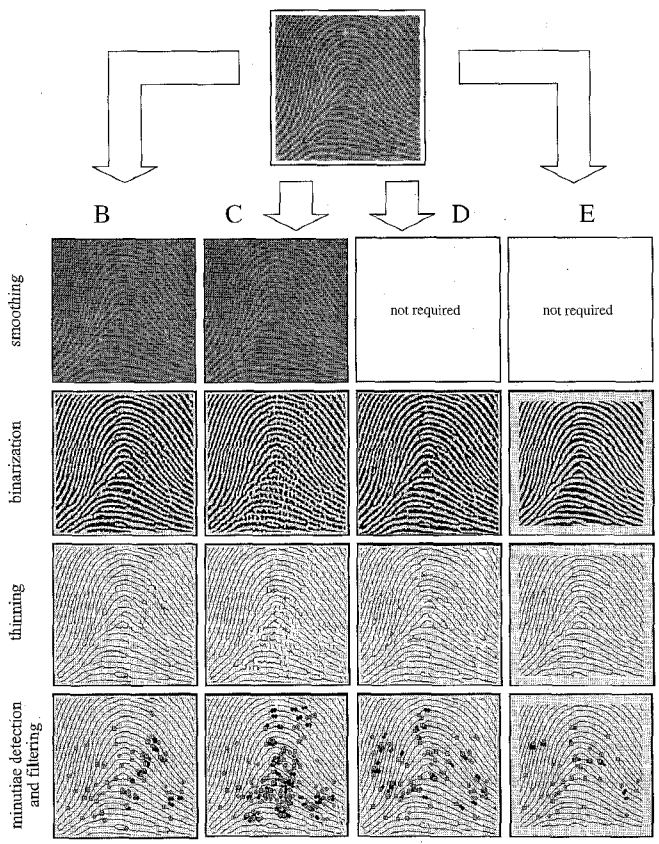


Fig. 18. Automatic minutiae detection in fingerprint no.1 (belonging to the class *good*) using approaches B, C, D and E. Each column shows the results of the processing steps of the corresponding approach. The minutiae are denoted by small white circles (termination minutiae) and squares (bifurcation minutiae). Both black squares and black circles denote filtered minutiae.

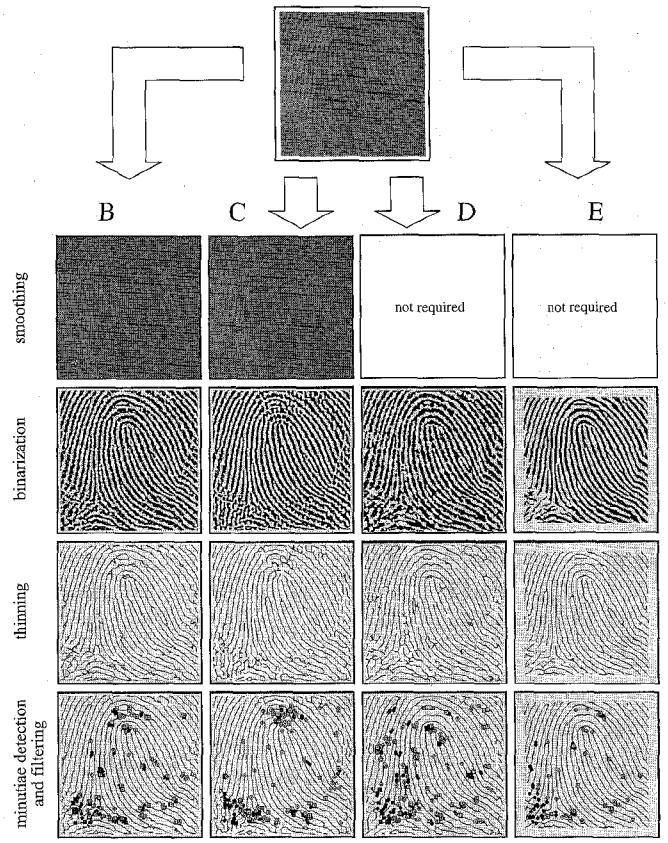


Fig. 19. Automatic minutiae detection in fingerprint no.13 (belonging to the class *poor*) using approaches B, C, D and E. Each column shows the results of the processing steps of the corresponding approach. The minutiae are denoted by small white circles (termination minutiae) and squares (bifurcation minutiae). Both black squares and black circles denote filtered minutiae.

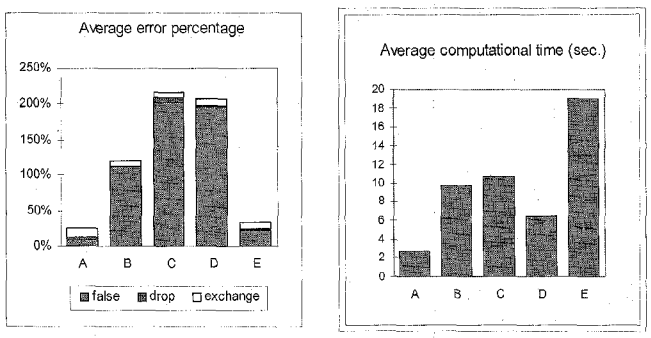


Fig. 20. A comparison between the average error percentage and the average computational times of the five different approaches.

TABLE 5
AVERAGE COMPUTATIONAL TIME (IN SECONDS) TAKEN FOR THE MINUTIAE DETECTION ON PC 80486-DX 50 MHZ ARCHITECTURE.

Average computational time (sec.)					
	A	**B**	**C**	**D**	**E**
directional image	0.51	-	-	-	0.51
smoothing	-	3.90	3.90	-	-
binarization	-	2.25	3.08	2.64	15.73
thinning	-	3.11	3.11	3.15	2.45
min. detection and filtering	2.22	0.51	0.67	0.66	0.33
total time	2.73	9.77	10.76	6.45	19.03

TABLE 6
ELEMENTARY OPERATIONS CARRIED OUT AT EACH FOLLOWING STEP. $(2\sigma+1)$ IS THE LENGTH OF THE SECTION SEGMENT

	sum	**multiplication**	**test**
1	2	2	-
2	$11+2(2\sigma + 1)$	4	3
3	$(2h + 1)(2\sigma + 1)$	$(2p + 1)(2\sigma + 1)$	-
4	-	-	$2(2\sigma - 1)$
5	3	1	7
6	$\varepsilon\,(2\mu + 7)$	$2\,\varepsilon$	$3\,\varepsilon$
7	12	11	-
Total	$28 + (2\sigma + 1)(2h + 3) + \varepsilon\,(2\mu + 7)$	$18 + (2p + 1)(2\sigma + 1) + 2\,\varepsilon$	$10 + 2(2\sigma - 1) + 3\,\varepsilon$

$(2\sigma + 1)$ is the length of the section segment, $(2h + 1)$ is the number of planes used by the first step of the regularization, $(2p + 1)$ is the dimension of the mask **d** *and ε is the thickness of the polygonal traces used to update* **T**.

Most of the errors of our approach are minutiae exchanges. These errors are mainly due to some termination minutiae which are detected as bifurcation minutiae. In particular, if a termination minutia is very close to another ridge line the algorithm may skip the termination and intersect the adjacent ridge line. A local analysis of the gray scale image in each minutia neighborhood could be adopted to substantially reduce the type-exchanged errors. To train a neural network to verify the type of minutiae detected could be a promising approach.

Let us now make some considerations about the computational complexity of the proposed technique. We assume, for simplicity, that a fingerprint pattern is made up of a set of straight horizontal segments, which are ξ-pixels thick and ξ-pixels distant from each other (Fig. 21). In fact, even if in reality the ridge line thickness may vary from about 4-5 pixel to 12-15 pixel (at the resolution we have used), ξ can be assumed to be the mean value. Under this assumption, the number of ridge lines in a $n \times n$ image is $n/2\xi$. If μ is the step of the ridge line following algorithm, n/μ steps are necessary to extract a whole ridge line. Therefore, the algorithm performs $n^2/(2\xi\mu)$ steps in order to extract all the ridge lines and then all the minutiae.

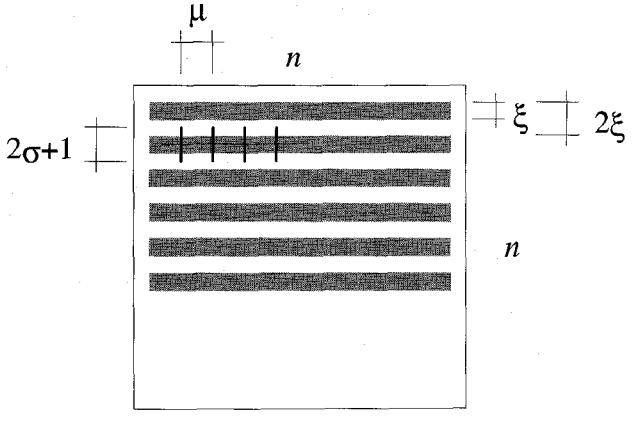


Fig. 21. A simplified fingerprint pattern.

At each step the algorithm performs the following operations:

1) computation of the new point (i_t, j_t),
2) construction of the section segment Ω,
3) convolution with the gaussian mask **d**,
4) maxima searching,
5) checking the stop criteria,
6) update of **T**,
7) computation of the new direction φ_c.

By using Bresenham's algorithm [4] to compute Ω and to update $\mathbf{T}$, and by representing the angles through discrete values, all the computations can be performed by means of integer arithmetic. Table 6 summarizes the elementary operations carried out at each ridge line following step.

All the parameter values can be approximately estimated according to the ridge thickness ξ; in the following we assume: $\sigma = 2p+1 = \xi$ and $\mu = \varepsilon = 2h+1 = \frac{1}{2}\xi$. The computational complexity of the technique presented (except for the computation of the directional image) is then:

$$n^2\left(\frac{3/2\,\xi^2+8\,\xi+30}{\xi^2}\right) \text{ sum}, \quad (\approx 3\,n^2 \quad \text{if } \xi=8)$$

$$n^2\left(\frac{2\,\xi^2+2\,\xi+18}{\xi^2}\right) \text{ multiplication}, \quad (\approx 2.5\,n^2 \quad \text{if } \xi=8)$$

$$n^2\left(\frac{11/2\,\xi+8}{\xi^2}\right) \text{ test}. \quad (\approx 0.8\,n^2 \quad \text{if } \xi=8)$$

Hence, the total number of operations executed for the direct gray scale minutiae detection is only a few times the number of pixels of the whole image. It is important to note that, by using techniques based on binarization and thinning, the binarization step alone requires n^2 pixel-neighborhoods to be processed, so that the time taken for the whole detection is undoubtedly longer than in our approach.

5 Conclusion

Minutiae represent the most discriminant features of fingerprints, so that they are employed by most automatic systems for fingerprint comparison. In this paper we have presented an original approach to automatic minutiae detection, which detects the minutiae directly in gray scale images. This technique is based on a ridge line following algorithm that follows the image ridge lines until a termination or a bifurcation occurs. In spite of a greater conceptual complexity, we have shown that our technique has less computational complexity than the complexity of the techniques which require binarization and thinning. The results achieved have been compared with those obtained through some known approaches. The comparison shows the superiority of our technique in terms of efficiency and robustness.

The low computational complexity makes this method particularly suitable for applications where efficiency is a primary issue (i.e., on-line access control, low-cost biometric-systems, etc.).

Future work in this direction will include the following topics:

- Investigating the feasibility of a neural network approach to verify the type of minutiae detected by the approach presented in this work.
- Developing a new ridge line following algorithm capable of dynamically adapting the parameters μ and σ according to the local ridge line width.
- Defining local criteria to evaluate the reliability of each minutia detection, in order to associate each minutia with a confidence value, which is particularly useful during fingerprint matching.
- Adopting more sophisticated identification models, for instance extending minutiae definition by including trifurcations, islands, spurs, bridges, etc.
- Analyzing the behavior of our approach on chanceprints, which are partial fingerprints, normally poor quality images, picked up from scene of crime. Processing chanceprints is very important for police investigation purposes, and is a real challenge.

Finally, we would like to point out that our ridge line following algorithm, though developed ad hoc to extract the fingerprint ridge lines, can also be more generally used for the detection of lines and curves in gray scale images.

Appendix A
Computation of the Tangent Direction

Let (i_0,j_0) be the pixel of the image $\mathbf{I}$ where the tangent direction φ_0 must be computed. Let the *tangent window* be a squared window centered in (i_0,j_0) with side length α pixels. For each pixel (i_h,j_k) belonging to the tangent window, a vector $\mathbf{n}_{hk}$ orthogonal to the surface $z = S(i, j)$ is defined. The tangent vector in each pixel (i_h,j_k) (when defined) lies on the ij-plane and is orthogonal to the corresponding vector $\mathbf{n}_{hk}$. The average tangent vector $\mathbf{t}$, which represents the required direction φ_0, is the unit vector lying on the ij-plane which is the "most orthogonal" to all the vectors $\mathbf{n}_{hk}$ computed.

Fig. 22 shows the tangent direction $\mathbf{t}$ on a surface S including a ridge parallel to direction j.

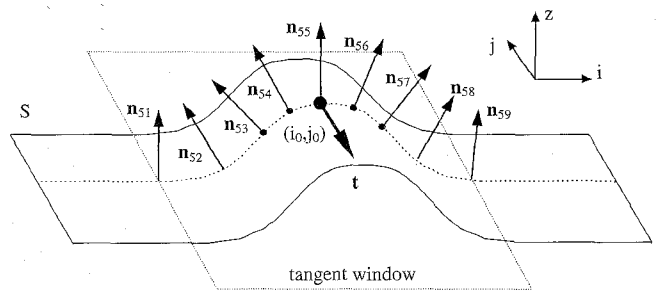


Fig. 22. A surface S including a ridge in direction j. The figure shows the tangent window (α = 9) centered in (i_0,j_0), the normal vectors $\mathbf{n}_{hk}$ and the average tangent vector $\mathbf{t}$. The normal vectors $\mathbf{n}_{hk}$ (computed for simplicity only in one row; h=5, k=1,...9) are the normal vectors to the surface S. The average tangent vector $\mathbf{t}$ is the "most orthogonal" to the normal vectors $\mathbf{n}_1$,...$\mathbf{n}_9$ of the unit vectors lying on the ij-plane.

The average tangent vector computation takes place as follows:

Let (i_{h+1}, j_{k+1}), (i_{h-1}, j_{k+1}), (i_{h-1}, j_{k-1}) and (i_{h+1}, j_{k-1}) be the pixels belonging to the 2×2 pixel neighborhood of each tangent-window pixel (i_h, j_k). Let $a_1=gray(i_{h+1}, j_{k+1})$, $a_2=gray(i_{h-1}, j_{k+1})$, $a_3=gray(i_{h-1}, j_{k-1})$, and $a_4=gray(i_{h+1}, j_{k-1})$. For each neighborhood the normal vector $\mathbf{n}_{hk}$ to the plane surface determined by (a_1, a_2, a_3, a_4) can be computed via least-squares minimization:

$$\mathbf{n}_{hk} = [a_{hk}, b_{hk}, 1]$$

where

$$a_{hk} = \frac{-a_1 + a_2 + a_3 - a_4}{4}, \qquad b_{hk} = \frac{-a_1 - a_2 + a_3 + a_4}{4}$$

Then, the average tangent vector $\mathbf{t}$ is determined as the unit vector lying on the ij-plane which is the "most orthogonal" to all the normal vectors $\mathbf{n}_{hk}$, h=1,...α, k=1,...α, computed over the tangent window. Let $\mathbf{v}_{hk}=(a_{hk}, b_{hk})$, h=1,...$\alpha$, k=1,...$\alpha$ be the vectors obtained by removing the z component from the corresponding normal vectors $\mathbf{n}_{hk}$ and let $\mathbf{t} = (t_1, t_2)$. Formally it is a least-squares minimization:

$$min \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} \mathbf{v}_{hk} \cdot \mathbf{t}^2 \quad \text{subject to} \quad \mathbf{t} = 1$$

Neglecting the mathematical details, which can be found in [8]:

$$A = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} (a_{hk})^2 , \quad B = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} (b_{hk})^2 , \quad C = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} a_{hk}\, b_{hk}$$

$$\mathbf{t} = \begin{cases} \left[1, \ \frac{B - A}{2C} - \text{sgn}(C)\sqrt{\left(\frac{B - A}{2C}\right)^2 + 1}\right] & \text{if } C \neq 0 \\ [1, 0] & \text{if } C = 0, \ A \leq B \\ [0, 1] & \text{if } C = 0, \ A > B \end{cases}$$

finally, the tangent direction φ_0 can be simply computed as:

$$j_0 = \begin{cases} \arctan\left(\frac{t_2}{t_1}\right) & \text{if } t_1 \neq 0 \\ \frac{p}{2} & \text{otherwise} \end{cases}$$

REFERENCES

[1] American National Standards Institute, Fingerprint Identification—Data Format for Information Interchange. New York, 1986.

[2] C. Arcelli and G.S.D. Baja, "A Width Independent Fast Thinning Algorithm," IEEE Trans. Pattern Analysis Machine Intelligence, vol. 7, no. 4, pp. 463-474, 1984.

[3] O. Baruch, "Line Thinning by Line Following," Pattern Recognition Letters, vol. 8, no. 4, pp. 271-276, 1988.

[4] J. Bresenham, IBM System J., vol. 4, no. 1, pp. 25-30, 1965.

[5] C.E. Chapel., Fingerprinting—A manual of Identification. New York: Coward McCann, 1971.

[6] L. Coetzee and E.C. Botha, "Fingerprint Recognition in Low Quality Images," Pattern Recognition, vol. 26, no. 10, pp. 1,441-1,460, 1993.

[7] C.R. Dyer and A. Rosenfeld, "Thinning Algorithms for Gray-Scale Pictures," IEEE Trans. Pattern Analysis Machine Intelligence , vol. 1, no. 1, pp. 88-89, 1979.

[8] M.J. Donahue and S.I. Rokhlin, "On the Use of Level Curves in Image Analysis," Image Understanding, vol. 57, no. 2, pp. 185-203, 1993.

[9] F. Galton, Finger Prints. London: Macmillan, 1892.

[10] F.T. Gamble, L.M. Frye and D.R. Grieser, "Real-Time Fingerprint Verification System," Applied Optics, vol. 31 no. 5, pp. 652-655, 1992.

[11] E.R. Henry, Classification and Uses of Finger Prints. London: Routledge, 1900.

[12] J. Hollingum, "Automated Fingerprint Analysis Offers Fast Verification," Sensor Review, vol. 12, no. 3, pp. 12-15, 1992.

[13] D.C.D. Hung, "Enhancement and Feature Purification of Fingerprint Images," Pattern Recognition, vol. 26, no. 11, pp. 1,661-1,671, 1993.

[14] M. Kawagoe and A. Tojo "Fingerprint Pattern Classification," Pattern Recognition vol. 17, no. 3, pp. 295-303, 1984.

[15] S. Igaki, S. Eguchi, F. Yamagishi, H. Ikeda, and T. Inagaki, "Real-Time Fingerprint Sensor Using a Hologram," Applied Optics, vol. 31 no. 11, pp. 1,794-1,802, 1992.

[16] L. Leboucher, T. Irinopoulou, and S. Hazout, "Grey-Tone Skeletons of Elongated Objects Using the Concept of Morphological Automaton. Application to Images of DNA Molecules," Pattern Recognition Letters, vol. 15, no. 3, pp. 309-315, 1994.

[17] M.T. Leung, W.E. Engeler, and P. Frank, "Fingerprint Image Processing Using Neural Network," Proc. Tenth Conf. Computer and Communication Systems, pp. 582-586, Hong Kong, 1990.

[18] W.F. Leung, S.H. Leung, W.H. Lau, and A. Luk, "Fingerprint Recognition Using Neural Network," Proc. IEEE Workshop Neural Network for Signal Processing, pp. 226-235, 1991.

[19] C.H. Lin, J.H. Liu, J.W. Ostenberg, and J.D. Nicol, "Fingerprint Comparison I: Similarity of Fingerprint," J. Forensic Science, no. 27, pp. 290-304, 1982.

[20] B.M. Mehtre, "Fingerprint Image Analysis for Automatic Identification," Machine Vision and Applications, vol. 6, no. 2-3, pp. 124-139, 1993.

[21] B.M. Mehtre, N.N. Murthy, S. Kapoor, and B. Chatterjee, "Segmentation of Fingerprint Images Using the Directional Image," Pattern Recognition, vol. 20, no. 4, pp. 429-435, 1987.

[22] B.M. Mehtre and N.N. Murthy, "A Minutia Based Fingerprint Identification System," Proc. Second Int'l Conf. Advances in Pattern Recognition and Digital Techniques, Calcutta 1986.

[23] B. Moayer and K. Fu, "A Tree System Approach for Fingerprint Pattern Recognition," IEEE Trans. Pattwen Analysis and Machine Intelligence, , vol. 8, no. 3, pp. 376-388, 1986.

[24] A.A. Moenssens, Fingerprint Techniques. Philadelphia: Chilton Book Company, 1971.

[25] L. O'Gorman and J.V. Nickerson, "An Approach to Fingerprint Filter Design," Pattern Recognition, vol. 22, no. 1, pp. 29-38, 1989.

[26] F. Pernus, S. Kovacic and L. Gyergyek, "Minutiae-Based Fingerprint Recognition," Proc. Fifth Int'l Conf. Pattern Recognition, pp. 1380-1382, 1980.

[27] B.G. Sherlock, D.M. Monro, and K. Millard, "Algorithm for Enhancing Fingerprint Images," Electronics Letters, vol. 28, no. 18, pp. 1,720-1,721, 1992.

[28] B.G. Sherlock, D.M. Monro, and K. Millard, "Fingerprint Enhancement by Directional Fourier Filtering," Proc. Conf. Vision, Image and Signal Processing, pp. 87-94, 1994.

[29] R.M. Stock and C.W. Swonger, "Development and Evaluation of a Reader of Fingerprint Minutiae," Cornell Aeronautical Laboratory, Technical Report CAL No. XM-2478-X-1:13-17, 1969.

[30] E.N. Székely and V. Székely, "Image Recognition Problems of Fingerprint Identification," Microprocessors and Microsystems, vol. 17, no. 4, pp. 215, 1993.

[31] M.R. Verma, A.K. Majumdar, and B. Chatterjee, "Edge Detection in Fingerprints," Pattern Recognition, vol. 20, no. 5, pp. 513-523, 1987.

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

[33] L. Wang and T. Pavlidis, "Detection of Curved and Straight Segments from Gray Scale Topography," Image Understanding, vol. 58, no. 3, pp. 352-365, 1993.

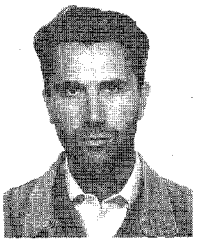
[34] C.I. Watson and C.L. Wilson, Fingerprint Database. National Institute of Standards and Technology, Special Database 4, April 18, 1992.

[35] D.M. Weber, "A Cost Effective Fingerprint Verification Algorithm for Commercial Applications," Proc. 1992 South African Symposium on Communication and Signal Processing, pp. 99-104, 1992.

[36] J.H. Wegstein, "An Automated Fingerprint Identification System," US Government Publication, Washington, 1982.

[37] Q. Xiao and H. Raafat, "Fingerprint Image Postprocessing: A Combined Statistical and Structural Approach," Pattern Recognition, vol. 24, no. 10, pp. 985-992, 1991.

Dario Maio is a full professor with the Computer Science Department of the University of Bologna, Italy. He has published in the fields of distributed computer systems, computer performance evaluation, database design, information systems, neural networks, biometric systems, and autonomous agents. Before joining the university, Dr. Maio received a fellowship from the Italian National Research Council (CNR) for participation in the Air Traffic Control Project. He received his degree in electronic engineering from the Univeristy of Bologna in 1975, and is a member of the IEEE.



Davide Maltoni received the degree in computer science from the University of Bologna, Italy in 1993. Since November 1994, he has been a PhD student at the Faculty of Computer Science of the University of Bologna researching biometric systems. His research interests also include autonomous agents, pattern recognition, and neural nets.