International Journal of Electronics Signals and Systems

Volume 2 | Issue 3 Article 12

January 2013

FPGA IMPLEMENTATION OF RED ALGORITHM FOR HIGH SPEED PUPIL ISOLATION

PRASHANTH M. ULLAGADDI
Dept of ECE, Dayananda Sagar College Of Engineering, Bangalore-78, pullagaddi@gmail.com

K. N. PUSHPALATHA
Dept of ECE, Dayananda Sagar College Of Engineering, Bangalore-78, knpdrs@gmail.com

ARAVIND KUMAR GAUTAM
SD college Of Engineering, Muzaffar Nagar, U.P-54, drakgautam@gmail.com

FPGA IMPLEMENTATION OF RED ALGORITHM FOR HIGH SPEED PUPIL ISOLATION

[1]PRASHANTH M ULLAGADDI, [2]K N PUSHPALATHA & [3]ARAVIND KUMAR GAUTAM

[1&2] Dept of ECE, Dayananda Sagar College Of Engineering, Bangalore-78
[3]SD college Of Engineering, Muzaffar Nagar, U.P-54
E-mail : pullagaddi@gmail.com, knpdrs@gmail.com & drakgautam@gmail.com

Abstract - Iris recognition is an automated method of biometric identification that uses mathematical pattern-recognition techniques on video images of the irises of an individual's eyes, whose complex random patterns are unique and can be seen from some distance. Modern iris recognition algorithms can be computationally intensive, yet are designed for traditional sequential processing elements, such as a personal computer. However, a parallel processing alternative using Field Programmable Gate Array offers an opportunity to speed up iris recognition. Within the means of this project, iris template generation with directional filtering, which is a computationally expensive, yet parallel portion of a modern iris recognition algorithm, is parallelized on an FPGA system. An algorithm that is both accurate and fast in a hardware design that is small and transportable are crucial to the implementation of this tool. As part of an ongoing effort to meet these criteria, this method improves a iris recognition algorithm, namely pupil isolation. A significant speed-up of pupil isolation by implementing this portion of the algorithm on a Field Programmable Gate Array.

I. INTRODUCTION

Daugman created a very strong algorithm for iris detection based on Gabor wavelets .An alternate design, Ridge Energy Direction is based on spatial domain directional filters. This application has great military interest, and a small system on a chip design using a Field Programmable Gate Array (FPGA) would be ideal for carrying into the field or storing in a backpack [2]. FPGA devices also provide an attractive solution to computationally intensive applications because of their high density, high performance and complete configurability to support specific applications. An FPGA chip offers a combination of the flexibility of general purpose computers and hardware-based real-time processing of Application Specific Integrated Circuits (ASICs). An architecture design for FPGA technology can fully exploit the data and I/O parallelism in most image processing applications.

An FPGA chip offers a combination of the flexibility of general purpose computers and hardware-based real-time processing of ASICs. An architecture design for FPGA technology can fully exploit the data and I/O parallelism in most image processing applications. Our ultimate goal is to build each section of the RED algorithm using the most efficient method and combine them to create a single fast and accurate system for iris recognition.

The iris is a thin circular diaphragm, which lies between the cornea and the lens of the human eye. A front-on view of the iris is shown in Figure 1. The iris is perforated close to its center by a circular aperture known as the pupil. Formation of the iris begins during the third month of embryonic life. The unique pattern on the surface of the iris is formed during the first year of life, and pigmentation of the stroma takes place for the first few years. Formation of the unique patterns of the iris is random and not related to any genetic factors. Segmentation isolates the actual iris region in a digital eye image. The success of Segmentation depends on the imaging quality of eye images.

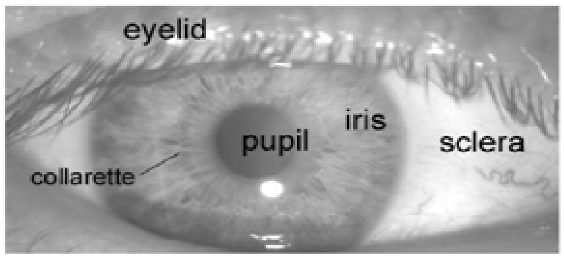


Figure 1. Front on view of iris

Segmentation normalizes the inconsistency caused by pupil dilation, varying imaging distance, rotation of the camera, head tilt, and rotation of the eye within the eye socket.

The most discriminating information present in an iris pattern is extracted by feature encoding. Only the significant features of the iris must be encoded so that comparisons between templates can be made. Matching the template that is generated in the feature encoding process will also need a corresponding matching metric, which gives a measure of similarity between two iris templates.

After determining the inner and outer boundaries and centre of the pupil, the iris is again transformed into polar Coordinates with the centre of the pupil as the point of reference, into a 120 row by 180 column image. In this process, the radial extent of the iris is normalized in order to account for pupil dilation. Each row in the unwrapped iris image represents an annular region surrounding the pupil, and the columns represent radial form information.

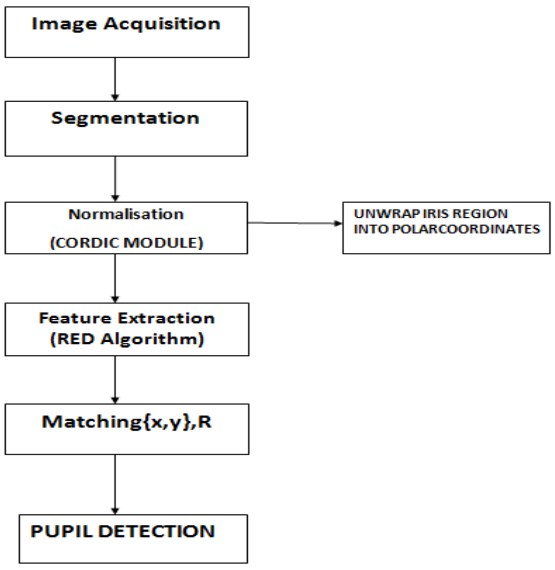


Image Acquisition
Segmentation
Normalisation (CORDIC MODULE)
UNWRAP IRIS REGION INTO POLARCOORDINATES
Feature Extraction (RED Algorithm)
Matching{x,y},R
PUPIL DETECTION

Figure 2: Processing flow diagram

II. RED ALGORITHM STEPS

Iris recognition requires four main steps:

i. Image capture;

ii. Pre-processing, which includes segmentation (isolating the iris from the image of the eye area), and usually a polar coordinate transform of the annular iris region into a rectangular image;

iii. Feature extraction, which generates an iris template; and

iv. Comparison of iris templates and a recognition (matching) decision. A number of methods of pre-processing and comparison are present.

A. Canny Edge Detection

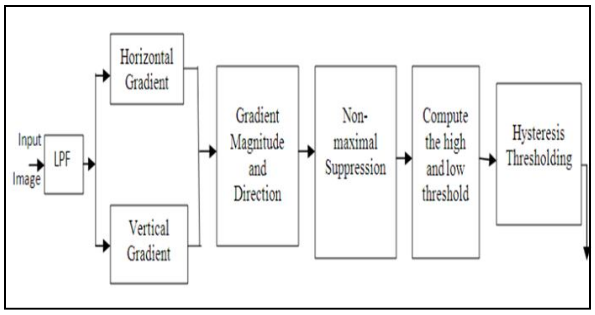


Figure 3. Block diagram of the canny edge detection algorithm

The Canny edge detector is predominantly used in many real-world applications due to its ability to extract significant edges with good detection and good localization performance. Unfortunately, the Canny edge detection algorithm contains extensive pre-processing and post-processing steps and is more computationally complex than other edge detection algorithms, such as Roberts, Prewitt and Sobel algorithms. Furthermore, it performs hysteresis thresholding which requires computing high and low thresholds based on the entire image statistics.[6] This places heavy requirements on memory and results in large latency hindering real-time implementation of the Canny edge detection algorithm.

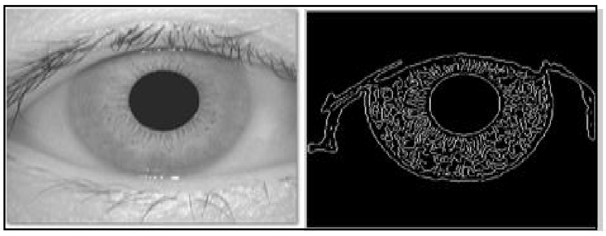


Figure 4. Photograph of the eye captured by a digital camera (a)and after Canny edge detection (b).

There are several steps required to achieve this end state. From the infrared image of the eye of a compliant participant (Fig. 4a), the iris must be extracted as accurately as possible. Several factors can cause interference such as the presence of the eyelid and eyelashes, reflections, and different angles of the eyeball. The application must be able to isolate and extract the pupil from within the iris. One of the means to do this in the RED algorithm is described as follows. The first step is to apply Canny edge detection (Fig. 4b). [1]Then the outline of the pupil is located using a search for circles of various radii. This paper speeds up the process of finding the inner iris border, using an FPGA to parallelize the search. This paper has been described a computational approach to edge detection. [4]The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behaviour of the detector while making minimal assumptions about the form of the solution.

The main steps included are the one dimensional formulation. In the above steps three performance criteria are followed, Good detection, Good localization, Only one response, [8] Finding optimal detection by numerical optimization, A detector for step edge, [4]. An efficient approximation, two or more dimensions, need of multiple width, need of directional operators

Here the defined detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functional on the operator impulse response. We have also found that the first two criteria are not "tight" enough, and that it is necessary to add a third criterion to circumvent the possibility of multiple responses to a single edge. Using numerical optimization, we derive optimal operators for ridge and roof edges. We will then specialize the criteria for step edges and give a parametric closed form for the solution.

In the process we will discover that there is an uncertainty principle relating detection and localization of noisy step edges, and that there is a direct tradeoff between the two. One consequence of this relationship is that there is a single unique "shape" of impulse response for an optimal edge detector, and that the tradeoff between detection and localization [9] can be varied by changing the spatial width of the detector. Several examples of the detector performance on real images will be given.

There was a direct tradeoff in detection performance versus localization, and this was determined by the spatial width. A detector was proposed which used adaptive thresholding with hysteresis to eliminate streaking of edge contours. The thresholds were set according to the amount of noise in the image, as determined by a noise estimation scheme. [9]This detector made use of several operator widths to cope with varying image signal-to-noise ratios, and operator outputs were combined using a method called feature synthesis, where the responses of the smaller operators were used to predict the large operator response.

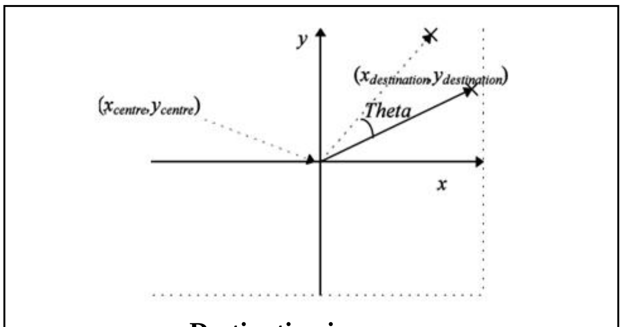
The performance of the canny algorithm depends heavily on the adjustable parameters, σ, which is the standard deviation for the Gaussian filter, and the threshold values, ‘T1’ and ‘T2’. σ also controls the size of the Gaussian filter. The bigger the value for σ, the larger the size of the Gaussian filter becomes. This implies more blurring, necessary for noisy images, as well as detecting larger edges. [4][6]As expected, however, the larger the scale of the Gaussian, the less accurate is the localization of the edge. Smaller values of σ imply a smaller Gaussian filter which limits the amount of blurring, maintaining finer edges in the image. Canny’s edge detection algorithm is computationally more expensive compared to Sobel, Prewitt and Robert’s operator. However, the Canny’s edge detection algorithm performs better than all these operators under almost all scenarios.

B. Red Algorithm

The “energy” of the unwrapped iris image after contrast-limited adaptive histogram equalization is considered. Here, “energy” loosely refers to the prominence (pixel values) of the ridges that appear in the histogram equalized image, higher value reflects higher energy. This “energy” image is passed into each of two different 11 x 11 directional filters (a vertical filter and a horizontal filter). These filters are used to indicate the presence of strong ridges, and the orientation of these ridges.[3]

At every pixel location in the filtered image, the filter which provides the largest value of output is recorded and encoded with one bit to represent the identity of this directional filter. The iris image is thus transformed into a one bit template that is the same size as the image in polar coordinates (120 rows by 180 columns). In some portions of the image input to the filters, the energy may be too low to reliably determine if a ridge is present. For this reason, each template is accompanied by a binary mask, with a 1 indicating presence of a ridge and a 0 indicating no ridge being detected.[3] For future implementations of the RED algorithm, detection of eyelids, eyelashes and specularities will be incorporated into the segmentation, so that the mask will also be used to identify these non-iris areas as well as iris regions without prominent ridges. The template generation process is outlined in Fig. 6.

C. Cordic Modul


y
$(x_{destination}, y_{destination})$
(x_{centre}, y_{centre})
Theta
x

Destination image

Figure 5: The points created by the CORDIC modules in one clock cycle

Creating a Circle with a Given Center-point and Radius. Given an (*x, y*) coordinate from the search area and an even radius between 10 and 126 pixels, a list of address points for a circle with center-point (*x, y*) and radius *r is created.* The circle plot area is a 256 by 256 pixel space taken from the edge detected image. The address points are computed by implementing a version of the CORDIC algorithm.[1] The CORDIC module is copied 91 times in parallel to create the first 91 points on a circle from t=0 to t=90 in one clock cycle.

$$X_0=X_{center}+rcos(t) \qquad (1a)$$

$$Y_0=Y_{center}+rcos(t) \qquad (1b)$$

In the memory controller these points will be used to form the full circle. After the 91 points are created they go into an array and are compared to remove any repeat points created due to rounding. Once this is complete, the points are ready to move into the memory controller. The memory controller takes three points at a time and rotates them into the second, third and fourth addresses for simultaneous reads. Each pixel location (*x1, y1*) in the first quadrant is used to generate the corresponding three coordinates (*x2, y2*), (*x3, y3*), (*x4, y4*) in the second, third and fourth quadrants.

Every time the memory controller fetches three data points from the CORDIC module, it accesses 12 memory locations in one clock cycle. To support this processing rate, six copies of the edge image are stored and dedicated to each CORDIC module. The 12 bits are read and sent to the third part of the architecture, the accumulator.

D. Template Matching

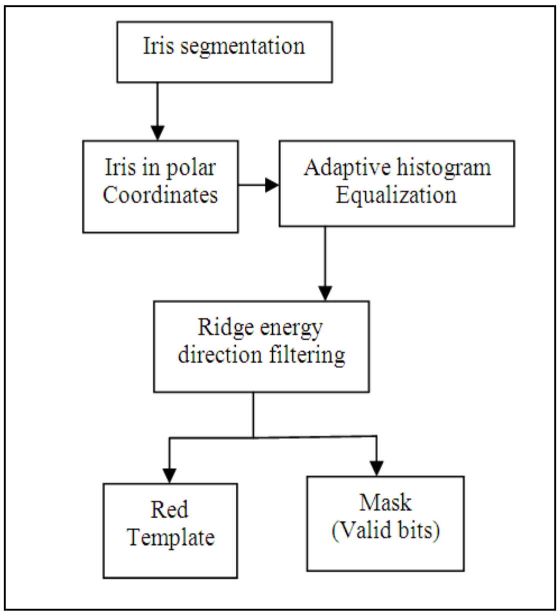

Iris segmentation
Iris in polar Coordinates
Adaptive histogram Equalization
Ridge energy direction filtering
Red Template
Mask (Valid bits)

Figure 6: Template Matching

For matching, this template can now be compared to a stored template using fractional Hamming distance (HD) as the measure of closeness (template A template B) mask A mask B

$$HD = \frac{\|(\text{template A} \otimes \text{template B}) \cap \text{mask A} \cap \text{mask B}\|}{\|\text{mask A} \cap \text{mask B}\|}$$

The operator is the binary exclusive-or operation to detect disagreement between the bits that represent the directions in the two templates, is the binary AND function, ||.|| is a summation, and masks A and B are the associated binary masks for each template. [5] The denominator ensures that only the bits that matter are included in the calculation, after non-ridge areas are discounted. Rotation mismatch between irises (due to head-tilt) is handled with left right shifts of the template to determine the minimum HD. [3]For example, with 120 x 180 templates, each column represents of angular resolution and a shift of 12 (6 columns) is performed in each direction (left/right). The resulting fractional Hamming distances (similarity scores) representing genuine matches (i.e., comparisons of the same eye) and imposter matches (comparisons of different eyes) generated the results presented in the next section.

III. FPGA IMPLEMENTATION

A. Parallelizing iris recognition

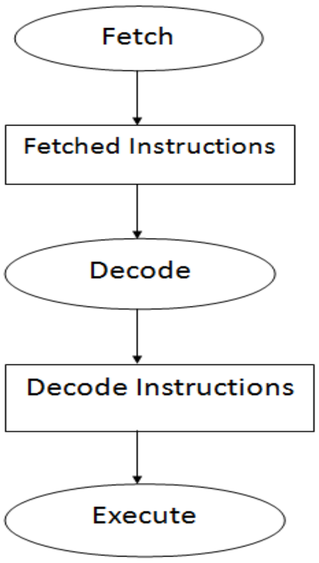


Figure 7: Simple processor pipeline with four ALUs

The iris recognition algorithms are currently implemented on general purpose sequential processing systems, such as generic central processing units (CPUs). In this work, we present a more direct and parallel processing alternative using field-programmable gate arrays (FPGAs), offering an opportunity to increase speed and potentially alter the form factor of the resulting system. Within the means of this project, the most time-consuming operations of a modern iris recognition algorithm are deconstructed and directly parallelized. In particular, portions of iris segmentation, template creation, and template matching are parallelized on an FPGA-based system, with a demonstrated speedup of 9.6, 324, and 19 times, respectively, when compared to a state-of-the-art CPU-based version.

Figure 8 illustrates a simple processor pipeline. Instructions are first fetched, decoded, and finally executed by more than one ALU. Therefore, multiple instructions can be executed in parallel. However, modern processors are limited in the number of ALUs they possess, and most of today's CPUs do not have more than four logic units or ALUs. State-of-the-art processors contain more than one ALU. Therefore, multiple instructions can be executed in parallel. The focus of this research is on parallelizing key portions of the iris recognition algorithm using an FPGA. This work demonstrates this by making the following contributions, Introduction and calculation of the theoretical best performance of CPU-based machines executing key components of an iris recognition algorithm.

Measurements of CPU performance of key components of an iris recognition algorithm on a state-of-the-art computer. Deconstruction and novel parallelization of three key iris recognition components, Iris segmentation using local kurtosis, Iris template creation via filtering, and Template matching via hamming distance Evaluation of the benefits of parallelization in terms of performance and size.

After twelve points are read from memory they are sent directly into the accumulator. In one clock cycle, the 12 fell on the created circle. As the memory controller reads the next three points from the list of 91, rotates them and sends the addresses into memory, the accumulator adds the previous cycle's data. Once all points have been read and added together a flag indicates the circle has been read and a percentage of the circle is calculated. [1] A best percentage is stored in the module, each new percentage being compared to the current best. Each of the 36 parallel paths keeps its own best percentage. Once all center points and radii have been tested, the best of each module goes through a tree of comparators and the overall best match becomes the output of the system. The center-point, radius and percent match are output. This ends the process for the given image. The overall layout is shown in Fig. 8 for the parallel path.

B. Architecture

We utilize Field-Programmable Gate Arrays (FPGAs) to parallelize directional filtering. FPGAs are complex programmable logic devices that are essentially a "blank slate" integrated circuit from the manufacturer and can be programmed with nearly any parallel logic function. They are fully customizable and the designer can prototype, simulate and implement a parallel logic function without the costly process of having a new integrated circuit manufactured from scratch. In Figure 8 an FPGA [3] has been programmed with many ALUs. FPGAs are commonly programmed via VHDL (VHSIC Hardware Description Language). VHDL statements are inherently parallel, not sequential. VHDL allows the programmer to dictate the type of hardware that is synthesized on an FPGA. For example, if you would like to have 2,048 XOR logic gates that execute in parallel, then you program this directly in the VHDL code

Our results were performed on a modest-sized Spartan-3 FPGA, and we expect future generations of FPGAs to increase in performance greater than future generation CPU chips. We also believe other components of iris recognition can be ported to a parallel system. Iris recognition is becoming more and more popular, and an accurate, timely system is our goal.

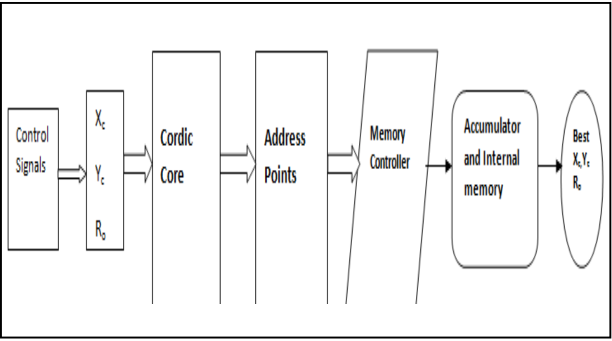


Figure: 8 layout of the parallel implementation of algorithm

IV. RESULTS AND ANALYSIS

A. *Hardware Details*

The CPU experiment is executed on an Intel Xeon X5355 workstation class machine. The processor is equipped with eight cores, 2.66-GHz clock, and an 8-MB L2 cache. While there are eight cores available, only one core is used to perform this test, therefore allowing all cache and memory resources for the code under test. The HD code was compiled under Windows XP using the Visual Studio software suite. The code has been fully optimized to enhance performance. Additionally, millions of matches were executed to ensure that the templates are fully cached in the on-chip L2 cache. The RED algorithm optimized C++ code time is faster than some of the times reported in the literature for other commercial iris recognition implementations. For example, the template matching time earlier was 10m, whereas our template matching time is 383 ns. We attribute this difference to: 1) our use of a faster, more modern CPU operating at 2.66 GHz vice 300 MHz; 2) the RED algorithm is fully optimized;

	Percentage of logic elements on cyclic	Percentage of logic elements estimated on stratix	comment
Kurtosis	9%	.9%	Utilizes faster on-chip multipliers. After multipliers are exhausted, each parallel instantiation of kurtosis consumes 9%.
Digital filtering	71%	7.1%	This algorithm is hardware intensive. It also consumes 17% of on chip memory.
Hamming distance	73%	7.3%	Storing a template consumes 0.7% of cyclone memory. Therefore approximately 230 irises can be stored on cyclone. A tenfold increase can be stored on the stratix.

Figure 9: FPGA Hardware Usage Results

An extrapolation of the execution times of on a modern computing environment at 2.66 GHz would leave it still slower, but closer to the RED execution times. Our goal here is not to provide a direct comparison of the execution times of these two iris recognition algorithms. chip is necessary to execute our algorithm. We are able to determine the size required of our program on the FPGA, and the resulting execution time.

B. Comparisons

Speedup: All VHDL code is fully synthesizable and is downloaded onto our FPGA for direct hardware execution. As discussed above, our code is fully contained within a "process" statement. The process statement is only initiated when a signal in its sensitivity list changes values. The sensitivity list of our process contains the clock signal and, therefore, the code is executed once per clock cycle. In this code, the clock signal is drawn from our FPGA board which contains a 50-MHz clock. Therefore, every 20 ns, our calculation is computed. Note that for template generation, this process takes many clock cycles, and therefore, the total time for template generation with our FPGA is 98.6 us. Fig. 9 illustrates the execution times and acceleration achieved for our implemented FPGA.For example, the optimized C++ version takes 96.8 ns per match while the FPGA takes 20 ns per match for the kurtosis function.

The parallel algorithm on our FPGA is approximately 2.3, 4.7, and 7.5 times calculations, respectively. If Intel were to design a much faster microprocessor with a perfect compiler, it still would be orders of magnitude slower than an off-the-shelf inexpensive low-end FPGA.

V. CONCLUSION

The hardware implementation of RED is still in progress. This segment of the implementation of the RED algorithm can be improved to maximize clock frequency by adjusting the pipeline. The next implementation will include the Canny edge detection algorithm on the FPGA as well. [2]We also expect to find a speed-up on this part. There are other portions of RED that could be improved using a hardware design. Some of these include other parts of the segmentation process as well as the tedious bit-matching algorithm for finding a match within a large database.

This method carries with it several assumptions. First, it is assumed that the iris images are orthogonal, such that the eye is looking directly at the camera. In conjunction with this, it is assumed that the pupil and the limbic boundary of the iris is circular, which is not always accurate. The eyes are assumed to be wide open in that the presence of eyelids or eyelashes within the determined boundaries of the iris is not considered when extracting the ridge features, which serves to detriment the system performance. Overall, the algorithm has several areas that can be addressed to improve performance; these are discussed in the next section. Considerable gains in performance are expected with these modifications Iris template generation with directional filtering can be computationally intensive, yet the algorithms are currently designed for traditional sequential processing elements, such as a personal computer. We have presented a parallel processing alternative using field programmable gate arrays to speed up template generation. [5]

Iris recognition as a biometric technology has great advantage such as variability, stability and security. Thus it will have a variety of application. Here an image is analyzed by calculation of histogram and then it is converted to binary image for that purpose a reasonable threshold value is chosen. And then edge detection is done using a canny edge detector. The Canny's edge detection algorithm performs better than all other operators under almost all scenarios.

REFERENCE :

[1] Using an FPGA to Accelerate Pupil Isolation in Iris Recognition Jennifer L. Shafer, Hau Ngo, and Robert W. Ives United States Naval Academy Department of Electrical and Computer Engineering Annapolis, MD 21402 USA

[2] J. Daugman, "Probing the uniqueness and randomness of IrisCodes: Results from 200 billion iris pair comparisons." *Proceedings of the IEEE,* vol. 94, no. 11, pp 1927-1935

[3] Robert W. Ives, Randy Broussard, Lauren Kennell, Ryan Rakvic and Delores Etter, "Iris Recognition Using the Ridge Energy Direction (RED) Algorithm," *Proceedings of the 42nd Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA,Nov. 2008.

[4] John Canny, "A Computational Approach to Edge Detection," *IEEE Transactions of Pattern Analysis and Machine Intelligence,* vol. PAMI-8, no. 6,

[5] Ryan N. Rakvic, Bradley J. Ulis, Randy P. Broussard, and Robert W. Ives, "Iris Template Generation with Parallel Logic," *Proceedings of the 42nd Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Nov. 2008.

[6] A Distributed Canny Edge Detector And Its Implementation On Fpga *Qian Xu, Chaitali Chakrabarti and Lina J. Karam* School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ qianxu@asu.edu, chaitali@asu.edu, karam @asu.edu

[7] W. He and K. Yuan, "An improved Canny edge detector and its realization on FPGA," *WCICA*, pp. 6561 –6564, Jun. 2008.

[8] Monro, D. M., Rakshit, S., and Zhang, D, University of Bath, U.K. Iris Image Database, http://www.bath.ac.uk/ elec eng/pages/sipg/ irisweb

[9] A fast method for iris localization R. Meenakshi Sundaram Dept. of Information Technology Jadavpur University.2011

[10] Altera, "TriMatrix Embedded Memory Blocks in Stratix IV Devices," *Alter Corporation Stratix IV Device Handbook,* vol 1, March 2010.

◈◈◈