Efficient Hardware Implementation For Fingerprint Image Enhancement Using Anisotropic Gaussian Filter

Tariq Mahmood Khan, *Member, IEEE*, Donald G. Bailey, *Senior Member, IEEE*, Mohammad A. U. Khan, *Member, IEEE*, and Yinan Kong, *Member, IEEE*

***Abstract*— A real-time image filtering technique is proposed which could result in faster implementation for fingerprint image enhancement. One major hurdle associated with fingerprint filtering techniques is the expensive nature of their hardware implementations. To circumvent this, a modified anisotropic Gaussian filter is efficiently adopted in hardware by decomposing the filter into two orthogonal Gaussians and an oriented line Gaussian. An architecture is developed for dynamically controlling the orientation of the line Gaussian filter. To further improve the performance of the filter, the input image is homogenized by a local image normalization. In the proposed structure, for a middle-range reconfigurable FPGA, both parallel compute-intensive and real-time demands were achieved. We manage to efficiently speed up the image-processing time and improve the resource utilization of the FPGA. Test results show an improved speed for its hardware architecture while maintaining reasonable enhancement benchmarks.**

***Index Terms*— Fingerprint image enhancement, FPGA, oriented gaussian filter.**

I. INTRODUCTION

BIOMETRICS is a fast-progressing science that deals with human identification by using their traits or characteristics. The science is especially interested in measurable, albeit distinctive, characteristics to label and to some extent describe individuals [1]. These characteristics are often categorized as behavioral versus physiological. Physiological characteristics deal with the shape of the body. Examples include face, DNA, palm print, hand geometry, retina, iris and fingerprints. Behavioral characteristics are related to the pattern of behavior of an individual, including but not limited to typing rhythm, gait, and voice. Among all the characteristics, fingerprints are the most widely used.

Fingerprint identification is one of the oldest biometric techniques and has proven its worth in numerous applications [2]. Every person has a unique and immutable fingerprint that can be acquired by a scanner. A fingerprint consists of parallel ridges and furrows. However, at some points one ridge splits into two (ridge bifurcations) and, at other points, may even terminate and continue no more (ridge endings). These local ridge singularities (deviations from normal parallel behavior), also known as minutiae points, are distinctive [3] and it is these that are primarily utilized for identifying individuals.

Although associating identity with a fingerprint impression can be accomplished through image correlation-based methods [4], more commonly minutiae points are matched [5], [6]. The minutiae-based representation consists of the set of ridge endings and bifurcations along with their spatial location and direction on the fingerprint. Having a small template size and high accuracy, the minutiae-based representation is considered better by many experts than correlation-based methods [7]. Minutiae-based fingerprint matching is widely used by both machines and human experts. This representation has now become a standard, mostly used by forensic experts, for the exchange of information between different systems across the world [8].

To make use of a minutiae-based representation, it is essential that the minutiae are extracted accurately. Extraction starts with the acquisition of fingerprints by a scanner. Acquired fingerprint images often show important variations, with poor contrast in some areas and gaps in ridge and valley regions. These occur due to the very process of scanning a finger. The finger's surface is not flat. Consequently there is more pressure on the middle of the finger than of the edges, giving better contrast in the center relative to the edges [5]. This results in a background variation for different regions of the same image, disturbing the ridge and valley contrast. Since the ridge and valley pattern is identified by its gray-level profile, this effect may adversely affect the performance of the subsequent fingerprint recognition algorithms.

Fingerprint image normalization and enhancement is therefore essential pre-processing before minutiae extraction. General-purpose noise-reduction techniques, including local averaging, were not found to be as successful as expected. Their failure could well be attributed to the non-stationary nature [9] of a fingerprint. Filtering at its most abstract level can be considered as applying some prior knowledge to improve the signal-to-noise ratio. For images, a classic prior is smoothness, which implies the application of a low-pass filter to smooth the image. However, a fingerprint has a regular texture with well-defined local orientation and frequency.

Manuscript received January 19, 2016; revised July 5, 2016 and January 7, 2017; accepted February 6, 2017. Date of publication February 17, 2017; date of current version March 27, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Gang Hua.

T. M. Khan and Y. Kong are with the Department of Engineering, Macquarie University, Sydney, NSW 2109, Australia (e-mail: tariq045@gmail.com).

D. G. Bailey is with the School of Engineering and Advanced Technology, Massey University, Palmerston North 4442, New Zealand.

M. A. U. Khan is with the Biometric and Sensor Laboratory, Effat University, Jeddah 22332, Saudi Arabia (e-mail: makhan@effatuniversity.edu.sa).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2671781

To exploit this prior, techniques emerged for fingerprint enhancement that include local neighborhood information during noise filtering. Notable among them is the Gabor filter [1], parameterized with orientation and frequency. The even-symmetric two-dimensional Gabor filter kernel is given by:

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

where

$$x_\theta = x\cos\theta + y\sin\theta$$

$$y_\theta = y\cos\theta - x\sin\theta,$$

θ is the orientation of the ridges and f is the local ridge frequency. x_θ and y_θ respectively are the rotated coordinates with fixed σ_x and σ_y as standard deviations parallel and perpendicular to the ridges respectively. The Gabor filter is a band-pass filter with center frequency f, and bandwidth defined by σ_x and σ_y.

Hong [1] introduced Gabor filtering to enhance fingerprint images and provided a systematic approach to set its parameters in a local neighborhood. There are several studies extended from [10], and most of these focus on improving performance. For example Yang [11] introduced a modified Gabor filter (MGF) resulting in better fingerprint verification.

Unfortunately, Gabor filtering has a high computational complexity. In [12], a set of 8 fixed-orientation-based separable 2-D Gabor filters is introduced to reduce the computational complexity of the conventional Gabor filters used in [1]. Nevertheless, their enhanced image quality was only marginally inferior to the traditional 2-D Gabor filter. Areekul et al. [10] proposed a generalized separable Gabor filter for any orientation based on Hong's work, resulting in lower computational complexity, better enhancement, and less memory space for the Gabor filtering process. They kept the frequency constant because the frequency does not change much throughout the fingerprint image.

One key limitation of Gabor filtering is that it assumes a given ridge frequency. Although this is true for most of the image, it is not true around minutiae points. Consequently, in maintaining the regular texture of the ridges and valleys, the minutiae points are distorted, and this can affect the accuracy of matching. For fingerprint images, the exact estimation of local ridge frequency is a very difficult and time-consuming task, especially in noisy regions and in regions where minutiae and singular points exist. In fact, these singularities, where the curvature is large, do not have any frequency and a wrong assumption can create spurious ridge structures.

To avoid ridge frequency dependence altogether, the prior for fingerprints becomes limited to its high value of anisotropy, a fingerprint can be enhanced by using an anisotropic filter. The pioneering work on anisotropic filters was done by Perona and Malik [13]. They suggested employing the heat equation in a heterogeneous medium for edge enhancement. The scheme allowed both steering and scaling of an anisotropic Gaussian. However, the number of basis filters is large, and the basis filters are nonseparable, increasing the high computational cost. Geusebroek [14] proposed decomposing the anisotropic Gaussian into two Gaussian line filters in non-orthogonal directions. Choosing an axis to decompose the filter along turns out to be extremely efficient from a computing perspective. In a practical setting, not knowing the axis of orientation for each pixel poses a problem. Therefore, a large number of filters are usually applied at different scales and orientations, and the maximum response per pixel over all the filters is accumulated.

Applying a large number of filters commonly requires a significant amount of computing resources. The best way to achieve good real-time performance is to implement it in hardware utilizing parallel processing. Although several efficient FPGA implementations have been presented in the literature for separable as well as non-separable filters, research on the oriented-filter implementation on an FPGA is limited. Fons [15]–[18], Alilla [19], Qin [20] and Garcia [21] used conventional Gabor filters for FPGA-based fingerprint image enhancement. Qin used a 16×16 block for Gabor filter implementation. Fons used a directional 7×7 Gabor filter. For image normalization, both Qin and Fons used a two-pass local mean and a local variance-based normalization method.

For large windows, several decompositions can be used, for example, [22] approximates a large circularly symmetric filter by octagons. Joginipelly [23] proposed an efficient implementation of an oriented Gaussian smoother on an FPGA. They decomposed the 2-D filter into 1-D filters and then used pipelining to obtain higher throughput. Their implementation only has a single orientation; for multiple orientations, they require multiple filters in parallel. This limits the applicability of these filters for fingerprint image enhancement.

In this paper, a decomposition of the anisotropic Gaussian is used in which the image is diffused by first applying a small two-dimensional isotropic Gaussian filter followed by a relatively large anisotropic Gaussian line filter aligned to the ridge direction. We extend the work of Joginipelly by being able to change the orientation of the final line Gaussian on a pixel-by-pixel basis. To implement a real-time system, these algorithms are efficiently enhanced for fixed-point representation and optimized for memory and computational capacity. The significance, or the innovation, of this work is a novel architecture to steer the orientation of an anisotropic Gaussian filter on a pixel-by-pixel basis. We also implement a fingerprint image-enhancement algorithm on an FPGA, which is well suited for real-time systems. The proposed implementation is faster and consumes fewer resources than existing methods.

The rest of the paper is organized as follows. The proposed fingerprint enhancement algorithm is described in Section II. In Section III, the FPGA implementation of the proposed structure is presented. The experimental results, and performance with several datasets, of the proposed method are illustrated in Section IV. In Section V, we make some conclusions and suggest future work.

II. Proposed Method

A. *Image Normalization*

In fingerprints, the input images that are obtained from sensors may have imperfections or poor quality due to

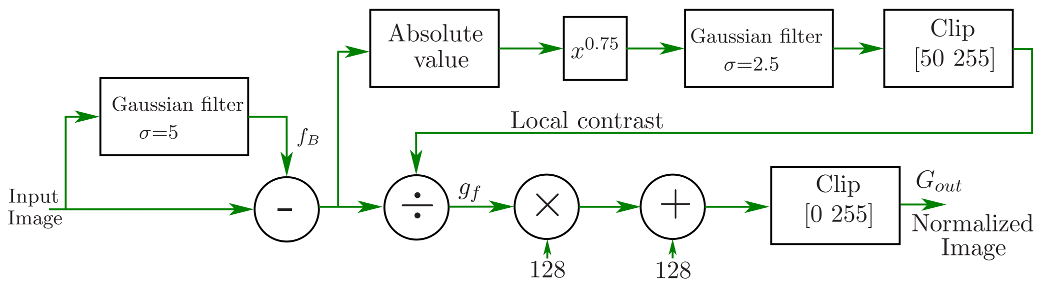

Fig. 1. Block diagram of the proposed local-normalization algorithm.

non-uniformity. The accuracy of fingerprint recognition can be improved considerably by normalizing the image for background variations and contrast before filtering. In the literature, most researchers implement an adaptive normalization algorithm based on a local mean and variance [15]–[18], [20]. This type of normalization requires two processing passes of the whole image [15], [17], [18]. Therefore, a frame buffer is required to perform this task. To eliminate multiple image passes, a new algorithm is proposed, which is not only well suited for hardware implementation but also gives much better results than the local-property-based normalization. A block diagram of the proposed algorithm is shown in Fig. 1.

A Gaussian filter with a window size of 3.5σ is used, where the chosen $\sigma = 5$ is the average ridge width of the fingerprint. The application of the filter results in blending the dominating structure with the background and results in a blurred image that contains the slowly-varying illumination pattern. Then the difference D between the image f and the background estimation image f_B is calculated for every pixel:

$$D(x, y) = f(x, y) - f_B(x, y) \tag{2}$$

This background subtraction process is effectively a high-pass filter that allows the structure of interest to pass. Though the filter provides us with a uniform background, its contrast can vary significantly throughout the image. Therefore, contrast enhancement has to be performed next to normalize object intensities about the background. The magnitude is used to estimate the local contrast. Then a power-law transformation (with $\gamma = 0.75$) compresses the high-contrast pixels about those with low contrast. Then another Gaussian filter, with $\sigma = 2.5$, is applied on the power-law transformed image to average that locally. The resultant image is clipped in between [50-255] to avoid over-enhancing noise and to retain the relative strengths of already high-contrast regions. This provides a measure of the local contrast within the image. The difference image is normalized by this local contrast. The result of this division is scaled by a factor of 128 and offset by 128 to enable negative values to be elevated for accommodation in the dynamic range of the monitor. In the end, the image is clipped to the allowed pixel range [0-255] to give the normalized output image.

B. Orientation Estimation

The patterns present in a fingerprint image demonstrate strong local directionality that has to be taken into account when filtering the image for enhancement. The local ridge orientations $\theta_{x,y}$ are to be estimated from the neighborhood. In the literature, many researchers propose different techniques to derive robust orientation estimation [2], [24], [25]. Although these techniques give excellent orientation estimation they are iterative, making them less suited to hardware implementation. In this paper, the procedure outlined in [26] was adopted for this purpose. First discrete derivatives G_x and G_y in the x and y directions are calculated by employing a Gaussian smoothed kernel, with a small standard deviation to mitigate noise. Then, covariance matrix data for the fingerprint image were calculated for each pixel as $G_{xx} = G_x^2$, $G_{xy} = G_x G_y$, and $G_{yy} = G_y^2$. The covariance matrix elements were further smoothed with a Gaussian having $\sigma = 1$. Since the gradient vectors on each side of a ridge are opposite to each other, if we smooth the orientation by taking the average of gradient angles directly in a local block, the opposite gradients would cancel each other. To solve this problem, Kass and Witkin [27] proposed a simple and clever idea of doubling the gradient angles before averaging. In practice, 2θ are the angles of squared gradient vectors and are related to the covariance matrix elements by [26]:

$$\sin(2\theta) = \frac{2G_{xy}}{\sqrt{4G_{xy}^2 + (G_{xx} - G_{yy})^2}} \tag{3}$$

$$\cos(2\theta) = \frac{G_{xx} - G_{yy}}{\sqrt{4G_{xy}^2 + (G_{xx} - G_{yy})^2}} \tag{4}$$

These doubled angles are smoothed with a Gaussian of $\sigma = 7$. Finally, the orientation is estimated by

$$\theta = \frac{\pi}{2} + \frac{\arctan\left(\frac{\cos(2\theta)}{\sin(2\theta)}\right)}{2} \tag{5}$$

The visual inspection of the estimated orientations provide a good match with local ridge directions as depicted in Fig. 18.

C. Steerable Gaussian filter

In this paper, an anisotropic directional Gaussian filter is used to enhance the ridge structure and reduce noise. Its kernel is given by

$$G_{dir}(x, y; \theta, f, \sigma_x, \sigma_y) = e^{-\left(\frac{x_\theta^2}{2\sigma_x^2} + \frac{y_\theta^2}{2\sigma_y^2}\right)}. \tag{6}$$

To make the implementation process simpler, G_{dir} is decomposed into two filters. Since $\sigma_y^2 \ll \sigma_x^2$, the filter can be decomposed into a small isotropic filter

$$G_{iso}\left(y; \sigma_y\right) = e^{-\frac{x^2+y^2}{2\sigma_y^2}} \tag{7}$$

and an anisotropic filter

$$G_{ani}\left(x_\theta; \theta, \sigma_\theta\right) = e^{-\frac{x_\theta^2}{2\sigma_\theta^2}} \tag{8}$$

where $\sigma_\theta^2 = \sigma_x^2 - \sigma_y^2 \approx \sigma_x^2$ and $x_\theta = x\cos\theta + y\sin\theta$. The oriented Gaussian filter is implemented by first convolving with an isotropic 2-D Gaussian filter of size σ_y. The resulting image is then convolved with a 1-D Gaussian in the θ direction. Allowing θ to vary from pixel to pixel gives a steerable Gaussian filter, which yields an anisotropically smoothed image.

III. FPGA Implementation

Real-time image processing systems are hard to design using software. The reason is that a large data set is required to represent an image, and complex operations are needed to perform a certain task. Consider that with a video rate of 60 frames per second, a single operation performed on every pixel of a 640 × 480 color image (VGA) equates to 18.4 million operations per second. Thus, the alternative is to make use of hardware design, and realising it on an FPGA. FPGAs offer a compromise between the flexibility of general-purpose processors and the hardware-based speed of an integrated-circuit design. One of the main advantages of using FPGAs for the implementation of image-processing applications is that their structure can exploit spatial and temporal parallelism. FPGA implementations have the potential to be parallel using a mixture of these two forms. For example, the FPGA could be configured to partition the image and distribute the resulting sections to multiple pipelines, all of which could process data concurrently. Such parallelization is subject to the processing mode and hardware constraints of the system. Converting the software design to an efficient hardware design is one of the most difficult steps in embedded system design.

In software, usually one operation is performed at a time with intermediate results stored in RAM for the next operation. This is the reason why it takes longer to perform tasks which comprise multiple sequential operations, while in hardware these components can often be pipelined to create parallel computing structures [28]. Almost all image-processing algorithms contain operations that execute in sequence. This is a form of temporal parallelism [28]. Hence, this structure is ideal to have a separate processor for each operation, as a pipelined architecture. When processing images, data can usually begin to be output from an operation long before the complete image has been processed by that operation. The time between when data is first input to an operation and the corresponding output is available is the latency of that operation. When each operation only uses input pixel values from a small, local neighborhood then its latency is lowest. This is because each output only requires data from a few input pixel values. Operation pipelining at the pixel level can give significant performance improvement when all of the operations have low latency, because a downstream processor may begin performing its operation before the upstream processors have completed.

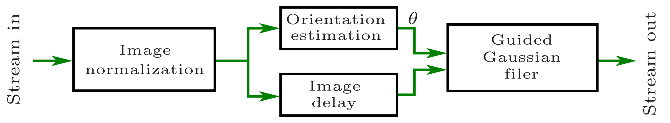

Fig. 2. Block diagram of the proposed hardware for fingerprint image enhancement.

The proposed algorithm for fingerprint image enhancement contains different operations that execute in sequence. Temporal or task-level parallelism creates a different processor for each operation. In general, each processor performs an operation on an M × N window. Each clock cycle, a new pixel is input and processing begins for the current window position. When processing is complete, after some latency, an output pixel is produced each clock cycle. In this way, scanning a window through the image is equivalent to streaming the image through a processor, one pixel per clock cycle. Processing a window requires more than one pixel input. Pixels from the previous M-1 columns and N-1 rows are cached using row buffers as described in [28]. One problem with filtering is managing what happens when the window is not completely within the input image [28]. Very few papers consider these boundary conditions since the design, to handle them properly, can take more effort than to manage the normal case where all of the data is available. In this paper, to tackle these boundary conditions border pixel duplication is used.

Fig. 2 shows a block diagram of the proposed hardware structure, where an input image stream is normalized and enhanced directly as it is streamed to the output. In the proposed structure, each processor or block is designed carefully so that the minimum latency can be achieved and the overall system can take full advantage of the pipelined parallel computing structures.

A. Image Normalization

This step is typically carried out to reduce the variation in the gray-scale image without changing the image structure or texture information. Fig. 3 shows a block diagram of the proposed local-normalization algorithm hardware. The input image is first smoothed through a large 2-D Gaussian filter of $\sigma = 5$. The 2D Gaussian filter is implemented as a cascade of one-dimensional Gaussian filters (1 × 19 and 19 × 1). The details of the Gaussian filter implementation are given in the next subsection. Then the output of this Gaussian filter is subtracted from the delayed image. Appropriate X and Y delays are used, compensating for the latency of the Gaussian filter. The power-law transformation is performed on the absolute value of the difference image. The power-law transformation is implemented in hardware by using a lookup table. This power-law transformed image is smoothed by a second Gaussian filter of $\sigma = 2.5$. The delayed image is divided by the output of the Gaussian filter. The division is rescaled by a factor of 128, offset by 128, and clipped to give the normalized image.

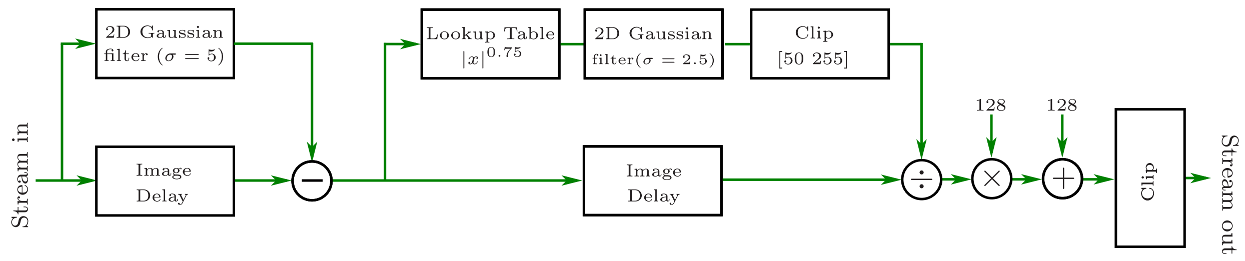
Stream in
2D Gaussian filter ($\sigma = 5$)
Image Delay
Lookup Table $|x|^{0.75}$
2D Gaussian filter($\sigma = 2.5$)
Clip [50 255]
Image Delay
128
128
Clip
Stream out

Fig. 3. Block diagram of the proposed local-normalization algorithm for hardware implementation.

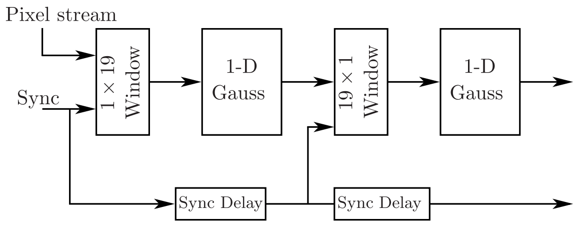

Fig. 4. Block diagram of a 2D Gaussian filter implementation.

1) Gaussian-Filter Implementation: A Gaussian filter is used for image blurring and removing noise or high-frequency components of an image. In two dimensions, the Gaussian filter kernel is:

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (9)$$

For large σ, the size of the filter increases significantly, which makes a hardware implementation expensive. For the hardware implementation, a size of $\sigma = 5$ is used with a 19×19 mask. Truncating the window to this size limits the stopband attenuation of the Gaussian filter at high frequencies to 50 dB.

As the Gaussian is separable, this allows the filter to be implemented as a cascade of one-dimensional Gaussian filters (1×19 and 19×1). Although the filter can be decomposed to use only adders [28]–[30], the need for such a decomposition is less important on modern FPGAs, where high-speed pipelined multipliers are plentiful. The symmetrical nature of the Gaussian filter allows the number of multipliers to be halved by folding the data path and performing the addition before the multiplication, as shown in Fig. 4. For the output additions, an adder-tree is used to minimise the latency.

B. Orientation Estimation

Reliable orientation-field (OF) estimation plays a vital role in fingerprint image enhancement. It is one of the most important preprocessing steps. The performance of the proposed guided Gaussian filter is dependent on the orientation field as well as two additional tuning parameters: σ_x and σ_y, the standard deviations of the Gaussian envelope. For this purpose, the non-iterative gradient-based method [26] described in section II B is implemented in hardware. A block diagram of the proposed hardware is shown in Fig. 6.

The input-stream data is first smoothed by a Gaussian filter of $\sigma = 1$ (with a 5×5 window) to reduce the noise before calculating the discrete derivatives, G_x and G_y, using a Sobel filter, as shown in Fig. 7. The covariance matrix data G_x^2,

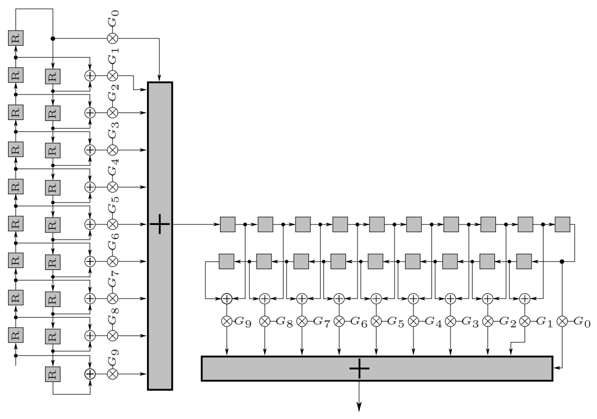

Fig. 5. Hardware implementation of a (1×19 and 19×1) Gaussian filter with $\sigma = 5$. The G_x are filter coefficients. For the vertical filter, the boxes represent row buffers.

G_xG_y and G_y^2 are then calculated and filtered by another Gaussian filter with $\sigma = 1$. Then $\sin(2\theta)$ and $\cos(2\theta)$ are calculated using a modified CORDIC to calculate both Eq. 3 and Eq. 4 in one operation (avoiding the need for square roots and divisions), as shown in Fig. 8. This efficiently uses CORDIC in vectoring mode to determine 2θ, combined with a second CORDIC in rotation mode to calculate $\sin(2\theta)$ and $\cos(2\theta)$. Since the angle determined by the first CORDIC is used to rotate the vector for the second CORDIC, both iterations can be combined, as shown in Fig. 9. The double angles are further smoothed by a larger Gaussian of $\sigma = 7$ (using a 29×29 window). The final orientation is estimated by the arctangent of these double angles by a CORDIC unit operating in vectoring mode. Fig. 10 shows a block diagram of the unrolled CORDIC iteration. The Gaussian filters are decomposed and implemented as a cascade in a similar manner to that shown in Fig. 4 and Fig. 5.

C. Guided-Line Gaussian filter

The aim of this work is to design an efficient algorithm for fingerprint image enhancement that best suits a real-time hardware implementation. For this purpose, the guided Gaussian function is further decomposed into a 2-D isotropic filter and a 1-D anisotropic filter. For the 2-D filter, $\sigma_y = 0.5$ using a 5×5 window.

D. Oriented-Line Gaussian Filter Implementation

The next step is to design the hardware for the oriented Gaussian. Usually, $\sigma_x > \sigma_y$ so a larger filter size

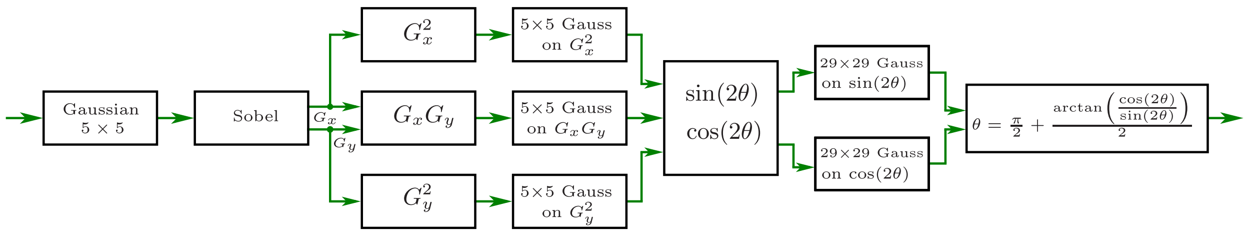

Fig. 6. Block diagram of orientation-field estimation hardware implementation.

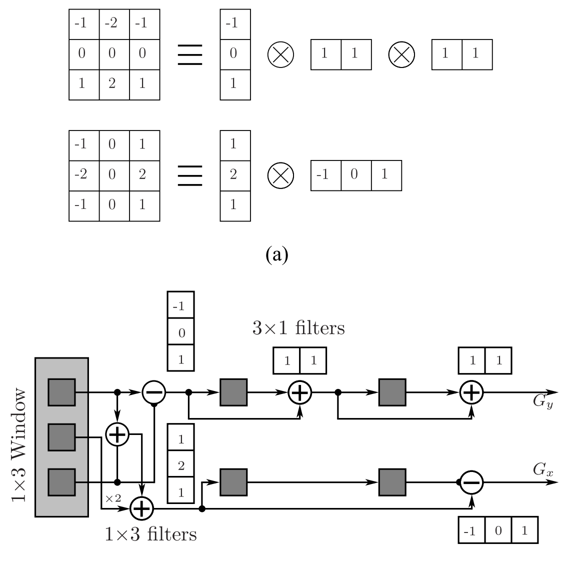

Fig. 7. Sobel filter implementation. (a) Filter decomposition. (b) Implementation.

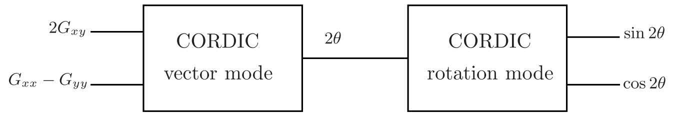

Fig. 8. Block diagram of the modified CORDIC for calculating $\sin 2\theta$ and $\cos 2\theta$.

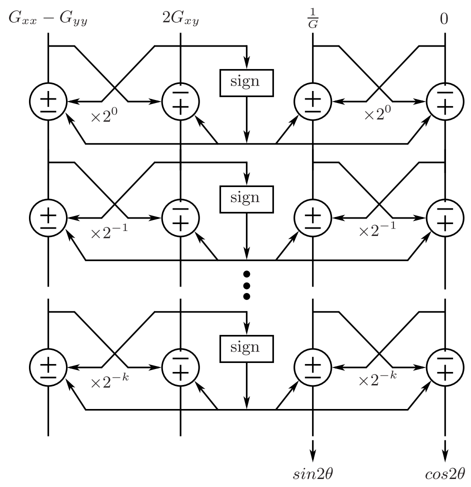

Fig. 9. Iterations of the combined CORDIC for calculating $\sin 2\theta$ and $\cos 2\theta$. The vector mode on the left directly controls the rotation mode on the right. The $\frac{1}{G}$ compensates for the CORDIC gain.

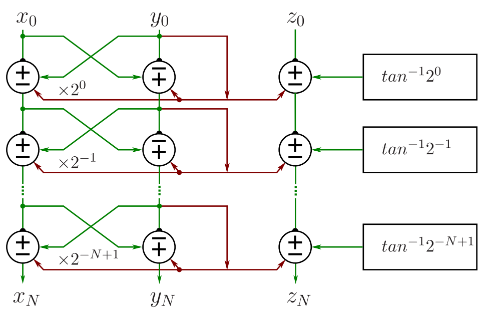

Fig. 10. Block diagram of the unrolled vector mode CORDIC iteration.

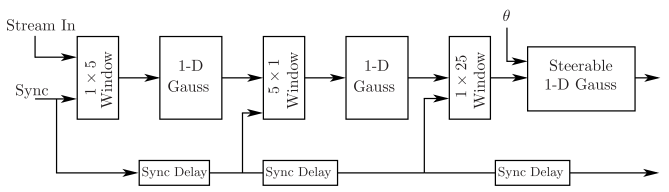

Fig. 11. Block diagram of Oriented Gaussian filter implementation.

is required. For fingerprint enhancement, we use $\sigma_x = 4$, using a 25 × 25 window. Fig. 12 shows a line Gaussian filter directed at an angle of 45°. A direct 2-D implementation of a steerable filter would require a 25 × 25 window where the filter coefficients change every pixel depending on θ. If we analyze Fig. 12, it can be observed that, out of 625 pixels, only 25 pixels have non-zero values. To implement this oriented-line-Gaussian filter efficiently, this filter is converted into a 1-D line Gaussian filter. It is observed that, for a 25 × 25 window, if an angle tolerance of 5.625° (180°/32) is used then it does not significantly affect the smoothing performance of the filter. A larger window may require better angle resolution. This tolerance, combined with the isotropic Gaussian pre-filter, enables nearest-neighbor interpolation to be used to select each pixel from the window to filter. This significantly reduces the hardware complexity.

To convert the 2-D filter into 1-D, the window is divided into two sub-windows hwind and vwind, which are used to filter angles that are primarily horizontal and vertical respectively, as shown in Fig. 13. With nearest-neighbor interpolation, angles within vwind require one pixel from each row within the window while those in hwind require one from each column. For hwind, with streamed image data, the columns within the window arrive in successive clock cycles, leading naturally into a transpose filter structure as shown in Fig. 15.

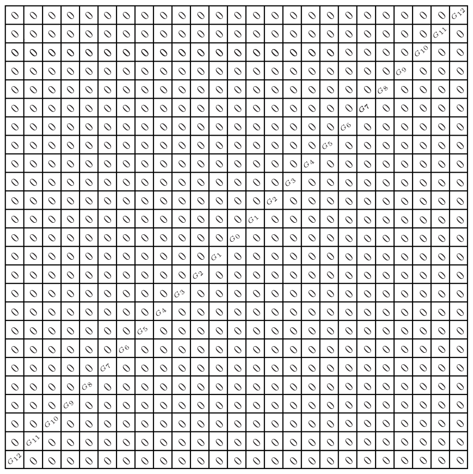

Fig. 12. Line Gaussian filter orientated at 45°.

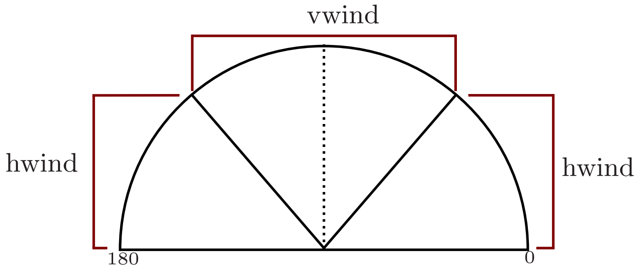

Fig. 13. Window selection as a function of Gaussian filter orientation.

Multiplexers select the appropriate row for each coefficient depending on the angle corresponding to the center of the window. For vwind, one pixel is taken from each window row. The orientation is controlled by implementing variable delays for each window row. These allow a different delay to be selected at each clock cycle. The pixels corresponding to the required delays are selected and then multiplied by the Gaussian filter weights. These are summed to get the filter output value. The number of multipliers can be reduced to half in vwind by exploiting symmetry as shown in Fig. 16(b). Conceptually the variable delays are implemented using a multiplexer as shown in Fig. 16(a). Finally, either the output of vwind or hwind is selected based on the dominant direction of the oriented Gaussian, as shown in Fig. 17.

IV. Experimental Results

The experiments were performed on the FVC2004 database [31] that consists of four sub-bases, in which images are captured with four different sensors. DB1_A, DB2_A, DB3_A, DB4_A. This database was used to enable comparison with existing algorithms. Each sub-base includes 800 fingerprints, 8 for each person. Each sub-base has different fingerprint image sizes:

- In DB1_A the images are 640 × 480
- In DB2_A the images are 328 × 364
- In DB3_A the images are 300 × 480
- In DB4_A the images are 288 × 384

For simulation, the program was written in MATLAB and run on a 3.40 GHz Core i7 processor with 16 GB of memory.

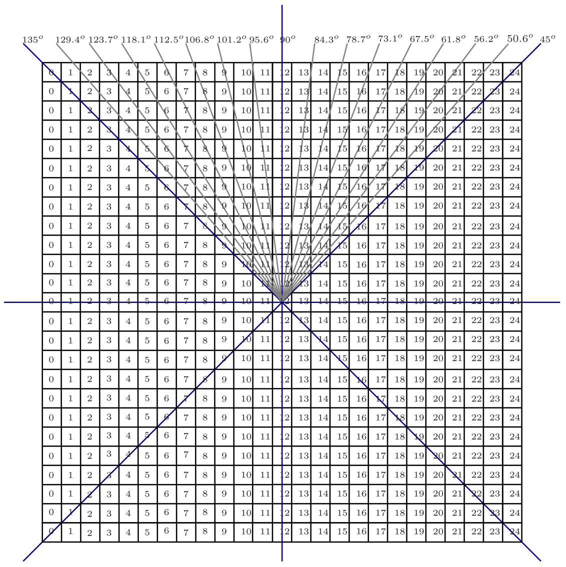

Fig. 14. List of lines with rational slopes used to calculate the variable delays of vwind.

TABLE I

Equal Error Rate EER (%) Assessment of Proposed Algorithm With Existing Algorithms

FVC2004	Compared algorithms	EER(%)
DB1_A	Oriented linear diffused + Hong's normalized	9.74
	Oriented linear diffused+2×2 block normalized	9.08
	Oriented linear diffused+4×4 block normalized	9.93
	Oriented linear diffused+8×8 block normalized	11.53
	2 stage enhanced (Yang)+2×2 block normalized	7.50
	2 stage enhanced (Yang)+4×4 block normalized	7.17
	2 stage enhanced (Yang)+8×8 block normalized	7.26
	Proposed Method	9.03
DB2_A	Oriented linear diffusion + Hong's normalized	11.02
	Oriented linear diffused + 2×2 block normalized	9.99
	Oriented linear diffused + 4×4 block normalized	11.27
	Oriented linear diffused + 8×8 block normalized	13.62
	2 stage enhanced (Yang)+2×2 block normalized	7.75
	2 stage enhanced (Yang)+4×4 block normalized	7.70
	2 stage enhanced (Yang)+8×8 block normalized	8.97
	Proposed Method	9.71

For real-time implementation, VHDL is used and the design is tested on a Cyclone III FPGA using Quartus II, and simulated using ModelSim. To assess the efficiency of the verification system the FAR (false acceptance rate) and FRR (false rejection rate) are calculated by

$$FAR = \frac{Number\ of\ rejected\ genuine\ claims}{Total\ number\ of\ genuine\ accesses} \tag{10}$$

$$FRR = \frac{Number\ of\ accepted\ imposter\ claims}{Total\ number\ of\ imposter\ accesses} \tag{11}$$

Finally, the EER (equal error rate) was used as a success-rate indicator, marking the point where FRR and FAR are equal [30].

$$EER = \frac{FAR + FRR}{2}, \quad if\ FAR = FRR \tag{12}$$

Table I shows the equal error rate EER (%), assessment of the proposed algorithm compared with similar existing

TABLE II
PROCESSING SPEED (IN seconds) of Proposed FPGA Based Algorithm With Proposed PC-Based Structure

FVC2004 Database Type	Proposed MATLAB code	Proposed C code	Proposed FPGA	Speedup relative to MATLAB	Speedup relative to C
FVC_DB1_A	4.78	0.419	0.0140	340×	30×
FVC_DB2_A	1.86	0.165	0.00594	313×	28×
FVC_DB3_A	2.24	0.201	0.00712	315×	28×
FVC_DB4_A	1.72	0.156	0.00558	308×	28×

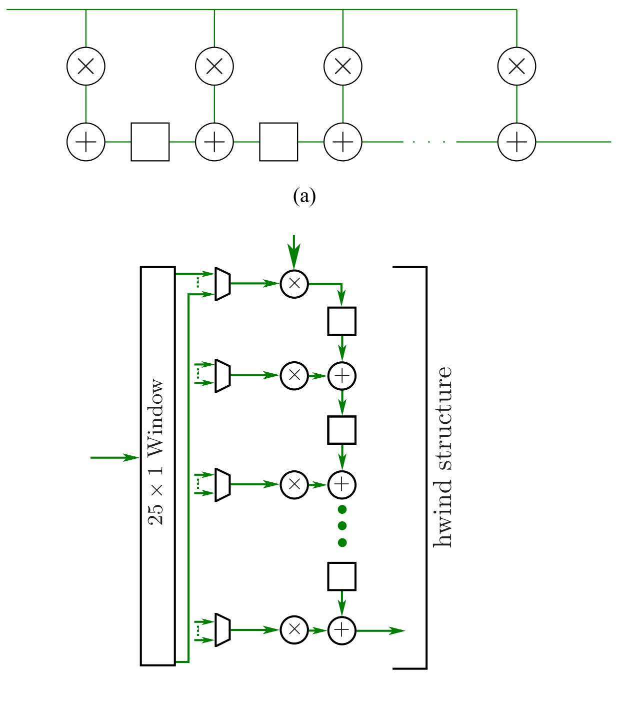

Fig. 15. Proposed hwind structure. a) Pipelined transpose filter structure. b) Adapted for steerable filter. The multiplexers select inputs from the appropriate rows depending on the orientation.

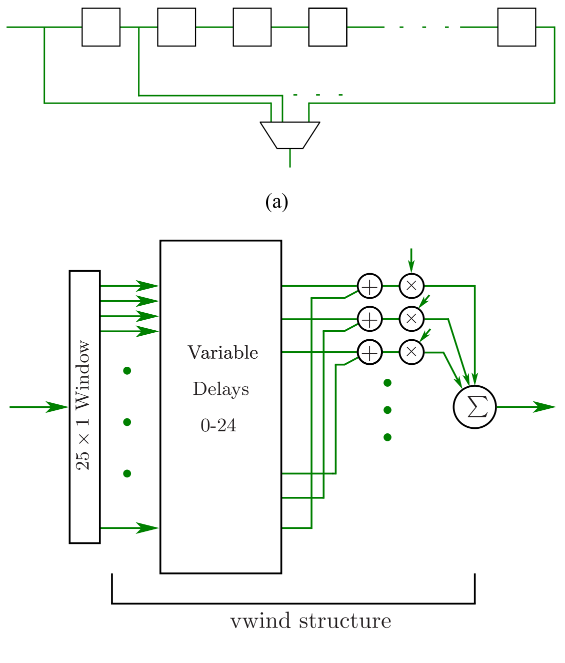

Fig. 16. Proposed vwind structure: a) Variable delay selected by a multiplier, b) structure of vwind using variable delays for each row, and exploiting symmetry for the multiplications.

algorithms. Kocevar [32] proposed a block-local normalization and normalized the image with 3 different block sizes: 2×2, 4×4 and 8×8. He claimed that 8×8 block-local normalization is best suited for real-time application, only if the required accuracy of the system is not too high. For fingerprint image enhancement, both algorithms are used as a pre-processing step with either the Yang algorithm [9] or oriented linear diffusion [33] for enhancement. Although the EER of the proposed algorithm is slightly higher than with Yang's algorithm, it is still very good for a FPGA-based real-time system. Yang's algorithm is less suited to an FPGA implementation because it uses a Gabor filter and the enhancement is based on two stages, requiring more computation and resources. Also, in this paper, we put more emphasis on the speed as well as the parallel compute-intensive demands of the fingerprint image-enhancement process with respect to an FPGA rather than on its EER.

Fig. 18 depicts the results of the proposed fingerprint image-enhancement algorithm on a noisy image. In column 1, three noisy fingerprint images are chosen. The proposed local-normalization algorithm is applied to these images. The second

TABLE III
COMPARISON OF THE PROPOSED METHOD WITH SOME EXISTING FPGA-BASED NORMALIZATION METHODS ON A SAMPLE IMAGE OF SIZE 512×256

Method	Processing time (ms)	Registers	Logic cells (s)	Memory bits
Garcia [21]	522	N/A	N/A	N/A
Qin [20]	165	N/A	N/A	N/A
Fons [15]	60	12811	28756	289k
Proposed	6.9	9590	18160	329k

column of Fig. 18 clearly depicts that the nonuniformity of the poor-quality images is removed in an efficient way. Column 3 shows the orientation map of the local normalized images, calculated by using Eq. 5. Finally, column 4 shows the result of an oriented-line Gaussian filter on the noisy images. The second row shows that it is necessary to remove the non-fingerprint background because the noise from enhancing the contrast can result in false ridges and false features. It is also essential to accurately estimate the orientation field because errors can blur the fingerprint pattern, potentially resulting in false minutiae points. In Table II, the relative processing speed

TABLE IV
DETAILED HARDWARE RESOURCES COMPARISON OF PROPOSED METHOD WITH FONS [34] ON A SAMPLE IMAGE OF SIZE 512 × 256

Process	Method	Processing time (ms)	Registers	Logic cells	Memory bits
Normalization	Fons	25	1384	4729	38912
	Proposed Method	6.4	917	2286	59522
Orientation Estimation	Fons	25	1513	9123	36864
	Proposed Method	6.6	3335	7597	164864
Filtering	Fons	35	5022	11639	65536
	Proposed Method	5.4	5334	8268	58464

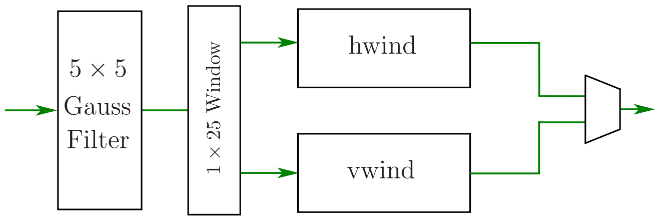

Fig. 17. Proposed oriented-line Gaussian filter.

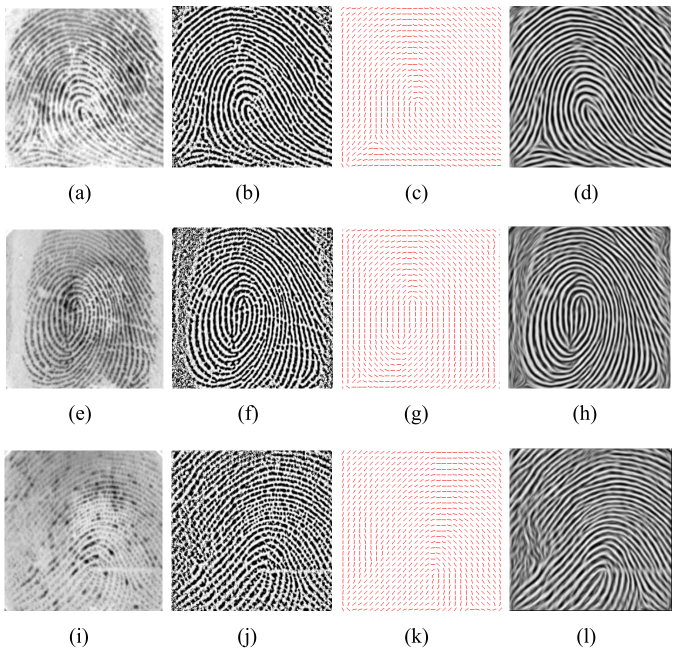

Fig. 18. Impact of proposed proposed method on low-contrast fingerprint images. Column 1 shows three fingerprint images. Column 2 shows the output of proposed local normalization on those images. Column 3 shows the ridge orientation map and column 4 shows the result of a oriented-line Gaussian filter on noisy images.

of the proposed FPGA-based implementation and the proposed MATLAB-based algorithm are presented. The proposed FPGA implementation is over 300 times faster than the MATLAB-based implementation on a PC and 28 times faster than its corresponding C implementation. The reason for the high speed is the efficient use of parallelism in the FPGA.

A. Comparison to Other Hardware Architectures

For complete fingerprint image enhancement, the proposed method is compared with two prominent FPGA-based fingerprint image-enhancement algorithms, one proposed by Fons [15] and the other proposed by Qin [20]. In the literature, Fons was the fastest and most efficient algorithm for fingerprint image enhancement. Table III compares the processing time of the proposed algorithm with the Fons and Qin algorithms. The proposed FPGA implementation is approximately 9 times faster than the Fons FPGA implementation. There are two reasons for this speed improvement. The first is the use of an oriented-line Gaussian filter instead of the Gabor filter used by Fons. Second, our whole algorithm is single-pass while the segmentation and normalization components of Fons are two-pass. One disadvantage of the two-pass process is that it requires the complete image to be stored off-chip; large images cannot be processed completely on an FPGA due to a lack of sufficient on-chip memory. To process a single pixel a two-pass algorithm requires a minimum of 2 clock cycles. If we compare the proposed method with Fons, the proposed method requires fewer registers and logic cells. Table IV gives a detailed hardware resource comparison of the proposed method with Fons [34]. A sample image of size 512 × 256 is used for this comparison. Again, the proposed normalization requires fewer registers as well as logic cells than the Fons method. We require more memory than the Fons method because of the use of a Gaussian filter twice in the normalization. However, these figures are a little misleading, because our architecture does not require off-chip memory which Fons does (not listed in the table). In terms of performance, the proposed normalization performance is much better than Fons. For orientation estimation, Fons used Hong's method that does not tackle the noise present in the orientation. In the proposed method, a large Gaussian filter is used to mitigate the effects of noise present in the orientations. Although in terms of resources this is bit costly, as can be seen in Table IV, it gives better smoothing than Fons. For the final filtering stage the proposed method's performance is much better than Fons. It consumes fewer memory bits and logic cells. This is achieved by using the oriented-line Gaussian and its separability property along with efficient hardware implementation using parallelism of streamed data. Although the individual operations are only 4-5 times those of Fons, pipelining without a frame buffer enables us to more effectively overlap operations, giving a 9 times speedup overall.

V. CONCLUSION AND FUTURE WORK

In this article, we address several challenging problems for real-time fingerprint image enhancement. A new architecture for anisotropic diffusion is presented, decomposing it into an isotropic and an anisotropic filter. By this decomposition, we manage to efficiently speed up the image processing and improve the resource utilization of the FPGA. To further improve the performance of the filter, the input image is homogenized by a local image normalization. The proposed normalization method not only best suits hardware implementation but also gives much better results than existing methods. Although the EER of the proposed algorithm is slightly higher than some existing algorithms, our main achievement is the speed as well as the parallel and compute-intensive demands of the fingerprint image-enhancement process with the FPGA rather than its EER. As far as the authors know, the proposed structure is the fastest, most cost-effective and most efficient FPGA-based structures. As a future work, the authors aim to develop the remaining stage of an algorithm to reach a complete embedded automatic fingerprint identification system (AFIS). We also plan to investigate further optimization of the orientation estimation and anisotropic filtering stages of the algorithm.

REFERENCES

[1] L. Hong, Y. Wan, and A. Jain, "Fingerprint image enhancement: Algorithm and performance evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 777–789, Aug. 1998.

[2] T. M. Khan, M. A. U. Khan, and Y. Kong, "Fingerprint image enhancement using multi-scale DDFB based diffusion filters and modified Hong filters," *Optik-Int. J. Light Electron Opt.*, vol. 124, no. 16, pp. 4206–4214, Aug. 2014.

[3] S. Pankanti, S. Prabhakar, and A. K. Jain, "On the individuality of fingerprints," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 8, pp. 1010–1025, Aug. 2002.

[4] A. M. Bazen, G. T. B. Verwaaijen, S. H. Gerez, L. P. J. Veelenturf, and B. J. van der Zwaag, "A correlation-based fingerprint verification system," in *Proc. Workshop Circuits, Syst. Signal Process. (ProRISC)*, Veldhoven, The Netherlands, 2000, pp. 205–213.

[5] M. A. U. Khan, T. M. Khan, D. G. Bailey, and Y. Kong, "A spatial domain scar removal strategy for fingerprint image enhancement," *Pattern Recognit.*, vol. 60, pp. 258–274, Dec. 2016.

[6] T. M. Khan, M. A. U. Khan, O. Kittaneh, and Y. Kong, "Stopping criterion for linear anisotropic image diffusion: A fingerprint image enhancement case," *EURASIP J. Image Video Process.*, vol. 2016, no. 1, p. 6, Dec. 2016, doi: 10.1186/s13640-016-0105-x.

[7] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "FVC2002: Second fingerprint verification competition," in *Proc. 16th Int. Conf. Pattern Recognit.*, vol. 3. 2002, pp. 811–814.

[8] American National Standards Institute. *Common Biometric Exchange File Format*, accessed on Jan. 3, 2002. [Online]. Available: http://www.itl.nist.gov/div895/isis/bc/cbeff/

[9] J. Yang, N. Xiong, and A. V. Vasilakos, "Two-stage enhancement scheme for low-quality fingerprint images by learning from the images," *IEEE Trans. Human–Mach. Syst.*, vol. 43, no. 2, pp. 235–248, Mar. 2013.

[10] V. Areekul, U. Watchareeruetai, K. Suppasriwasuseth, and S. Tantaratana, "Separable Gabor filter realization for fast fingerprint enhancement," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3. Sep. 2005, pp. III-253–III-256.

[11] J. Yang, L. Liu, T. Jiang, and Y. Fan, "A modified Gabor filter design method for fingerprint image enhancement," *Pattern Recognit. Lett.*, vol. 24, no. 12, pp. 1805–1817, 2003.

[12] V. Areekul, U. Watchareeruetai, and S. Tantaratana, "Fast separable Gabor filter for fingerprint enhancement," in *Proc. Int. Conf. Biometric Authentication (ICBA)*, 2004, pp. 403–409.

[13] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, Jul. 1990.

[14] J.-M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast anisotropic Gauss filtering," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 938–943, Aug. 2003.

[15] F. Fons, M. Fons, and E. Cantó, "Approaching fingerprint image enhancement through reconfigurable hardware accelerators," in *Proc. IEEE Int. Symp. Intell. Signal Process. (WISP)*, Oct. 2007, pp. 1–6.

[16] M. Fons, F. Fons, and E. Cantó, "Fingerprint image processing acceleration through run-time reconfigurable hardware," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 12, pp. 991–995, Dec. 2010.

[17] M. Fons, F. Fons, E. Cantó, and M. López, "FPGA-based personal authentication using fingerprints," *J. Signal Process. Syst.*, vol. 66, no. 2, pp. 153–189, 2012.

[18] F. Fons, M. Fons, E. Cantó, and M. López, "Real-time embedded systems powered by FPGA dynamic partial self-reconfiguration: A case study oriented to biometric recognition applications," *J. Real-Time Image Process.*, vol. 8, no. 3, pp. 229–251, 2013.

[19] A. Alilla, M. Faccio, T. Vali, G. Marotta, and L. DeSantis, "A new low cost fingerprint recognition system on FPGA," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Feb. 2013, pp. 988–993.

[20] M. Qin, "A fast and low cost SIMD architecture for fingerprint image enhancement," M.S. thesis, Dept. Elect. Eng., Delft Univ. Technol., Delft, The Netherlands, 2007.

[21] M. L. Garcia and E. F. C. Navarro, "FPGA implementation of a ridge extraction fingerprint algorithm based on microblaze and hardware coprocessor," in *Proc. Int. Conf. Field Program. Logic Appl.*, 2006, pp. 1–5.

[22] S. Kawada and T. Maruyama, "An approach for applying large filters on large images using FPGA," in *Proc. Int. Conf. Field Program. Technol.*, Kitakyushu, Japan, 2007, pp. 201–208.

[23] A. Joginipelly, A. Varela, D. Charalampidis, R. Schott, and Z. Fitzsimmons, "Efficient FPGA implementation of steerable Gaussian smoothers," in *Proc. 44th IEEE Southeastern Symp. Syst. Theory*, Mar. 2012, pp. 78–83.

[24] P. Perona, "Orientation diffusions," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 457–467, Mar. 1998.

[25] C. Gottschlich, P. Mihailescu, and A. Munk, "Robust orientation field estimation and extrapolation using semilocal line sensors," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 4, pp. 802–811, Dec. 2009.

[26] Y. Wang, J. Hu, and F. Han, "Enhanced gradient-based algorithm for the estimation of fingerprint orientation fields," *Appl. Math. Comput.*, vol. 185, pp. 823–833, Feb. 2007.

[27] M. Kass and A. Witkin, "Analyzing oriented patterns," *Comput. Vis., Graph., Image Process.*, vol. 37, pp. 362–385, Mar. 1987.

[28] D. G. Bailey, *Design for Embedded Image Processing on FPGAs*. Hoboken, NJ, USA: Wiley, 2011.

[29] T. M. Khan, D. G. Bailey, M. A. U. Khan, and Y. Kong, "Real-time edge detection and range finding using FPGAs," *Optik-Int. J. Light Electron Opt.*, vol. 126, no. 17, pp. 1545–1550, 2015.

[30] T. M. Khan, D. G. Bailey, M. A. U. Khan, and Y. Kong, "Efficient hardware implementation strategy for local normalization of fingerprint images," *J. Real-Time Image Process.*, vol. 1, pp. 1–13, Jul. 2016, doi: 10.1007/s11554-016-0625-8.

[31] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "FVC2004: Third fingerprint verification competition," in *Proc. Int. Conf. Biometric Authentication*, Hong Kong, 2004, pp. 1–7.

[32] M. Kočevar, B. Kotnik, A. Chowdhury, and Z. Kačič, "Real-time fingerprint image enhancement with a two-stage algorithm and block–local normalization," *J. Real-Time Image Process.*, vol. 1, pp. 1–10, Jul. 2014, doi: 10.1007/s11554-014-0440-z.

[33] C. Gottschlich and C. B. Schonlieb, "Oriented diffusion filtering for enhancing low-quality fingerprint images," *IET Biometrics*, vol. 1, no. 2, pp. 105–113, Jun. 2012.

[34] F. Fons, M. Fons, E. Cantó, and M. López, "Flexible hardware for fingerprint image processing," in *Proc. Res. Microelectron. Electron. Conf. (PRIME)*, 2007, pp. 169–172.

Tariq Mahmood Khan received the B.S. degree in computer engineering from the COMSATS Institute of Information & Technology, Islamabad, Pakistan, the M.Sc. degree in computer engineering from the University of Engineering & Technology, Taxila, Pakistan, and the Ph.D. degree from Macquarie University Sydney, Australia, in 2016. He is interested in both digital image processing (with an emphasis on biometrics) and VLSI. His research interests include most aspects of image enhancement, pattern recognition, and image analysis. One area of particular interest is the application of FPGAs to biometric processing algorithms.

Donald G. Bailey received the B.E. degree (Hons.) in electrical engineering and the Ph.D. degree in electrical and electronic engineering from the University of Canterbury, New Zealand, in 1982 and 1985, respectively. From 1985 to 1987, he applied image analysis to the wool and paper industries within New Zealand. From 1987 to 1989, he was a Visiting Research Engineer with the University of California at Santa Barbara. In 1989, he joined Massey University, Palmerston North, New Zealand, as the Director of the Image Analysis Unit. He is currently a Professor with the School of Engineering and Advanced Technology, and a Leader of the Image and Signal Processing Research Group. His primary research interests include applications of image analysis, machine vision, and robot vision. One area of particular interest is the application of FPGAs to implementing image processing algorithms.

Mohammad A. U. Khan is currently an Associate Professor of Signal and Image Processing with Effat University, Jeddah, Saudi Arabia. He is also the Director of the Biometric and Sensor Laboratory, Effat University. He is teaching in the areas of signal processing and image processing. His research interests include most aspects of image enhancement, pattern recognition, and image analysis.

Yinan Kong received the Ph.D. degree from the University of Adelaide, Australia. He currently leads the VLSI Research Group with Macquarie University, Sydney, Australia. He is also the Director of Higher Degree Research, Department of Engineering, Macquarie University. During his research with the VLSI Research Group, Macquarie University, and the Center for High Performance Integrated Technologies and Systems, University of Adelaide, he focused on digital VLSI design at an arithmetic level for specific applications like cryptography and DSP.