6 Fingerprint Image Processing

There are two different approaches to fingerprint verification. One can detect global structures and perform a *pattern match* between two fingerprint images. This is done with the aid of correlation operators [Chan1999, Jain1999d] or general neural network classifiers [Blue1994, Coet1992]. The more prominent technique is to extract the local structures, the so-called minutiae or Galton details introduced in chapter 5.4. Both techniques require an automated system to preprocess a fingerprint image. Characteristic data can be extracted automatically from the enhanced image. Raw images generated from an inked fingerprint or with a live scanner are processed as shown in the following figure.

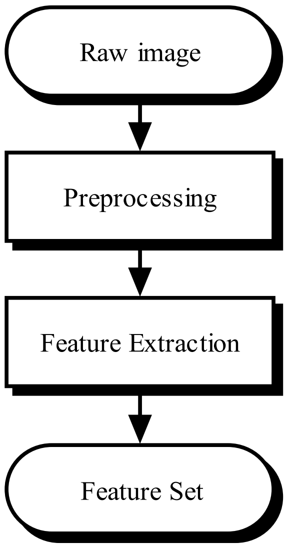


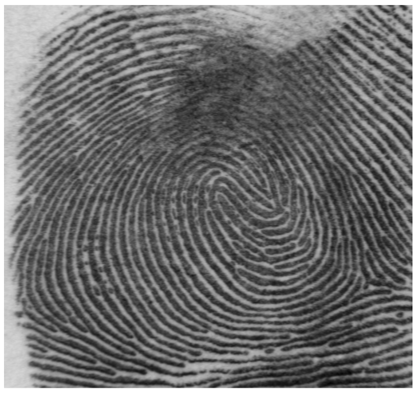
Figure 6.1: Feature extraction flowchart

Image enhancement is a critical step in automated fingerprint recognition systems. This section describes the image processing and feature extraction techniques used in the remainder of this work for fingerprint verification. Digital image processing requires the source images to come in a digital format. This is achieved either by a *frame grabber*, that converts sensed analog video signals or a direct digital sensor (e.g. a CMOS camera). A fingerprint verification system operates on a gray-scale image produced by a live scanner. Inked fingerprints would give similar pictures when scanned but are not considered here. Usually, the sensor delivers an 8-bit gray-scale image with 500 dpi resolution. For a comprehensive introduction in image processing, the reader is referred to [Gonz1993, Rose1982, Phil1994, Bank1990, Lewi1990, Cast1996].

This section covers the image processing tools required to extract local features with good stability from live fingerprint images. Correlation-based fingerprint matching is not addressed in this thesis. The interested reader is advised to study [Sing1998, Coet1992, Chan1999, Jain2000b, Karu1996, Hrec1990, Blue1994].

6.1 Spatial Domain Techniques

Spatial domain techniques play an important role in image processing, pattern recognition and image compression applications. They operate directly on the pixels of an image and have many applications. In digital image processing, spatial domain techniques are particularly useful for image enhancement. The standard fingerprint image is a gray-scale image, represented as a matrix. The gray levels usually are not evenly distributed in such an image. The discrete distribution of the gray values in an image is referred to as *luminance spectrum*. The graph depicting this distribution is called *histogram*. A fingerprint image and its histogram are shown in Figure 6.2. It plots the luminance spectrum as a stair-stepped function by simply counting the occurrences of every gray value.



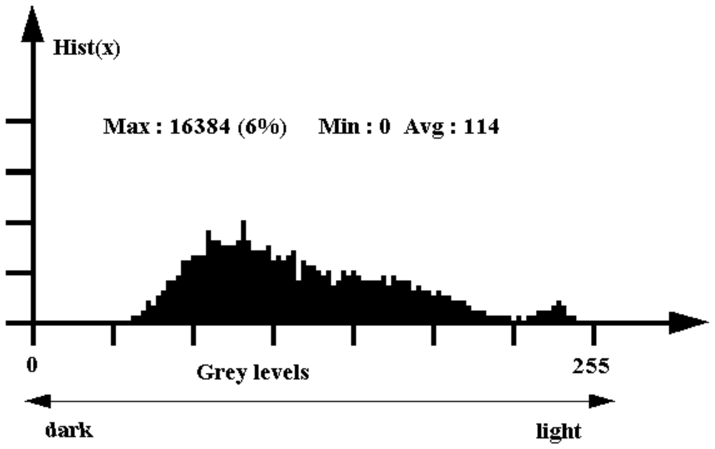


Figure 6.2: Sample image and histogram, from [Emir1997]

6.1.1 Histogram stretching

A commonly used technique to increase an image's contrast is *histogram stretching* [Bank1990]. It scales all the gray levels within an image between the minimum and maximum gray levels used. The columns of the histogram are distributed over the full spectrum. Consider the input image I as an $R{\times}C$ matrix, where $I(i,j)$ represents the gray level at the *ith* row and *jth* column. Denote *G* the output image, f_{max} and f_{min} the maximum and minimum gray levels in the input image *I* respectively and g_{max} the maximum gray level for the given image format[31]. The following equation is a linear transformation for histogram stretching [Cast1996]:

$$\forall i \in \{1..R\}, j \in \{1..C\}: \quad G(i,j) = \frac{g_{\max}}{f_{\max} - f_{\min}} (I(i,j) - f_{\min}) \tag{6.1}$$

6.1.2 Normalization

A more general image processing algorithm is the so-called *normalization*. The variance and mean gray level of an image can be normalized to desired values with this procedure. First define the mean gray level *M*(*I*) and the variance *V*(*I*) of the image *I* (as above) [Hong1996a]:

$$M(I) = \frac{1}{RC} \sum_{i=0}^{R-1} \sum_{j=0}^{C-1} I(i,j) \tag{6.2}$$

$$V(I) = \frac{1}{RC} \sum_{i=0}^{R-1} \sum_{j=0}^{C-1} \left(I(i,j) - M(I) \right)^2 \tag{6.3}$$

Be furthermore M_0 and V_0 the desired mean gray level and variance respectively, then the normalized output image *G* is defined as follows [Hong1996a]:

$$\forall i \in \{1..R\}, j \in \{1..C\}: \qquad G(i,j) = \begin{cases} M_0 + \sqrt{\dfrac{V_0 \left(I(i,j) - M \right)^2}{V(I)}} & \text{if } I(i,j) > M \\ M_0 - \sqrt{\dfrac{V_0 \left(I(i,j) - M \right)^2}{V(I)}} & \text{otherwise} \end{cases} \tag{6.4}$$

The effect of normalization is evident in the following figure.

[31] Usually, sensing devices provide 8-bit gray-scale images, which would result in $g_{\max} = 255$.

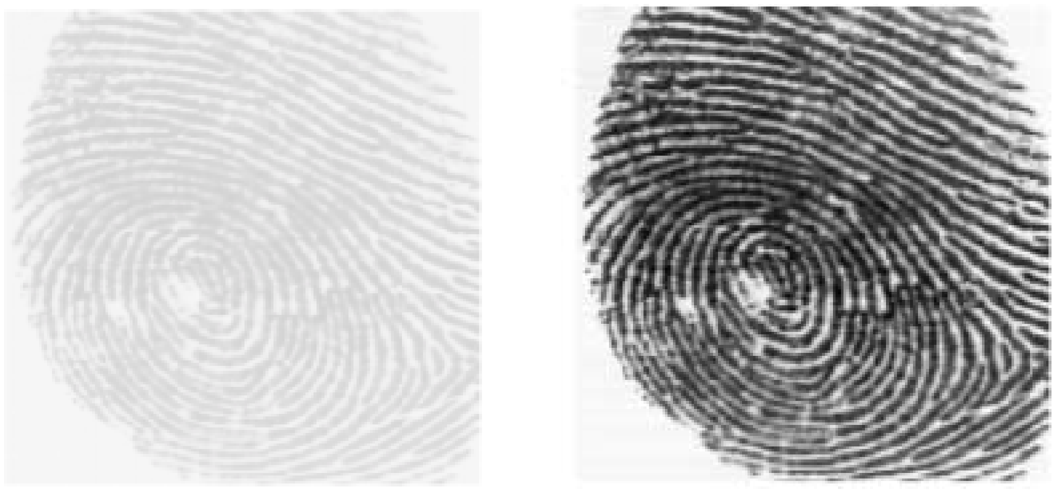


Figure 6.3: Raw image and normalized representation

6.1.3 Image smoothing

Image smoothing techniques are used for noise reduction and are often referred to as *low pass filtering* [Cast1996]. They can be implemented either in the spatial domain or in the frequency domain[32]. In the spatial domain, there are two commonly used smoothing techniques which are *neighborhood averaging* and *median filtering*. In neighborhood averaging or *mean filtering*, each pixel of the image is replaced by the average of 3×3, 5×5 or any other masks. Be g_0 the gray value of a pixel p_0 to be smoothed, g_1 .. g_k the gray values of its k neighbors in a $n \times n$ region –which means $k = n^2-1$– respectively and g_s denotes the gray value of the smoothed image. The mean averaging filter operates by computing the following sum:

$$g_s \quad = \quad \sum_{i=0}^{n^2-1} g_i \tag{6.5}$$

In median filtering, each pixel of the image is replaced by the median value of 3×3, 5×5 or other odd-sized masks. The median value is obtained by sorting the values in ascending or descending order and taking the value at the center position. For example, in a 3×3 mask, the median value is the 5th largest value, in a 5×5 mask the median value is the 13th largest value. Denote med(A) the median value in A, the following is a definition of median filtering using an $n \times n$ mask:

$$g_s \quad = \quad \mathrm{med}\{g_i,\ i = 0..\, n^2 - 1\} \tag{6.6}$$

The neighbors of a pixel p_0 are shown in figure 6.4a for a 3×3 and figure 6.4b for a 5×5 mask.

[32] Frequency domain filtering is addressed in subsection 6.4 below.

p_2	p_3	p_4
p_1	$\mathbf{p_0}$	p_5
p_8	p_7	p_6

(a)

p_{12}	p_{13}	p_{14}	p_{15}	p_{16}
p_{11}	p_2	p_3	p_4	p_{17}
p_{10}	p_1	$\mathbf{p_0}$	p_5	p_{18}
p_9	p_8	p_7	p_6	p_{19}
p_{24}	p_{23}	p_{22}	p_{21}	p_{20}

(b)

Figure 6.4: Neighbor pixels in an image

The smoothing filter may also take the gray values of a pixel's neighbors into account by computing a weighted average. Be g_c the gray value of a pixel p_0 to be smoothed, g_1 .. g_{24} the gray values of its 24 neighbors in a 5×5 region respectively, w_0, w_1 and w_2 weighting constants and g_s denotes the gray value of the smoothed image. The following formula is a useful generalization of (6.4):

$$g_s \;=\; w_0 \cdot g_0 + w_1 \cdot \sum_{i=1}^{8} g_i \;+\; w_2 \cdot \sum_{i=9}^{24} g_i \tag{6.7}$$

6.2 Flow Orientation Field

Considering the fingerprint as a textured image, it is possible to define a *directional image* according to the local direction of the ridge a pixel resides on. The concept of directional images was developed by Mehtre [Meht1987]. The directional image of a fingerprint is useful for filtering, segmentation and classification. Consider again the input image I as an $R \times C$ matrix, where $I(i,j)$ represents the gray level at the *ith* row and *jth* column. Denote D the directional image, n the number of pixels to use for direction estimation and d the direction, which is quantized at N possible levels. The following formula gives a definition for the directional image D with $I_d(i_k, j_k)$ the gray value of I in a particular direction [Meht1987]:

$$\forall i \in \{1..R\}, j \in \{1..C\}: \quad D(i,j) = \min_{d=0..N-1} \sum_{k=1}^{n} \left| I(i,j) - I_d(i_k, j_k) \right| \tag{6.8}$$

Figure 6.5 shows a sample fingerprint image and its directional image. Every gray-scale value in the directional image represents a certain direction.

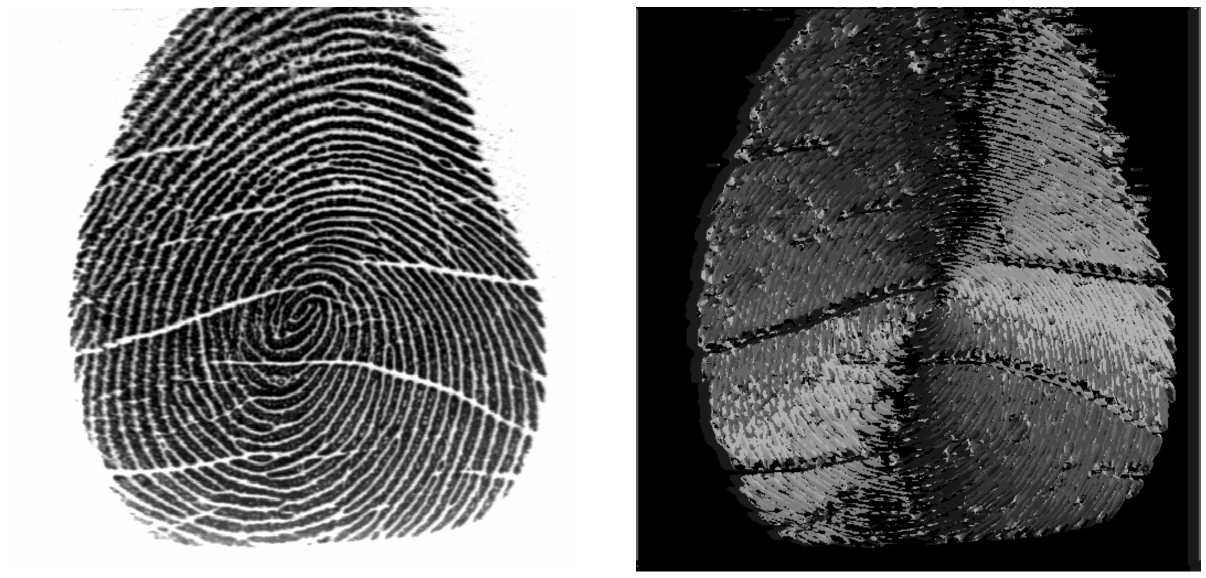


Figure 6.5: Raw image and pixel-wise directional image[33]

This *pixel-wise directional image* is computationally complex for subsequent filter operations. To speed up filtering, a *block-wise directional image* is advantageous [Rath1995]. The directional image is divided into blocks of 16×16 or 32×32 pixels and the orientation of the block is set to the most common direction of its pixels. A least mean square estimation algorithm for the block directional image is proposed in [Hong1996a]. The outcome, also referred to as *flow orientation field* [Rath1995b], may need some post-processing in terms of smoothing the directions for noisy regions with a median filter. It is also possible to reduce the window size locally, if the directions from neighboring segments show large deviations, which means that more precision is needed [Wats1994].

A sample raw image and its block directional image are shown in figure 6.6. The smoothed flow orientation field computed with a 3×3 median post-processing is shown superimposed on an original fingerprint image in the figure 6.7. In the mapped image, a grid of width 16×16 is laid over the image to make clear the block direction for every tile.

[33] A 3×3 median smoothing filter was applied to the raw image before computation of the pixel-wise directional image. Dark pixels represent directions close to 0, light pixels correspond to directions close to π.

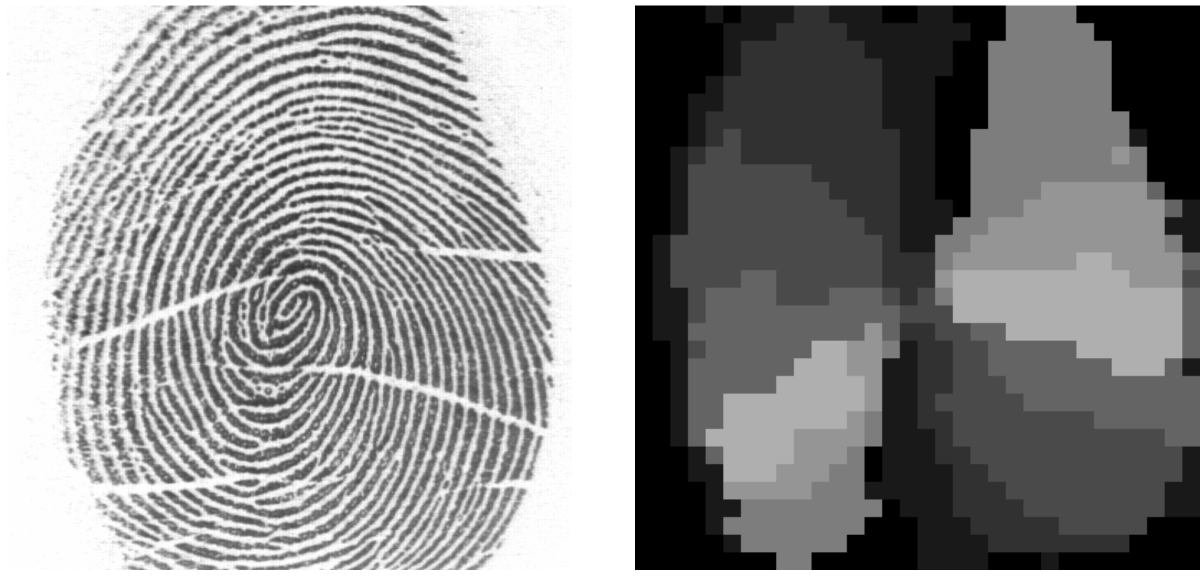


Figure 6.6: Raw image and block directional image

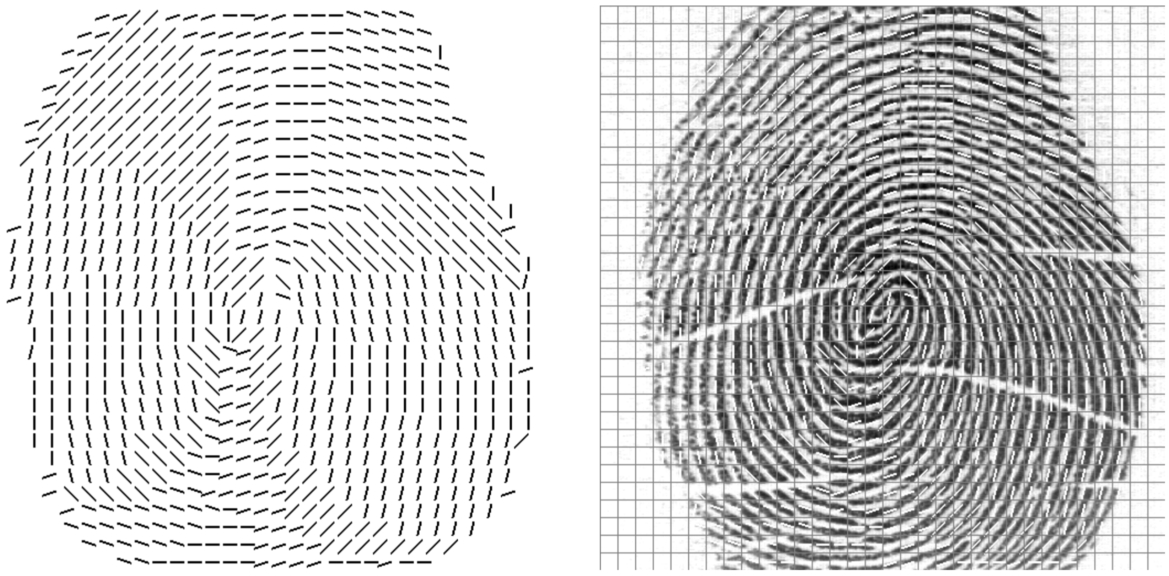


Figure 6.7: Smoothed orientation field superimposed on original image

6.3 Foreground/Background Segmentation

The *region of interest* (ROI) in a fingerprint image is where the finger actually was sensed [Wats1994]. It has to be separated from the background, which contains dirt, noise and other disturbance factors. To find the region of interest, the average contrast in each segment is estimated by computing the gray level variance. If this measure of local image quality exceeds a threshold, the segment is assigned to the ROI. An example of an ROI can be seen in figure

6.8. The ROI is also suitable for saving execution time, because expensive filter operations need not to be applied to the background. The background is also referred to as *unrecoverable region* [Hong1998b].

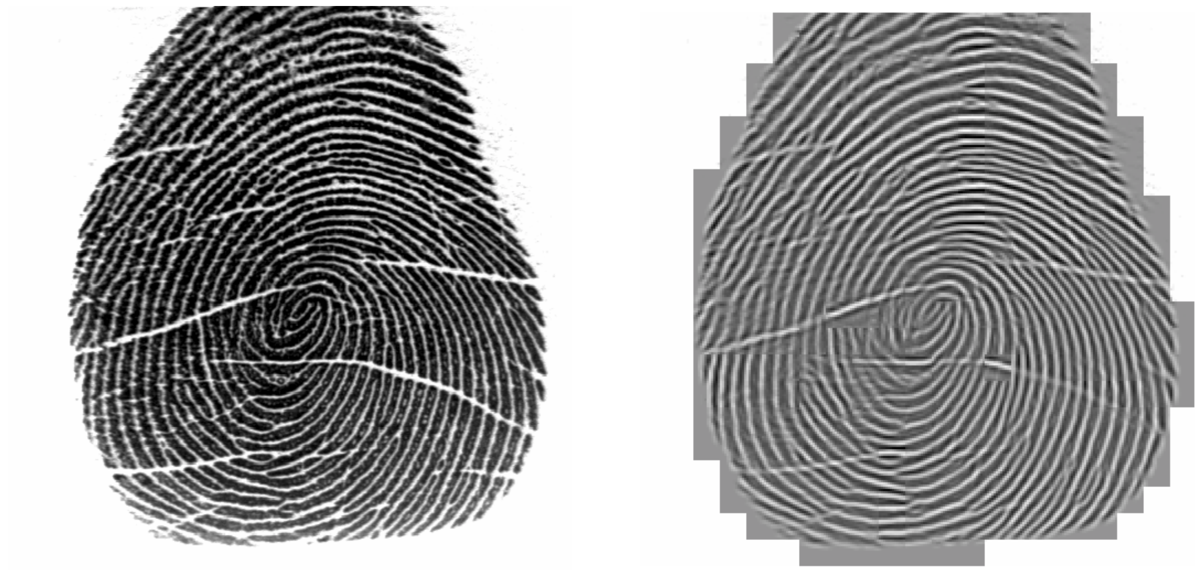


Figure 6.8: Raw image and enhanced image with ROI

6.4 Fourier Domain Filtering

Considering the fingerprint image as a texture, it is striking, that the density of the ridges and valleys has only minor variations throughout the whole image. To take advantage of this characteristic, the target is to enhance the typical frequencies of ridges and to suppress the frequencies belonging to background and noise.

A common technique in image processing, the *Fourier transformation* transforms an image into the frequency domain, where quantization and filter operations can be applied specifically to high and low frequencies. After filtering, the image is transformed back into the spatial domain [Pres1992a, Gonz1993, Wats1994].

6.4.1 Fourier transformation

Consider the digital input image as a $M \times N$ matrix A of real numbers. The *discrete Fourier transform* (DCT) is defined as follows[34] (compare [Wats1994]):

$$\forall (j,k) \in \{1..M\} \times \{1..N\}:$$

$$X_{jk} + iY_{jk} \quad = \quad \sum_{m=1}^{M} \sum_{n=1}^{N} \left(A_{mn} + B_{mn}\right) \; e^{-2\pi i \left(\frac{(j-1)(m-1)}{M} + \frac{(k-1)(n-1)}{N} \right)} \tag{6.9}$$

[34] *B* is initially set to zeros.

Rather than using formula (6.9) directly, the *Fast Fourier Transformation* (FFT) is used to compute the frequency domain image. The reader is addressed to [Pres1992a] or [Knut1969] for a detailed description. The following figure shows an original image and the frequency image computed with the FFT.

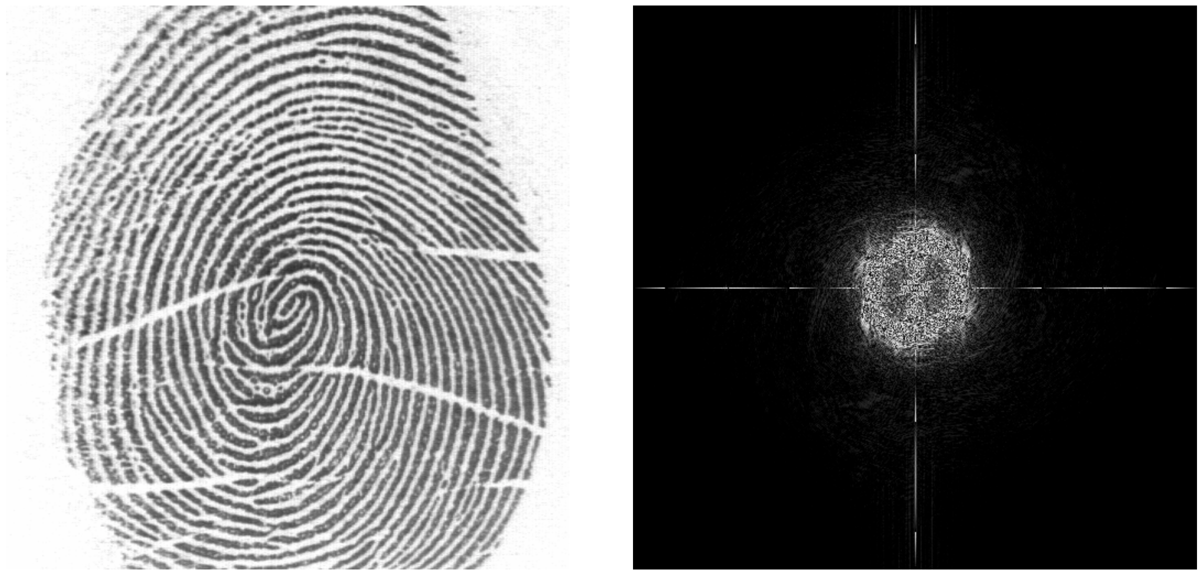


Figure 6.9: Raw image and Fourier transformed image

6.4.2 Low pass filter

An ideal *low pass filter* passes all frequency components unmodified from zero to a predefined cut-off frequency [Cast1996]. Any frequency component above the cut-off frequency is set to zero. Be $D(x, y)$ the Euclidian distance from a point (x, y) to the origin of the frequency plane and D_0 the cut-off frequency. A two-dimensional ideal low pass filter H_{low} is defined by the following equations [Gonz1993]:

$$D(x,y) = \sqrt{x^2 + y^2} \tag{6.10}$$

$$H_{low}(x,y) = \begin{cases} 1 & if\ D(x,y) \le D_0 \\ 0 & if\ D(x,y) > D_0 \end{cases} \tag{6.11}$$

The filter function H_{low} is applied pixel-wise to the transformed image in frequency domain. When the image is transformed back using the inverse Fourier transformation, any high frequencies considered as noise have been extinguished. In digital technology, low pass filters are used to clarify signals. For digital image processing in particular, applying a low pass filter means removing any single isolated points and hairlines considered as noise from the input image.

6.4.3 Band pass filter

The *band pass filter* is a useful generalization of the low pass filter and passes a band of frequency components bounded by minimum and maximum frequencies D_{min} and D_{max} unchanged. Any frequency component above D_{max} or below D_{min} is set to zero. With $D(x, y)$ as described above, the band pass filter H_{band} can be expressed as a formula [Gonz1993]:

$$H_{band}(x,y) = \begin{cases} 0 & if\ D(x,y) < D_{\min} \\ 1 & if\ D_{\min} \le D(x,y) \le D_{\max} \\ 0 & if\ D(x,y) > D_{\max} \end{cases} \tag{6.12}$$

The band pass filter is useful in enhancing a digital image by removing hairlines and speckles corresponding to high frequencies and at the same time removing global shadows corresponding to the lowest frequencies.

6.4.4 Butterworth filter

Besides the simple filter design described above, there are a number of filter operations used in digital image processing. An important functional extension is the *Butterworth filter*, which enhances frequencies in the input image scaled by a given frequency D_0. The n^{th} order low pass Butterworth filter B_n can be expressed as [Gonz1993]:

$$B_n(x,y) = \frac{1}{1+[D(x,y)/D_0]^{2n}} \tag{6.13}$$

A choice of other interesting filter operations is found in [Gonz1993, Cat1996, Bank1990] for example. Generalization of Butterworth filters is addressed in [Sele1996].

6.4.5 Dominant frequency filter

Consider equation (6.9) for the Fourier transformation again. The *power spectrum* of the DCT is defined as follows [Wats1994]:

$$P_{jk} = X_{jk}^{\ 2} + Y_{jk}^{\ 2} \tag{6.14}$$

To enhance the dominant frequencies in an image, the elements $X_{jk} + iY_{jk}$ in the frequency domain are multiplied by a power α of the power spectrum elements P_{jk} [Wats1994]:

$$\begin{aligned} U_{jk} &= P_{jk}^{\ \alpha} X_{jk} \\ V_{jk} &= P_{jk}^{\ \alpha} Y_{jk} \end{aligned} \tag{6.15}$$

The inverse Fourier transformation is applied on the new elements $U_{jk} + iV_{jk}$ and the real part of the result becomes the filtered image. This filter type has the ability to highlight the dominant frequencies in an image. Therefore, it is particularly useful for textures.

6.5 Directional Fourier Filtering

Recalling the flow orientation field from section 6.2, the local ridge flow can be used to specifically filter the image in the frequency domain. A directional band pass filter passes frequency components within an oriented area and sets any other components to zero. There are two different approaches to perform directional Fourier filtering. The first possibility is to perform transformation, filtering and inverse transformation on every tile[35] of the size 32×32 pixels for example. The second approach computes the directional filtered images in the frequency domain from the whole image for a bank of directional filters. A voting algorithm based on local direction is applied to choose, which filter response will be used for the composition of the filtered image [Wats1994]. In spite of greater computational complexity, the second method is implemented in this thesis, because it produces considerably better images. In particular, it has turned out that the global algorithm retains minutiae information and does not produce artifacts.

The directional filter uses a predefined orientation mask designed to filter the fingerprint image in a primary ridge direction. The directional mask is rotated to a fixed number of distinct orientations and applied to the FFT of the image, creating different images with the ridge flow enhanced directionally. The inverse FFT for each directionally filtered image is computed. The filtered image is then reconstructed using recorded smoothed block directions to determine from which directionally filtered image to select each pixel value. The following flowchart pictures the procedure with a sample image. Every intermediate filter response is generated with a directional filter on the frequency image. The directional filters are quantized at 16 levels, which corresponds to 8 filter masks. The mathematical description of directional Fourier filtering is given in appendix 11.2. The reader is referred to [Berg1996, Ikon1985, Kunt1985, Sher1989, Sher1994, Wats1996, Capp1999b] for other implementations of directional filters.

[35] Note, that the image was already segmented into tiles for the block directional image estimation.

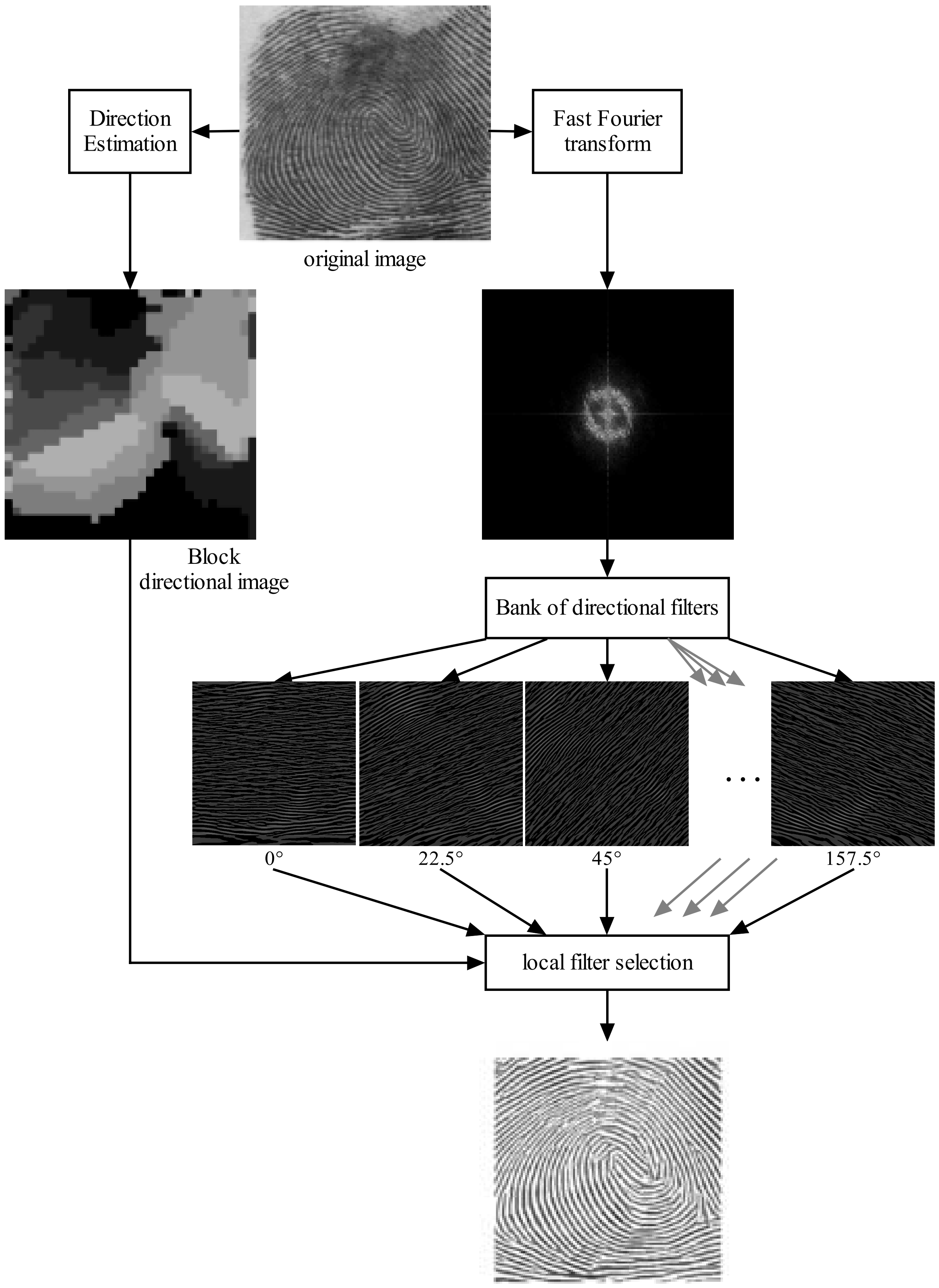


Figure 6.10: Directional Fourier filter

An interesting alternative to the described filters is given in [Hong1996]. Here, a bank of *Gabor filters* is used to enhance the image in terms of removing noise while preserving the ridge structure. Gabor filters have both frequency-selective and orientation-selective properties. They offer optimal joint resolution in frequency and spatial domain. The implemented method from Sherlock [Sher1989], however, already provided very good results both in terms of image enhancement and performance. Therefore, the more complex Gabor filters were not implemented in this thesis.

6.6 Binarization

In order to extract the minutia features of the fingerprint, it is necessary to convert the gray-scale representation into a binary image. Therefore, we simply define a threshold and assign any pixel, the gray value of which exceeds it, as white. All other pixels become black in this process named *binarization* [OGor1999]. An improvement is to use not a fixed threshold according to the image histogram, but an adaptive threshold for any segment defined above. Therefore, the median gray value of every tile is computed and serves as the threshold for that specific tile. Because luminous efficacy and light intensity are not equally distributed in common live scan images, this technique called *regional average thresholding* (RAT) [Akha1995b] produces better binary images than a global method. The best results can be obtained, when the threshold determination is based on slightly wider windows than the binarization, as proposed in the original publication [Akha1995b]. The binary image extracted with this technique from an original fingerprint is depicted in the following figure:

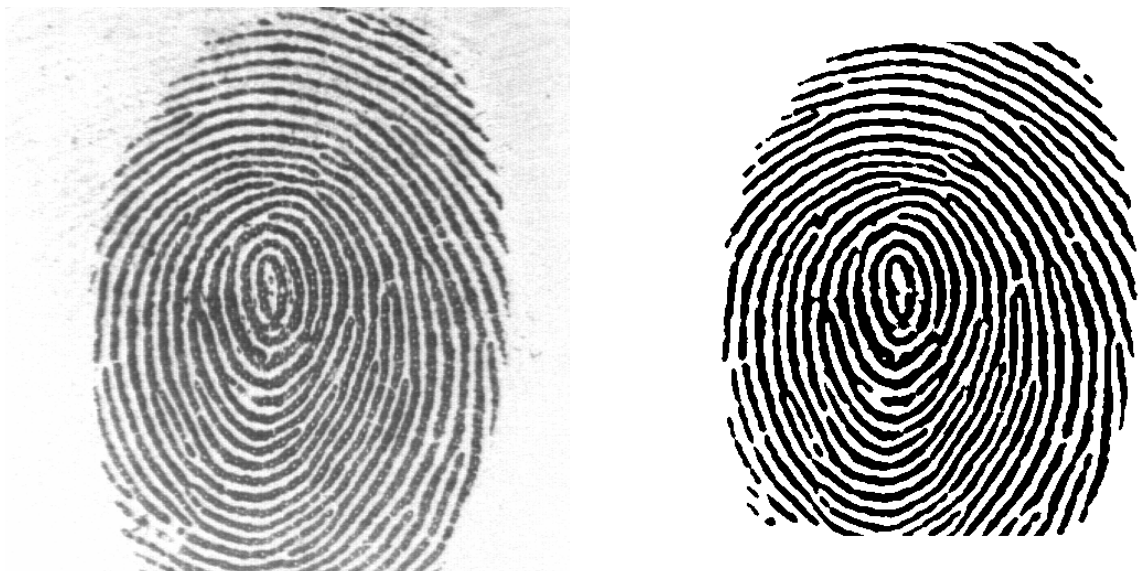


Figure 6.11: Raw image and binarized image

6.7 Thinning

The result of thresholding is a binary image with ridge ends and bifurcations that are visible to the naked eye. To decide automatically, whether a pixel is an end or branch point, the *skeleton*[36] is extracted from the image using a *thinning* algorithm [OGor1999]. Thinning is a standard procedure in image processing. It is important, to maintain the connectivity structure of the objects within the binary image. An iterative thinning algorithm from [Stef1971] works as follows:

Recalling the neighbor pixels from figure 6.4a, functions to measure the sum ϑ of nonzero pixels and the transition sum τ are defined by (be S the binary image):

$$\vartheta(p_0) = \sum_{n=1}^{8} S(p_n) \tag{6.16}$$

$$\tau(p_0) = \sum_{n=1}^{8} S(p_n) - S(p_{n+1}) \tag{6.17}$$

ϑ is to the sum of the nonzero pixels and τ the number of black and white transitions in the ordered neighbor set. Starting from the binary image in S, the following rules are applied to all pixels p_0 in S and iterated until a one-pixel wide skeleton rests:

1) $1 < \vartheta(p_0) < 7$

2) $\tau(p_0) = 2$

3) $p_1 p_5 p_7 = 0$ or $\tau(p_7) = 2$

4) $p_3 p_5 p_7 = 0$ or $\tau(p_5) = 2$.

If all four rules are valid for a ridge pixel in S, it is assigned a background pixel. Otherwise, the algorithm proceeds with the next pixel.

A sample fingerprint and the thinned ridge map extracted after the steps described above is pictured in the following figure. *Connectivity post-processing*[37] and *spike removal* (see below) were carried out on the raw skeleton [Maio1995, Stos1994].

[36] Also referred to as *ridge map* [Jain1997a].

[37] It is very important to maintain the connectivity structure in the fingerprint image. Post-processing means that certain rules are used to correct the raw skeleton. Neighboring branches, that seem to belong together, are connected and links, considered to be artificial, are disconnected.

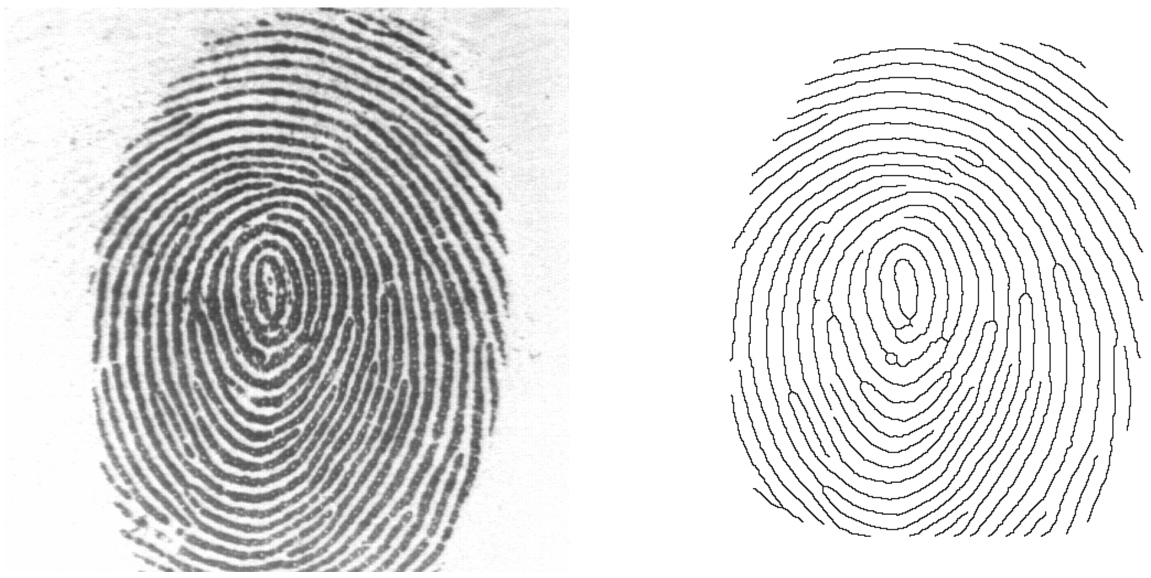


Figure 6.12: Raw image with thinned ridge map

In the above picture, the ridges have been thinned, which is the standard procedure in fingerprint image processing. Black ridges on white background is also the common setting for live scanners. Sometimes, it is necessary to thin the valleys as will be outlined in chapter 7. Therefore, the thinning algorithm is applied to the inverted image. The result of thinning the valleys from the raw image above is depicted in the following figure. End points correspond to ridge bifurcations and branches correspond to ridge ends in this map. On the right hand side of the figure, the valley skeleton is superimposed in red color on the ridge skeleton.

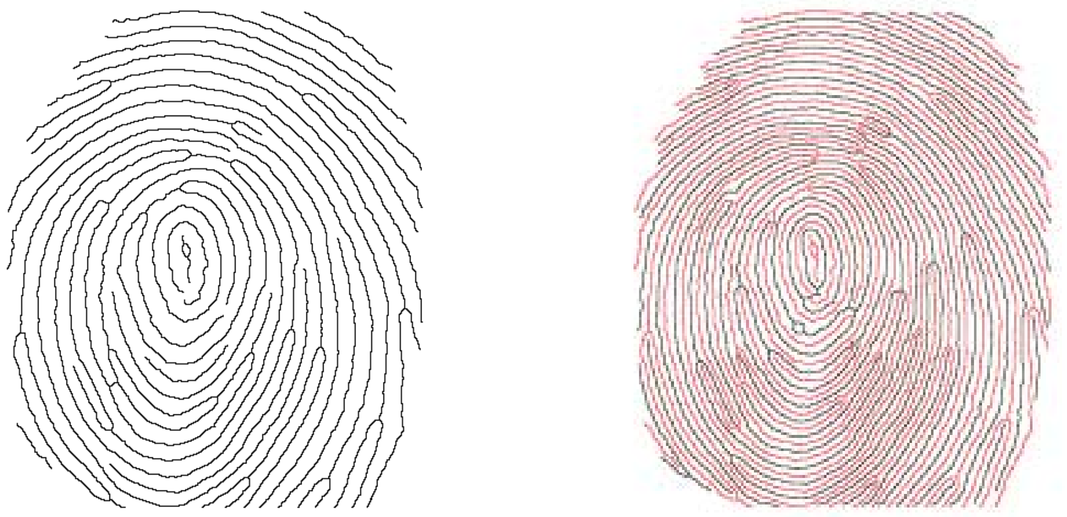


Figure 6.13: Valley skeleton (left), superimposed red on ridge skeleton (right)

An alternative to the thinning algorithm described above is given in [Akha1995a]. Thinning is carried out by a ridge line following in this approach. The results of the implementation are comparable to the iterative algorithm with a slightly better performance.

6.8 Feature Extraction

It is a trivial task to find the minutia in the thinned fingerprint image, when looking at the eight surrounding neighbor pixels. Obviously, every pixel with one and only one neighbor black[38] must be a ridge end. Consequently, if a pixel has three or more black neighbors, it is a bifurcation. The following figure (6.14) shows a fragment from the ridge map with one bifurcation and one ridge end.

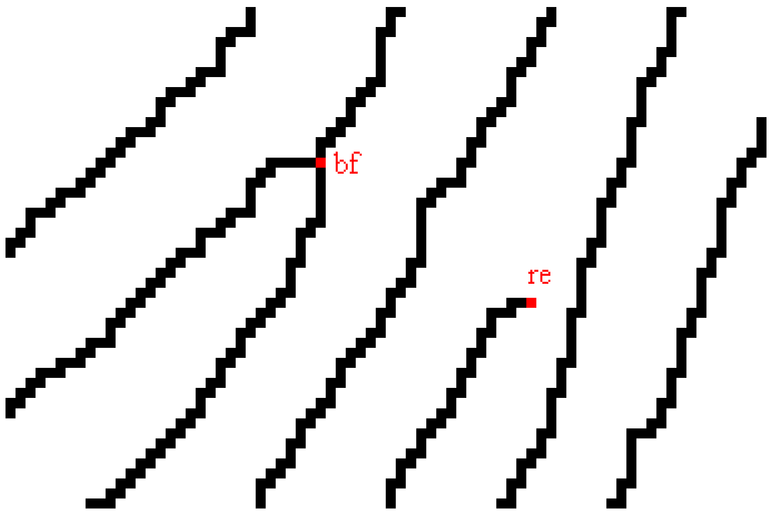


Figure 6.14: A magnified skeleton fragment with a bifurcation (bf) and a ridge end (re)

Some post-processing heuristics must be applied to the minutiae candidates to produce a suitable skeleton. From a close inspection of the thinned image it is striking that the thinning algorithm introduced some *spurious minutiae* that are no true end or branch points in the fingerprint. By tracking from every candidate end point along the ridge, the starting candidate will only be kept, if the path length exceeds a threshold value before another candidate minutia is reached.

Other common factors of disturbance are the cuts and wrinkles in a fingerprint. They produce artificial ridge ends that have to be eliminated. Every pair of minutia candidates that lies close together with the ridge ends pointing at each other must be removed. The above tracking procedure is also applied to the bifurcation candidates to remove ridge breaks. All three directions must be considered here. Groups of ridge ends lying on a straight line are also considered to stem from cuts and removed from the candidate list.

[38] Without loss of generality, it is assumed that black pixels represent ridges and white pixels correspond to background or valleys.

The procedures are called *spike removal* and *ridge break elimination* [Rath1995b]. A sample ridge map with many spikes and ridge breaks is shown in the following figure.

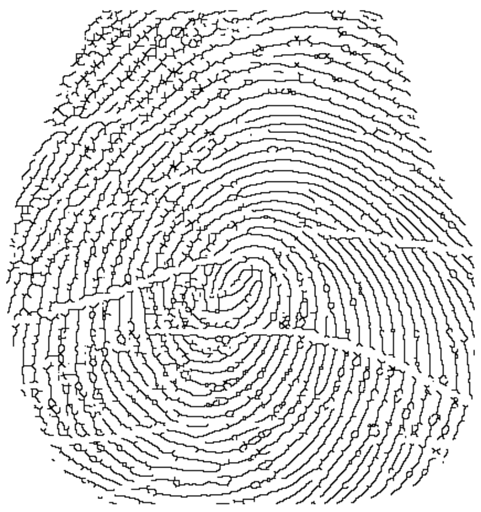


Figure 6.15: A thinned ridge map with spikes and ridge breaks

The minutia direction is computed using a subpart of the ridge it resides on. The tangent is determined and the angle it forms with the horizontal axis becomes the minutia's orientation [Jain1997a]. The mean direction of the segment, in which the minutia lies, serves as a non-oriented approximation. For a ridge end, the tangent is the elongation of the ridge pointing away from it. A bifurcation got three arms meeting in one point. One forks up into two shoulders encompassing an acute angle. The tangent is built as the elongation of the furrow enclosed by these ridges. For the sake of simplicity, the bifurcation could also be considered as an end point in the inverted image. The minutia direction is always measured as the angle that the tangent forms with the horizontal direction[39]. It is suitable to quantize the minutia direction to 32 or 64 levels [Jain1997a].

Another approach to minutia extraction is presented in [Maio1996]. Maio and Maltoni extract the ridge ends and bifurcations directly from the gray-scale image by "sailing" along the ridges until they terminate or interfere. This technique is claimed to be superior both in terms of accuracy and performance [Maio1996]. It was not implemented in this work for two reasons. First of all, the main focus of this thesis is not feature detection but embedded matching and robust feature extraction was also possible with the algorithms used. Secondly, the concepts for pore extraction are designed for the methods described above and do not apply to direct gray-scale minutia extraction.

Ridge counts can be extracted for the minutiae found by inspecting the binarized image or the skeleton. The number of black and white transitions of every line linking two minutiae

[39] See also figure 5.8 in previous chapter.

represents a ridge count [Guna1991, Cost1994]. The most stable ridge counts cross the papillary ridges orthogonally. A common problem with automated ridge count determination is overestimation due to crossing of pores. The pore detection algorithms presented in chapter 7 will be used to enhance the results of ridge count evaluation. To cut down computation effort, redundant counts are only determined once. The total ridge counts are stored in a triangular matrix. If the template must be stored in a smart token with limited memory, only the ridge counts to nearest neighbors are used.

6.9 Singular Point Detection

Sometimes, it is necessary to automatically detect the singular points introduced in subsection 5.2 in a fingerprint image. As already mentioned, the main application of core and delta points is fingerprint classification. The fingerprint verification algorithms developed is thesis do not require classification of fingerprints. The core point instead will be used as a reference point to align the minutiae point pattern in subsection 8.5. It is therefore necessary to understand the singular point detection algorithm.

Several algorithms have been proposed for automated singular point detection [Sher1993, Hong1999a, Rao1978, Rao1990, Karu1995, Hung1996, Srin1992, Jian1991]. Most of the concepts rely on the directional image to estimate the center of maximum curvature, which is considered to be the core of a fingerprint.

The following is a brief description of the singular point detection algorithm presented in [Hong1999a], which was implemented in this thesis. The *Poincaré index* is used to define core and delta points, that have to be distinguished from ordinary points in the fingerprint. Let $\psi_x(.)$ and $\psi_y(.)$ represent the x and y coordinates of a closed digital curve of length N_ψ. The Poincaré index at pixel (i, j) is defined as follows [Sher1993, Hong1999a]:

$$Poincaré(i,j) = \frac{1}{2\pi}\sum_{k=0}^{N_\Psi}\Delta(k) \tag{6.17}$$

where

$$\begin{aligned}\Delta(k) &= \begin{cases}\delta(k), & \text{if } |\delta(k)| < \frac{\pi}{2}\\ \pi+\delta(k), & \text{if } \delta(k) \le -\frac{\pi}{2}\\ \pi-\delta(k), & \text{otherwise,}\end{cases}\\ \delta(k) &= \mathrm{O}(\Psi_x(i'),\Psi_y(i')) - \mathrm{O}(\Psi_x(i),\Psi_y(i)), \qquad (6.18)\\ i' &= (i+1)\bmod N_\Psi.\end{aligned}$$

and $\mathrm{O}(i, j)$ represents the smoothed orientation field at pixel (i, j).

In simple words, the Poincaré index measures the change in direction at a certain point in the fingerprint image using its smoothed directional image. The change of direction naturally is a measure for the curvature at that point. The following figure illustrates the Poincaré indices of the distinguished point classes.

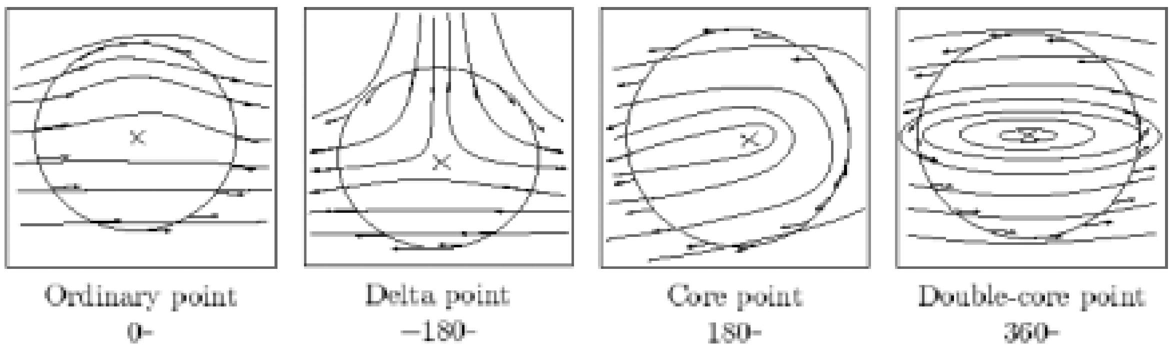


Figure 6.16: Poincaré index and definition of ordinary, core and delta point, from [Karu1995]

To determine the positions of cores and deltas, the algorithm proceeds as follows. The first step is to compute the Poincaré index for each pixel in the smoothed orientation field. Pixels with an index of ½ or -½ are marked as core or delta candidates respectively. The centroid of a connected component with all pixels marked is then considered as the position of the core or delta if its size exceeds 7 pixels [Hong1999a]. Extra rules are to assume two cores, if the connected component is larger than 20 elements and that the smoothing of the directional image is repeated iteratively, if more than two cores or deltas are detected.

If live images are used instead of rolled fingerprints, usually not the whole image is covered with the fingerprint image. It may therefore occur, that artificial core or delta points are detected in the border of the fingerprint impression. An enhancement to the algorithm for the use with live fingerprints is to operate only in the previously determined region of interest as defined is subsection 6.3 above. The following figure depicts sample fingerprint images with the detected core and delta points. Cores are highlighted with circles and deltas are marked with triangles. The resulting classification[40] of every fingerprint is given below the image, respectively.

[40] Fingerprint classification was introduced in section 5.3. The relative positions of cores and deltas in a fingerprint are sufficient to solve the classification problem into five basic classes.

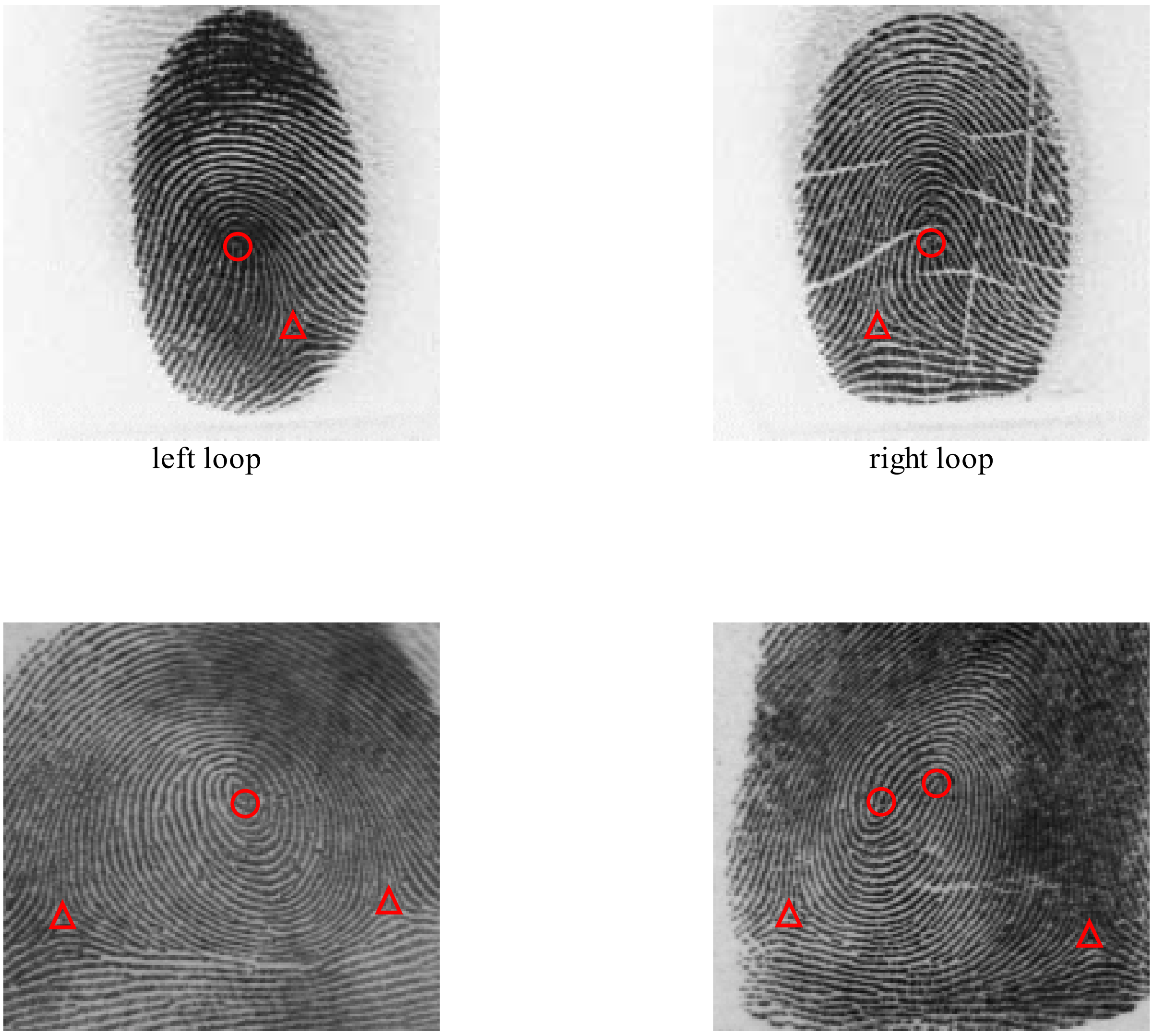


Figure 6.17: Fingerprint images with core(O) and delta (Δ) points marked

A hierarchical approach to singular point detection is proposed in [Hung1996]. Here, the block directional image is compared with fixed local orientation masks for core and delta to define regions for a search with more precision. This process is iterated until squares of size 2×2 rest, that represent the positions of the cores and deltas. Another method is to estimate normal vectors for every fingerprint ridge. The normal vectors are then tracked to the center of maximum curvature [Rao1990]. The reader is referred to [Hung1996] and [Rao1990] for detailed descriptions of the alternative singular point detection algorithms.

The methods described above either had a lack of stability or were computationally too complex. An efficient method for core detection will be presented in chapter 8.5.