Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Dedicated hardware architecture for localizing iris in VW images

Vineet Kumar *, Abhijit Asati [1], Anu Gupta [1]

Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science Pilani, Pilani 333031, India

ARTICLE INFO

Article history:
Received 10 August 2020
Revised 17 October 2020
Accepted 8 November 2020
Available online xxxx

Keywords:
Dedicated hardware architecture
FPGA implementation
Iris localization

ABSTRACT

This study presents dedicated hardware for iris localization that can be used as a coprocessor in the development of real-time and low-cost embedded iris biometric systems. Though the hardware architecture is described for iris localization in the visible wavelength (VW) images, the concept used can be applied to near infrared (NIR) images as well. In general, the architecture can be used for a class of iris localization algorithms based on the edge-map generation and circular Hough transform (CHT). The architecture presented here generates the edge-maps for limbic and pupil boundary detection using median filtering followed by Sobel edge detection; however, an additional reflection removal module is used for pupil boundary detection. Further, the CHT hardware module detects circle in each edge-map. The proposed architecture was implemented in programmable logic of the Zynq-7000 SoC device from Xilinx. This hardware implementation gives an iris localization accuracy of 98.43% and average processing time of 5.148 ms for UBIRIS.v1 VW database images (200 × 150 pixel). The algorithm used is suitable for less unconstrained and frontal-view iris images captured with subjects' active participation; however, the images may contain non-ideal issues such as reflection and occlusion by eyelids and eyelashes.

1. Introduction

Iris localization, which deals with the detection of iris boundaries in an image, is the most time-consuming stage among all the stages of an iris biometric algorithm (Fig. 1) running either on a personal computer (PC) platform (Basit and Javed, 2007; López et al., 2011) or on a microprocessor-based embedded system (Grabowski and Napieralski, 2011) with limited database size (N). The dedicated hardware for an iris localization task can be very useful for achieving real-time performance in embedded iris biometric systems that use low-speed central processing units (CPUs).

The software implementation of iris recognition algorithms is usually carried out on platforms consisting of high-performance serial microprocessors (working at clock frequencies in the GHz range), cache memory, high-speed communication buses and additional units that facilitate rapid execution of complex algorithms. However, such software implementations could restrict the application of biometrics to specific markets because of the microprocessor cost, complete system cost, system size and high-power consumption. In order to spread the biometric security, it is needed to develop the embedded biometric systems that can be easily integrated into any kind of product for access control. Besides this, they are also developed as stand-alone portable systems, giving the advantages of small size, lightweight, low power and low cost. Such portable systems are available for fingerprint biometrics, for example, personal biometric token having size about today's pen drives (Basit and Javed, 2007; Liu-Jimenez et al., 2011). Nevertheless, these tokens are available for the fingerprint biometrics due to its higher acceptability rate (Jain et al., 1997); they are not yet developed for iris biometrics (Grabowski and Napieralski, 2011).

From the literature review, it was found that the main works towards development of embedded systems for iris recognition were presented in Rakvic et al. (2009), López et al. (2011), Grabowski and Napieralski (2011), Liu-Jimenez et al. (2011). These works are reviewed here briefly with emphasis on iris segmentation stage and this review shows the dedicated hardware for iris segmentation as a research gap in the development of embedded system for iris recognition.

In Rakvic et al. (2009), the authors demonstrate parallelization of iris recognition and they parallelized portions of iris segmentation, template creation and template matching on a field

* Corresponding author at: Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani Campus, India.
E-mail address: vineet.bitsp@gmail.com (V. Kumar).
[1] Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani Campus, India.

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2020.11.004
1319-1578/

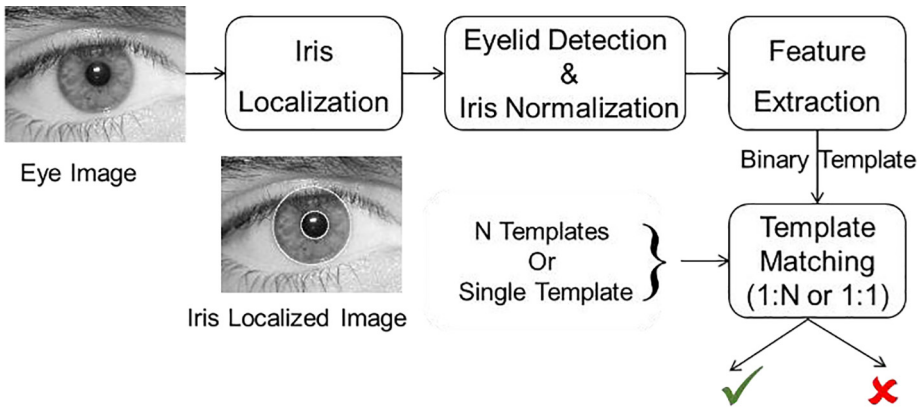


Fig. 1. Stages of iris biometric system.

programmable logic array (FPGA) based system and demonstrated speedup of 9.6, 324 and 19 times respectively as compared to a state-of-the-art CPU-based version. However, this work uses ridge energy detection (RED) algorithm and speedup results indicate that there is a scope for achieving better speedup factor for iris segmentation by using an algorithm other than RED algorithm.

The work in López et al. (2011) has set one of the best examples of embedded system design for iris recognition, which is based on hardware-software co-design. The authors implemented their hardware-software co-design on a low-cost Spartan 3 FPGA and the results showed that the execution time of the entire iris recognition algorithm was 522.6 ms for an image of 640 × 480 pixels with a clock frequency of 40 MHz. In this co-design, the major portion of the iris segmentation stage was implemented as dedicated hardware, as this stage was taking more than 71% of the total (all stages') execution time in software. The iris segmentation hardware took an execution time of 159 ms, but neither its hardware architecture design nor its accuracy evaluation was revealed. Moreover, execution time also seems non-optimized and there exists a scope for improvements in terms of a suitable selection of algorithm to achieve better speed and may be accuracy.

The authors in Grabowski and Napieralski (2011) describe the implementation of a complete iris identification system (1: N) on a multicore embedded system. This system architecture is mainly composed of digital signal processors (DSPs) and FPGAs. Their work shows that the iris segmentation is the most time-consuming task when executed on the PC based platforms, whereas it takes very less computation time when it was realized using multicore embedded system described in Grabowski and Napieralski (2011).

In Liu-Jimenez et al. (2011), the authors presented two implementations of iris biometrics on two different platforms used for embedded system development: (1) a microprocessor-based architecture; and (2) a dedicated hardware design. However, these implementations do not include iris segmentation stage of iris recognition due to its complexity and include only feature extraction and template matching. The second implementation (FPGA-based dedicated hardware solution) gives the best processing time, which shows 200 times speedup over microprocessor-based solutions. Both implementations exhibit benefits of data security, system size and cost as compared to the general-purpose computer systems.

The near infrared (NIR) iris images are traditional input to an iris recognition system, but in recent years, biometricians turned their attention to visible wavelength (VW) iris images acquired in the visible band of electromagnetic spectrum (Proença, 2010). This trend is supported by a variety of factors: (1) optical cameras in visible range are cheap and characterized by a very high resolution; (2) these cameras may capture face or iris images from a longer distance, which further may be used to authenticate a suspicious or violent individual; (3) today's smartphones would not need additional NIR camera for running iris recognition application; (4) the NIR wavelength may be hazardous because the eye does not instinctively respond with its natural mechanisms (aversion, blinking, and pupil contraction) (Proença, 2010).

The related works (López et al., 2011; Kumar et al., 2018; Giacometto et al., 2011; Ngo et al., 2014; Khan et al., 2019) on dedicated hardware implementation for the iris localization task are shown in Table 1, but unfortunately, they all are meant for NIR images and cannot be applied to VW images. The Table 1 mentions their salient features and performance characteristics, whereas Table 2 summarizes the limitations of these works. The symbol '--' in these tables indicates that the corresponding data is not available. The algorithm used in López et al. (2011) work approximates the iris boundaries' parameters using Daugman's IDO and then they are fine-tuned using active contours. However, only the integer part of active contours is implemented in hardware, whereas floating part is executed by the microprocessor. Both hardware architecture and accuracy of iris localization are not revealed by the authors in López et al. (2011). The accuracy is also not revealed by the authors in Giacometto et al. (2011). Moreover, (Giacometto et al., 2011) was tested only with CASIA-Iris-version 1.0 dataset, which is considered as one of the most ideal data set for NIR images. Ngo et al. (2014) have presented CHT hardware implementation for limbic boundary detection, but the work does not include hardware implementations of edge-map generation and pupil boundary detection. In a recent work, Khan et al. (2019) used an algorithm based on thresholding, morphological operations and connected component analysis. The algorithm was tested with multiple NIR data sets, but it cannot be applied to VW images. The proposed work targets VW images, however, the proposed architecture concept can be applied to NIR images as well, which means that mainly by changing edge-map generation techniques, the proposed architecture can be used for NIR iris localization.

In this paper, a hardware architecture for an iris localization task has been presented, which implements an algorithm based on the edge-map generation and circular Hough transform (CHT) to localize irises in VW images. The Xilinx's Zynq FPGA platform was used and performance was evaluated using UBIRIS.v1 database images. The difference between our previous work (Kumar et al., 2018) and this work is shown in Table 3. The term "consolidated architecture" in the table refers to the architecture module that accepts image as input and outputs the circle parameters of iris's outer and inner boundaries after its execution. Such an architecture was not presented in Kumar et al. (2018); only its components were presented without integrating them.

Table 1
Dedicated hardware implementations for iris localization (LUT: Look up table, FF: Flip flop, Reg: Register).

	FPGA Device	Hardware resource utilization	Method used	Image size (Pixel)	Accuracy (%)	Time (ms)	f_{max} (MHz)
López et al. (2011)	Spartan 3 XC3S2000	LUT = 1985, FF = 1765	IDO and active contour model (partial)	640 × 480	–	159	40
Giacometto et al. (2011)	Virtex-5 LX50T	LUT = 1594, Reg = 2805, BRAM = 2, DSP = 24	Intensity gradient image and CHT	320 × 280	–	56.59	100
Ngo et al. (2014)	A 40 nm FPGA	LUT = 9688, Reg = 12165, BRAM = 256	Edge-map and CHT	320 × 240	>92	5.15	215
Khan et al. (2019)	Cyclone IV EP4CE115	LUT = 4653, Reg = 2778, Multipliers = 22, Mem = 1641.3kbits	Region property	320 × 240	97.37–99.01	6.5	111
Kumar et al. (2018)	Zynq xc7z020-1clg484	LUT = 13691, Reg = 7115, BRAM = 68	Edge-map and CHT	320 × 240	95.06, 97.70	3.85	203

Table 2
Limitations of the related works listed in Table 1.

Iris localization hardware	Input type	Input data set used	Limitations
López et al. (2011)	NIR	–	• Hardware architecture and accuracy is not revealed
Giacometto et al. (2011)	NIR	CASIA Iris version 1.0	• Works only with ideal NIR images • Accuracy is not revealed
Ngo et al. (2014)	NIR	ICE-2005	• Iris should be visible as full circle • It detects only outer boundary
Khan et al. (2019)	NIR	MMU v1.0, ICE-2005, CASIA-IrisV3-Lamp, ND-IRIS-0405	• Does not work for images having dense eyelids and eyelashesocclusion or dark illumination • It does not work for VW images
Kumar et al. (2018)	NIR	CASIA Iris-Thousand, version 4.0, CASIA Iris-Lamp, version 3.0	• Components are not integrated as a single iris localization module • It does not work for VW images • It works for frontal-view images only
Proposed	VW	UBIRIS.v1	• Works for less unconstrained and frontal-view VW images

Table 3
Difference with our previous work (Kumar et al., 2018).

S. No.	Kumar et al. (2018)	This work
1	Works for NIR images	Works for VW images
2	Consolidated architecture for iris localization was not presented	It presents consolidated architecture and this idea would work for (Kumar et al., 2018) also
3	Hardware components presented are not sufficient for VW iris localization	Additional components, such as reflection removal and morphological dilation are presented
4	Edge-map generation techniques are different	

The remainder of this paper consists of sections as follows: Section 2 discusses the selection of algorithm and proposed approach targeting VW images. Section 3 presents the proposed hardware implementation. The synthesis and performance results are discussed in Section 4. Lastly, Section 5 concludes the work.

2. Algorithm selection and overview of proposed approach

In this section, the existing state-of-the-art algorithms for iris localization task are reviewed with respect to dedicated hardware implementation and an algorithm suitable for hardware implementation is proposed. The use of dedicated hardware is an alternative for implementing operations that require high-speed parallel processing. Designing a dedicated hardware solution may not be justifiable always specifically for the algorithms that majorly contain sequential operations and hinder the parallel and pipelined implementation. Therefore, algorithm selection becomes a crucial exercise before starting the design of dedicated hardware for an application.

Undoubtedly, integro-differential operator (IDO) and CHT based algorithms have been the most commonly used algorithms for iris localization (Proença, 2010; Roy et al., 2011; Shah and Ross, 2009). With respect to parallel implementation on FPGA, the IDO is found to be less suitable compared to CHT. In order to impart parallelism in IDO implementation, k copies of input image are required to store in memory corresponding to k radii for the circle detection since the IDO requires to read pixel intensity values of the image, whereas a single edge-map image is needed in CHT to perform parallel processing corresponding to k radii since the CHT does not require pixel intensity values of the image.

The performance (accuracy and time) of IDO and CHT based algorithms depends on image preprocessing steps, which precede the application of IDO or CHT, such as reflection removal, identifying potential circle centers, selection of pixels or region of interest and optimal edge-map generation techniques etc. (Kumar et al., 2016, 2015). In Kumar et al. (2018), the results show that the CHT accuracy degrades significantly with increase in variance of white Gaussian noise in the images due to increase in number of edge-points in edge-maps on which CHT is applied. Researchers have proposed new IDO or CHT based algorithms by proposing new image preprocessing techniques that improve overall performance of an iris localization algorithm. The details of some recent algorithms (Abdullah et al., 2017; Jan, 2017; Radman et al., 2013; Llano, 2018) for VW images are discussed in Section 4.4.

The edge-map generation and CHT based approach is chosen for hardware implementation due to parallelism offered by CHT as mentioned a short while ago. While applying this approach to NIR images, the pupil is detected prior to limbic boundary, whereas the reverse is done for VW images (Fig. 2), as limbic boundary has higher contrast than pupil boundary in VW images. The algorithm

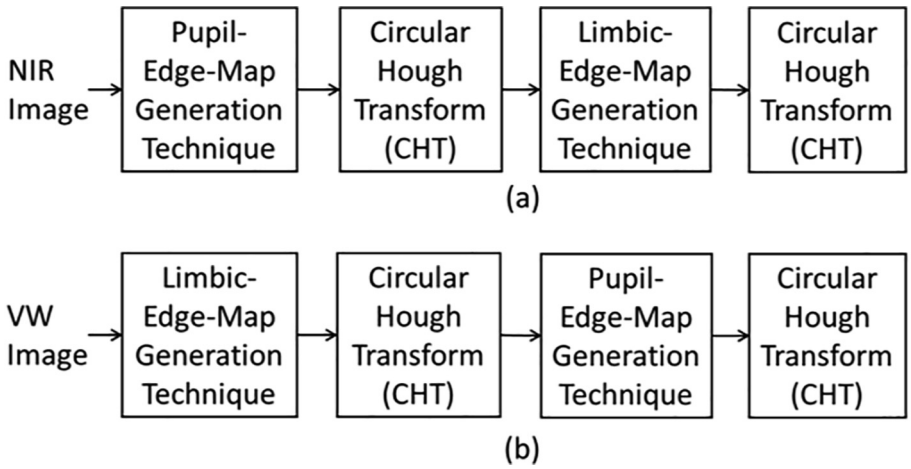


Fig. 2. Edge-map and CHT based iris localization for: (a) NIR images; (b) VW images.

for NIR images is described in our prior work (Kumar et al., 2018), whereas the algorithm for VW images is described here.

The occlusion by eyelids and eyelashes is a major problem in case of NIR images, as we have to localize first the small-size pupil in the presence of dense eyelids (please see Figs. 5 and 6 in (Kumar et al., 2016)) and then the iris outer boundary using pupil parameters. In contrast to this, in case of VW images (the presented paper), the iris outer boundary is localized first, whose visible circular contour is sufficiently large enough for detection by CHT in the presence of dense eyelids (see Fig. 13). However, to some extent, the dense eyelid occlusion could be a serious problem for VW images as well, for example, partially closed eyes or in non-frontal-view iris images.

2.1. Proposed approach

The proposed algorithm is suitable for less unconstrained VW images captured with subjects' active participation; however, the images may contain non-ideal issues such as reflection and occlusion by eyelids and eyelashes. The limbic or pupil boundary detection is a two-step process: (1) Edge-map generation; and (2) Circle detection in the edge-map using CHT. The proposed limbic and pupil boundary detection is described next in individual subsections.

2.1.1. Limbic boundary detection

The process of limbic boundary detection is illustrated in Fig. 3. The eye image is filtered with median filter prior to applying Sobel edge detection, which reduces false edges in the edge-map of eye image. The size of the median filter kernel is 5×5, which was chosen after experiments on given database images. The vertical Sobel edge detector is applied to the median-filtered image to detect vertical edges, as the horizontal limbic boundary-edges are not visible mostly in the images due to occlusions by eyelids and eyelashes. The Sobel kernel used for vertical edge detection is given in (Kumar et al., 2018), which computes the gradient magnitude using only x-derivative component of image gradients. Now, CHT is applied to the edge-map to detect the limbic boundary. The pseudo code for CHT algorithm and its implementation is presented in our previous work (Kumar et al., 2018).

2.1.2. Pupil boundary detection

The center of the limbic boundary circle defines the region of interest (ROI) for pupil boundary detection shown in Fig. 4(a). The lighting reflections in the pupil region cause hindrance in pupil boundary detection, which are therefore, removed (explained in detail shortly) before the edge detection step. After reflection removal, the edge-map is generated using median filtering and Sobel edge detection, as shown in Fig. 4(e) and Fig. 4(f) respectively. Both vertical and horizontal Sobel edge detection is used to detect the pupil boundary edges, as the pupil is mostly available as full circle in the images. The pupil circle is detected in the edge-map using CHT, as did in the previous subsection.

The reflections are first localized using thresholding operation on grayscale images and the localized reflections are shown as black pixels in the binary image in Fig. 4(b). This reflection map is enhanced through morphological dilation operation for black objects as shown in Fig. 4(c). In this case, black pixels are considered an object with a white background. The dilation operation increases the areas of localized reflections, so that the reflection map contains the reflections together with glows around them. The dilation operation uses 'disk' type structuring element of radius two. The coordinates of reflections in the enhanced reflection map (Fig. 4(c)) are determined and the pixel values of these coordinates in the original intensity image are replaced by a low pixel value, I_L. The intensity value, I_L is determined automatically by finding minimum pixel value (M) in the sub image (ROI) and adding 25 to M. The pupil is the darkest region in the ROI and the number, 25 accounts for reflection glows that increase the pixel values inside the pupil. The value 25 was determined heuristically by experiments on images for given database.

3. Proposed architecture for iris localization

This section describes the hardware implementation of the proposed approach discussed in the previous section. This paper uses the standard hardware design methodology, for example, after identifying main functions of the algorithm, the hardware architecture was drawn followed by Verilog modelling and simulation (an iterative process). The synthesis results were obtained targeting Xilinx's Zynq device. The proposed iris localization hardware module is shown in Fig. 5, whose input is an eye image applied as a pixel-stream (one pixel per clock cycle) and outputs are two sets of circle parameters (r, x_c, y_c) for limbic and pupil boundaries. The internal blocks of this module are shown in Fig. 6.

The proposed hardware achieves iris localization in two phases: (a) Limbic boundary detection; and (b) Pupil boundary detection. The 1-bit control signal, PD/LD' is used to switch from the first phase to second phase and it remains zero during the first phase (Fig. 6).

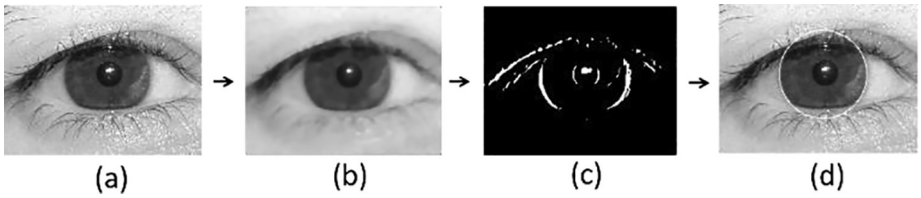


Fig. 3. Limbic boundary detection: (a) Eye image from UBIRIS.v1; (b) 5 × 5 median-filtered image; (c) Edge-map without edge thinning; (d) Limbic boundary detected after applying CHT on (c).

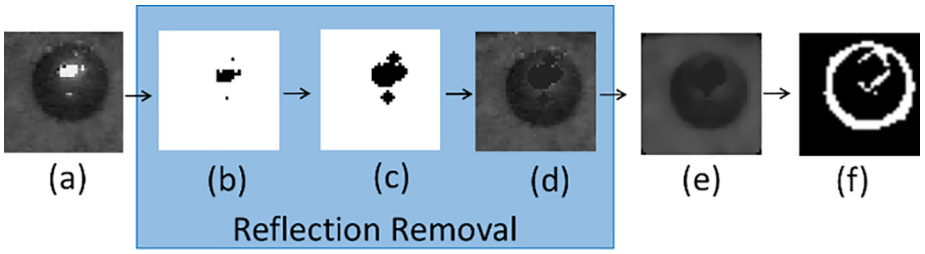


Fig. 4. Pupil boundary detection: (a) Region of interest (ROI); (b) Reflection map; (c) Enhanced reflection map; (d) Scrubbed reflections; (e) 5 × 5 median-filtered image; (f) Edge-map for CHT.

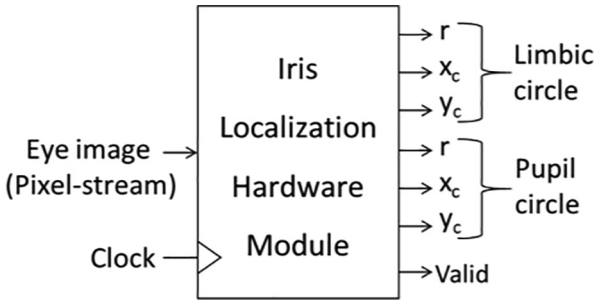


Fig. 5. Proposed iris localization module.

For PD/LD' = 0, the pixel-stream of input eye image directly passes through median filter and Sobel edge detector modules to generate edge-map for limbic boundary detection and at the same time the input image is also stored in BRAM0 (block RAM 0) for pupil boundary detection. The coordinates of edge pixels are stored in BRAM1 and BRAM2, which are then read by the CHT module to detect limbic boundary circle.

For PD/LD' = 1, the input eye image stored in BRAM0 is read (one pixel per clock cycle) and the image pixel-stream after passing through reflection removal module passes again through median filter and Sobel edge detector modules in order to generate edge-map for pupil boundary detection. The coordinates of new edge pixels are stored in BRAM1 and BRAM2 by overwriting the previous ones, which are then read by the CHT hardware module to detect pupil boundary circle.

The Sobel edge detector module works as 'vertical edge detector' during limbic boundary detection and it works as 'vertical plus horizontal edge detector' during pupil boundary detection. Moreover, threshold value (T) is also different for limbic and pupil boundary detection, which is selected using the PD/LD'. The control signal, PD/LD' also selects between a range of radii for limbic and pupil boundary detection inside the CHT module. The main

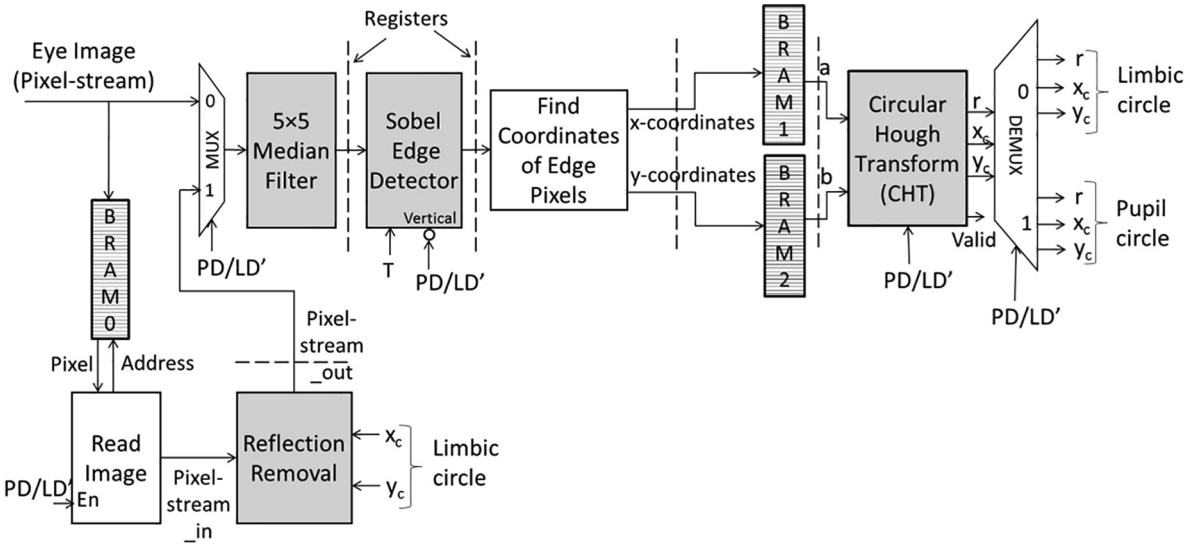


Fig. 6. Proposed iris localization architecture (internal block diagram of Fig. 5).

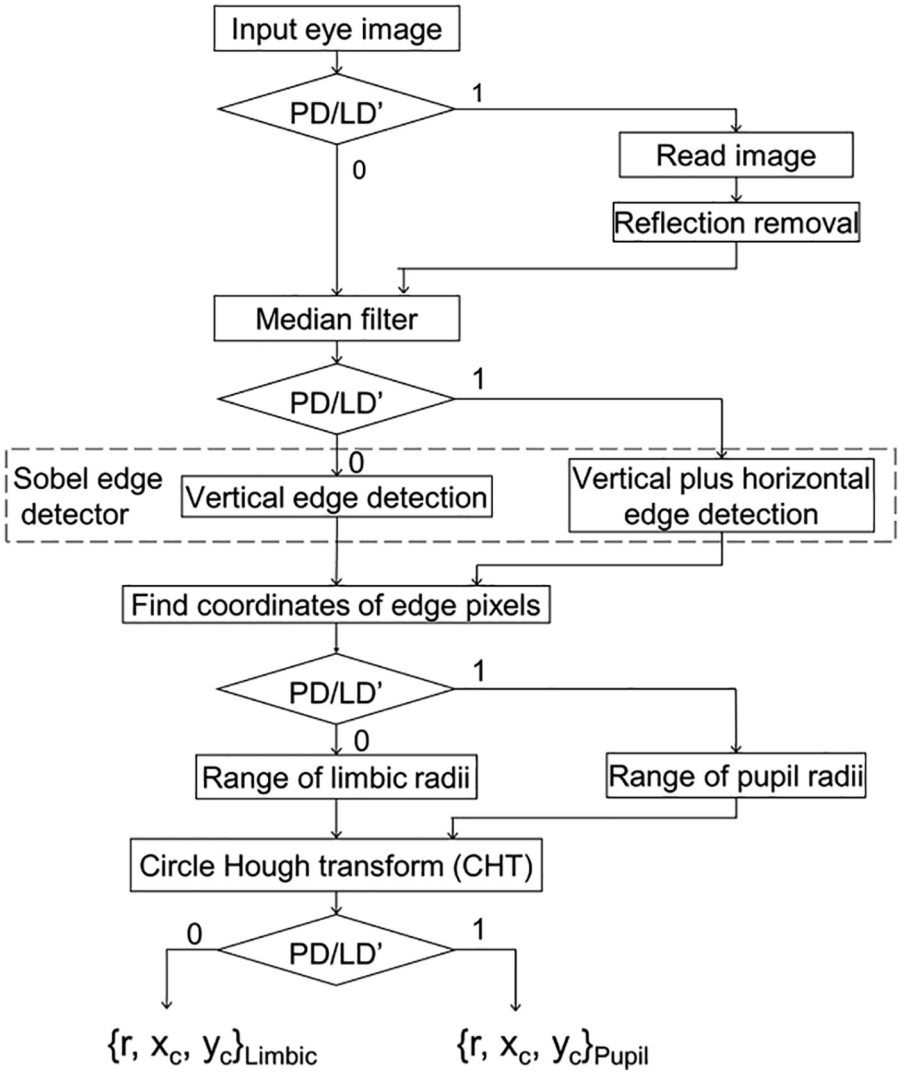


Fig. 7. Hardware control flow of the proposed architecture.

hardware components of Fig. 6 are 5 × 5 median filter, Sobel edge detector, reflection removal and CHT, whose architecture design and implementation are described in the next subsections. Fig. 7 shows the hardware control flow of the architecture presented in Fig. 6. The figure shows that '0′ value of PD/LD' follows a path leading to limbic boundary detection and '1′ value chooses the right-side path leading to pupil detection.

The proposed architecture concept can be used for NIR images also, but pupil and limbic edge-map generation techniques would be different. First PD/LD' would be set to 1 to detect pupil boundary and then to 0 to detect limbic boundary. The hardware implementations of edge-map generation techniques for NIR images are described in our previous work (Kumar et al., 2018).

3.1. 5 × 5 median filter

The hardware implementation of 5 × 5 median filter is described in our previous works (Kumar et al., 2018, 2017) and the readers are requested to refer to Section 4.2 in (Kumar et al., 2018) for this component. At every clock cycle, the median filter core accepts a new 5 × 5 pixel window of the input image and finds the median of it. The new window is generated by the sliding-window architecture (Kumar et al., 2018).

3.2. Sobel edge detector

The proposed Sobel edge detector architecture shown in Fig. 8 is different than (Kumar et al., 2018), in the sense that it can be used either for vertical edge detection or for vertical plus horizontal edge detection using the control signal, *Vertical*. The Sobel filter core computes G_x and G_y derivatives of the image gradients in parallel and then computes the gradient magnitude (GM) as shown in Fig. 8. The Sobel kernels used to compute the image gradients, G_x and G_y are given in (Kumar et al., 2018). The registers are used in this architecture to implement pipelining and obtain throughput of one edge pixel (output) per clock cycle. The upper half of the diagram shows a 3 × 3 sliding-window architecture, which provides a 3 × 3 pixel of the input image to the Sobel filter core every clock cycle after an initial latency of (2 W + 3) clock cycles for an image width, W.

3.3. Reflection removal

The proposed hardware architecture is shown in Fig. 9 that implements the reflection removal technique described in Section 2.1.2. This architecture accepts a pixel-stream of image at input 'Pixel-stream_in' and produces pixel-stream of the reflection-free image at output 'Pixel-stream_out' as shown in

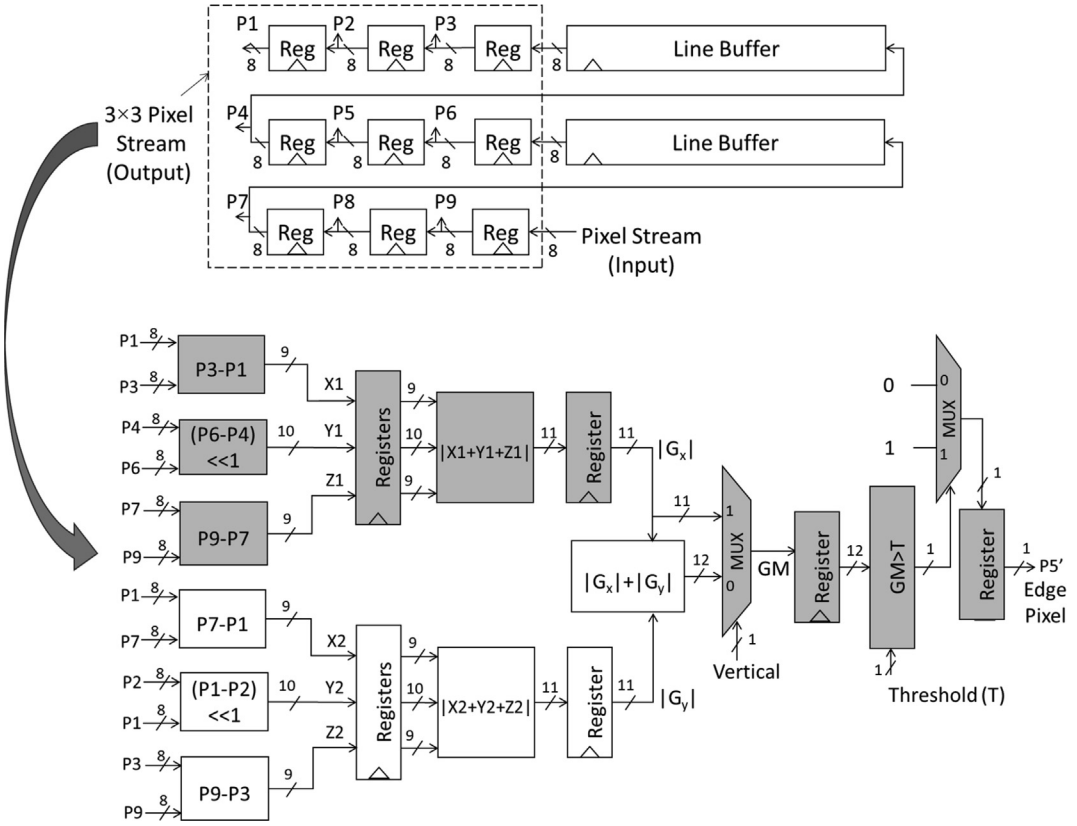


Fig. 8. Proposed Sobel edge detector architecture (the gray colored portion performs only the vertical edge detection).

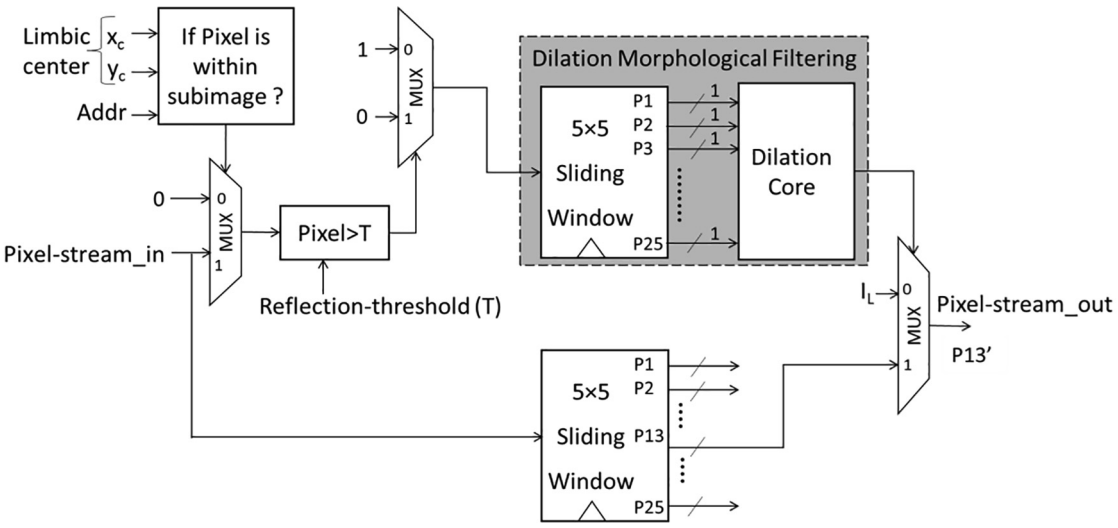


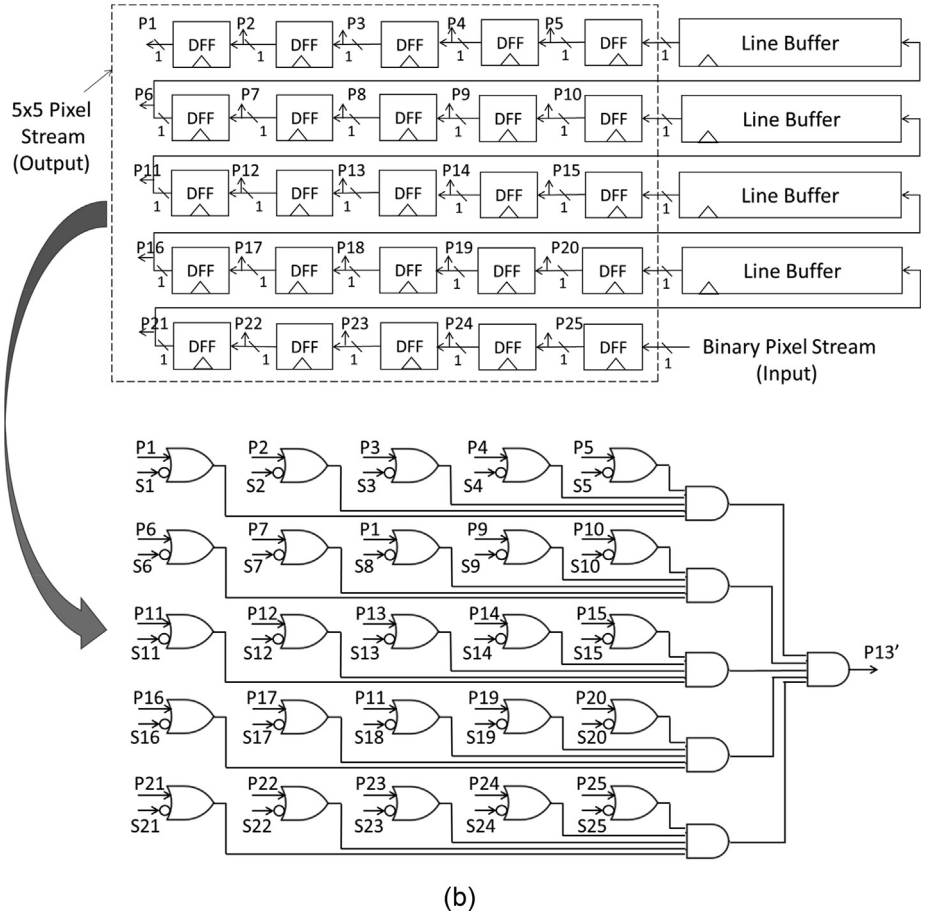
Fig. 9. The proposed architecture for reflection removal.

the figure, after an initial latency caused by the sliding-window component. This architecture is used to remove lighting reflections inside the pupil region. The region of interest (subimage) is located in the input image using the center (x_c,y_c) of the limbic boundary circle as discussed in Section 2.1.2.

The architecture in Fig. 9 reads the input image as a serial pixel-stream and the pixels that lie within the subimage are compared with a threshold value (T) to detect reflection pixels. The localized reflection area is increased using dilation morphological filtering for black objects, which uses a disk-type structuring element. The output of dilation filtering is either a zero or a one, where zero value represents the reflections (Fig. 9). These reflection pixels are replaced by a low intensity value, I_L (discussed in Section 2.1.2) in the input image using a multiplexer and a 5 × 5 sliding window as shown in Fig. 9. The proposed architecture for dilation filtering is shown in Fig. 10, which uses a 5 × 5 disk-type structuring element shown in Fig. 10(a). However, the architecture is programmable in the sense that it can be used for different types of structuring elements just by changing the values of S1, S2, and so on corresponding to a structuring element. The 5 × 5 sliding-window

$$\begin{bmatrix} S1 & S2 & S3 & S4 & S5 \\ S6 & S7 & S1 & S9 & S10 \\ S11 & S12 & S13 & S14 & S15 \\ S16 & S17 & S11 & S19 & S20 \\ S21 & S22 & S23 & S24 & S25 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

(a)



(b)

Fig. 10. Dilation morphological filtering: (a) Structuring element; (b) The proposed architecture with programmable structuring element.

architecture, shown in the upper half of the Fig. 10(b), uses 25 D flip-flops (DFFs) instead of 8-bit registers, as it generates pixel window of binary image rather than intensity image. The line buffer in this architecture is an image row buffer, which is essentially a 'first in, first out' register set of length, W-5, where W designates the image width.

3.4. Circular Hough transform (CHT)

The CHT hardware implementation was presented in our previous work (Kumar et al., 2018), but minor changes have been made here in the CHT architecture as compared to (Kumar et al., 2018), which is discussed as follows. In order to detect a circle in a 200 × 150 pixel image, the CHT architecture presented here uses three accumulator arrays of (200/m)×(150/n), 200 × 1 and 150 × 1 cells, therein reducing the memory requirement by a factor of m × n (approximately) compared to the direct CHT implementation, which uses a single accumulator array of 200 × 150 cells. The proposed CHT, presented here, chooses m = n = 2 for image-size of 200 × 150 pixels because accuracy starts degrading for higher values of *m* and *n*, whereas the (Kumar et al., 2018) uses m = n = 4 for image-size of 320 × 240 pixels. Thus, the memory requirement is reduced to 25% (approx.) or in other words, a memory reduction of 75% (approx.) is achieved by the proposed CHT compared to the direct CHT without any accuracy degradation.

4. Results and discussion

This section presents FPGA implementation results, the accuracy results and processing time of the proposed iris localization architecture. Furthermore, the results of the proposed work are compared with the published results, which show significant improvements.

4.1. FPGA implementation results

The proposed iris localization architecture was implemented with Verilog HDL to target Xilinx's 7 series Zynq FPGA. The Zynq device was selected as it contains CPU cores, which can be used in future to execute other less-compute-intensive stages of iris recognition when implementing the complete system. It is very difficult and not wise to implement dedicated FPGA hardware for complete iris recognition algorithm. Table 4 shows the synthesis results of the complete architecture presented in Fig. 6 for 200 × 150 pixel image. The maximum frequency, f_{max} obtained is 109 MHz. The target FPGA contains 36 K size block RAMs and each block RAM can be configured as a dual port RAM of aspect ratio 2 K × 16-bits. Each block can also be used as two independent 1 K × 16-bits dual port RAMs. The block RAMs are used mainly to realize line buffers of sliding-window architectures and the CHT accumulator arrays.

Table 4
Synthesis results of the proposed architecture (Fig. 6) for Zynq xc7z020-1clg484 device (f_{max} = 109 MHz).

Logic utilization	Used	Available on device
Number of slice registers	9132	106,400
Number of slice LUTs	15,476	53,200
Number of block RAMs	106	140

4.2. Iris localization accuracy

In order to evaluate the accuracy performance of the proposed iris localization hardware, the eye images were taken from UBIRIS.v1 database that contains JPEG images of size 200 × 150 pixels collected under VW light (Proença and Alexandre, 2005). The UBIRIS.v1 database is composed of 1877 images collected from 241 persons in two distinct sessions. The first image capture session of 1214 images was carried out with active participation from the subjects and contains less noise factors, especially those relative to reflections, luminosity and contrast. The second session includes heterogeneous images with respect to reflections, contrast, luminosity and focus problems. Moreover, the images were collected with minimal active participation from the subjects, adding several noise problems. In this paper, we have done experimentations with session 1 images, as our method works with less unconstrained images.

An additional module 'Draw Circles' was written in Verilog HDL that draws two white circles on the input eye image to show (indicate) iris boundary localization. It stores a copy of the input image in block RAM and takes input the circle parameters that are provided by the iris localization module as shown in Fig. 11. Both the iris localization hardware and 'Draw Circles' modules were ported on FPGA, which gives an iris localized image as output as shown in Fig. 12. The Fig. 12 is the set-up that was used to test and evaluate iris localization hardware on the FPGA, which uses the 'Xilinx's system generator for DSP' tool of VIVADO v2014.4 and ZedBoard (Zynq FPGA board).

A Simulink model was created and its snapshot is shown in Fig. 12, which uses Simulink blocksets for reading image files from computer's hard disk and displaying the output image. The 'Read Input Image' is a subsystem made-up of Simulink blocks that imports an image into the model from computer's hard disk, converts the image into 1D array and provide image pixel-stream to the hardware under test. The central block 'JTAG Cosim' in the Simulink model contains the bitstream file (.bit) of iris localization hardware along with 'Draw Circles' module for configuration of Zynq xc7z020-1clg484 FPGA on ZedBoard. The ZedBoard connects to the computer using JTAG cable. When the model runs first time, the FPGA device on ZedBoard gets programmed, receives the input pixel-stream from the computer, processes the pixels and sends the output pixel-stream back to the computer. The output pixel-stream from Zedboard is converted from a 1D array to 2D array (image) and it is displayed using video viewer by another Simulink subsystem 'View Iris Localized Image'.

Now, to test next image, the new image path is specified in 'Read Input Image' subsystem and the iris localization hardware runs again on FPGA, but this time model is run with skipping the FPGA configuration as FPGA is already programmed.

The accuracy results of iris localization are shown in Table 5 for all 1214 images of the session-1 from UBIRIS.v1 database. The presented work mostly used integer operations only and identical precision were used for both hardware and software implementations. However, at one place in CHT hardware implementation (Kumar et al., 2018), two 32-bit fixed-point multipliers were used for circle point generation (r × cosθ, r × sinθ), in contrast to floating-point multipliers in MATLAB implementation, but it did not affect the accuracy. The accuracy was calculated by manual inspection of output images using the formula, $(N_I/N_T) \times 100$, as shown in Table 5. The correct iris localized image in the table means that both limbic and pupil boundaries are localized correctly. The results in Table 5 show that both the MATLAB and hardware implementations have same accuracies.

The sample images of the correct iris localization are shown in Fig. 13 along with the edge-maps for limbic and pupil boundary detection. The correct iris localization means both pupil and limbic borders are detected accurately.

4.3. Processing time

The maximum frequency of operation was 109 MHz obtained from the synthesis of the proposed architecture, but the processing time is calculated for a clock of 100 MHz. The processing time was obtained by multiplying clock period with the number of clock cycles taken in completing the iris localization task. The average processing time per image is 5.148 ms calculated from 100 images of 200 × 150 pixel size. The computation time in the presented work is roughly directly proportional to the number of edge-points (N) in the edge-maps of iris image. These edge-points are inputs for the CHT module and time taken by the CHT module is directly proportional to the edge-points, as given by equation (5) in Kumar et al. (2018). Therefore, if the image size is doubled, that is 400 × 300 pixels, and assuming that the edge-points are also getting doubled, the computation time would be 10 ms approximately. However, this time can be reduced by reducing false edge-points in the edge-map of the image by using various techniques such as a large-sized median filter.

The MATLAB code of the proposed approach was written using the MATLAB's built-in functions, such as edge() for edge detection, ordfilt2() for median filter, im2bw() for thresholding and dilate() for dilation etc. The execution time is given in Table 6 when the code runs using MATLAB (version 8.4) installed on a laptop computer with Intel i5 CPU @ 2.40 GHz, 8 GB RAM and Windows 7 operating system.

The MATLAB code is not suitable to show a speedup of hardware over software. It is well-proved fact that the dedicated hardware of an algorithm always gives speedup over its software implementation (compiled machine code), which is also shown in one of our previous works Kumar et al. (2017), where hardware implementation of an edge-map generation technique for pupil detection achieves 206x speedup over software implemented in C++. Therefore, the authors feel that C/C++ implementation can be avoided here in order to show the speedup, which is a kind of additional information for the presented work.

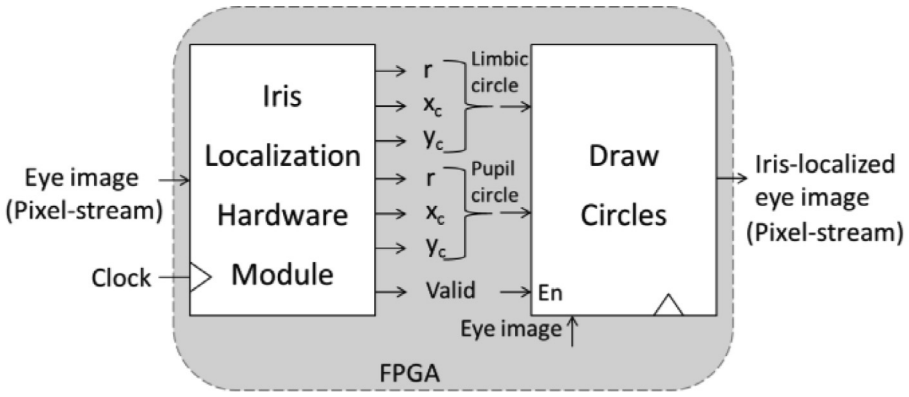


Fig. 11. Testing of iris localization module on FPGA.

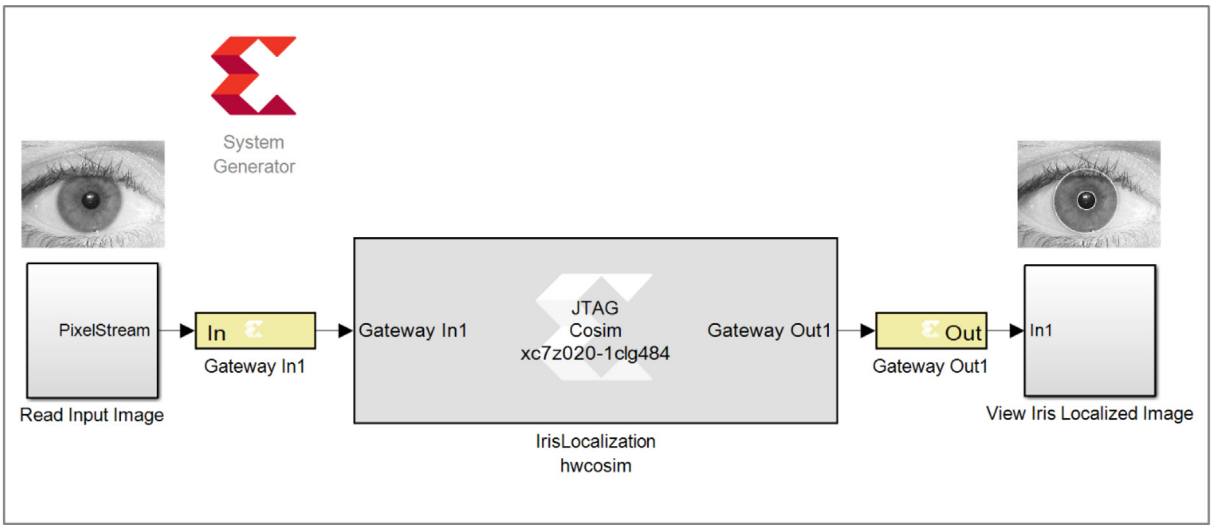


Fig. 12. Set-up to test and evaluate accuracy of iris localization hardware on FPGA.

Table 5
Accuracy results of iris localization for UBIRIS.v1 (Identical precision were used in both implementations).

Implementation	Number of images taken for testing (N_T)	Number of correct iris localized images (N_I)	Accuracy (%) = (N_I/N_T) × 100
MATLAB	1214	1195	98.43
FPGA	1214	1195	98.43

4.4. Comparison with previous works

The comparison in Table 7 shows that the accuracy of the proposed method is better than some of the state-of-the-art methods for UBIRIS.v1 data set images. However, the proposed method has limitation of working with less unconstrained images only. In Radman et al. (2013), the algorithm based on circular Gabor filter and IDO was developed to target near ideal VW images such as UBIRIS.v1, whereas (Jan, 2017) targets non-ideal VW images using a method based on reflection removal, IDO, image intensity and Fourier series. In Abdullah et al. (2017), a robust algorithm based on active contour force, is presented, which can localize irises in non-ideal images captured under either NIR or VW illuminations.

The proposed hardware architecture is not compared with previous works, as the published work on hardware implementation of VW iris localization is not available to the best of our knowledge.

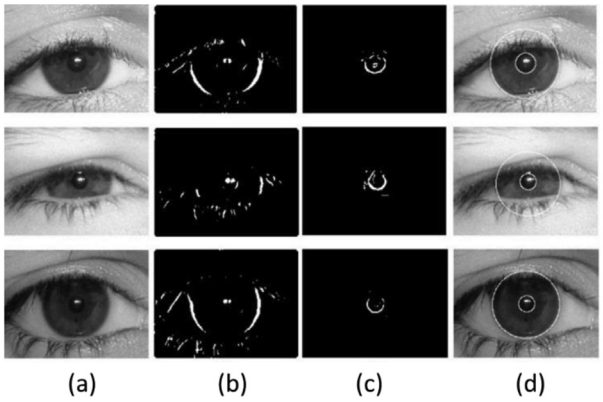


Fig. 13. Sample images obtained using the proposed work: (a) Original image; (b) Edge-map for limbic boundary detection; (c) Edge-map for pupil boundary detection; (d) Iris localized image.

Table 6
Processing time for 200 × 150 pixel image.

Process	Hardware module@100 MHz	MATLAB code CPU@2.40 GHz
Iris localization	5.148 ms	1.809 sec

Table 7
Comparison with published results for UBIRIS.v1 images.

Method	Accuracy (%)
Radman et al. (2013)	97.36
Jan (2017)	98.00
Abdullah et al. (2017)	96.50
Proposed	**98.43**

Moreover, it is not fair to compare the presented work with the related works listed in Table 1 due to different image sizes, devices and image type used.

5. Conclusion

The work presented in this paper provides a solution to the problem of dedicated hardware implementation for the iris localization task. This work is also useful for real-time iris localization in video-based biometric system that deals with video of high frame rate. All previously published works on dedicated hardware for iris localization are meant for NIR images, whereas the proposed work can localize irises in VW images. The proposed architecture concept can be used to integrate the hardware modules of edge-map generation and circle detection presented in our previous work (Kumar et al., 2018), in order to build a single iris localization module for NIR images. Moreover, some of the hardware components used in Kumar et al. (2018) are also being used in this paper, such as Sobel edge detection, median filter and CHT module. Therefore, these components can be reused if we combine these both works (this paper and (Kumar et al., 2018), and a new iris localization module can be developed, which would work for both NIR and VW images. There is also a scope for improving the proposed algorithm (for VW images) by using additional preprocessing, such as adaptive thresholding, so that it can work with unconstrained images as well. Currently, the proposed method works for frontal-view iris images and user has to look ahead towards the camera while capturing the iris images.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We would like to thank Prof. Hugo Proenca (Department of Computer Science, University of Beira Interior) for providing us UBIRIS.v1 database that was used in this research. This database was developed by University of Beira Interior, Portugal.

References

Basit, A., Javed, M.Y., 2007. Localization of iris in gray scale images using intensity gradient. Opt. Lasers Eng. 45 (12), 1107–1114.
López, M., Daugman, J., Cantó, E., 2011. Hardware-software co-design of an iris recognition algorithm. IET Inf. Secur. 5 (1), 60–68.
Grabowski, K., Napieralski, A., 2011. Hardware architecture optimized for iris recognition. IEEE Trans. Circuits Syst. Video Technol. 21 (9), 1293–1303.
Liu-Jimenez, J., Sanchez-Reillo, R., Fernandez-Saavedra, B., 2011. "Iris Biometrics for Embedded Systems", IEEE Trans. Very Large Scale Integr. Syst. 19 (2), 274–282.
Jain, A.K., Hong, L., Pankanti, S., Bolle, R., 1997. An identity-authentication system using fingerprints. Proc. IEEE 85 (9), 1365–1388.
Rakvic, R.N., Ulis, B.J., Broussard, R.P., Ives, R.W., Steiner, N., 2009. Parallelizing iris recognition. IEEE Trans. Inf. Forensics Secur. 4 (4), 812–823.
Proença, H., 2010. Iris recognition: on the segmentation of degraded images acquired in the visible wavelength. IEEE Trans. Pattern Anal. Mach. Intell. 32 (8), 1502–1516.
Kumar, V., Asati, A., Gupta, A., 2018. Hardware accelerators for iris localization. J. Signal Process. Syst. 90 (4), 655–671.
Giacometto, F.J., Vilardy, J.M., Torres, C.O., Mattos, L., 2011. Design and implementation of an algorithm for creating templates for the purpose of iris biometric authentication through the analysis of textures implemented on a FPGA. J. Phys. 274 (1), 1–13.
Ngo, H., Rakvic, R., Broussard, R., Ives, R., 2014. Resource-aware architecture design and implementation of Hough transform for a real-time iris boundary detection system. IEEE Trans. Consum. Electron. 60 (3), 485–492.
Khan, T.M., Bailey, D.G., Khan, M.A.U., Kong, Y., 2019. Real-time iris segmentation and its implementation on FPGA. J. Real-Time Image Proc. https://doi.org/10.1007/s11554-019-00859-w.
Roy, K., Bhattacharya, P., Suen, C.Y., 2011. Iris segmentation using variational level set method. Opt. Lasers Eng. 49 (4), 578–588.
Shah, S., Ross, A., 2009. Iris segmentation using geodesic active contours. IEEE Trans. Inf. Forensics Secur. 4 (4), 824–836.
Kumar, V., Asati, A., Gupta, A., 2016. A novel edge-map creation approach for highly accurate pupil localization in unconstrained infrared iris images. J. Electr. Comp. Eng. 2016, 10.
Kumar, V., Asati, A., Gupta, A., An Iris Localization Method for Noisy Infrared Iris Images, In: Proceedings of IEEE International Conference on Signal and Image Processing Applications, Kuala Lumpur, Malaysia, pp. 208–213, Oct 2015.
Kumar, V., Asati, A., Gupta, A., 2018. Memory-efficient architecture of circle Hough transform and its FPGA implementation for iris localization. IET Image Proc. 12 (10), 1753–1761.
Abdullah, M.A.M., Dlay, S.S., Woo, W.L., Chambers, J.A., 2017. Robust iris segmentation method based on a new active contour force with a noncircular normalization. IEEE Trans. Syst., Man, Cybernet.: Syst. 47 (12), 3128–3141.
Jan, F., 2017. Non-circular iris contours localization in the visible wavelength eye images. Comput. Electr. Eng. 62, 166–177.
Radman, A., Zainal, N., Jumari, K., 2013. Fast and reliable iris segmentation algorithm. IET Image Process. 7 (1), 42–49.
Llano, E.G. et al., 2018. Optimized robust multi-sensor scheme for simultaneous video and image iris recognition. Pattern Recogn. Lett. 101, 44–51.
Kumar, V., Asati, A., Gupta, A., 2017. Low-latency median filter core for hardware implementation of 5 × 5 median filtering. IET Image Proc. 11 (10), 927–934.
Kumar, V., Asati, A., Gupta, A., 2017. Hardware implementation of a novel edge-map generation technique for pupil detection in NIR images. Int. J. Eng. Sci. Technol. 20 (2), 694–704.
Proença, H., Alexandre, L.A., 2005. UBIRIS: A noisy iris image database. In: Proceedings of 13th International Conference on Image Analysis and Processing, pp. 970–977.