Speed-optimized Fingerprint Image Enhancement for Embedded Systems

Cs.Z. Kertész

Dept. of Electronics & Computers, Transilvania University Braşov, Romania
e-mail: csaba.kertesz@vega.unitbv.ro

***Abstract* — Fingerprint recognition is the most widely used form of biometric identification. Many high performance recognition algorithms exists, which can reliably identify even poor fingerprint images, but most of these algorithms require great processing power implying the use of some computer systems to do the actual recognition. However, more and more applications require fingerprint recognition to be done in small embedded system with low processing power but within an acceptable timeframe. This paper focuses on some possible speed-up techniques for image enhancement and their effect on recognition performance.**

I. Introduction

Fingerprint recognition is regarded as a highly reliable form of personal identification and, as such, is finding its way into many areas of our life. Initially, automatic fingerprint recognition was done by high performance computers needing expensive tools and was used only in small circles.

With the spread of low-cost powerful microcontrollers, fingerprint recognition is finding its ways into low-cost independent embedded systems, without the need of computers executing the recognition algorithms in the background.

However, these embedded systems still present large deficiencies compared to computers, resulting in less reliable and slower recognition. In this paper, I will examine some methods of speeding up the execution of these algorithms and their impact on recognition performance.

The most important characteristic of microcontrollers, which should be taken into account, is the lack of a floating point unit. This means that floating point calculations must be emulated in software and, as such, are very time consuming. A fixed point representation can be used instead to speed up calculations on the expense of lower precision. Even so, complex calculation like trigonometric functions or exponentials are rather slow, so the number of such calculations must be kept at a minimum.

The most popular approach for fingerprint recognition with the lowest computational complexity is based on minutiae-matching. Local features of the fingerprint image such as ridge ending and ridge bifurcation – called minutiae – are compared with a stored template of minutiae similarity value [1]. The main steps of this approach are presented on Fig. 1.

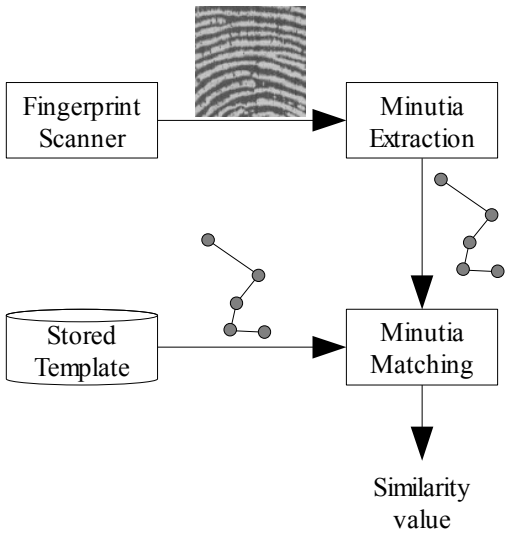

Figure 1. Fingerprint recognition by minutia matching

Minutiae extraction is usually done on the binary skeleton of the fingerprint image, therefore not causing performance degradation due to the lack of floating point unit. However, captured fingerprint images present a large quantity of noise mainly caused by sweat pores on ridge lines and also noise introduced by the fingerprint sensor (a typical fingerprint image captured with a capacitive sensor is presented on Fig. 2). Ridge skeleton cannot be reliably extracted from such images so a

Figure 2. Fingerprint image captured with a capacitive sensor

preceding image enhancement step is needed to regularize ridge lines.

Unfortunately the enhancement step implies a series of complex operation including trigonometric and exponential calculation. This means the enhancement step must be seriously optimized to be practical for implementing in microcontrollers.

II. Fingerprint image enhancement

Fingerprint images due to the nature of ridge and valley structures present a directional flow-like pattern. The best method to enhance these images is to use a filter, which takes into account the textural features, namely the ridge orientation and the ridge frequency.

Many oriented filtering methods were proposed to perform fingerprint image enhancement, the most widespread of them being the use of even-symmetric Gabor filter introduced by Hong et al [2].

The Hong enhancement is thought to be giving the best result for fingerprint image enhancement, and although many improvements were proposed for increasing the output reliability [3][4][5], the method is basically formed from several steps presented on Fig. 3.

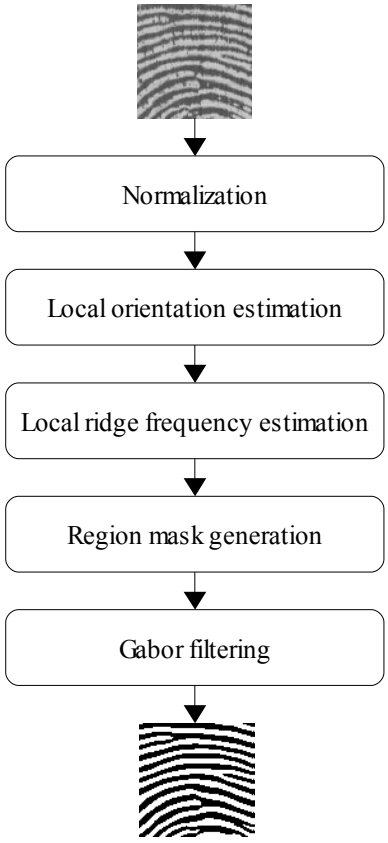

Figure 3. Gabor filtering method

The fingerprint image is divided into non-overlapping blocks of *w*x*w* (usually 8×8 or 16×16) pixels, for which a local orientation and a local ridge frequency is calculated.

The orientation θ for every block *(i,j)* is calculated by using the least mean square approach proposed by Rao et. al. This is based on the calculation of local gradients (G_x and G_y) for every pixel in the block by the use of some kind of gradient operator like the Sobel mask. After that, the local orientation is calculated by (1) (2) and (3):

$$V_x(i,j)=\sum_{x=i-\frac{w}{2}}^{i+\frac{w}{2}}\sum_{y=j-\frac{w}{2}}^{j+\frac{w}{2}} 2\cdot G_x(x,y)\cdot G_y(x,y) \qquad (1)$$

$$V_y(i,j)=\sum_{x=i-\frac{w}{2}}^{i+\frac{w}{2}}\sum_{y=j-\frac{w}{2}}^{j+\frac{w}{2}} \left[G_x^2(x,y)-G_y^2(x,y)\right] \qquad (2)$$

$$\theta(i,j)=\frac{1}{2}\cdot\arctan\left(\frac{V_y(i,j)}{V_x(i,j)}\right) \qquad (3)$$

The use of double angles is necessary to avoid wrapping problems at 90° and 180°.

The extraction of correct ridge orientation might be compromised in some blocks, especially the ones including some minutiae, making the entire enhancement process to be unreliable exactly in the areas of most significance. Knowing the fact that ridge orientation is varying slowly, a further smoothing step can be included on the local orientations.

The smoothing can be done by using a lowpass filter on a continuous vector field given by (4) and (5):

$$\Phi_x(i,j)=\cos(2\theta(i,j)) \qquad (4)$$

$$\Phi_y(i,j)=\sin(2\theta(i,j)) \qquad (5)$$

The lowpass filtering can be done using any type of filter kernel (for example a 5×5 gaussian kernel). The resulted vectors are converted back to orientation with (6)

$$\theta(i,j)=\frac{1}{2}\cdot\arctan\left(\frac{\Phi_y'(i,j)}{\Phi_x'(j,j)}\right) \qquad (6)$$

For calculating ridge frequency, a 1-D x-signature calculated in a *l*×*w* window, orthogonal to local ridge orientation, can be used (7). The coordinates in the oriented window are calculated with (8) and (9).

$$X(k)=\frac{1}{w}\sum_{d=0}^{w-1} I(u,v) \qquad (7)$$

$$u=i+(d-\frac{w}{2})\cos(\theta(i,j))+(k-\frac{l}{2})\sin(\theta(i,j)) \qquad (8)$$

$$v=j+(d-\frac{w}{2})\sin(\theta(i,j))-(k-\frac{l}{2})\cos(\theta(i,j)) \qquad (9)$$

For a well defined ridge structure, the x-signature gives sinusoidal shape wave. The frequency of this wave can be easily determined by measuring the period of the signal. However, in the blocks containing minutia or other distortions, the x-signature will be distorted as well, so a smoothing step of the frequency image is necessary as in the case of the orientation image.

Gabor filters have both frequency-selective and orientation-selective properties and, therefore, are appropriate to be used as bandpass filters to remove noise but preserve ridge structure. [2]

The even-symmetric Gabor filter has the form (10)

$$G(x,y,\theta,f)=\exp\left\{-\frac{1}{2}\left(\frac{x_\theta^2}{\sigma_x^2}+\frac{y_\theta^2}{\sigma_y^2}\right)\right\}\cdot\cos(2\pi f x_\theta) \quad (10)$$

where:

$$x_\theta = x\sin\theta + y\cos\theta \quad (11)$$

$$y_\theta = y\sin\theta - x\cos\theta \quad (12)$$

σ_x and σ_y are the standard deviations of the Gaussian envelope and are experimentally determined for a given type of fingerprint images; x and y are the indices in the filter kernel ranging $-\frac{W}{2},+\frac{W}{2}$, W being the kernel size.

The result of Gabor-filtering a fingerprint image is presented in Fig. 4.

Figure 4. Result of Gabor filtering

Unfortunately the straightforward implementation of this method is not very suitable for embedded systems because of the large number for trigonometric and exponential operations.

Another approach to Gabor-filtering is proposed in [6], offering better performance while eliminating some of these operations.

A filter bank consisting of 8 Gabor kernels is used to filter the input image. The filters are tuned to 8 evenly distributed orientations at 22.5° apart and a single frequency.

The output image is then calculated from these 8 filtered images using a rather computationally intensive voting algorithm. First a coarse ridge map is extracted from all 8 images, then the local orientation of these ridge maps is calculated. For every pixel an orientation is selected based on the highest local variance in the filtered images and ridge maps. Finally the output image is constructed pixel-by-pixel from the filtered images corresponding to the estimated local orientation.

Although Gabor kernel calculation is avoided by this algorithm, the increased amount of simple calculations (like the 8 filtering and the voting algorithm) makes this approach as unsuitable for embedded systems as the first one.

However, a combination of the two methods, where the local orientation is calculated, after which fixed Gabor kernels are used for filtering, can be successfully employed to reduce the execution time of the image enhancement.

III. Local Orientation Estimation

As we have seen, the local orientation can be calculated from the local gradients with the use of the arctangent function. This is very time consuming in embedded systems without dedicated floating point unit.

However, if fixed orientation Gabor kernels are used for filtering, then there is no need for the exact orientation to be calculated.

The 8 filter kernels are tuned to orientations of -90°, -67.5°, -45°, -22.5°, 0°, 22,5°, 45° and 67,5°. As stated before, when calculating the local orientation from averaging the gradients, a double angle is used to have a proper rolling over at -90°/90°. This means the angles -180°, -135°, -90°, -45°, 0°, 45°, 90° and 135° must be checked against the gradients.

Table 1 shows these angles and the corresponding x- and y-gradients ($\cos\theta$ and $\sin\theta$).

Table 1.
Gradients of the predefined angles

θ	2θ	$\cos(2\theta)$	$\sin(2\theta)$
-90°	-180°	-1	0
-67.5°	-135°	$-\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$
-45°	-90°	0	-1
-22.5°	-45°	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$
0°	0°	1	0
22.5°	45°	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$
45°	90°	0	1
67.5°	135°	$-\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$

If the intermediate angles are approximated by the predefined value, three simple comparison of the

gradients are sufficient for determining the corresponding angle without the need of an arctangent calculation.

The sign of the x- and y-gradients has to be checked and their absolute value to be compared. For example, orientation is considered to be -90° (i.e. between -90° – -67.5°), if both gradients are negative and the absolute value of the x-gradient is lower than the absolute value of the y-gradient. The complete comparison chart is presented on Figure 5.

Also, it has to be noted that the gradient values can be used to determine a region mask: if both gradients in absolute value are below a threshold, then the image block contains no variation and, hence, no useful ridge information. Such blocks can be excluded for further processing.

Another advantage of this method is that all the gradients can be calculated and stored without loss of precision in 32 bit integers, so there is no need for costly floating point representation.

IV. Gabor Filtering

Using predefined Gabor kernels for filtering reduces the execution time because there is no need for the exponential and cosine calculus from (10) and sine and cosine from (11) and (12).

Also, as demonstrated in [7], fixed point computation can be adopted instead of floating point with acceptable errors: in case of 15.16 format fixed point representation, the errors are below 1%, which is more than acceptable if we consider that the output of the Gabor filtering is a binary image (or binarized in the next step).

The main disadvantage of this approach is the use of a single ridge frequency for filtering. Problems occur when the filtering frequency is a harmonic of the actual ridge frequency. In this case a ridge doubling appears introducing 2 false minutiae (ridge bifurcation). To avoid this, filtering kernels were designed for 4 different frequencies (the most probable frequencies for the specific sensor).

Calculating the frequencies for the predefined orientations is much simplified as, for every orientation, a separate pattern for x-signature calculation can be used.

Gabor filtering is done for every image block by choosing the right kernel and executing the 2-D convolution on every pixel in that block.

To further speed-up the filtering, separability of the filtering kernel is applied.

For orientations -90° and 0°, the filter kernel is fully separable: equations (11) and (12) become (13) for -90°, respective (14) for 0°.

$$x_\theta = -x,\ y_\theta = -y \tag{13}$$

$$x_\theta = y,\ y_\theta = -x \tag{14}$$

The filter kernel will then be separable into 1-D filters (15) and (16).

$$G_x(x, f) = \exp\left(-\frac{1}{2}\cdot\frac{x^2}{\sigma_x^2}\right)\cdot\cos(2\pi f x) \tag{15}$$

$$G_y(y, f) = \exp\left(-\frac{1}{2}\cdot\frac{y^2}{\sigma_y^2}\right) \tag{16}$$

In this case, W+W multiplications and additions are necessary instead of W×W, meaning a 5.5 times speed-up in case of a kernel size W=11.

The kernels for other orientations are not fully separable, but the approach presented in [8] can be applied simply for orientations -45° and 45°. This approach assumes filtering with the kernels for -90° and 0°, but on an image rotated by 45°. With 45° rotation, no interpolation is necessary, the filtering will be done on the diagonals with a filtering kernel resampled to $\sqrt{2}$ times larger displacements.

For other orientations rotating image is not practical, as interpolation is necessary, which introduces too much calculations due to the trigonometric functions involved.

For these orientations singular value decomposition (SVD) of the filter kernels can be used for reducing the number of calculations. All of the Gabor kernels tested proved to be of rank 2 meaning 2×(W+W) calculations are necessary instead of W×W (64% faster filtering).

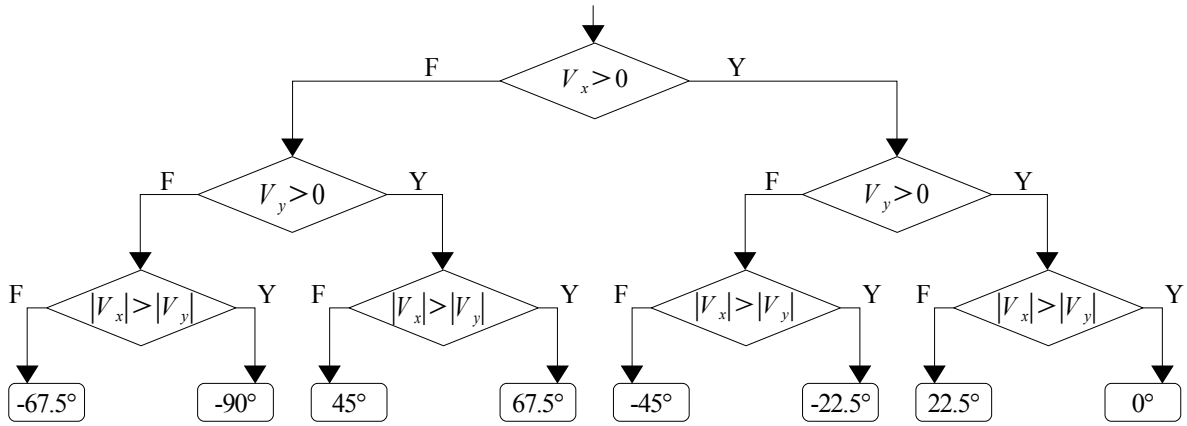

Figure 5. Comparison chart for orientation estimation

Figure 6. Result of fixed kernel filtering

The result of this fixed kernel filtering is presented in Fig. 6. As it can be seen, the result is similar to the Gabor filtered image from Fig. 4, but there are more ridges connected instead of independent ridge portions. This is due to the fact that larger standard deviations of the Gaussian were used to compensate the lack of exact ridge frequency.

V. Experimental results

Experiments were carried out on the MDFP200 development kit containing:

- MBF200 capacitive fingerprint sensor
- MB91F302 microcontroller
- 8MB SDRAM and 2MB Flash

The MBF200 fingerprint sensor is a 500dpi resolution 256x300 pixel capacitive sensor with integrated ADC, offering good quality fingerprints. The captured images have sufficient similarity (like gray level distribution and ridge characteristics), making the use of fixed kernel filtering easy.

The microcontroller used is a 32 bit RISC type processor with external SDRAM interface running at 68 MHz. The 32 bit architecture offers a good precision in 15.16 format fixed point calculation and enough word size to avoid overflow in integer additions.

The algorithms were implemented in C (for high portability) and without extreme optimizations.

Execution times of the original Gabor filtering algorithm and of the fixed kernel filtering are presented in Table 2. The performance of the algorithms were also evaluated using the Goodness Index (GI) [2], which is basically the ratio of the difference of the correct and false minutiae to the total number of minutiae, as compared with the minutiae identified by a human expert. The higher the GI the better the performance. A GI=1 means that all the minutiae in the fingerprint were correctly extracted and no false minutiae were found.

Table 2.
Execution times of the discussed algorithms

Algorithm	Average execution time	GI
Gabor filtering (floating point)	54.98 s	0.60
Gabor filtering (fixed point)	8.30s	0.59
Fixed kernel filtering	0.22 s	0.53

The execution times were averaged for 40 different fingerprint impressions, but they are still highly dependent on the contents of the image, so these values are only estimative, giving only an idea of the speed-up caused by the use of fixed kernel filtering.

VI. Conclusions

The execution times for image enhancement are greatly reduced in the case of fixed kernel filtering compared to dynamic kernel filtering. However, the reliability of the filtering is also somewhat lowered, resulting in a higher number of false minutiae. Some of the false minutiae can be removed in a later step; however, this increases the overall execution time of the fingerprint recognition.

The speedup resulted from this technique still outweighs the loss of reliability if we consider the intended application area: small embedded system for fingerprint identification where almost instantaneous response time is more important than the occasional false reject, e.g. simple door control system.

Although serious speedup of the enhancement was achieved, the algorithm was implemented in C, leaving some space for instruction level optimizations for the final product. Also, the memory consumption was not taken into account; minimizing it involves future work.

References

[1] Maltoni, D., Maio, D., Jain, A., Prabhakar, S., "Handbook of Fingerprint Recognition", 2002

[2] Hong, L., Wan, Y., Jain, A.K., "Fingerprint Image Enhancement: Algorithm and Performance Evaluation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 777-789, August, 1998

[3] Greenberg, S., Aladjem, M., Kogan, D., Dimitrov, I., "Fingerprint Image Enhancement using Filtering Techniques", *15th International Conference on Pattern Recognition*, vol. 3, Freeman, WT, 2000

[4] Kim, B.G., Kim, H.J., Park, D.J., "New Enhancement Algorithm for Fingerprint Images", *16th International Conference on Pattern Recognition*, vol. III, Washington, DC, 2002

[5] Yang, J., Liu, L. Jiang, T., Fan, J., "A modified Gabor filter design method for fingerprint images enhancement", *Pattern Recognition Letters*, vol. 24, pp. 1805-1817, August, 2003

[6] Hong, L., Jain, A.K., Pankanti, S., Bolle, R., "Fingerprint Enhancement", *IEEE Workshop on Applications of Computer Vision*, Sanratosa, FL, 1996

[7] Chen, J., Moon, J.S., Fong, K.F., "Efficient Fingerprint Enhancement for Mobile Embedded Systems", *Biometric Authentication Workshop*, Prague, 2004

[8] Areekul, V. et al., "Separable Gabor Filter Realization for Fast Fingerprint Enhancement", *International Conference on Image Processing*, vol. III, Genova, 2005