Iris Localization in Iris Recognition System: Algorithms and Hardware Implementation

THESIS

Submitted in partial fulfilment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

VINEET KUMAR

Under the Supervision of

Dr. Abhijit Asati

Prof. Anu Gupta

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE PILANI
PILANI - 333031 (RAJASTHAN) INDIA
2016

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the thesis entitled **"Iris Localization in Iris Recognition System: Algorithms and Hardware Implementation"** submitted by **Vineet Kumar,** ID. No. **2008PHXF433P** for award of Ph.D. of the Institute embodies original work done by him under my supervision.

(Signature of the Supervisor)

DR. ABHIJIT ASATI

Assistant Professor

Birla Institute of Technology & Science Pilani

Pilani - 333031 (Rajasthan) INDIA

(Signature of the Co-Supervisor)

PROF. ANU GUPTA

Associate Professor

Birla Institute of Technology & Science Pilani

Pilani - 333031 (Rajasthan) INDIA

Date:

Abstract

The iris recognition is now becoming a common authentication method in handheld consumer electronics devices, such as cellphones and tablets. The iris being a biometric parameter is a way better than password protection because of its uniqueness for each individual. The embedded iris recognition system is in demand, which will give the advantages of portability, light weight and small size. This kind of embedded system would be using in general the low speed and low power CPU, which may not give the real-time performance for the iris recognition as it involves computational-intensive image processing tasks, such as the iris localization. The iris localization is first step in the iris recognition, which is followed by the feature extraction and the iris-template matching, but the iris localization is most time consuming task in an iris recognition system with not having too big database size. Therefore, the hardware acceleration for iris localization can be very useful for obtaining the real-time performance. The field programmable gate array (FPGA) technology is preferred for realizing such a hardware accelerator.

The accuracy of iris recognition is most important for the applications demanding high security, where confidential data/resources must not be compromised. In order to make iris recognition algorithms more accurate, in general, the complexity of the processing is increased by adding more computational stages. However, when implementing these complex signal processing algorithms on embedded systems with limited processing resources, the challenge of achieving real-time performance arises.

Iris localization is an important stage in an iris recognition system, which involves image processing. The accuracy and speed of iris recognition depends on the iris localization algorithm. In this thesis, two aspects of iris localization are addressed: (a) Accurate and fast algorithms for the iris localization; and (b) FPGA based hardware accelerator for a selected iris localization algorithm. The objective of this research is to improve the performance of iris localization algorithms and implement the dedicated hardware for iris localization task without compromising accuracy. This dedicated hardware can be used to accelerate the iris localization task in more accurate and more affordable embedded iris recognition systems. Implementing dedicated

hardware for iris localization algorithm requires several embedded memory blocks (block RAMs) of FPGA, which makes it memory-intensive. This thesis work proposes a new hardware architecture, which is memory-efficient. Moreover, to improve accuracy, additional image preprocessing steps are implemented in the hardware, which further increases hardware complexity. Although, the iris localization algorithms proposed in the thesis use standard techniques, but additional image preprocessing steps have been introduced, which improves the overall performance.

This thesis is composed of seven chapters. Chapter 1 introduces the research field of the thesis and discusses about motivation behind carrying out this thesis work along with scope and objectives of the thesis. After presenting literature review in chapter 2, the main research work is exposed in chapter 3, 4, 5 and 6. Chapter 7 presents conclusion, main contributions derived from the work and future directions.

Acknowledgements

Foremost, I would like to express my humble gratitude and sincere thanks to my supervisors Dr. Abhijit Asati and Dr. Anu Gupta for their valuable guidance, encouragement, suggestions, and moral support throughout the period of this research work. It has been a privilege for me to work and learn under them.

Next, I would like to express my deep sense of satisfaction to BITS Pilani, Deemed University, for providing all the necessary facilities and support to complete the research work. My special thanks to Prof. V.S. Rao, Acting Vice Chancellor, BITS Pilani, and Prof. A.K. Sarkar, Director, BITS Pilani, Pilani campus, for giving me an opportunity to pursue my research successfully.

I am thankful to Prof. S.K. Verma, Dean, and Prof. Hemant R. Jadav, Associate Dean, Academic Research (Ph.D. Programme) for providing all necessary guidelines and extending full support which were crucial for the completion of this thesis.

I would like to thank members of Doctoral Advisory Committee, Prof. S. Gurunarayanan and Prof. Surekha Bhanot, for their critical comments, which helped me improve the quality of manuscript.

Thanks are due to the faculties of Dept. of Electrical & Electronics Engineering, especially Prof. Chandrashekhar, Prof. Vinod Kumar Chaubey and Prof. Navneet Gupta, for their constant motivation and encouragement.

List of Acronyms

1D	:	One-Dimensional
2D	:	Two-Dimensional
3D	:	Three-Dimensional
ASIC	:	Application Specific Integrated Circuit
BRAM	:	Block RAM
CASIA	:	Chinese Academy of Sciences Institute of Automation
CHT	:	Circular Hough Transform
CILV3	:	CASIA-Iris-Lamp, Version 3.0
CITHV4	:	CASIA-Iris-Thousand, Version 4.0
CIV1	:	CASIA-Iris, Version 1.0
CPU	:	Central Processing Unit
FIFO	:	First In, First Out
FPGA	:	Field Programmable Gate Array
HDL	:	Hardware Description Language
HT	:	Hough Transform
IDO	:	Integro-Differential Operator
MMUV1	:	Multimedia University, Version 1.0
NIR	:	Near Infrared
OTP	:	One Time Password
PC	:	Personal Computer
ROM	:	Read Only Memory
UBIRIS	:	University of Beira Interior Iris Database
USB	:	Universal Serial Bus
VW	:	Visible Wavelength

Table of Contents

List of Figures

List of Tables

This page was
left blank
intentionally.

Chapter 1
Introduction

1.1 Iris Biometric System

In the recent years, the field of automated human identification have grown with advent of different types of biometric systems, such as fingerprint, face, iris, voice and hand etc., which use sensors, image processing, pattern recognition and computers [Jain and Kumar, 2012], [Faundez-Zanuy, 2006]. These systems identify or authenticate the individuals for security and access control purposes based on their biometric characteristics (physiological or behavioral). The authentication systems based on biometrics determine the user's identity on the principle that some physiological or behavioral characteristics are unique for each person and hence, provide absolute authentication. For example, no two persons have the same iris-patterns even the identical twins; also, the left and right eye irises of the same person are different. Moreover, the iris pattern remains stable and does not change throughout one's life provided no injury happens to the eye. In contrast to biometrics, traditional methods of authenticating persons based on (a) what they carry, such as identity card, driving license, passport and magnetic stripe card; and (b) what they know, such as username, passwords and pin number are less secured methods because they can be stolen, lost, forgotten, misplaced or forged. Automated biometric systems such as fingerprint or iris recognition have been successfully deployed in several large-scale public applications, increasing reliability and convenience for users and reducing identity fraud [Jain and Kumar, 2012].

An iris recognition system is a computer-assisted system that identifies individuals based on comparisons of iris-patterns [Ross, 2010]. The iris recognition is considered one of the most secure and reliable technologies among currently existing biometric modalities and finds application where high level of security is required, such as countering terrorism or providing access control for accessing important information. The iris recognition system consists of an automatic iris segmentation system that extracts the iris region from eye image. The extracted iris

region is then unwrapped into a rectangular block with constant dimensions. The iris features are extracted with a feature extraction method to encode the unique pattern of the iris into biometric template. The generated iris template is compared with the previously stored (recorded) iris templates to find a match based on a match-threshold.

The two processes are involved in the use of biometric systems as shown in Figure 1.1: (a) enrollment and (b) recognition (verification). The enrollment is a process of creating the biometric template and storing it in the template store (database), whereas comparison (template matching) is done in recognition process in addition to the template creation. The enrollment for a user is generally done once and recognition is done every time a security check is performed.

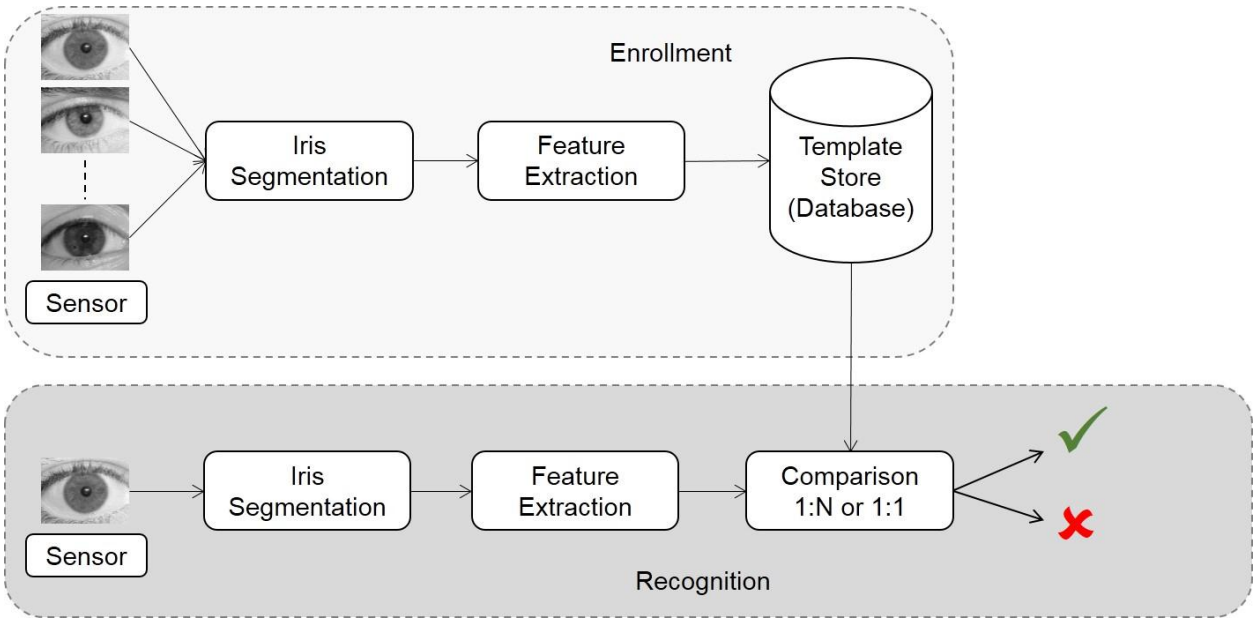


Figure 1.1. Functioning of iris biometric system.

A biometric system is used to perform two kinds of recognition tasks [Liu-Jimenez et al., 2011], [Faundez-Zanuy, 2006], [Bowyer et al., 2008]: (a) identification; and (b) authentication. These tasks are also known as modes of operation of a biometric system. In identification mode, the user does not provide any user identity (e.g. card or identification (ID) number) and the biometric system finds the user from a database of biometric data (templates). This process carries out a maximum of 1:N comparisons of the input template (biometric data) with the N templates stored in the database. In contrast to identification, authentication mode involves 1:1 comparison to recognize a user, but the user has to provide either his biometric data in form of a smart card (biometric token) or an ID number referring to his template stored in the database. Therefore, a biometric system for authentication task can be designed using two approaches [Liu-

Jimenez et al., 2011]: (a) online, which requires communication with central databases to access biometric data; and (b) offline, wherein biometric data is stored on personal biometric tokens. The online approach must deal with serious security and privacy issues, as the communication between the system and the central database can be attacked, and the identity may be stolen or altered. For this reason, offline systems are recommended, as long as the data is kept securely in the personal biometric token.

1.2 Motivation and Scope

A) Motivation

The iris recognition systems are available as both stationary devices (usually equipped with standard personal computers (PCs)) and portable devices, but usually the implementation of iris recognition algorithms is carried out using high-performance serial microprocessors working at clock frequencies in the GHz range. These devices are designed with an advanced architecture based on several pipeline stages, 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 such as mobile phones, automatic teller machines, laptops, personal digital assistant devices, access control systems, etc. It is worth to emphasize that realization of iris recognition using reliable and handheld embedded systems remains an open problem in the biometrics because of the complexity of the computational tasks [Grabowski and Napieralski, 2011].

Devices available in the low-cost consumer market are generally too slow for applications requiring intensive computations. For example, an iris recognition algorithm running on an ARM922 T at 160 MHz executes in 3162 ms, which is about 80 times slower than the execution of the same code on a high-performance microprocessor [Lopez et al., 2011]. The use of dedicated hardware is an alternative for implementing operations that require high-speed parallel processing. For example, under certain conditions, an image enhancement routine usually employed in a fingerprint recognition algorithm can be processed in dedicated hardware faster than on a Pentium clocked at a frequency 30 times higher [Lopez et al., 2011]. However, the speed benefits are obtained at the cost of chip area and design efforts. 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, sometimes, an embedded system architecture based on dedicated hardware units and serial microprocessor called as hardware-software (HW-SW) co-design can provide the optimized solution as described in [Lopez et al., 2011] for iris recognition application. However, sometimes, the complete algorithm can also be implemented using dedicated hardware units without using HW-SW co-design approach [Liu-Jimenez et al., 2011].

(a) (b)

Figure 1.2. Personal biometric tokens: **(a)** Techshino fingerprint USB token; **(b)** Fingerprint-based HYPR Biometric OTP.

Sources: (a) http://www.techshinobiometrics.com (b) https://www.hypr.com

An important and interesting application of small size, light weight and low power embedded iris recognition systems is token-based biometric systems that falls in the category of the offline biometric authentication, where the biometric data is stored in a smart card (token) instead of the centralized database [Liu-Jimenez et al., 2011]. The two different types of personal biometric tokens are used in offline biometric authentication: 1) the token only provides the biometric template of a person stored in it and does not perform the task of authentication; and 2) the token not only provides the biometric template, but also performs the authentication (verification) task and supplies the result. The tokens of second type avoid the external access to the user's biometric template, which is a better strategy for security and privacy point of view. Such tokens (second type) for fingerprint biometrics are available in the market having Universal serial bus (USB) interface or Bluetooth enabled communication with host machine (Figure 1.2). The Bluetooth enabled token is also known as biometric one time password (OTP), which is used for log in purpose as an alternative of traditional username and password-based log in. These are being used for access control in the applications such as online banking, mobile payments, electronic trading and computer unlocking. These tokens provide password-less authentication while improving users' experience, but they are also being used as an additional security check

along with traditional username and password, personal identification number (PIN) or OTP based authentication. For example, OTP is generated only when biometric check is passed using the token. USB token and biometric OTP (Bluetooth dongle) produced by Techshino and HYPR companies respectively are shown in Figure 1.2 having about size of today's pen drives. USB token is a compact plug and play design; can be used instantly upon insertion, empowered with on-device template matching and extendable fingerprint template storage that is can be used more than one persons.

These fingerprint tokens (Figure 1.2) are embedded system comprising of mainly a sensor and low power microprocessors to do the computations. Such biometric tokens for fingerprint biometrics are available in market, but are not yet developed for iris biometrics; the reason being that the fingerprint trait has higher acceptability rate [Jain et al., 1997]. The acceptability measures the people's acceptance level for the use of a biometric trait in their daily lives. The development of biometric token based on iris trait is current topic of research. The authors in [Liu-Jimenez et al., 2011] have proposed an optimized hardware solution for development of such iris biometric tokens, which is a dedicated hardware design on field programmable logic array (FPGA) as discussed later in chapter 2. However, it has limitations of reduced accuracy and not implementing iris segmentation task. The issues in the development of tokens-based complete biometric systems are low computational power (processing capability), less space and limited resources.

In the development of aforementioned embedded iris recognition systems, the usage of FPGA based platform plays an important role due to inexpensive nature for development, optimizing feasibility, reconfigurability and shorter time-to-market.

B) Scope

This thesis focuses on the iris localization stage of iris recognition system. The work carried out in the thesis can be divided in two parts. The first part of thesis describes the iris localization algorithms for localizing irises in less constrained images captured under near infrared (NIR) light. These algorithms show improved performance in terms of accuracy and time performance. The second part of thesis presents the dedicated hardware implementation of iris localization task on FPGA. This dedicated hardware for iris localization can work as a hardware accelerator in the embedded iris recognition systems, such as a biometric token, which may be having a low speed and low power CPUs operating in MHz. The hardware module for iris localization can be used in

such systems to meet the real-time performance. The thesis also presents preliminary work towards hardware implementation of iris localization for visible wavelength (VW) images.

Iris localization was chosen because it is the slowest process among all the stages in an iris recognition algorithm provided that the template database is not too large, otherwise template matching stage may become the slowest process [Lopez et al., 2011], [Grabowski and Napieralski, 2011]. The iris localization is very first stage in iris recognition that is performed on the captured eye image, therefore, its accuracy affects the stages following it and a wrong iris localization will fail the whole process of recognition. This thesis describes the algorithms for iris localization that are accurate and fast. However, only one selected algorithm is used for hardware implementation due to its reduced complexity in terms of memory requirement and yet offering more accurate and faster localization. This thesis work offers solution to one of the problems that is a dedicated hardware implementation for iris localization task, which can help in improving the speed and accuracy of iris recognition systems.

1.3 Objectives of Thesis

The objectives of this thesis were set after literature review and identification of research gap in the existing work, which is described in chapter 2. The main objectives of this thesis are:

1. To explore and improve the performance (speed and accuracy) of iris localization algorithms for NIR images in presence of image noise, such as eyelids and eyelashes occlusion, lighting reflections, non-uniform illumination, low contrast and eyebrow hair.
2. Comparison of results with previous iris localization algorithms and testing of the algorithms for different NIR image databases.
3. To implement selected algorithm on FPGA after algorithm optimization, which will work as dedicated hardware or hardware accelerator for iris localization. Comparison with previous work for hardware implementation of iris localization.
4. Preliminary work towards hardware implementation of iris localization for VW images.

1.4 Workflow

The task of algorithm development and its hardware implementation in this thesis was accomplished using the flow described in Figure 1.3. This flow was used for all the tasks as simple as edge-detection or as complex as circle detection in an image. Three main stages were used to carry out the work in this thesis:

1. In the first stage, a thorough study of existing algorithms was done for the iris localization task and then we have proposed algorithms to achieve better accuracy and time performance. The implementations of these algorithms were done using high-level computing and programming language; and MATLAB was used for algorithm coding, data visualization and performance evaluation under a personal computer platform. The built-in MATLAB functions for image processing, such as 'edge ( )' for edge detection were also used while coding the algorithm. After the initial composition of the algorithm, as set of iterative loops is carried out in order to tune the algorithm to the properties of accuracy and speed.

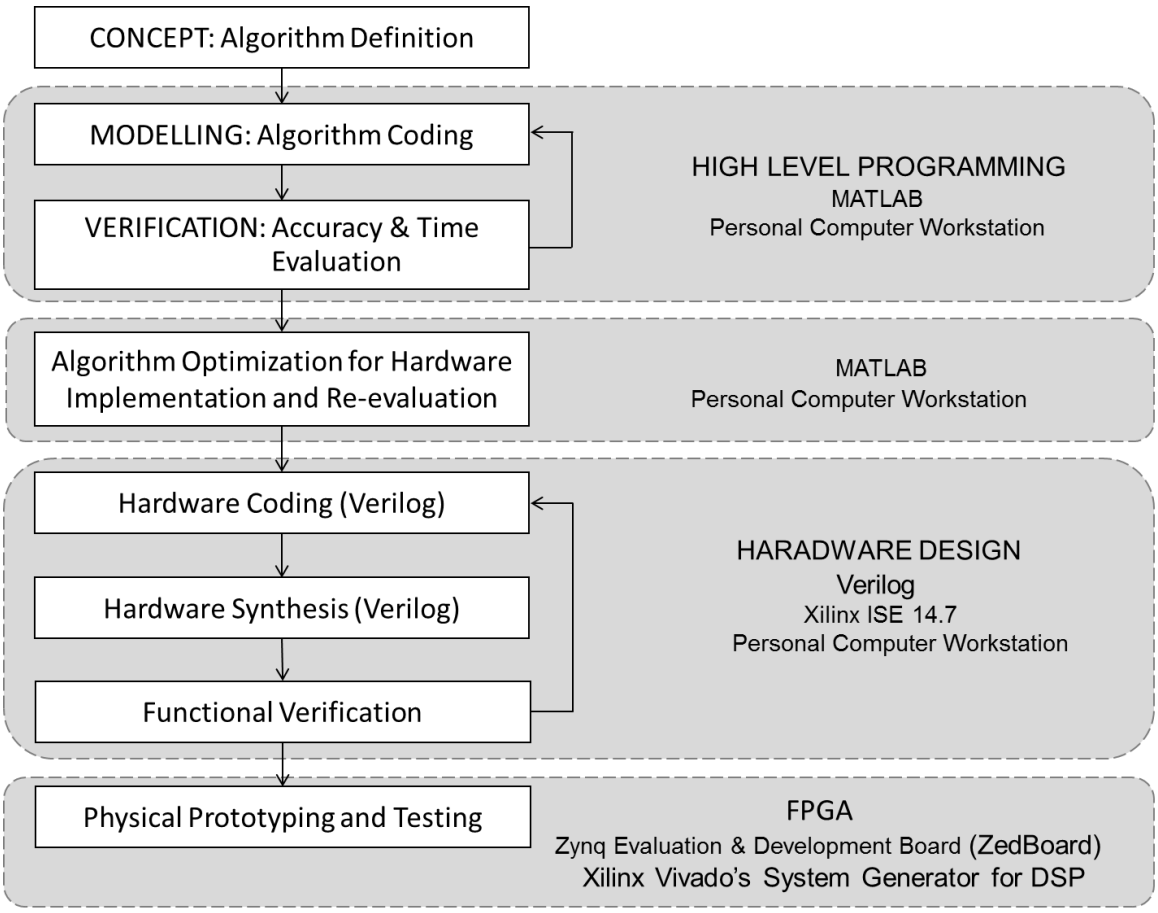


Figure 1.3. Algorithms and hardware development flow of thesis.

2. In second stage, the algorithm proposed in stage 1 was optimized for its hardware implementation by using steps, such as the optimized algorithm may be using reduced filter size for making the hardware resource efficient. The optimized algorithm was coded in MATLAB and its accuracy was evaluated to compare with the original algorithm proposed in stage 1. The optimized algorithm was again coded in MATLAB, but this time

the built-in functions of MATLAB were not used. For example, we wrote the code for convolution operation for performing edge detection on image instead of using built-in function 'edge ( )' of MATLAB.

3. In third stage, the flowchart of the algorithm was explored, main functions (operations) were identified and its hardware architecture was drawn with pen and paper, which was then modelled using Verilog Hardware Description Language (HDL). The hardware design had to go through a number of iterations to meet the functionality of the algorithm. After that, synthesizable design was ported on the FPGA for testing.

1.5 Organization of Thesis

This thesis is organized in seven chapters. After the introductory chapter, which presents the research topic and the main scope of this work, the thesis addresses each one of the following aspects in the next chapters.

Chapter 2 provides the literature review on iris recognition and embedded system development for iris recognition; and explores research gap, which helps in framing objectives of this thesis. This chapter also describes in detail the existing algorithms of iris localization and the main techniques used in iris localization. The existing work on FPGA based hardware implementation of iris localization has also been reviewed.

Chapter 3 discusses the iris localization based on integro-differential operator (IDO) and describes a method proposed by us that uses IDO to localize iris boundaries in the images captured under NIR light and contain significant noise. The method has been evaluated for its accuracy and time performance, and also compared with the previous methods. The algorithm was coded and tested in MATLAB installed on a personal computer (PC).

Chapter 4 describes the iris localization based on circular Hough transform (CHT). This chapter discusses implementation of Wildes' approach [Wildes, 1997] and a new method proposed by us for less constrained NIR images. The extensive evaluation of the algorithms has been done on different databases and results are compared with the existing algorithms. The algorithms were coded and tested using MATLAB on a PC platform.

Chapter 5 is devoted to the hardware implementation of an iris localization algorithm based on CHT for NIR images. The algorithm was first optimized for hardware implementation and impact of optimization has been evaluated. The hardware simulation and synthesis results targeting Xilinx's Zynq 7000 FPGA device are presented.

Chapter 6 describes the preliminary work towards hardware implementation of iris localization for VW images. We have proposed an iris localization algorithm for VW images, which is optimized for hardware implementation.

Chapter 7 concludes the thesis and presents the research contributions of this thesis. This chapter also discusses about the future directions of the work presented in the previous chapters.

1.6 Concluding Remarks

This chapter has introduced the research field of this thesis. The scope of development of new embedded systems for iris recognition, such as a biometric token, has motivated to carry out this thesis work to support such systems. The scope of this thesis has been discussed, which is about iris localization stage. Finally, the work methodology used in this thesis and focus of the different chapters was described.

Chapter 2
Literature Review

This chapter first discusses the literature review on iris recognition, the various stages of iris recognition and embedded systems for iris recognition. This preliminary literature review helped in identifying iris localization stage as a main research gap, which is the major hurdle in improving the performance of the embedded iris recognition system; and also helped in setting up the objectives of the thesis. Subsequently, a detailed literature review on iris localization algorithms and its hardware implementation has been presented.

2.1 Iris Recognition

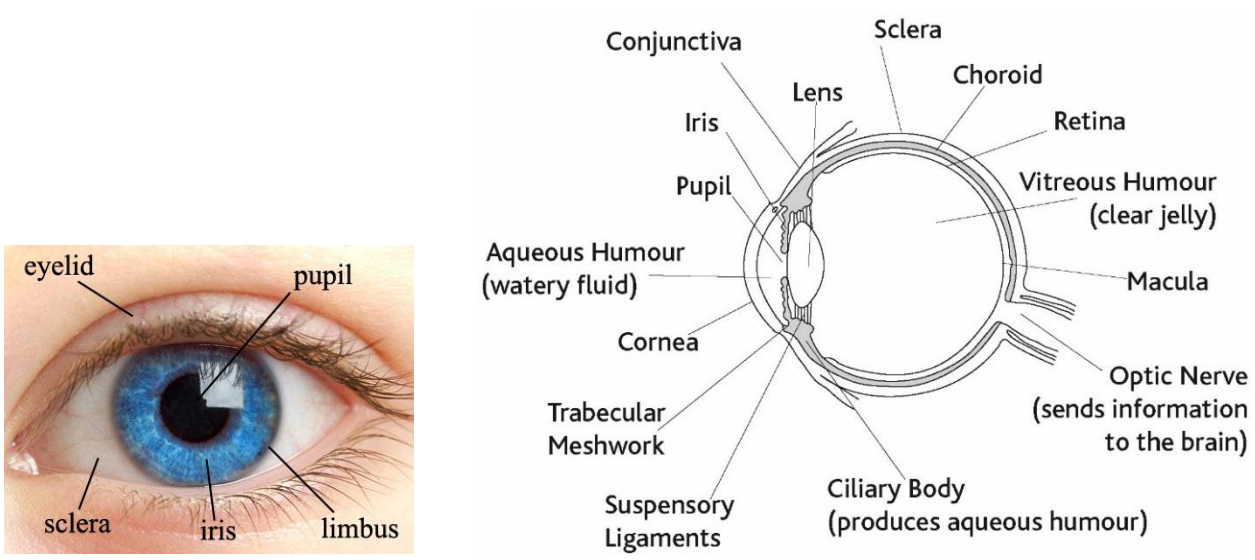
The iris recognition technique is used for automatic identification (or authentication) of persons by converting their irises into mathematical representations. An iris has features that make this modality appropriate for recognition purposes. This section presents an introduction to iris recognition and provides a literature review on the techniques that are being used in iris recognition systems.

2.1.1 Structures of Eye and Iris

In this subsection, we start with the description of the human eye structure, followed by a detailed description of the iris. The detail of the iris structure is necessary to know, since it becomes important to understand these for an iris based automatic identification process. Figure 2.1(a) shows the location of iris in a captured eye image. The iris is a ring-like structure fitted between pupillary and limbic boundaries and it is surrounded by sclera, pupil, eyelids and eyelashes.

A) Structure of Eye

The human eye is a ball-like structure as shown in Figure 2.1(b). The sclera is a tough, white and opaque outer coat of eye and the surface of sclera is covered by a thin skin layer called the conjunctiva. The front part of the eye, where the outer coat of eye is transparent, is called the cornea and behind the cornea is the iris. The iris is the colored part of the eye and it has the pupil at its center forming a hole. The most important function of the iris is that it controls the size of the pupil with help of its muscles. The size of pupil is sensitive to the light entering the eye. The pupil becomes larger in the dim light and smaller in the bright light. The light passes through the lens that helps focus the light from the pupil onto the retina. The light that has passed through the cornea and pupil now passes through the lens that helps focus the light onto the retina. The retina is light sensitive layer inside the back of the eye on which what is being seen is focused. The retina has high density of light-sensitive cells at its center and this part of retina is called macula. The photoreceptor cells of retina coverts light into a series of electrical signals. These signals pass to the brain via optic nerve, where the final image is processed.



(a) (b)

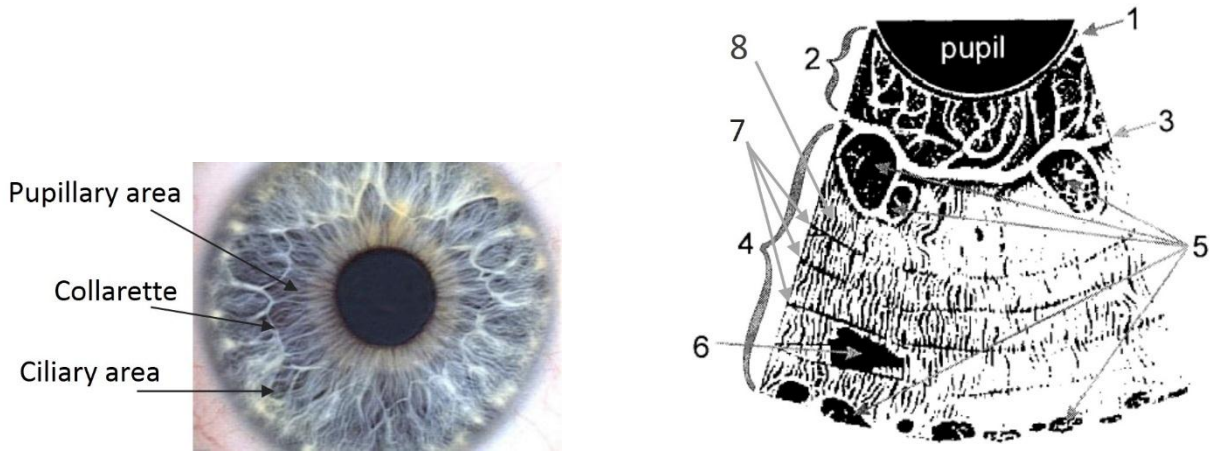
Figure 2.1. Human eye: **(a)** Location of iris in eye image; **(b)** Structure of human eye.
Source: http://www.glaucoma-association.com/about-glaucoma/the-eye

The space between the retina and lens is filled with vitreous humour through which the light passes to the retina. Also, there is an aqueous humour filled between cornea and the lens

that maintains the pressure of eye. The aqueous humour is produced by ciliary body. The layer of eye under the retina is choroid that provides the blood supply to the retina cells.

B) Visible Features of Iris

The tissue of the iris is soft and loosely woven and it is called stroma [Muron and Pospisil, 2000]. The density of stroma is one of the factors that determine the color of iris. The visible pattern of iris displays various distinctive features. The whole surface of the human iris is divided into the pupillary area and the ciliary area as shown in Figure 2.2(a). The collarette represent the zigzag boundary between the pupillary area the ciliary area and it is the thickest part of the iris. A few of the visible features of the human iris are shown in Figure 2.2(b), which are important features used for identifying a person. These are mainly pigment related features, features controlling the size of the pupil, pigment frill and collarette.



(a) (b)

Figure 2.2. Human iris: **(a)** Picture of human iris.; **(b)** Visible features on surface of the human iris: 1-Pigment frill, 2-Pupillary area, 3-Collarette, 4-Ciliary area, 5-Crypts, 6-Pigment spot, 7-Concentric furrows, 8-Radial furrows.
Sources: (a) John Daugman's home page: https://www.cl.cam.ac.uk/~jgd1000; (b) [Muron and Pospisil, 2000].

The crypts and the pigment spots belong to the pigment related features. The crypts are very dark colored areas and the iris is relatively thin in these areas. They appear like sharply demarcated excavations and their presence is near the collarette or/and the periphery of the iris. The pigment spots are called as moles and freckles having nearly black colour. They are random

concentrations of pigment cells and generally appear in the ciliary area [Muron and Pospisil, 2000].

The iris features that control the size of the pupil are radial and concentric furrows. These furrows are called contraction furrows. The radial furrows may start near the pupil and extend in the ciliary area through the collarette. The concentric furrows are generally circular and concentric with the pupil. These furrows appear in the ciliary area and near the periphery of the iris. The pigment frill represents the boundary between the pupil and the iris.

The iris grows from the ciliary body and its colour is given by the amount of pigment and by the density of the iris tissue, which means from blue to black.

2.1.2 Iris as a Biometric Trait

The properties of the iris that enhance its suitability for use in high confidence identification systems are discussed in this section [Daugman, 1993], [Daugman, 2004]. The advantages and disadvantages of using iris as a biometric identifier are discussed below.

Advantages:

1. Iris is a highly protected, internal organ of the eye. It is inherently isolated and protected from the external environment. It is impossible to surgically modify it without unacceptable risk to vision.
2. Iris is stable throughout the person's life. The human iris begins to form during the third month of gestation. The structure is complete by the eighth month of gestation. However, the pigmentation continues into the first year after birth.
3. There is high level of randomness in iris pattern (structure), which makes the iris unique for each person and hence, a reliable automatic biometric identifier.
4. A live detection test of iris can be performed with ease to avoid spoofing of the system using a photograph or video of iris, because the iris responds to the light and the changing size of pupil confirms natural physiology.
5. Images of the iris are adequate for personal identification with very high confidence. They can be acquired from distances of up to about 3 feet.
6. The iris image is captured at some distance from a subject without physical contact, which makes the identification process unobtrusive.
7. There is no genetic determination of iris pattern. The identical twins have different iris patterns; and the left and right eye irises of the same person are also different.

Disadvantages:

1. Iris is a small target to capture from a distance (>1m).
2. Iris is a moving target in non-cooperative iris recognition, which is difficult to acquire.
3. Iris is obstructed by eyelids, eyelashes and reflections that make the automatic iris recognition algorithms more complex.
4. NIR illumination is preferred to capture the iris; illumination should not be visible or bright.

2.1.3 Iris Image Acquisition

Image acquisition set up determines quality of the iris image data. Different choices of the camera sensor, lens, illumination and capture distance will result in data of varying quality [Wildes, 1997]. Almost all commercial iris recognition systems are using near infrared (NIR) illumination sources and NIR cameras. This is because color cameras are not as effective as NIR cameras in terms of capturing the textural information in iris. Visually dark brown (almost black) irises do not display much color variation and thus contain almost no information useful for iris recognition. Due to the requirement to be low cost, the iris recognition systems often involve inexpensive and thus low quality sensors and optics. This considerably limits capture distances as well as quality of captured data. In order to protect the eye from overheating and thus from damage, the strength of the NIR illumination is required not to exceed a predetermined threshold [Matey et al., 2006]. As was mentioned before, the NIR iris images are traditional input to 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 [Proenca et al., 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) [Proenca, 2010].

2.1.4 Iris Image Database

This section describes a number of iris image databases that are public and free available online. These databases are being used in the research on iris recognition to evaluate the performance of

algorithms developed for this application. These databases have been developed under two different types of illuminations: (1) NIR; and (2) VW. The images in different databases contain different level of obstructions and noises, such that an algorithm giving accurate results for one database may not be accurate for other database [Bowyer et al., 2008].

Some of the databases have been developed under constrained environment and with user's full cooperation. For example, these database images have minimum obstructions by eyelids and eyelashes, reflections confined to pupil area only, uniform illumination and frontal view that is user was looking straight towards camera while capturing the image, etc.

A few databases have been developed under unconstrained environment and contain non-ideal images, such as the images having the reflection spots appearing anywhere in the image, eyeglasses, non-uniform illumination, low contrast and heavy obstructions by eyelids, eyelashes and eyebrow etc. One more type of non-ideality is that the images have the off-angled irises (non-frontal view), which are captured while the user is not looking straight towards the camera.

The quality of the iris images decides both complexity and accuracy of the iris recognition algorithms [Bowyer et al., 2008]. The most popular NIR and VW image databases that are being used to evaluate today's iris recognition algorithms are described below.

A) NIR Image Databases

Most of the available iris image databases are the NIR images [Bowyer et al., 2008] because iris patterns are well pronounced in these images. The most commonly used NIR image databases are the CASIA Iris Image Databases (CASIA-Iris), which were developed by the Center of Biometrics and Security Research group in China [CASIA Iris Database]. The four versions (sets) of databases, CASIA-IrisV1 to CASIA-IrisV4, have been released by this research group since 2002 and each set has different subsets also. Up to thousands of images are contained in a subset, for example a subset CASIA-Iris-Thousand, version 4.0 (CITHV4) database contains 20000 images collected from 1000 persons and a subset CASIA-Iris-Lamp, version 3.0 (CILV3) database contains 16212 images from 811 persons.

The examples of other NIR databases are Multimedia University (MMU), Iris Challenge Evaluation (ICE)-2005, West Virginia University (WVU) and Indian Institute of Technology Delhi (IITD) databases. The summary of these databases except IITD database are provided in [Bowyer et al., 2008], where the information regarding the image capturing device (camera), size of database, way to get the database and links to download the database are given. A few sample images from these databases are shown in Figure 2.3. The two versions of the MMU database are

available: (1) Multimedia University, version 1.0 (MMUV1); and (2) Multimedia University, version 2.0 (MMUV2).

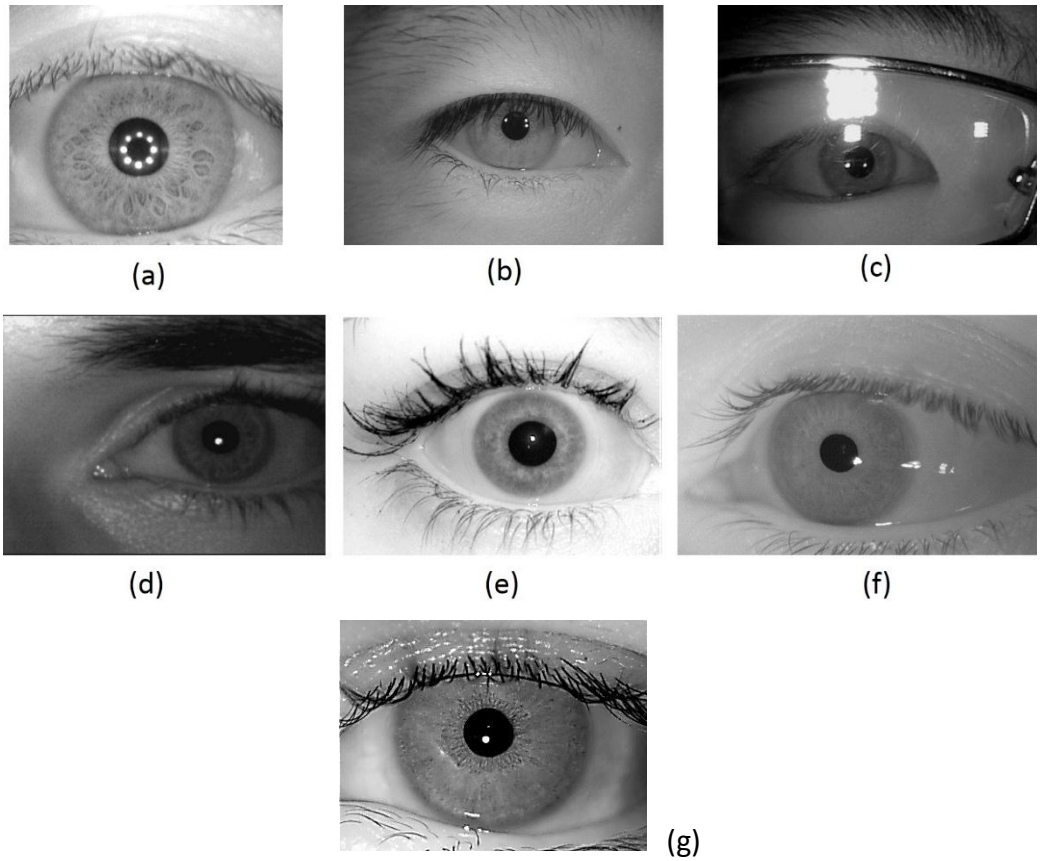


Figure 2.3. Sample images from NIR databases: **(a)** CASIA-Iris-Interval, version 3.0; **(b)** CASIA-Iris-Lamp, version 3.0; **(c)** CASIA-Iris-Thousand, version 4.0; **(d)** MMU, version 2.0; **(e)** ICE-2005; **(f)** WVU (off-angle image); **(g)** IITD, version 1.0.

The iris images from the NIR databases can be categorized into two types: (1) High close-up and constrained iris images, such as CASIA-Iris-Interval, version 3.0 database images (Figure 2.3(a)); and (2) Medium close-up and less constrained iris images, such as CITHV4 database images (Figure 2.3(c)).

B) VW Image Databases

The most famous iris image database developed under visible wavelength illumination is University of Beira Interior iris database (UBIRIS) [UBIRIS Image Database]. It was developed with the purpose that this database would work as a helping tool to evaluate the feasibility of visible wavelength iris recognition. First version of UBIRIS was released in 2004 and is called

UBIRIS.v1 [Proença and Alexandre, 2005], whereas the second version, UBIRIS.v2 [Proenca et al. 2010] was developed in 2009. These databases are public and free available. The example images from these databases are shown in Figure 2.4.

Figure 2.4. Sample images from VW databases: **(a)** UBIRIS.v1; **(b)** UBIRIS.v2.

The UBIRIS.v1 database is composed of 1877 images collected from 241 eyes. These images were acquired under less constrained imaging conditions as compared to UBIRIS.v2. The UBIRIS.v2 database has over 11000 images (and continuously growing) and it contains more realistic noise factors. The images in this version were captured at a distance and on the move.

2.1.5 Stages of Iris Recognition

The typical stages of an iris recognition algorithm are shown in Figure 2.5. The rectangular boxes in this figure denote the different stages and function performed in each stage is written in the box. The first stage of an iris recognition algorithm is iris localization, which takes input from image acquisition system that deals with capturing the eye (iris) image using a digital camera as discussed before. In the iris localization stage, the inner (pupillary) and outer (limbic) boundaries of the iris are detected. The localized iris is then converted to a fixed size rectangular strip to compensate variations in pupil size and in the image capturing distances. This step is known as iris normalization. The iris is generally occluded by the eyelid (s) in the captured image; hence,

an eyelid detection step is carried out after the iris localization, so that the eyelid(s) can be avoided in further processing. The eyelid detection step combined together with the iris localization stage is also called iris segmentation. The normalized iris acts as the input for feature extraction stage. The feature extraction is a process that creates the binary template of iris. This process is known as isolation of distinctive features and the encoding phase. The final stage of iris recognition is template matching in which the comparison between the iris templates is made, producing a numeric dissimilarity value (d). If this value (d) is lower than a threshold (t), the system outputs a decision of match found, which means that both templates were extracted from the same iris. Otherwise, the system outputs a decision of match not found, meaning that the templates belong to different irises. The matching stage is required to do 1:1 (one is to one) comparison or 1:N (one is to many) comparisons depending on the mode of operation of iris recognition system that is verification or identification respectively as discussed in section 1.1 of chapter 1.

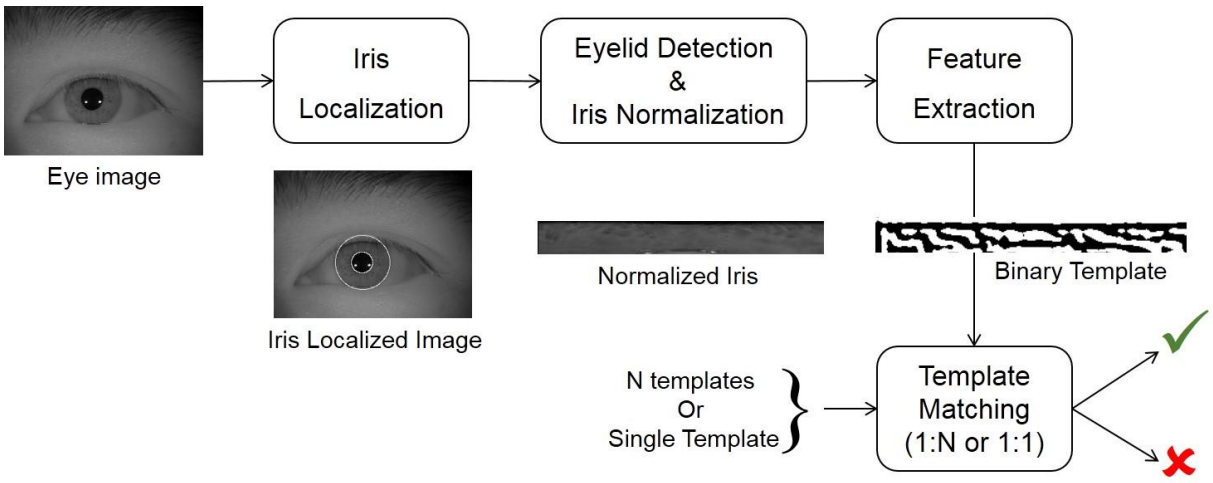


Figure 2.5. Stages of an iris recognition system.

Having defined the different stages of iris recognition system, these stages are described next in detail and a review on the most relevant approaches to perform each stage has been discussed.

A) Iris Localization

The purpose of iris localization is to locate the iris in the eye image for further processing. Iris localization is a very important stage in iris recognition systems because all the subsequent stages

depend on its accuracy. The pupillary and limbic boundaries of iris are considered as two circles in the frontal view (zero off angle) images, but these circles are not necessarily concentric. The frontal view images are obtained when user looks ahead towards the camera while capturing the images. Most of the iris localization methods model iris as a set of two non-concentric circles, but presence of different kinds of noise in the eye images, such as lighting reflections, occlusions by the eyelids and eyelashes, eyeglasses, low contrast and eyebrow hair etc., make iris localization difficult.

The iris localization methods in the commercial iris recognition systems are typically based on Daugman and/or Wildes's algorithms [Daugman, 1993], [Wildes, 1997]. Daugman proposed an integro-differential operator (IDO) that acts as a circular edge detector to localize the iris boundaries, whereas Wides' method uses the Hough transform (HT) to detect the circular edges. However, these iris localization methods work under the controlled environments and their performance deteriorates when dealing with the noisy data [Jan et al., 2014]. After Daugman and Wildes's methods, the researchers have proposed several iris localization methods with the improved accuracy and speed for different databases. Some of the methods are tabulated in [Shah and Ross, 2009], which shows that most of the iris localization methods still use the HT and IDO based techniques along with additional steps to improve the performance of previous algorithms. Some of the recent methods to localize the irises in the NIR and VW images are described in [Jan et al. 2012], [Jan et al., 2014], [Wang et al., 2014], [Zuo and Schmid, 2010] and [Li et al., 2010], [Proenca, 2010], [Sahmoud and Abuhaiba, 2013] respectively.

The more detailed review on existing iris localization methods has been presented in section 2.4 later in this chapter.

B) Eyelid Detection and Iris Normalization

i) Eyelid Detection

The upper and/or lower eyelids(s) may be occluding the iris in the eye images. One of the solutions to this problem is that exclude the upper and lower most parts of each iris that is expected to be covered by eyelids, which is done after the iris localization step. The eyelids are checked for their presence within the localized iris, so the region of interest for eyelid detection is inside the limbic boundary. The Daugman's IDO can also be used to detect the eyelids by changing path of contour search from circular to arculate [Daugman, 1993]. The upper and lower eyelids are also modelled as parabolic arcs as suggested by [Wildes, 1997], where the eyelids are

detected by using a HT based technique. The other methods for eyelids detection are based on parabola polynomial fitting [Basit and Javed, 2007], line fitting and line detection based on HT approach [Masek and Kovesi, 2003].

ii) Iris Normalization

The purpose of iris normalization step is to prepare appropriate input for the feature extraction stage. The different images of the same eye have different sizes of the iris because the distance between the user and image capturing device (camera) can vary. Moreover, the pupil size also varies on varying intensity or position of lighting source, which causes variations in the iris patterns of same eye. Therefore, a proper normalization technique is required to compensate these variations. The circular ring-like iris is unwrapped into a fixed size rectangular strip in the iris normalization step, but the iris boundaries are not concentric circles, which increases the complexity of this process. The most common methods for iris normalization are based on rubber sheet model [Daugman, 2004], which is used to counter the size variations of irises as discussed above.

C) Feature Extraction

Feature extraction or encoding is performed to extract salient features representing an iris. This step ensures that data are compactly represented and also ensures that noise introduced by acquisition system is suppressed or removed. Therefore, an image preprocessing of normalized iris is generally done to enhance the contrast and remove noise due to eyelashes and uneven pixel intensities.

Researchers have suggested many different filters to extract iris features, such as Gabor filters [Daugman, 2004], modified Log-Gabor filters [Yao et al., 2006], Gaussian filters [Lu and Lu, 2005], median filters [Zuo et al., 2008], dyadic wavelet transform [Ma et al., 2004] and discrete cosine transform [Monro et al., 2007] etc. The extracted features were quantized to two levels to obtain the binary iris codes. The techniques other than filter-based techniques can also be used to construct the binary iris codes, such as local histogram equalization and a quotient thresholding [Thoonsaengngam et al., 2006].

Instead of creating binary iris codes, the real-valued feature vectors can be generated using independent component analysis (ICA), principal component analysis (PCA) and linear discriminant analysis (LDA) etc. [Dorairaj et al., 2005], [Zuo et al., 2008].

D) Template Matching

The most commonly used matching technique is performed using Hamming distance [Daugman, 2006]. This method measures the similarity/difference of two iris codes. Hamming distance method for comparing two iris codes is more reliable than the methods based on Euclidean distance and Zero-crossing distance [Lopez et al., 2011]. Hamming distance is a fractional value that ranges between 0 and 1 and represents the extent of similarity or disagreement between the two iris codes. A lower value of the Hamming distance represents a greater similarity and a high value represents dissimilar irises. Access to the system is granted only when the distance is lower than a certain threshold. The threshold value depends on the application for which the biometric system is used. The Hamming distance is computed by performing XOR (Exclusive-OR) for every bit of the two iris codes. The comparison is done as shown in Equation (2.1) for calculating Hamming distance.

$$HD(y,p) = \frac{1}{L}\sum_{i=1}^{L}(y_i \oplus p_i) \tag{2.1}$$

Where

L = dimensions of the vector

y_i = i^{th} component of the sample feature vector

p_i = i^{th} component of the template feature vector

XOR is an inequality operation that gives a high output if the bits under comparison are disagreeing and a low output if the bits under comparison agree in their values.

2.2 Embedded System for Iris Recognition

Today, most of the iris recognition systems are using PC based platforms as they do not need hard real time deadlines, which makes these systems bulky and costly; restricting their use in certain application domains. In order to spread the biometric security all over the world, it is needed to develop the embedded biometric systems that can be easily integrated into any kind of product such as mobile phones, automatic teller machines, laptops, personal digital assistant devices, access control systems, etc. This section describes about embedded system and then discusses the commonly used architecture alternatives for embedded system based design for iris recognition along with its detailed literature review.

2.2.1 Definition of Embedded System

An embedded system is a generalized term for many systems including biometric systems, which satisfy all or at least most of the following requirements [Berger, 2001].

1. An embedded system in general is designed for a specific purpose, whereas PCs are designed to perform several tasks and therefore, are considered as general-purpose devices.
2. An embedded system is typically small in dimensions due to their performance restrictions. In the PC case, several peripherals are required such as a screen, mouse and keyboard, hard disk, video enhancement boards and cooling mechanisms, whereas only the necessary peripherals are used in embedded systems.
3. The cost of embedded systems is lower than that of a general-purpose machine.
4. The embedded systems prefer ROM memories to store their programs instead of hard disks or any other big storage device. The ROM memories reduce the storage capability and therefore, most of the systems use real-time operating systems (RTOS) or an embedded code called firmware, where its size is much smaller than the general-purpose operating systems.
5. Most of the embedded systems perform under real time constraints due to the applications that range from time sensitive to time critical. For example, time is important in case of mobile telephones, but not restrictive as opposed to the instantaneous response required from a car airbag control system.
6. There are several applications for these type of systems. They may form part of a larger system. An embedded system may be a stand-alone device or a co-processor. When the system is stand-alone, it is typically a battery-powered device.

The large deployment of these systems has been motivated by the reduction in the area of the hardware required to implement complex systems. The increase in density of transistors on silicon chip has helped to develop more complex systems within a reduced space. Todays' embedded systems provide increased performance within a smaller space.

2.2.2 Architectures for Embedded Systems

Several architectures may be used for designing embedded systems, which depends on the central element and the scheme to be followed [Noegaard, 2005]. This section discusses about some of these architectures along with their main advantages and disadvantages.

The most commonly used solution is based on the use of a microprocessor or a microcontroller as the central processor unit. Several peripherals surround the microprocessor

and are required to perform the desired function. Microprocessors or microcontrollers require code to be executed due to which this type of solution is often referred as a software design for an embedded system. These solutions perform the code sequentially and therefore, several tasks are not time-optimized or must wait to be performed, as the microprocessor is only able to perform one task at a time. These architectures are quite flexible, and at the same time, are relatively easy to work with. The main disadvantage to this type of system is that they are not as fast as others are. It is also important to highlight systems that use more than one microprocessor, one as a central unit and the rest as co-processors used for different purposes. These are used in high performance systems, where several microprocessors are used for different tasks.

Application-specific integrated circuits (ASICs) or full custom circuits are the best option when massive production and high performance is desired. The main problem with this type of solutions is related to their fixed costs and designing a solution of this type requires not only an experienced designer, which is more expensive than other personnel, but also specific developing tools, longer design time and more complex facilities when compared to other more straightforward architectures. However, once designed, these systems are cheaper than other solutions with respect to the manufacturing process. The investment can be recovered by the massive production of these systems, as the cost per unit is greatly reduced when compared to microprocessor architectures. At the same time, the hardware area required for these systems is smaller than other solutions that are used to perform the same task; this makes this solution suitable for small devices and helps to reduce the cost per unit. Because of the above-mentioned problems, full custom solutions are being used less and less each day, and are only being used in embedded systems where the performance is applicable to different systems and therefore, can be commercialized for different purposes.

Hardware-software architectures are a half way solution when considering the previously discussed alternatives. These architectures are characterized by the use of both possible solutions in order to obtain the benefits from both types of systems; they use dedicated hardware, to perform some tasks and a microprocessor to perform others. By using this combination, the inherent advantages of both systems are obtained: such as reduced time, reduced area and also low power consumption. This architecture is more complex than others are and requires knowledge of both hardware and software. In spite of the difficulties that arise, co-design provides several advantages when compared to other alternative architectures. When the computational load of the system is high and it is under real-time restrictions, the alternative of using both approaches becomes popular and effective. By combining both solutions, advantages from both techniques are obtained: the simplicity of using software for sequential processes, and

the use of dedicated hardware to perform specific tasks, which require high performance. As a result, process acceleration is achieved. However, several parts of the design process become more complicated, requiring additional tedious tests. Not only problems from each platform should be solved but also those related to the connections and communication between them.

Among all the potential architectures, we would like to highlight the differences between conventional hardware-software solutions and System on Chip (SoC) solutions. In the first case, the microprocessor and the hardware are located on different chips. However, due to the transistor integration capacity, many chips that contain both a microprocessor and dedicated hardware are becoming increasingly popular. These chips are a half way solution between the conventional microprocessors and the full custom chip solutions.

FPGA Based Architectures

Among the different commercial alternatives Field Programmable Gate Arrays (FPGA) are currently regarded as the most interesting. A FPGA is a semiconductor device that can be configured by the customer or designer after manufacturing, hence the name 'field-programmable'. FPGAs are programmed using a logic circuit diagram or source code based on a HDL to specify how the chip is to operate. The FPGA can be used to implement any logical function that an ASIC can perform, but the ability to update the functionality after manufacturing offers advantages for many applications. FPGAs contain programmable logic components called 'logic blocks' and a hierarchy of reconfigurable interconnections that allow the blocks to be 'wired together'.

In today's system design approach, the system simulation is usually performed on a FPGA based platform, which is capable of combining both the hardware and software solutions. Both the microprocessors and dedicated hardware can be implemented on the same FPGA device, where debugging tools are provided by manufacturers for this purpose.

2.2.3 Existing Works on Embedded Iris Recognition Systems

A) Parallelizing Iris Recognition [Rakvic et al., 2009]

This research work [Rakvic et al., 2009] demonstrates parallelization of three major components of the Ridge Energy Detection (RED) iris recognition algorithm: (1) iris segmentation with local kurtosis; (2) template creation via filtering; (3) template matching via hamming distance. In

particular, the [Rakvic et al., 2009] authors parallelized portions of iris segmentation, template creation and template matching on an 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.

The FPGA experiment was executed by the authors on a DE2 board manufactured by Altera Corporation. The DE2 board consists of a Cyclone-II EP2C35 FPGA chip, as well as the required FPGA programming interface. The Cyclone-II family is designed for high-performance and low-power applications. It contains over 30000 logic elements and over 480000 embedded memory bits. The clock signal was drawn from DE2 board, which contains a 50 MHz clock.

The authors' parallel algorithm on FPGA greatly outperforms their calculated theoretical best Intel CPU design. The authors conclude that a full implementation of a very fast iris recognition algorithm on a state-of-the-art FPGA, is more than feasible, which would provide a small form-factor solution.

B) Hardware–Software Co-Design of an Iris Recognition Algorithm [López et al., 2011]

This work [Lopez et al., 2011] has set one of the best examples of embedded system design for iris recognition algorithm. The main purpose of this work was to implement an iris recognition algorithm using a low-cost FPGA. The architecture alternative used for implementing the iris recognition algorithm was based on a hardware-software co-design. The experimental results reported in this work were obtained using a Spartan-3 FPGA clocked at 40 MHz. The system architecture consists of a 32-bit soft-core microprocessor (Xilinx Microblaze) and several dedicated hardware units.

All the stages of the iris recognition algorithm, such as image preprocessing, iris segmentation, iris normalization, feature extraction and iris-template matching were implemented. Out of these stages, the less computation intensive tasks and those involving floating-point operations are executed by the microprocessor in software, whereas the tasks having higher computational cost are speed up by coprocessors, the hardware accelerators. The image preprocessing task for removing specular reflections and the iris localization (pupillary and limbic boundary detection) stage were found suitable for hardware implementation because they were taking 71% of the total (all stages') execution time in software. Moreover, the reflection removal and the iris localization involve integer arithmetic operations and they have an acceleration ratio of greater than 11 as compared to their software execution.

All the remaining stages of iris recognition other than iris localization, such as eyelid detection and iris normalization, feature extraction and iris-template matching, were executed on

the microprocessor because they mainly contain floating point operations and represent a small percentage of the total execution time (each one <5%) in software. In addition, their theoretical acceleration ratio was low. The active contours function was used to fine-tune pupillary and limbic boundaries detection. This function was partially implemented in hardware.

The [Lopez et al., 2011] 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. The best software solution for the iris recognition algorithm implemented on a microprocessor as Pentium 133 MHz gave an execution time of 1112 ms, which is about 2.12 times slower than Lopez et al. system operating at a clock frequency 3.3 times lower.

C) Hardware Architecture Optimized for Iris Recognition [Grabowski and Napieralski, 2011]

This work [Grabowski and Napieralski, 2011] describes the implementation of the complete iris identification system (1:N) on a multicore embedded system. This system architecture is mainly composed of digital signal processors (DSPs) and FPGAs. The algorithms for iris recognition used in this architecture are the algorithms developed by the authors on a PC platform in their previous work [Sankowski et al., 2010]. 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]. The feature extraction and the template matching stages of iris recognition were also implemented using DSPs. However, the authors plan to implement the template matching stage on dedicated hardware using FPGA in the future, as they found this stage most time consuming stage in their embedded system.

The [Grabowski and Napieralski, 2011] authors applied their hardware architecture to the biometric application BioServer platform that consists of two separate physical boards: (1) biometric system (BioSys) that consists a Virtex-5 FXT Xilinx ML510 platform forming the basis for an embedded system, which is based on two PowerPC 440 microprocessors; (2) the biometric computation unit (BioCU) that consists a Xilinx Spartan 3AN FPGA and four DSPs. The BioCU board is inserted into a 32-bit slot on the ML510 platform of BioSys.

The authors compared the performance of their multicore embedded system for iris recognition with the implementations on three different PC-based platforms: (1) PC1 based on Intel Core2 6600 2.4 GHz; (2) PC2 based on Intel Pentium4 3.4 GHz; (3) PC3 based on Intel

Core2 Duo T7100 1.79 GHz. For testing the performance, the size of the image was of 640×480 pixels, the template size was equal to 2048 bits and database size was N=10000 templates. The comparison with these implementations showed that their embedded system was fastest.

D) Iris Biometrics for Embedded Systems [Liu-Jimenez et al., 2011]

This work [Liu-Jimenez et al., 2011] 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 and include only feature extraction and template matching. The first implementation uses an ARM7TDMI microprocessor, which is a 16/32-bit RISC CPU and it is widely used in many commercial applications due to its high computational power and reduced cost. The second implementation that is the dedicated hardware uses Xilinx's Virtex4sx35 FPGA, which has been specially designed for digital signal processing, providing MAC units that reduce the implementation cost of complex transforms, such as fast wavelet transform (FWT). The microprocessor in the first implementation running @ 50 MHz takes 10 times more processing time than computer platform with CPU@2.6 GHz, but in relative term, the microprocessor gives optimized solution as it is clocked 50 times lower than the computer platform. The second implementation (FPGA-based dedicated hardware solution) gives the best processing time, which shows 200 times speed up over microprocessor-based solutions, and 20 times speed up over computer-based solution. Both the microprocessor and FPGA–based implementations exhibit benefits of data security, system size and cost as compared to the general-purpose computer systems. Selecting one of these two platforms depends on system and authentication application requirements.

2.3 Gaps in Existing Research

In [Lopez et al., 2011], when iris recognition algorithm is run on two different microprocessors, Intel Centrino 1.7GHz and 32-bit ARM922T 160MHz, the iris localization function takes execution time of 71% and 89% respectively of the total execution time of the overall algorithm. The literature review presented in the previous sections shows that iris localization is the most time consuming stage in the whole iris recognition algorithm either running on a PC platform [Lopez et al., 2011], [Basit and Javed, 2007] or on a microprocessor-based embedded system [Grabowski and Napieralski, 2011] with limited database size or when 1:1 comparison is

required. Therefore, iris localization is the best candidate for dedicated hardware implementation while developing embedded iris recognition systems. The iris localization was implemented on FPGA in [Lopez et al., 2011] and it took an execution time of 159 ms, but neither the hardware architecture design of iris localization nor its accuracy evaluation was revealed. Moreover, execution time also seems non-optimized and there exists a scope for improvements in terms of suitable selection of algorithm to achieve better speed and may be accuracy. In [Lopez et al., 2011], the IDO based algorithm was used for FPGA implementation, but we found the IDO less suitable compared to CHT for parallel implementation on FPGA. To impart parallelism in the IDO based algorithm, k copies of the input image are required to store in the 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 literature review also suggests that the FPGA based platforms containing single or multicore processors (CPUs) and the configurable logic in a single chip are best suited for development of embedded iris recognition systems. Therefore, implementing computation intensive tasks, such as iris localization on FPGA as a dedicated hardware, can offload the CPU by not executing the computation intensive tasks on it, which will improve the processing time of the whole system. The [Liu-Jimenez et al., 2011] authors implemented dedicated hardware for feature extraction and matching stages of iris recognition, but not for iris localization stage due to the complexity of this stage. Based on the literature review on iris recognition and its embedded systems, we found iris localization a research gap in development of fast and accurate embedded systems for iris recognition.

The accuracy of iris localization is most important because it is very first stage in an iris recognition algorithm and all the stages following this stage will fail if the iris localization goes wrong. Sometimes, the accuracy has to be compromised when designing a dedicated hardware of an algorithm in order to reduce the complexity of the algorithm. Therefore, the algorithm selection is also very important before going to start designing the dedicated hardware for an application.

This point onwards, the rest of thesis will focus on the iris localization stage from perspective of algorithms and its hardware implementation. After identifying iris localization as main thesis objective as described in chapter 1, the further literature review on iris localization algorithms and its hardware implementation is discussed in section 2.4 and section 2.5 respectively.

2.4 Algorithms for Iris Localization

2.4.1 Daugman's IDO

The first successful iris localization method was proposed by Daugman in [Daugman, 1993], which uses IDO shown in Equation (2.2). The IDO was used to localize iris, which considers iris's inner and outer boundaries as two circles. He applied the IDO to the image domain to search for iris's outer boundary first then within the iris's outer boundary to search for pupil. Daugman applied it on the images in which gray difference between iris and sclera is more than pupil and iris. This is true for images captured under VW light. Given a preprocessed image $I(x, y)$, the IDO can be used first to determine the iris's outer boundary. Daugman's IDO is mathematically expressed as below.

$$\max_{(r,x_o,y_o)} \left| G_\sigma(r) * \frac{\partial}{\partial r} \oint_{r,x_o,y_o} \frac{I(x,y)}{2\pi r} ds \right| \quad (2.2)$$

The operator searches over image domain (x,y) for maximum in the blurred partial derivative with respect to increasing radius *r* of the normalized contour integral of $I(x, y)$ along a circular arc ds of radius *r* and center coordinates (x_o,y_o). The symbol * denotes the convolution operation and $G_\sigma(r)$ is a smoothing function such as a Gaussian of scale σ (standard deviation). To normalize the circular integral with respect to its perimeter, it is divided by $2\pi r$. In short, the IDO behaves as circular edge detector blurred at a scale set by σ, which searches iteratively over image space through the parameter set $\{x_o, y_o, r\}$. First search is for iris's outer boundary with higher value of σ. Once the iris's outer boundary is localized, the search process with finer value of σ, for the iris inner boundary is carried out only within the pre-determined region. The computation time associated with an iris's outer boundary search process can be reduced by providing a range of estimates for the parameter *r* that are close to the actual boundary radius.

The additional details on Daugman's iris localization method have been provided in chapter 3: section 3.1.

2.4.2 Wildes' Method

Second famous segmentation method was proposed by Wildes in [Wildes, 1997], which is based on HT technique. He used HT to detect circles in an image. Given a preprocessed image I(x, y), the edges contained in the image are first determined using an edge detector. Consider the set of edge-points obtained by the edge detection algorithm to be (x_i,y_i), where i = 1, 2, . . . , n. Since

these edge-points could represent a non-continuous or non-circular contour, a voting procedure is used to fit a circle to the boundary. For this purpose, HT, a standard contour fitting algorithm, is used. The voting procedure in the HT technique is carried out in a parameter space, from which object candidates (in this case, circular contours) are obtained as local maxima in an accumulator space constructed by the algorithm. In the field of iris recognition, Wildes demonstrated the use of HT to determine the iris boundaries. For a given set of edge-points, (x_i,y_i), i = 1, 2, . . . , n, HT can be used to fit a circle with center (x_c,y_c), and radius *r* as follows:

$$H(x_c, y_c, r) = \sum_{i=1}^{n} h(x_i, y_i, x_c, y_c, r) \tag{2.3}$$

$$h(x_i, y_i, x_c, y_c, r) = \begin{cases} 1 & \text{if } g(x_i, y_i, x_c, y_c, r) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

$$g(x_i, y_i, x_c, y_c, r) = (x_i - x_c)^2 + (y_i - y_c)^2 - r^2 \tag{2.5}$$

For each edge-point contained in the set (x_i, y_i), $g(x_i, y_i, x_c, y_c, r)$ is considered to be 0, if the parameter triplet (x_c, y_c, r) represents a circle through that point. $H(x_c, y_c, r)$ shown in Equation (2.3) is an accumulator array and its values (indexed by discretized values for x_c, y_c and r) are incremented according to the Equation (2.4) and Equation (2.5). The parameter triplet that corresponds to the largest value in the accumulator array is considered to be the most suitable parameter set for the circle that fits the given contour.

The implementation and evaluation of Wildes' iris localization method has been described in chapter 4: section 4.1, which also discusses implementation of HT algorithm for circle detection that is Circular Hough Transform (CHT).

2.4.3 Other Methods

There are several iris segmentation methods available in relevant literature. Some of the state-of-the-art iris segmentation methods and their relevant main techniques are given in Table 2.1. Many of these methods in the table are based on HT (or CHT) and IDO techniques. However, along with HT and IDO, other processing steps are also used, such as reflection removal, edge-map creation of eye image using various techniques, thresholding, morphological operations, image filtering operations, image quality enhancement and active contours to fine-tune iris-boundaries etc. These image processing techniques become dominant when image-data are non-ideal having heterogeneous image characteristics, inconsistent illumination, low contrast, defocused and blurred images etc. The different methods listed in the table have been developed for specific image data, such as NIR or VW images, off angle iris images, colored VW images

and the non-ideal data for non-cooperative iris recognition etc. However, some of the techniques developed for NIR images can also be applied to VW images and vice versa with some modifications. For same type of image data, the newer methods show the improvement in computation time or accuracy performance or both over the previous methods.

In the iris localization of the NIR images, the pupil is generally localized prior to the iris's outer boundary because it is the dark compact region in the image, the pupil boundary is stronger than the iris's outer boundary and the pupil is visible as full circle in almost all the images. Having localized the pupil in the image, a sub-image around the pupil is processed to detect the iris's outer boundary [Jan et al., 2014], [Wang et al., 2014]. As opposed to NIR images, iris's outer boundary is detected prior to pupil in VW images as its boundary has more contrast than pupillary boundary [Chen et al., 2010], [Radman, 2013]. A few methods for noisy VW images, locate sclera region first in the images as it is pronounced white area in the eye images [Proenca, 2010].

The literature review reveals that the existing iris localization algorithms for the NIR images detect the pupil using either intensity thresholding [Khalighi et al., 2015], [Zuo and Schmid, 2010] or edge detection based segmentation techniques [Jan et al., 2012], [Hasan and Amin, 2014], [Marciniak et al., 2014], whereas iris's outer boundary is detected using either edge detection based techniques or IDO based approaches.

Table 2.1. Iris localization/segmentation methods

Method	Main technique(s) used
[Daugman, 1993]	Integro-differential operator (IDO)
[Wildes, 1997]	Edge detection and Hough transform (HT)
[Bole and Boashash, 1998]	Edge and contour detection
[Masek and Kovesi, 2003]	Edge detection and HT
[Ma et al., 2004]	Gray level information, canny edge detection and HT
[Liu et al., 2005]	Canny edge detection and HT
[Daugman, 2007]	Active contour model
[Basit and Javed, 2007]	Intensity gradient
[Schuckers et al., 2007]	IDO and angular deformation model
[He et al., 2009]	Pulling and pushing elastic model
[Shah and Ross, 2009]	Binarization, CHT and Geodesic Active contours
[Chen et al., 2010]	Edge detection and HT
[Tan et al., 2010]	IDO
[Proenca , 2010]	Feature extraction and polynomial fitting
[Sankowski et al., 2010]	Reflections localization and filling, and IDO
[Zuo and Schmid, 2010]	Thresholding, erosion-dilation and ellipse fitting
[Puhan et al., 2011]	Fourier spectral density
[Roy et al., 2011]	Variational level set-based curve evolution method

[Jan et al., 2012]	HT, grey level statistics, adaptive thresholding, geometrical transform, radial gradients and active contours
[Ibrahim et al., 2012]	Local histogram and other image statistics
[Radman, 2013]	Circular Gabor filter and IDO
[Jan et al., 2013]	HT, histogram-bisection and eccentricity
[Mehrotra et al., 2013]	Adaptive thresholding, hole filling and IDO like approach
[Wang et al., 2014]	Image inpainting using Navier-Stokes equations, Probable boundary (Pb) edge detection operator and CHT
[Jan et al., 2014]	Canny edge detection, CHT, histogram bisection, adaptive binarization, eccentricity, gray statistics, radial gradients and Fourier series
[Marciniak et al., 2014]	Canny edge detection and CHT
[Hasan and Amin, 2014]	Canny edge detection and CHT
[Khalighi et al., 2015]	Thresholding, eccentricity, dilation, Canny edge detection and CHT

In IDO based methods, the image preprocessing is very important to estimate the rough centers of pupil and iris so that the IDO can be applied on the selected pixels in the image or a subimage containing iris region can be extracted. These preprocessing steps make the iris localization fast and accurate. For example, [Radman, 2013] author, use circular Gabor filter for coarse detection of pupil and iris centers followed by IDO for fine detection of centers. The reflection removal is a common step in almost all the iris localization methods as IDO is very sensitive to the reflections and the reflections also mislead the circle detection in CHT based methods [Jan et al., 2013].

In the HT based algorithms, first optimal edge-maps of the iris image are generated that contain minimal false edges, so that the iris circles can be detected accurately and efficiently as demonstrated in [Jan et al., 2012] and [Hasan and Amin, 2014]. The generating optimal edge-maps get more challenging if the images are noisy such as CASIA-Iris-Thousand, version 4.0 (CITHV4) database images. The noisy images are first preprocessed to remove the noise such as lighting reflections, non-uniform illumination and low contrast as described in [Jan et al., 2012], [Jan et al., 2013], [Jan et al., 2014], [Wang et al., 2014], which improves the accuracy and time performance of the iris localization. The image inpainting techniques are used for removing the lighting reflection spots of the iris images and the histogram equalization is used for compensating the non-uniform illumination and low contrast. For the iris localization in noisy NIR images from CITHV4 database, [Wang et al., 2014] authors proposed an inpainting technique based on Navier-Stokes equations to remove the lighting reflection spots and Probable boundary (Pb) edge detection operator to counter the non-uniform illumination.

2.5 Hardware Implementation of Iris Localization

The iris localization was implemented on FPGA by [Lopez et al., 2011] authors and they obtained an execution time of 159 ms with a clock of 50 MHz. They used Daugman's algorithms based on IDO and active contour models for the hardware implementation of iris localization. The hardware architecture design of iris localization is not presented in the [Lopez et al., 2011] and only performance results are provided.

In a most recent work, [Ngo et al., 2014] authors have presented hardware implementation of CHT for iris localization, but with constraints that: 1) it localizes outer iris boundary only without detecting the inner iris-boundary and; 2) the iris should be fully visible so that it can be estimated as a full circle. Moreover, this work focuses on hardware architecture design and implementation of CHT for circle detection and does not include the hardware implementation of edge-map generation for CHT. Their CHT hardware takes 5 ms time in detecting circle in the edge-map of eye image, which is much faster as compared to IDO based hardware [Lopez et al., 2011] mentioned above. However, aforementioned constraints are main drawbacks of this work.

The iris localization can be implemented on FPGA by implementing the CHT algorithm [Pedersen, 2007], [Yuen et al., 1990] on FPGA. The CHT is a very computation-demanding and memory-demanding algorithm; and it is applied on the binary edge-map of the image. In the CHT algorithm, the circles are drawn at all edge-points in the edge-map with different radii and an accumulator array is used to store the count values every time a circle passes through it [Pedersen, 2007]. The peak in the accumulator array is used to find the center and radius of the circle. The standard sequential CHT algorithm [Yuen et al., 1990] uses a single 3-dimensional (3D) accumulator array for all radii, but for the parallel implementation on FPGA, the CHT algorithm requires k number of 2D accumulator arrays for k radius values. The size of each 2D accumulator array is same as the image in general (direct) CHT implementation. Therefore, the main hurdle in the FPGA implementation of the CHT algorithm is large memory requirement for realizing the 2D accumulator arrays, which may also exceed the block random access memories (RAMs) available in some low-end FPGA devices. The researchers have used the techniques to reduce the memory requirement, which is discussed below.

The authors in [Ngo et al., 2014] describe a CHT architecture design and its implementation on FPGA to localize outer iris-boundary, which reduces the memory requirement by a factor of n. In this method, the accuracy of the circle detection decreases as n increases beyond n=2. Their CHT architecture provides a large memory reduction of 93% as compared to

direct CHT implementation for n=16, but same time accuracy degrades by 8% and there is restriction that almost full circle should be visible in the edge-map.

The [Elhossini and Moussa, 2012] authors proposed a memory efficient FPGA implementation of HT to detect lines and circles. It reduces the memory requirement by sampling the HT voting space (accumulator array) and selected radii are taken for the circle to be detected. In this architecture, memory reduction is obtained at the expense of the accuracy. [Chen et al., 2012] authors proposed an FPGA architecture and implementation of HT to detect straight lines in which the image was divided into blocks to impart the parallelism. The [Chen et al., 2012] provides very efficient resource utilization by using incrementing property in HT but it was not developed for circle detection. [Zhou et al., 2013] presents FPGA-based implementation of HT for straight-line detection in image, which utilizes several DSP and block RAMs as the main resources.

In other work, [Ngo et al., 2012] authors provide FPGA-based real time iris segmentation, which implements the Canny edge detection and a circle detection algorithm other than the CHT. The algorithm requires estimation of circle location in the image and it is applied on the selected pixels that could be circle center.

2.6 Concluding Remarks

Iris recognition is a reliable biometric technique of identifying, authenticating or verifying persons for security purpose due to randomness of iris patterns and its implementation techniques in software are well explored. Several algorithms are available in the literature to perform the tasks involved in iris recognition process. However, iris recognition application still runs on PC-based platforms and development of small size and low-cost embedded systems for iris recognition is still a current topic of research. The researchers are trying to implement iris recognition process on FPGA based architectures to develop embedded iris recognition system as FPGAs facilitate the development of ASIC-like solutions at a lower cost and at a reduced development time.

The iris localization is the slowest stage of iris recognition process executing on PC based platforms. The fast iris localization algorithms are very essential to meet real-time performance in developing embedded system architectures for iris recognition. The fast iris localization implementation without compromising with the iris localization accuracy is identified as a gap in the present research, which can help in improving the speed and accuracy of iris recognition systems.

Chapter 3
IDO Based Iris Localization for NIR Images

The literature review on iris localization in chapter 2 has revealed that many of the existing iris localization methods and the commonly used methods in commercial iris recognition systems are based on Daugman's integro-differential operator (IDO) [Daugman, 1993]. But, these IDO based methods also require image preprocessing steps, such as the process of identifying potential circle centers (i.e. estimation of circle centers) in the image before applying the IDO to the image. These image preprocessing steps sometimes play a lead role in some of the iris localization methods especially for noisy data (images) and can be more complex than the IDO. The IDO performs worse (less accurate) in presence of lighting reflections in the images and other noise such as occlusion by eyelids and eyelashes, because it detects circle based on intensity gradient, which is maximum in case of reflections. In addition to this, if IDO is applied to each pixel in the image, it would take much longer time for circle detection. The image preprocessing is required before applying the IDO for better accuracy and time performance. The total time for iris localization is sum of time taken by preprocessing and IDO steps.

This chapter proposes an image preprocessing technique for the IDO based iris localization in near infrared (NIR) images that makes the iris localization more accurate and fast. The proposed method uses modified IDO for limbic boundary detection, which acts as circular arc detector.

3.1 Daugman's Method

The Daugman's iris localization method [Daugman, 1993] uses the IDO that is described in chapter 2 under subsection 2.4.1. The method involves applying the IDO on the Gaussian smoothed image to detect both the pupillary and limbic boundaries as shown in Figure 3.1(a). However, the [Daugman, 1993] does not provide enough details that are required to implement this method. For example, it says that the IDO searches the whole image for circular edges with a given range of radii, but if the IDO is applied on each pixel in the image, it takes too long in

detecting a circle. In another paper, [Daugman, 2004] says that the IDO is first applied for coarse search of circle center and then it is again applied for fine search (single-pixel precision) of the circle center as shown in Figure 3.1(b). However, the scheme of coarse or fine search was not provided in this paper. Both [Daugman, 1993] and [Daugman, 2004] state that once the IDO completes the process of limbic boundary detection, the IDO is again used for detecting the pupillary boundary, but [Daugman, 1993] applies this scheme to the images in which the limbic boundary is more pronounced than the pupillary boundary. However, for NIR images, the pupillary boundary should be detected first because it is more pronounced than the limbic boundary. Taking inputs from [Daugman, 1993] and [Daugman, 2004], we have proposed an iris localization method for NIR images that is based on the IDO as shown in Figure 3.1(c).

The proposed method is described in next section, which performs image preprocessing on the image in addition to applying the IDO. The image preprocessing does two tasks (a) reflection removal; (b) estimation of the potential circle centers (candidate pixels) in the image on which the IDO is applied instead of applying the IDO on whole image. Moreover, the IDO was also modified little while detecting the limbic boundary such that it searches for a set of two vertical arcs instead of a full circle. The purpose of the proposed method is to localize irises in the images that are captured under either constrained or less constrained environments. The less constrained images may have noise such as lighting reflection spots, iris obstruction by eyelids and eyelashes, low contrast and eyeglasses etc.

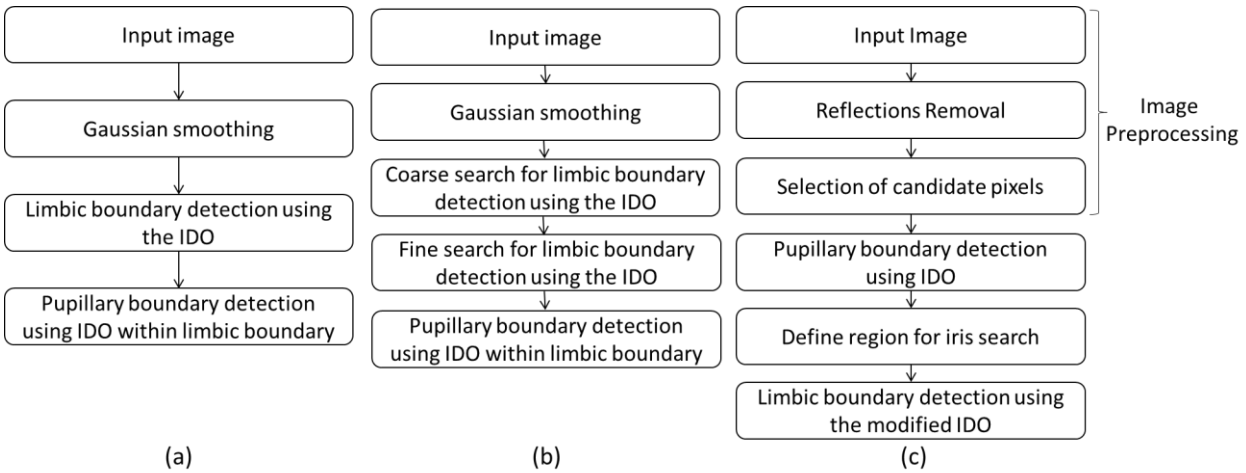


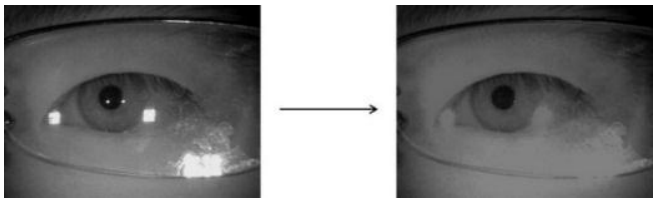
Figure 3.1. Iris localization based on IDO: **(a)** Original method [Daugman, 1993]; **(b)** Modified method [Daugman, 2004]; **(c)** Proposed method.

3.2 The Proposed Method

The pupillary boundary is detected prior to the limbic boundary because it is stronger than the limbic boundary in NIR images. The proposed method considers both the pupillary and limbic boundaries as perfect circles, which is true for the frontal view iris images. The iris localization is achieved in three phases. In first phase, the iris image is preprocessed to counter some of the non-ideal issues of it and make it suitable for the subsequent steps. In second and third phases, the pupillary and limbic boundaries are detected respectively. The steps involved in the proposed iris localization method are shown in Figure 3.1(c). The proposed method is described in detail below.

3.2.1 Reflections Removal

The way the IDO works, it cannot be tolerant to reflection spots and uneven high intensity pixels present in the iris image. The reflection spots in iris images cover portions of the image that causes hindrance in the iris detection process. The original pixel intensity values are replaced by the much higher intensity values in the parts of the image, affected by light reflections as shown in Figure 3.2(a). Therefore, the original information of these parts of the image has been lost. As apparent from Figure 3.2(a), there is much difference in intensity values between reflection spots and surrounding dark pixels. In order to avoid light reflection and uneven high intensity pixel values affecting the iris detection, a morphological operator is used which fills in the high intensity regions with the average of intensities of pixels from the region surrounding them. The MATLAB function ‘imfill’ with hole filling option is used as the morphological operator, which is applied on the complemented iris image. Taking the complement again, returns the original iris image in which the reflection spots and high intensity pixels are removed as shown in Figure 3.2(b). The image shown in Figure 3.2(b) is called preprocessed iris image, which is used as input image in detection of both the iris boundaries.



(a) (b)

Figure 3.2. Reflections removal: **(a)** A noisy image from CITHV4; **(b)** The preprocessed iris image obtained from (a).

3.2.2 Pupillary Boundary Detection

To localize pupil using the IDO, the image pixels on which the IDO is applied are identified first otherwise it may take much large time in localizing the pupil. These identified pixels are the potential centers of pupil circle (pupil-centers). The inputs given to the IDO for pupil localization are a range of pupil radii ($rmin_p$ to $rmax_p$) and a set of image pixels that are the potential pupil-centers. The set of image pixels on which the IDO is applied are obtained using the steps below, which reduces the false candidate pixels those cannot be potential pupil-centers. Steps for pupillary boundary detection are shown in Figure 3.3.

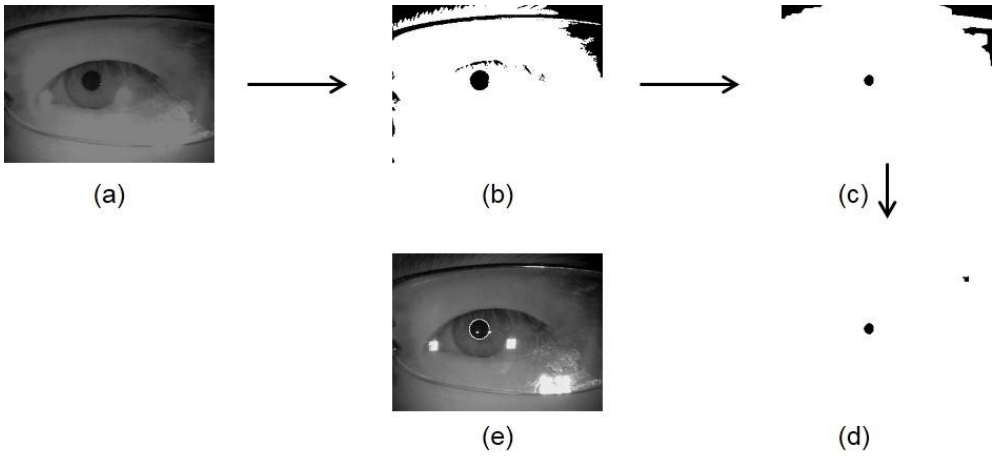


Figure 3.3. Pupillary boundary detection: **(a)** Preprocessed iris image; **(b)** Binarized-image obtained after thresholding; **(c)** Binary image after image cleanup step; **(d)** After removing pixels close to image border in (c); **(e)** Pupil localized image by the IDO.

A) Image Binarization

The pupil is segmented from the iris image using thresholding operation [Gonzalez et al., 2009], as the pupil is relatively much darker region as compared to the iris, sclera and the skin. The thresholding operation on an intensity image f(x,y) with global threshold value *T* results in binary image g(x,y) according to Equation (3.1).

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The value of *T* is determined by finding minimum pixel value (M) in the smoothed image and adding 40 to *M*. The pupil is considered as darkest region in the NIR image and the number,

40 indicates the variation of pixel values inside the pupil. The binary iris image, shown in Figure 3.3(b), is obtained by applying thresholding on the preprocessed iris image of Figure 3.3(a). The black regions in the binary iris image (see Figure 3.3(b)) correspond to pupil, eyelashes, eyelids, eyeglass frame and low illumination near the image border. The regions other than pupil in this image are noise and are false candidate pixels for pupil-center. Therefore, they are removed using morphological operation below.

B) Binary Image Cleanup

The noise in the binarized-image is removed using image erosion for black objects, which is a morphological operation [Gonzalez et al., 2009]. The erosion operation uses a structuring element of type disk because pupil is circular region. The disk radius value must be smaller than the minimum pupil radius in the selected iris image database. The disk radius is taken as 15 for CASIA-iris-thousand, version 4.0 (CITHV4) image database. A larger disk radius value may remove pupil completely and a smaller value may not remove false candidate pixels significantly. The image after erosion operation is shown in Figure 3.3(c), which removes the noise to a much extent and reduces the pupil size also. This step not only reduces the false candidate pixels for pupil-center due to the noise, but also removes pixels in the pupil, which cannot be the potential pupil-center pixels. The false candidate pixels are further reduced using step below.

C) Removing Pixels Close To Image Border

It is certain in every iris image that pupil cannot be touching the image border as pupil is surrounded by iris region. Therefore, some pixels closer to the image border can be discarded, as they cannot be potential pupil-centers. The range (distance) of discarded pixels from image border is taken as $\{k \times rmin_p\}$, where the k is a positive scalar greater than one whose value is estimated by visualizing and inspecting the iris database images. The value of k chosen for CITHV4 is 3.5 and $rmin_p$ is equal to 20. The value of k depends on the database chosen. This step is for a particular database (CASIA) only. Although removing this step will not affect the accuracy, only computation time will increase. The remaining pixels after removing the pixels close to image border are shown in Figure 3.3(d). The black pixels in Figure 3.3(d) are identified as the potential pupil-centers.

The reduced candidate pixels because of above discussed steps have been identified as the potential pupil-centers. Now, the IDO described in chapter 2 under subsection 2.4.1, is applied on

these pixels' coordinates in the preprocessed iris image, which gives the center and radius of pupil as output.

3.2.3 Limbic Boundary Detection

Limbic boundary detection may be hurdled by eyelids, eyelashes, reflections and low contrast between the iris and sclera. The reflections and uneven high intensity values have already been removed after the image preprocessing step. The eyelids and eyelashes occlusion as shown in Figure 3.4 is handled by modified IDO in Equation (3.2) that acts as the circular arc detector and it has been derived by taking motivation from the Daugman's IDO discussed in the previous chapter. The upper and/or lower eyelids occlude the iris in the noisy iris images, but the vertical iris contours are always visible, which are used for detecting the limbic boundary by taking partial integration (Equation (3.2)).

The modified IDO for limbic boundary detection is mathematically expressed as:

$$\max_{(r,x_o,y_o)} \left| G_\sigma(r) * \left[\frac{\partial}{\partial r} \left(\int_{-\pi/4}^{\pi/6} \frac{I(x,y)}{5\pi r/12} ds + \int_{5\pi/6}^{5\pi/4} \frac{I(x,y)}{5\pi r/12} ds \right) \right] \right| \quad (3.2)$$

The parameter *r* denotes the radius of the circular arc *ds* centered at (x_o,y_o). To normalize the intensity integral of arc-pixels with respect to arc length, it is divided by $5\pi r/12$. The modified IDO searches iteratively over a portion of the image domain for the maximum difference of sum of contour pixels intensities between two adjacent circular arcs one pixel radius apart and defined by {–π/4: π/6 and 5π/6: 5π/4} rad (see Figure 3.4). These ranges of angles were chosen after experiments on a number of images through MATLAB simulation and if we further reduce or increase these ranges, the accuracy suffers. The upper angle (above x-axis) is 30°, whereas lower angle is 45°, as upper eyelid occlusion is more than lower eyelid. If these angles are further reduced, computation time will reduce due to reduced partial integration, but accuracy also decreases, because the available arc information does not provide single-pixel precision in circle center detection in certain images.

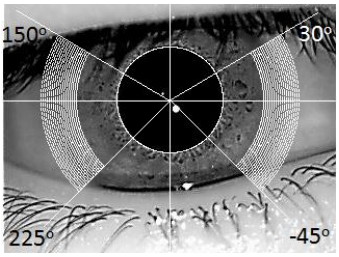


Figure 3.4. Selection of iris search region in the image.

A) Define Iris Search Region

The iris search area is defined by the pupil center, radii range of the limbic boundary and a small area around the pupil center that contains the potential candidate pixels for limbic boundary center. The white area on both sides of pupil in Figure 3.4 represents search region for the modified IDO with (x_o,y_o) equal to pupil center. As the pupillary and limbic boundary circles may not be concentric [Daugman, 2004], the modified IDO is applied on a rectangle of size 10 × 10 centered at the pupil center to find precise center and radius of limbic boundary circle. Iris localized image is shown in Figure 3.5(b).

(a) (b)

Figure 3.5. Limbic boundary detection: **(a)** Input is preprocessed iris image with pupil center known; **(b)** Iris localized image obtained using modified IDO.

3.2.4 Performance Evaluation

To evaluate performance of the proposed method, we have used MATLAB version 8.3 installed on a PC with Windows 7 Professional, Intel® Core™ i5 CPU @ 2.40 GHz, 8.00 GB RAM and a set of images from CITHV4 and Multimedia University, version 1.0 (MMUV1) databases.

A) Input Dataset

The short description of iris database taken for testing the proposed method is given below.

1. CITHV4 contains 20000 iris images collected from 1000 subjects. The main sources of intra-class variations in the database are eyeglasses and specular reflections. CITHV4 images are 8-bit gray-level JPEG files with resolution 640×480 pixel and collected under NIR illumination. Collectively, the images in the database contain non-ideal issues such as occlusions by eyelids, eyelashes, eyebrow hair, eyeglasses, low illumination, low contrast and reflections.

2. MMUV1 iris database contributes a total number of 450 images, which were taken using LG IrisAccess®2200. This camera is semi-automated and it operates at the range of 7-25 cm. The images are 24-bit gray-level BMP files with resolution of 320×240 pixels and all the images were collected under near infrared illumination.

The first 1000 images from CITHV4 were taken for testing the proposed method, whereas all the images of MMUV1 database were included in the experiments.

B) Results and Discussion

Figure 3.6 shows sample images from the CITHV4 database, whereas Figure 3.7 shows images from MMUV1 database with accurately localized irises by the proposed method.

The percentage accuracy of a localization method is defined as [Jan et al., 2013]:

$$\text{Accuracy } (\%) = \frac{N_I}{N_T} \times 100 \tag{3.3}$$

Where N_I is number of irises accurately localized and N_T is total number of iris images taken for testing. The accuracy of the proposed method was calculated using Equation (3.3) and the accuracy results are shown in Table 3.1. To find the average time cost per image, first 1000 images from the CITHV4 were taken. MATLAB timer functions 'tic' and 'toc' were used to find execution time of a code which runs to localize 1000 images and dividing the execution time by 1000 gives average time cost per image. The average time cost per image for MMUV1 database was also calculated in the same manner. The simulation results of the proposed localization method are shown in Table 3.1.

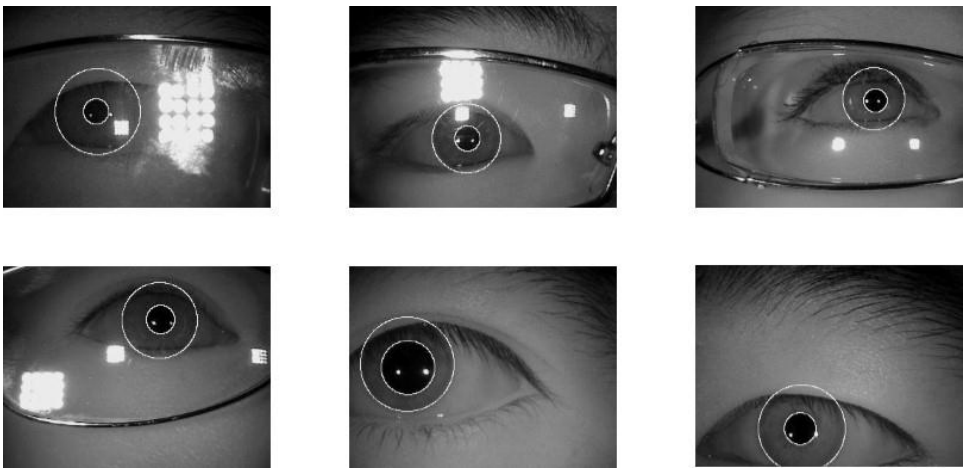


Figure 3.6. Examples of accurately localized irises in CITHV4 images using the proposed method.

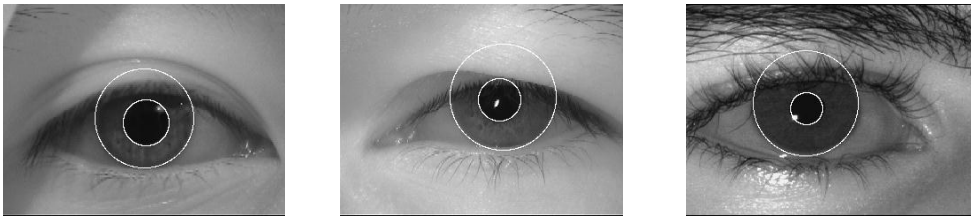


Figure 3.7. Examples of accurately localized irises in MMUV1 images using the proposed method.

Table 3.1. Simulation results

Iris Database	Performance of proposed method	
	Accuracy (%)	Average time cost per image (sec)
CITHV4 (640×480)	99.30	1.86
MMUV1 (320×240)	99.11	0.67

3.2.5 Comparison with Other Methods

To compare the proposed method with published iris localization methods in the literature, we chose [Jan et al., 2012] and [Jan et al., 2013] work because their iris localization methods were tested with the CITHV4 dataset and used similar platform, which we have used for testing the proposed method. Moreover, these methods are highly accurate methods in the literature as described in [Jan et al., 2012]. Table 3.2 shows that the accuracy of proposed method is similar to Jan et al. methods, but time cost performance is much better than Jan et al. methods. The major reason of reduced time cost of the proposed method is reduced number of pixels on which the IDO is applied during pupillary boundary detection. The use of modified IDO for limbic boundary detection further reduces time cost of iris localization, as it does not require drawing full circles.

Table 3.2. Comparison of the results of proposed method with the published results for CITHV4 database images

Method	CITHV4	
	Accuracy (%)	Time cost per image (sec)
[Jan et al., 2012]	99.5	4.93
[Jan et al., 2013]	99.23	3.4
Proposed method	**99.30**	**1.86**

Table 3.3 shows that the accuracy of the proposed method for MMUV1 is 99.11%, which is better than some state-of-the-art methods listed in the table. The accuracy results of the other methods in the table are taken from [Ibrahim et al., 2012], but this paper does not provide the computation time results. The reduction in the number of false candidate pixels that cannot be potential centers of pupil circle makes the proposed method more accurate. The use of modified IDO to localize limbic boundary is another reason for improvement in the accuracy performance of the proposed method.

Table 3.3. Comparison of accuracy of proposed method with published results (accuracy results in the table are taken from [Ibrahim et al., 2012])

Method	Accuracy (%) for MMUV1
[Daugman, 2004]	85.64
[Ma et al., 2004]	91.02
[Daugman, 2007]	98.23
Proposed method	**99.11**

3.3 Concluding Remarks

In this chapter, we have presented a method to localize irises in NIR iris images. The proposed method starts with image preprocessing followed by iris's inner and outer boundary detection using IDO and modified IDO respectively. A novel technique for pupil localization is proposed that reduces the number of false candidate pixels, which cannot be potential centers of pupil circle and makes the IDO based circle detection faster and accurate. The method also uses a modified IDO to detect iris's outer boundary, which increases the speed. The simulation results show that the proposed method is highly accurate and is tolerant to non-ideal issues in iris images such as reflections, low contrast, low illumination and occlusions by eyelids, eyelashes and eyebrow hair. The comparison of the proposed method with published work shows that it outperforms some of the iris localization methods in both accuracy and computation time.

Chapter 4
CHT Based Iris Localization for NIR Images

The circular Hough transform (CHT) based iris localization methods are most famous methods like integro-differential operator (IDO) based methods, which are also used in commercial iris recognition systems. The first CHT based method was proposed by Wildes in [Wildes, 1997], which is based on the edge detection and CHT. The CHT based methods perform iris localization mainly in two steps: (a) Edge-map generation; (b) Circle detection in the edge-map using CHT. These two steps are used for each of the pupillary and limbic boundary detection. However, the edge-map generation is not just an edge detection step; it also requires image preprocessing steps especially for the iris localization of less constrained images. For example, the image preprocessing, such as image inpainting techniques are used for removing the lighting reflection spots of the iris images and the histogram equalization is used for compensating the non-uniform illumination and low contrast [Wang et al., 2014]. The literature review in chapter 2 reveals that the edge-map generation techniques are the primary components in most of the CHT based iris localization methods for less constrained images, as they directly affect the overall accuracy and time performance of iris localization.

This chapter proposes the novel edge-map generation techniques for the CHT based iris localization in less constrained near infrared (NIR) images that makes the iris localization more accurate and fast. The chapter also proposes an adaptive CHT for limbic boundary detection, which is faster than CHT.

4.1 Wildes' Approach

Wildes' iris localization approach [Wildes, 1997] detects each of the pupillary and limbic boundaries using a two-step process: (a) Binary edge-map of the input image is generated; (b) CHT is used to detect a circle in the edge-map. The separate edge-maps are used for detection of the pupillary and limbic boundaries. However, the same CHT algorithm is applied twice, but with different range of radii to detect both the pupillary and limbic boundaries as shown in Figure

4.1(a). The Wildes' and the proposed approaches are shown in Figure 4.1. The proposed approach uses some additional operations than Wildes' approach, which are shown in bold in Figure 4.1(b). The proposed approach is described later in section 4.2, whereas Wildes' approach is discussed in current section.

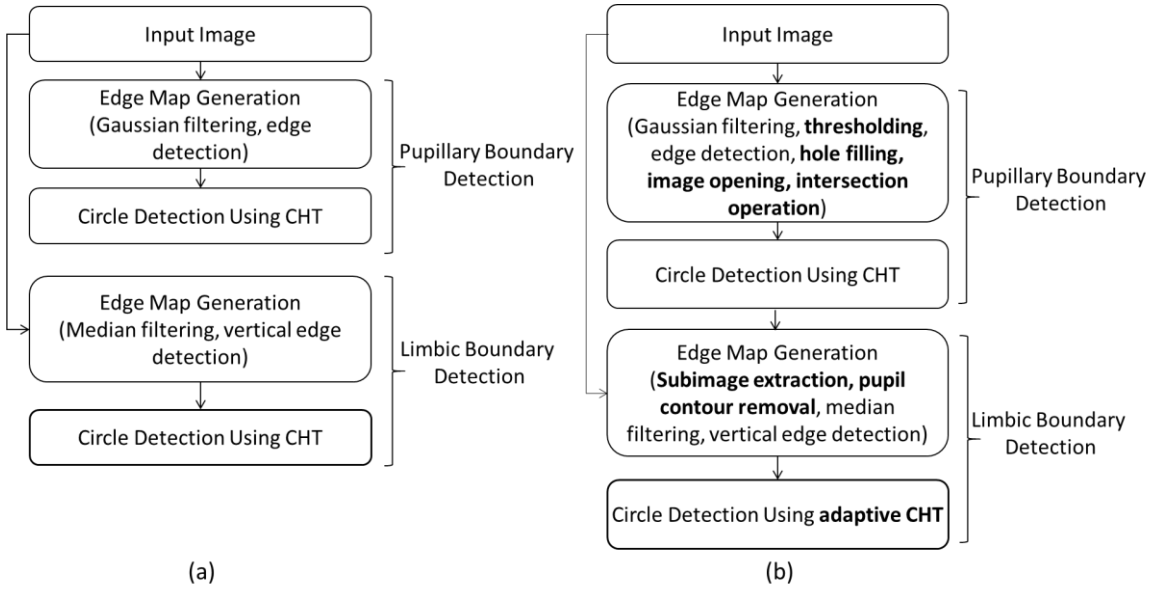


Figure 4.1. Iris localization: **(a)** Wildes' approach; **(b)** Proposed approach.

The detailed implementation of Wildes' approach (Figure 4.1(a)) is described next in two phases: (a) Pupillary boundary detection; (b) Limbic boundary detection.

4.1.1 Pupillary Boundary Detection

The two steps involved in the pupillary boundary detection are the edge-map generation and the circle detection using CHT. The edge-map is generated by applying Gaussian smoothing and Sobel edge detection on the input image.

The iris image is smoothed using the Gaussian filter [Gonzalez et al., 2009] of size 5×5 and sigma (σ) equal to 1.0. The larger filter size makes the image more blurred and reduces the pupil boundary contrast. The sigma (σ) equal to 1.0 makes the computer implementation of Gaussian filter simple. The smoothing of the iris image removes the random noise and the uneven intensities that may result in unnecessary false edges in the edge-map of iris image. The smoothed iris image is shown in Figure 4.2(b).

The edge-map is generated by applying Sobel edge detection with thinning operation on the smoothed image as shown in Figure 4.2(c). A suitable threshold value is chosen in the edge detection so that the pupil edges, which are among the strong edges in the image, are detected

and the faint edges are not. A lower threshold value may detect the iris's outer boundary edges, which we do not want to be detected at this stage. The initial value of threshold was determined by 'edge ( )' function of MATLAB, which was then adjusted by simulations of database images and visualization of edge-detected images in MATLAB. The higher threshold gives fewer edge pixels and lower threshold gives more edge pixels. The final value of threshold is chosen after a number of iterations of simulation and manual inspection of edge-detected images, and then it remains constant for a given database. Figure 4.2(c) shows the edge-detected image after the Sobel operator is applied on the smoothed iris image (Figure 4.2(b)) in both the horizontal and vertical directions with threshold value equal to 0.03 in 'edge ( )' function. The edge-detected image has the pupil edges and other (false) edges due to the lighting reflection spots, eyelids, eyelashes and eyeglass frames as shown in Figure 4.2(c).

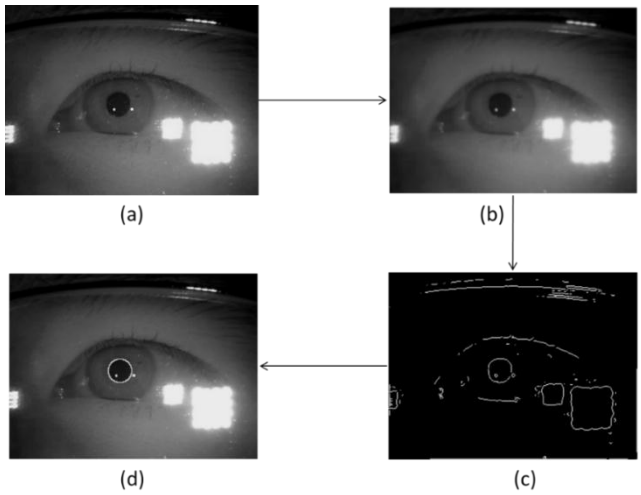


Figure 4.2. Pupillary boundary detection: **(a)** Input image; **(b)** Gaussian smoothed image; **(c)** Edge-map for pupillary boundary detection; **(d)** Pupil localization using CHT.

The CHT algorithm is applied on the edge-map in Figure 4.2(c) to detect the pupillary boundary circle. The CHT implementation is discussed next.

CHT Implementation

There are a number of different approaches that can be taken in the CHT implementation [Pedersen, 2007], [Yuen et al., 1990]. To meet the requirement of detecting a circle in the edge-map of iris image, we propose an implementation technique for CHT that detects a single strongest circle in an image. The proposed CHT implementation described in Algorithm 1 uses a 2D accumulator to store votes for one radius at a time, whereas the standard CHT requires a 3D accumulator to store votes for multiple radii that results in large storage requirements [Yuen et

al., 1990]. The 2D array is used to find maximum count value for a given radius, its maximum count is stored and for next radius, the same 2D array is used again to find the maximum count value, its maximum count is stored and so on. Since the same 2D array is used repetitively for all the radii, the space complexity of the algorithm reduces. At all the edge pixels (a,b), which are the white pixels in the edge-map, the virtual circles are drawn with different radii using Equation (4.1). A circle with radius *r* and center (a,b) can be described with parametric equations below.

$$\left.\begin{aligned} x &= a - r * \sin\theta \\ y &= b + r * \cos\theta \end{aligned}\right\} \tag{4.1}$$

When angle θ sweeps by full 360 degrees, the circle-points (x,y) lying on the perimeter of the circle are generated. A 2D accumulator array of size same as the image is initialized to zero. The cells' values in the array are incremented by one every time a circle passes through the cells; the process is known as accumulator voting as shown in Algorithm 1. The peak (maximum value) in the 2D accumulator array is determined for every radius. The maximum among all the peaks gives center and radius of the detected circle.

Algorithm 1. Proposed CHT implementation for pupillary boundary detection using 2D accumulator

Inputs: Edge map of iris image, minimum pupil radius (r_{minp}) and maximum pupil radius (r_{maxp})
Outputs: pupil circle radius (r_p) and center coordinates (x_p,y_p)

```
1.  for pupil_radius=rminp :1: rmaxp   do     // comments:
2.     A=zeros(rows,cols) ;                    // 2D accumulator of
                                                 iris image size
3.     for all "white pixels" in edge map of iris image   do
4.        for  θ =1 to 360°     do
5.          Calculate (x,y) using Equation (4.1)
6.          if (x,y) is in image bounds   do
7.            A(x,y) = A(x,y)+1 ;             // Accumulator-voting
                                                 step
8.          end if
9.        end for
10.     end for
11.     Find maximum value in A:
12.       M=A(x',y') ;                        //M is maximum value
                                                 in A
13.      Max_Array(pupil_radius)=M ;
14.      X_Array(pupil_radius)=x' ;
15.      Y_Array(pupil_radius)=y' ;
16.  end for
17. Find maximum in Max_Array:
18. M'=Max_Array(index)                      //M' is maximum value
                                                 in Max_Array
19.  rp = index;   xp = X_Array(index);
     yp = Y_Array(index)  ;                  // End of CHT algorithm
```

4.1.2 Limbic Boundary Detection

The limbic boundary detection may be hurdled by the eyelids, eyelashes, reflections and low contrast between the iris and sclera in the iris images. The input image is filtered using a median filter [Gonzalez et al., 2009] to suppress the noise, such as the eyelash hair and uneven pixel intensities, without damaging the edge structure as shown in Figure 4.3(b). The upper and/or lower eyelids occlude the iris mostly in the iris images, but the vertical iris contours are always visible. Therefore, to get the vertical edge pixels, the Sobel edge detection is applied in horizontal (x) direction only (Figure 4.3). A low threshold value of 0.0075 was set in the 'edge ( )' function, as the limbic boundary is of low-contrast. An edge thinning operation follows the edge detection.

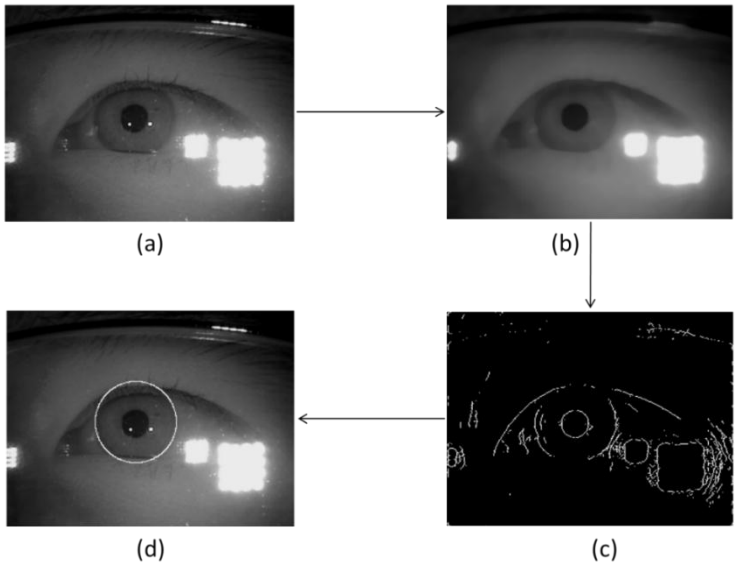


Figure 4.3. Limbic boundary detection: **(a)** Input image; **(b)** 9×9 median filtered image; **(c)** Edge-map for limbic boundary detection; **(d)** Limbic boundary localization using CHT.

Now, CHT is applied on the edge-map in Figure 4.3(c) to detect the limbic boundary, but [Wildes, 1997] suggests that the voting space of CHT is constrained to lie within the pupillary boundary. In this implementation, the CHT voting space in the accumulator is constrained to a 10×10 rectangle centered at the pupil center because the centers of the pupil and limbic boundary circles lie within a small window [Daugman, 2004]. The CHT that was used to detect the pupillary boundary is used to detect the limbic boundary also, but with different range of radii and different CHT voting space.

4.1.3 Performance Evaluation

The experiments on the datasets were done using a computer with Intel i5 CPU @ 2.40 GHz, 8 GB RAM and Windows 7 operating system. The proposed algorithm is implemented and tested with MATLAB (version 8.4) tool. Three different datasets were used in the experiments for performance evaluation. Collectively, the set of these databases contains noises, such as lighting reflections, non-uniform illuminations, eyeglasses, low contrast, defocus and the occlusions by eyelids, eyelashes and eyebrow hair etc. The following datasets were used:

1. CASIA-iris-thousand, version 4.0 (CITHV4): First 1000 images of this database were taken.
2. Multimedia University, version 1.0 (MMUV1): All 450 images of this database were taken.
3. CASIA-iris-lamp, version 3.0 (CILV3): This database contains images from 411 different subjects. The total number of the images in the database is 16212. For thorough experimentation with the database, 811 images were chosen selecting first left and first right eye image of each subject except 11 subjects. CILV3 contains 8-bit gray-level JPEG images with resolution of 640×480 pixels.

The accuracy of Wildes' approach was calculated using Equation (3.3) described in chapter 3, and the time cost per image was calculated using the MATLAB's timer functions 'tic' and 'toc', which was also discussed in chapter 3 under subsection 3.2.4.

Table 4.1. Performance of Wildes' Approach (Simulation results)

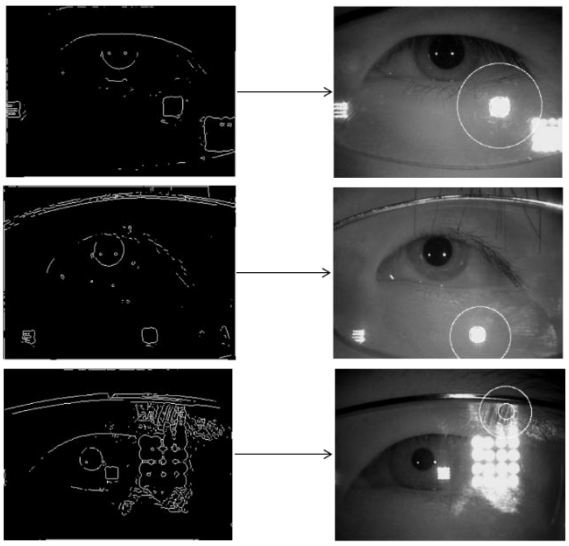
Method	Accuracy (%) & Average time cost per image (sec)		
	CITHV4 (640×480)	CILV3 (640×480)	MMUV1 (320×240)
[Jan et al., 2012]	99.5 & 6.4	98 & 4.93	--
[Jan et al., 2013]	99.23 & 3.4	99.21 & 3.35	--
Wildes' approach	**95.10 & 2.35**	**85.69 & 3.79**	**98.44 & 2.63**

*640×480 image is resized to 320×240 image before applying the iris localization method and resize time is included in the time cost of iris localization.

A) Accuracy

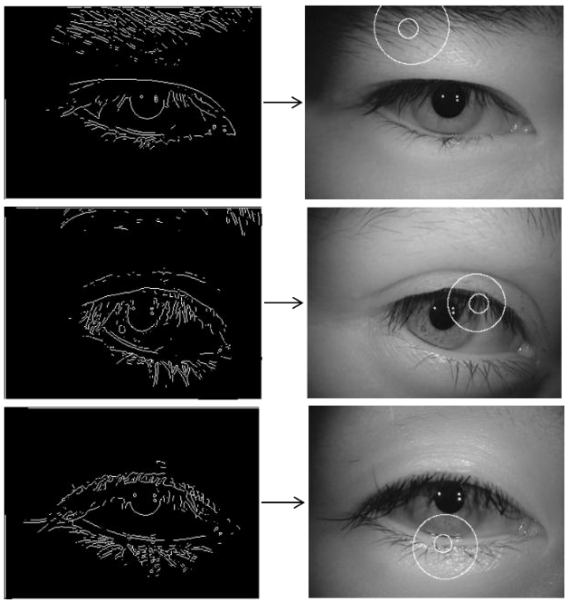
The accuracy results of Wildes' approach (Figure 4.1) are listed in Table 4.1. The MMUV1 has the highest accuracy among the three datasets because it contains less noisy images. The CILV3 dataset has the lowest accuracy (Table 4.1) because the occlusion by eyelids and eyelashes noise

is more prominent in these images, which hinders the iris boundaries detection. The lighting reflections cause incorrect iris localization in a few of the CITHV4 images. A few causes of the wrong iris localization in CITHV4 and CILV3 datasets are illustrated with help of the images in Figure 4.4, Figure 4.5 and Figure 4.6. These figures include only one of the two edge-maps, which is responsible for wrong localization.



(a) (b)

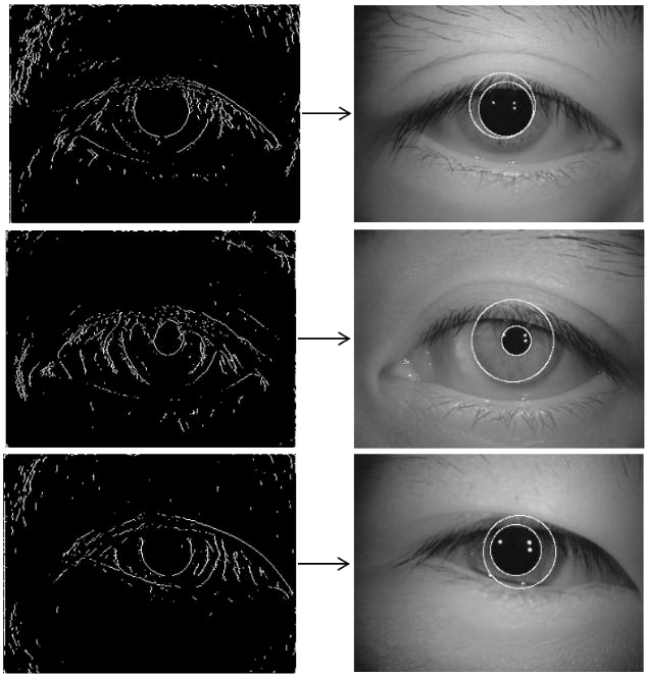
Figure 4.4. Examples of wrong iris localization in CITHV4 images due to lighting reflections: **(a)** Edge-map for pupillary boundary detection; **(b)** Wrong iris localization.



(a) (b)

Figure 4.5. Examples of wrong iris localization in CILV3 images due to eyelids, eyelashes and eyebrow hair: **(a)** Edge-map for pupillary boundary detection; **(b)** Wrong iris localization.

The Figure 4.4 and Figure 4.5 show that when the pupil is located wrongly in the image, the limbic boundary detection also fails, because the center of the pupil circle is used as an input in detecting the limbic boundary.



(a) (b)

Figure 4.6. Wrong limbic boundary detection: **(a)** Edge-map for limbic boundary detection; **(b)** Wrong limbic boundary detection, but correct pupillary boundary detection.

B) Computation Time

It was observed that the computation time (or time cost) of iris localization in an image is directly associated with the number of edge-points in the edge-maps. The edge-map should contain false edges as less as possible. Secondly, the range of radii specified to CHT algorithm should be chosen judiciously for a database; a wider range makes the circle detection slow. The edge-points close to the image border can be discarded because the pupil can not be touching the image border as it is surrounded by the iris. All the above mentioned parameters were taken into account while choosing the threshold values in the edge detection processes and applying the CHT for pupillary and limbic boundary detection. The time cost per image obtained for different datasets is listed in Table 4.1. The iris localization in CILV3 images is slower than CITHV4 or MMU1 because the edge-maps of CILV3 images contain more edge-points.

The iris localization results obtained with the Wildes' approach are moderate and there is a scope of improvement as today's algorithms provide iris localization accuracy as high as 99%

[Jan et al., 2014]. The results of Wildes' approach are compared with other methods in Table 4.1 to show its effectiveness. To improve these results, we need to improve the edge-maps, meaning that the edge-maps should contain as less as possible false edges. This will increase not only the accuracy of iris localization, but will improve the time performance also, because the CHT detects the circles more accurately and rapidly in the images whose edge-maps are optimized.

The proposed method described in next section discusses a novel edge-map generation technique for pupillary boundary detection, which reduces the false edges abruptly. The proposed method also introduces adaptive CHT for limbic boundary detection that makes the limbic boundary detection faster and more accurate. This method is also much faster than Wildes' approach as it processes a subimage surrounding the pupil for limbic boundary detection.

4.2 The Proposed Method

This method achieves iris localization for the NIR images in two phases: Phase 1) Pupillary boundary detection, and Phase 2) Limbic boundary detection. Each phase consists of two process steps, which are edge-map generation from iris image and circle detection in the edge-map as depicted earlier in Figure 4.1. This figure demonstrates how the proposed method differs the Wildes' approach. The goal of the edge-map generation is to prepare appropriate input for CHT so that the iris circles can be detected accurately and rapidly. The original iris image of size 640×480 pixels is scaled down to 320×240 pixels using a scaling factor of 0.5 to speed up the processing. The proposed algorithm is applied on the scaled iris image and the obtained circle's parameters are multiplied by two for mapping the parameters in the original iris image.

4.2.1 Pupillary Boundary Detection

For accurate iris localization in the NIR images, a high pupil localization accuracy is required because if the pupil were wrongly detected in the image, the iris's outer boundary would also be, as it requires pupil circle parameters as input for detection [Jan et al., 2014], [Wang et al., 2014]. Here, we propose a novel edge-map creation technique for the less constrained NIR images that reduces the false edges drastically so that the pupil can be localized accurately and rapidly using a general CHT algorithm. The two steps involved in the pupillary boundary detection are the edge-map generation and the CHT for pupillary boundary detection, which are discussed below.

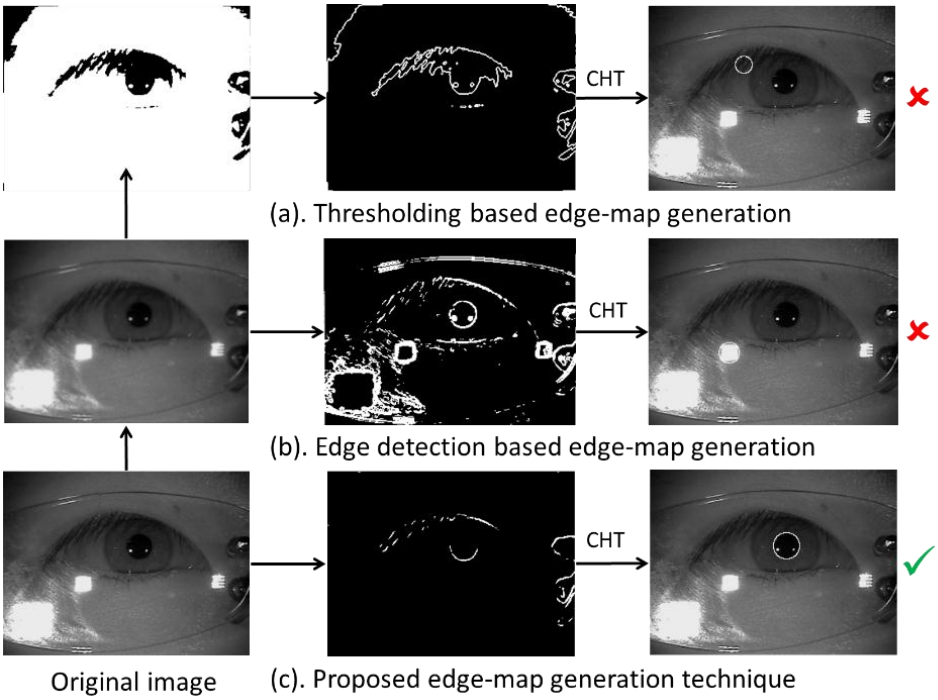


Figure 4.7. Edge-map generation techniques for pupillary boundary detection: **(a)** Thresholding based technique resulting in wrong pupil localization; **(b)** Edge detection based technique resulting in wrong pupil localization; **(c)** Proposed technique that combines the edge-maps of (a) and (b) using logical ANDing, which results in correct pupil localization.

A) Edge-Map Generation

The two standard edge-map generation techniques for pupillary boundary detection are: (a) Thresholding based; and (b) Edge detection based. In the proposed technique, the edge-map for pupillary boundary detection is generated by combining the two edge-maps obtained using (a) and (b), which improves the accuracy and speed of pupillary boundary detection as illustrated in Figure 4.7. This improved performance is obtained due to reduction of false edges in the edge-map that CHT uses for circle detection. The proposed edge-map generation technique is explained in detail using Figure 4.8 and Figure 4.9.

In the proposed technique (Figure 4.8), the idea of generating an optimal edge-map relies on combining two edge-maps obtained via two paths: Path 1 is applying intensity thresholding on the iris image to segment the pupil region followed by the edge detection; and Path 2 is applying the edge detection on the intensity iris image. Since both the edge-maps obtained via Path 1 and Path 2 have pupil contour in common, they are combined in a single edge-map using the intersection operation (logical AND), which minimizes the false edges due to noise such as eyelids, eyelashes and lighting reflections etc. significantly. The edge-map in Figure 4.8(e)

obtained via Path 1 excludes the effect of reflections, but contains the edges due to dark illumination, whereas the edge-map in Figure 4.8(f) obtained using Path 2 excludes the edges due to dark illumination, but contains the edges due to reflections. Therefore, the intersection operation on the two edge-maps (Figure 4.8(e) and Figure 4.8(f)) removes the effect of both reflections and dark illumination as shown in Figure 4.8(g). To get more advantage out of the intersection operation in reducing the false edges, the two morphological operations are also used in Path 1.

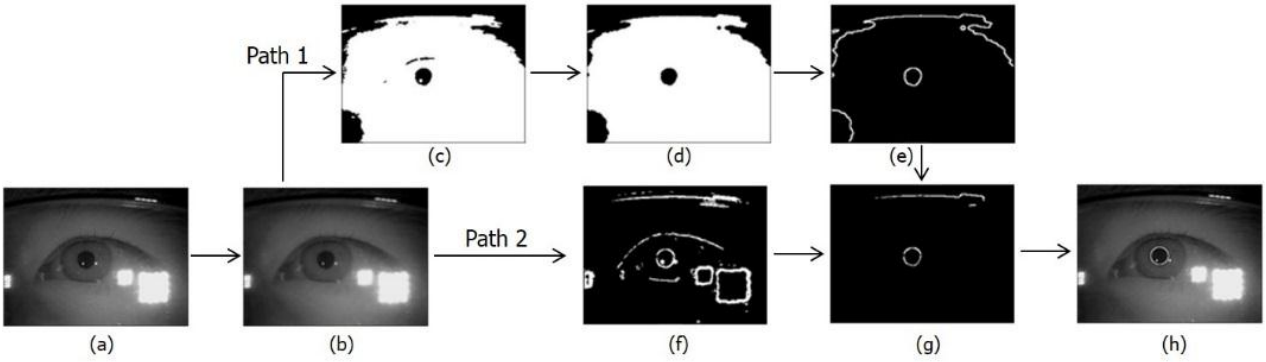


Figure 4.8. Edge-map generation for pupil boundary detection: **(a)** Iris image (320×240) from CITHV4; **(b)** Gaussian smoothed iris image (σ =1.0, k=5); **(c)** Binary image after applying intensity thresholding on (b); **(d)** Cleaned binary image obtained from (c) using hole filling followed by image opening (se='disk', k=7); **(e)** Edge image obtained after applying Sobel edge detector without thinning on (d); **(f)** Edge image obtained after applying Sobel edge detector without thinning on (b); **(g)** Edge-map obtained by intersection (logical AND) operation on (e) and (f); **(h)** Iris image with pupil detection (shown by white circle) obtained after applying CHT on (g).

The two morphological operations are applied on the binary image in Figure 4.8(c) to get the cleaned binary image shown in Figure 4.8(d); and the objective of these operations is reducing the noise-size so that the noise edges can be avoided in the intersection operation, which is illustrated later using Figure 4.9. First, a hole filling operation is applied on the binary image in Figure 4.8(c) to fill the white dots in the pupil region and then the image opening operation for black objects using a structuring element of type disc [Davies, 2012] is applied to reduce the size of the noise due to eyelids, eyelashes and eyebrow etc. Figure 4.8(d) shows the cleaned binary image in which the noise due to eyelids and eyelashes has been completely removed, but if the noise doesn't remove completely, its size reduces because the black regions of eyelids along with eyelashes in the binary image are not solid boundary compact objects like the pupil and the image opening operation removes the pixels at their boundaries.

The edge image of the cleaned binary image, shown in Figure 4.8(e), has the false edges due to dark illumination and eyeglass frame, but it could have contained other false edges due to the eyelids and eyelashes that can be removed or minimized by the intersection operation as illustrated using Figure 4.9.

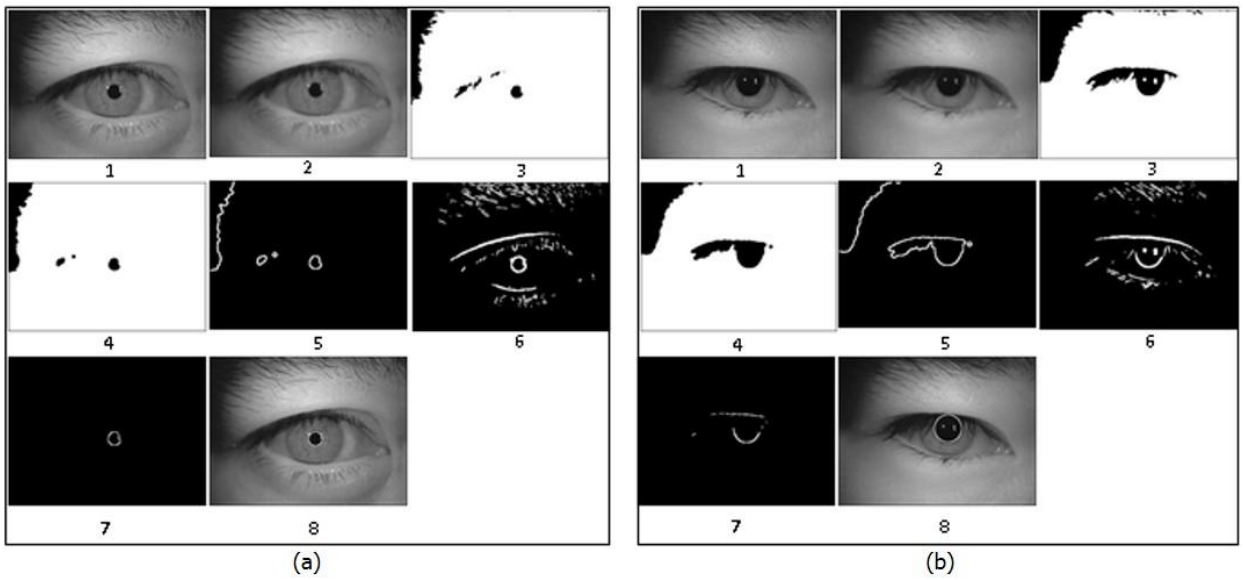


Figure 4.9. Edge-map generation for pupil boundary detection: **(a)** Ideal edge-map (image 7) that contains pupil boundary edges only; **(b)** Edge-map (image 7) that contains pupil boundary edges as well as false edges; [the images in (a) and (b) are: **1.** Iris image from CILV3; **2.** Smoothed iris image; **3.** Binary image after thresholding 2; **4.** Cleaned binary image obtained from 3; **5.** Edge image of 4; **6.** Edge image of 2; **7.** Edge-map obtained by intersection operation on 5 and 6; **8.** Pupil localized iris image obtained after applying CHT on 7].

The Figure 4.9 shows that the edge image of the cleaned binary image (image 5) contains the false edges due to eyelids and eyelashes, but these false edges are removed completely or partially after the intersection operation as shown in image 7. The image opening operation on the binary image (image 3) reduces the size of the noise due to the eyelids and eyelashes and hence, the reduced noise-size in the cleaned binary image (image 4) is not same as detected by the edge detection on the original iris image (image 6). Therefore, the intersection operation on the image 5 and the image 6 avoids the noise-edges completely or partially. Figure 4.9(a) shows an ideal situation, where the intersection operation removes the false edges completely (image 7), but the edge-map in Figure 4.9(b) has a few false edges also (image 7). The suitable threshold (T) for image binarization is chosen as discussed in chapter 3 under subsection 3.2.2, and threshold for edge detection on smoothed iris image kept same as Wildes' approach.

B) CHT for Pupillary Boundary Detection

The CHT that was implemented in subsection 4.1.1 and was used to detect the pupillary boundary in previously discussed Wildes' approach is used here also to detect the pupillary boundary. The 2D accumulator array after voting is shown in Figure 4.10 when the CHT is applied on the edge-map of Figure 4.8(g). In Figure 4.10, the radius (r) is equal to the pupil radius; therefore, the coordinates of the peak in the 2D accumulator array are the coordinates of the pupil center.

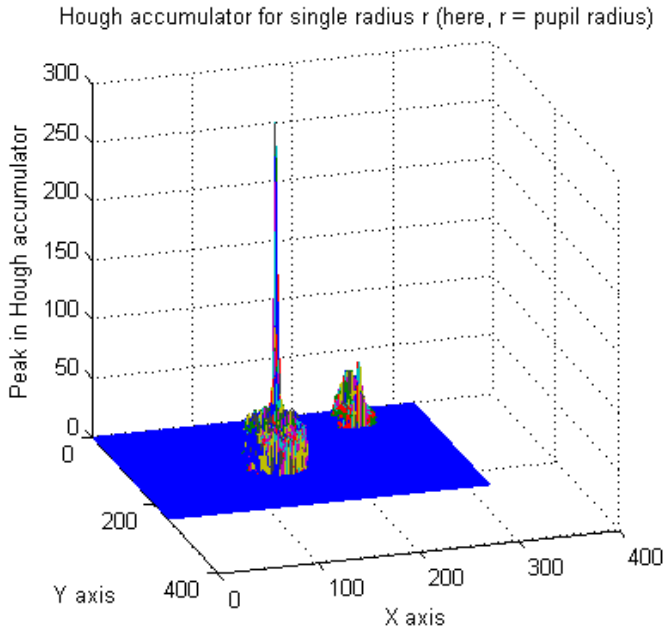


Figure 4.10. The surface plot of the 2D accumulator array corresponding to one radius after voting.

4.2.2 Limbic Boundary Detection

The center of the pupil circle is used as an input in detecting the limbic boundary as shown in Figure 4.11. The edge-map generation and adaptive CHT for limbic boundary detection are discussed below.

A) Edge-Map Generation

The limbic boundary detection may be hurdled by the eyelids, eyelashes, reflections and low contrast between the iris and sclera in the iris images [Jan et al., 2014]. A subimage is extracted from the iris image using a rectangle centered at the pupil center as shown in Figure 4.11(a) and Figure 4.11(b). The width of the rectangle (or subimage) is twice the maximum possible value of

the limbic boundary radius and the height is half of the width. The height of the rectangle can be increased further if the iris-occlusion by the eyelids and eyelashes is not much. The size of the rectangle remains constant for all the images from a database, but the location of the rectangle in the image changes as the rectangle is positioned using the pupil center.

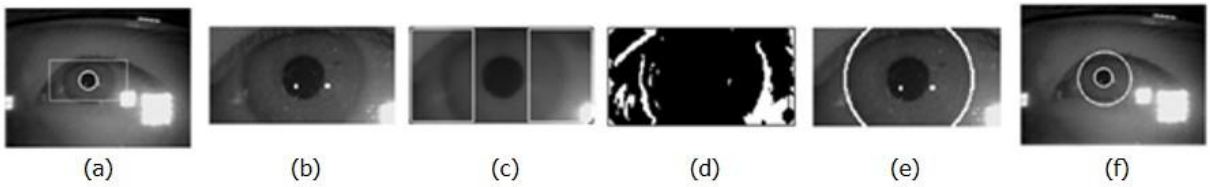


Figure 4.11. Limbic boundary detection: **(a)** Iris image (320×240) after pupil boundary detection; the rectangle in white indicates the size of subimage to be processed for limbic boundary detection; **(b)** The subimage (130×65) extracted from the iris image using the rectangle in (a); **(c)** Filtered subimage after applying a median filter of size 9×9 on (b); the two rectangles in white on left and right sides of the pupil are used to cover the iris's vertical contours; **(d)** Edge-map obtained after applying Sobel edge detection without thinning in horizontal direction inside the two rectangles in (c); **(e)** Circle detection after applying the adaptive CHT on (d); **(f)** Iris localized image (320×240).

The subimage in Figure 4.11(b) is filtered using a median filter [Gonzalez et al., 2009] to suppress the noise, such as the eyelash hair and uneven pixel intensities without damaging the edge structure. The upper and/or lower eyelids occlude the iris in the noisy iris images, but the vertical iris contours are always visible, which are used for detecting the limbic boundary. The vertical iris contours are covered using two rectangles that are placed as shown in Figure 4.11(c). The three sides of each rectangle touch the subimage borders and the fourth side of each rectangle is at a distance of pupil radius (rp) + 5 from the pupil center. To get the edge pixels, the Sobel edge detection without thinning operation is applied in the two rectangles in horizontal (x) direction only and threshold for edge detection kept same as Wildes' approach. Figure 4.11(d) shows the edge pixels that are used for the limbic boundary detection using the proposed adaptive CHT.

B) Adaptive CHT for Limbic Boundary Detection

Radman et al. in [Radman, 2013] had proposed an adaptive IDO for the limbic boundary detection, but here, we propose an adaptive CHT for the limbic boundary detection. In pupillary boundary detection, the CHT detects full circle and the circle-drawing angle sweeps by full 360 degree while detecting limbic boundary, the angle is changed to -45:45 and 135:225 degree. The

term “adaptive” means here that CHT has been modified to detect circular arcs instead of full circle. Instead of using the general CHT algorithm for the circle detection [Yuen et al., 1990], an adaptive CHT for the circular arc detection is applied on the edge-map shown in Figure 4.11(d). The adaptive CHT detects a structure of two circular arcs defined by -45:45 and 135:225 degree as shown in solid in Figure 4.12. The voting space in the adaptive CHT is limited to a small region around the pupil center instead of the whole image. The adaptive CHT for limbic boundary detection is useful for the images having iris-occlusions by the eyelids and eyelashes.

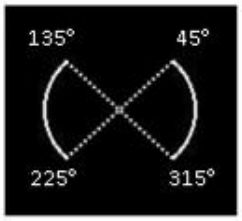


Figure 4.12. A set of two vertical arcs that the adaptive CHT finds in an image.

Algorithm 2. HT accumulator voting in adaptive CHT for limbic boundary detection

Compute: Center of subimage (x_o,y_o); imin=x_o-5; imax=x_o+5; jmin=y_o-5; jmax=y_o+5

```
Lines:  1.   - - -
        2.   A = zeros (rows,cols)                    // 2D accumulator
                                                         of subimage size
        3.     for all “white pixels” in edge map of subimage   do
        4.         for  θ = -45º to 45º   do
        5.           Calculate (x,y) using Equation (4.1)
        6.           if (imin ≤ x ≤ imax) and (jmin ≤ y ≤ jmax)   then
        7.           A(x,y) = A(x,y) +1;              // Accumulator-voting
                                                         step
        8.           end if
        9.         end for
        10.        for  θ =135º to 225º   do
        11.          repeat steps (lines) 5,6,7,8
        12.        end for
        13.    end for
```

The accumulator voting part of the adaptive CHT for limbic boundary detection is described in Algorithm 2. At all the white pixels (a,b) in the edge-map, the arcs’ structure shown in Figure 4.12 is drawn using the Equation (4.1) for a radius (r) and corresponding voting is done. The size of the 2D accumulator is same as the subimage, but voting space in the accumulator is limited to a 10×10 rectangle centered at the pupil center because the centers of the pupil and

limbic boundary circles lie within a small window [Daugman, 2004]. The peak in the 2D accumulator is determined corresponding to each radius and the maximum among the peaks gives the center and the radius of the limbic boundary circle. The 2D accumulator after voting is shown in Figure 4.13 when the adaptive CHT is applied on the edge-map of Figure 4.11(d). The Figure 4.13 shows the surface plot of the 2D accumulator corresponding to a radius equal to the limbic boundary radius and hence, the coordinates of the peak in the accumulator are the center coordinates of the limbic boundary circle. The adaptive CHT for limbic boundary detection is faster as it searches for half the circle length instead of a full circle, which requires only half the virtual circle length to be drawn at each edge pixel.

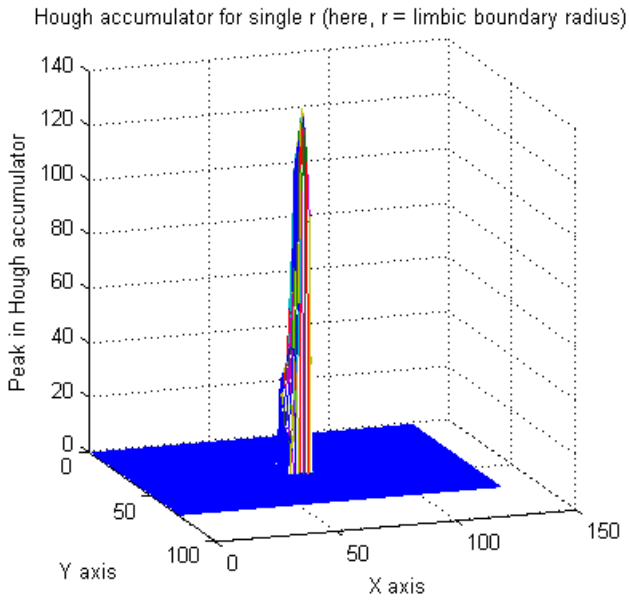


Figure 4.13. The surface plot of 2D accumulator array in the adaptive CHT corresponding to one radius after voting. [voting space is a 10×10 rectangle centered at pupil center]

4.2.3 Performance Evaluation

In this section, the performance of the proposed algorithm is evaluated by conducting experiments on iris databases, the iris localization results are summarized and the results are compared with some state-of-the-art iris localization methods in the literature. The datasets used in the experiments to evaluate the proposed algorithm are described below.

A) Datasets Used

The datasets are taken from two CASIA iris databases: CITHV4 and CILV3. These databases are chosen because they contain the noisy images having the noise such as reflections, non-uniform

illuminations, low contrast, eyeglasses and intrusions by the eyelids, eyelashes and eyebrow hair. Both CITHV4 and CILV3 contain 8-bit gray-level JPEG images with resolution of 640×480 pixels.

1. CITHV4 dataset: For extensive experimentation with this database, the images from all 1000 subjects are chosen. A total 5600 images are chosen which include all the images of the first 100 subjects and 3600 images from the rest of 900 different subjects (selecting 4 images from each subject).
2. CILV3 dataset: This database contains images from 411 different subjects [17]. The total number of the images in the database is 16212. For thorough experimentation with the database, 811 images were chosen selecting first left and first right eye image of each subject except 11 subjects.

The experiments on the datasets were done using a computer with Intel i5 CPU @ 2.40 GHz, 8 GB RAM and Windows 7 operating system. The proposed algorithm is implemented and tested with MATLAB (version 8.4) tool.

B) Results and Discussion

Figure 4.14 shows the sample images with accurately localized irises by the proposed algorithm. Figure 4.15 and Figure 4.16 are the examples, which illustrate that the standard edge-map generation techniques for pupil localization give wrong results, but they are corrected by the proposed technique. The wrong pupil localization fails the limbic boundary localization also as it uses the pupil circle parameters as inputs. The results of the proposed iris localization algorithm are summarized in Table 4.2.

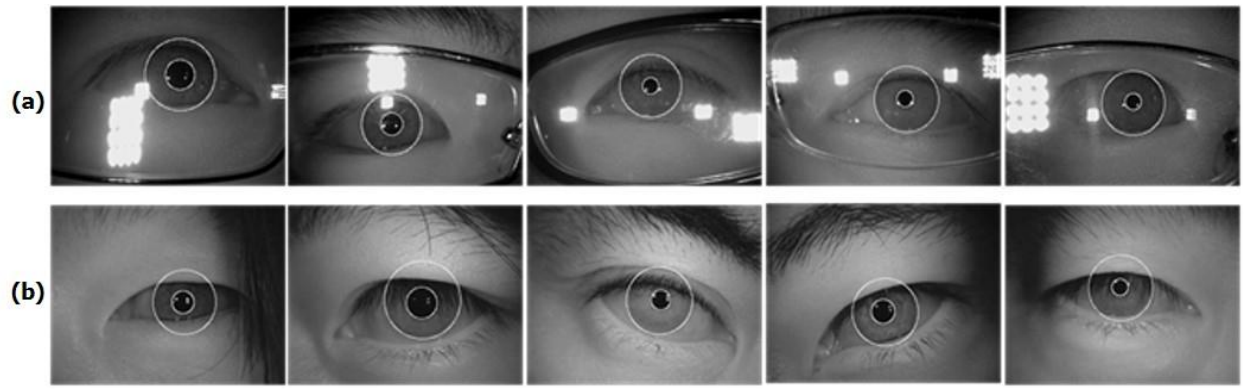


Figure 4.14. Accurately localized irises in the iris images from two CASIA databases: **(a)** CITHV4; **(b)** CILV3.

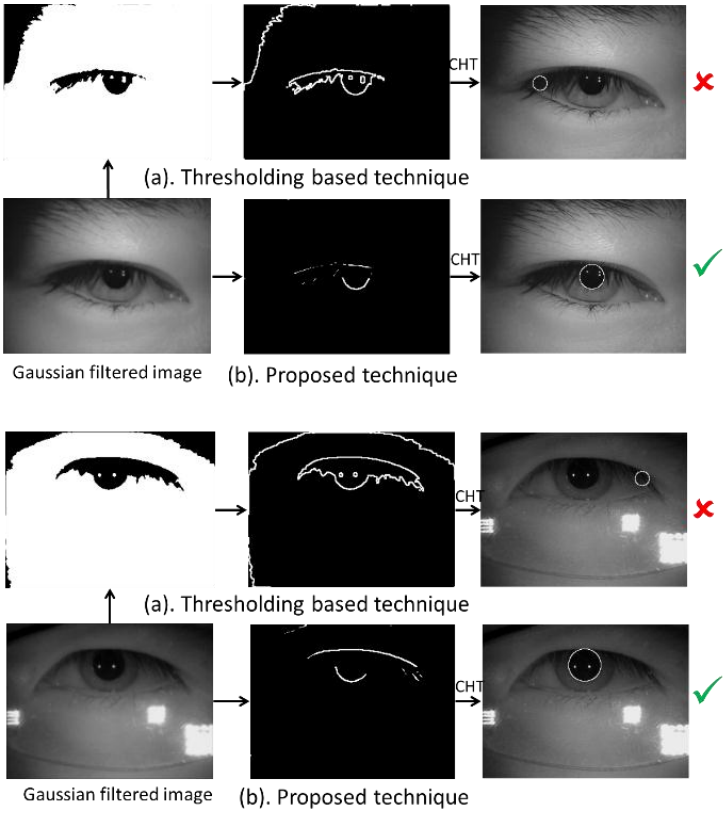


Figure 4.15. Examples of wrong pupil localization using thresholding based technique, which is corrected by the proposed technique.

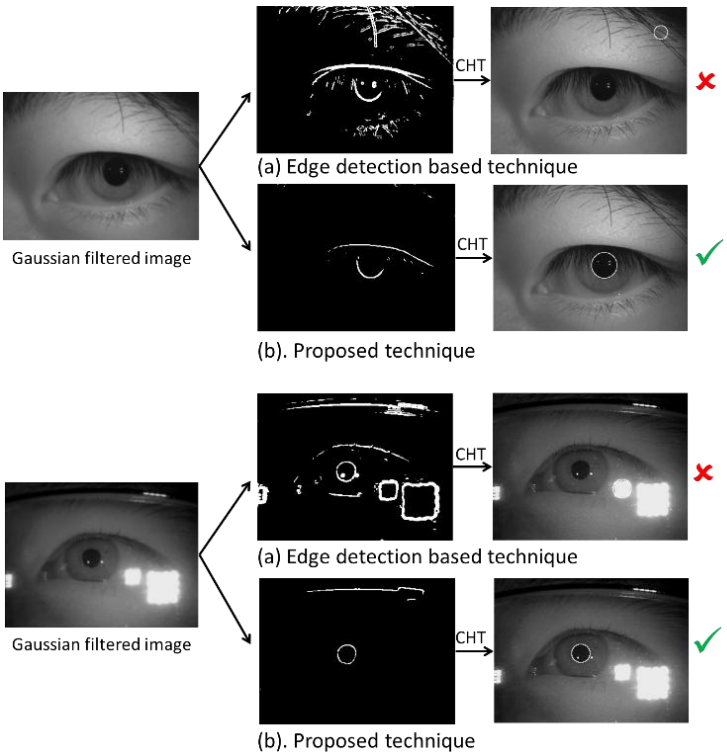


Figure 4.16. Examples of wrong pupil localization using edge detection based technique, which is corrected by the proposed technique.

Table 4.2. Performance of the proposed iris localization method (Simulation results)

Iris database	Number of images taken for testing (N_T)	Number of correct iris localized images (N_I)	Accuracy (%) = (N_I/N_T) ×100	Average time cost per image (sec)
CITHV4* (640×480)	5600	5583	99.70	0.92
CILV3* (640×480)	811	806	99.38	0.89
MMUV1 (320x240)	450	448	99.55	0.78

* 640×480 image is resized to 320×240 image before applying the iris localization method and resize time is included in the time cost of iris localization.

The accuracy of the proposed method is close to 100 percent as shown in Table 4.2. The accuracy of the circle detection in an image by the CHT depends on the amount of false edges the edge-map of the image contains. Fewer the false edges higher would be the accuracy of the CHT presented in subsection 4.1.1. Many false edges present in the edge-map of an image may form an imaginary arc/circle, which increases probability of false circle detection (Refer to Figure 4.5). The edge-map used for the pupil boundary detection in the proposed algorithm contains very less false edges due to the intersection operation as discussed in the subsection 4.2.1. The use of the adaptive CHT for limbic boundary detection to counter the iris-occlusions by the eyelids and eyelashes is another cause for high accuracy.

Table 4.2 also shows the time performance results of the proposed method. The average time cost is reported in the table, as the time taken by the CHT for circle detection is directly proportional to the number of edge pixels in the edge-map of the image. Fewer the false edges in the edge-map of iris image, lesser will be the time cost. The average time cost per image was calculated by randomly choosing 500 images from each individual database. The MATLAB's timer functions 'tic' and 'toc' were used to know the execution time of a code that runs to localize irises in 500 images. The execution time obtained was then divided by 500 to find the average time cost per image.

4.3 Comparison

The comparison of Wildes' iris localization approach discussed in section 4.1 and the proposed method described in section 4.2, is shown in Table 4.3, which shows the improved results for the proposed method in terms of both accuracy and time performance.

Table 4.3. Comparison of iris localization methods

Method	CITHV4		MMUV1		CILV3	
	Accuracy (%)	Time cost per image (sec)	Accuracy (%)	Time cost per image (sec)	Accuracy (%)	Time cost per image (sec)
Wildes	95.10	2.35	98.44	2.63	85.69	3.79
Proposed	**99.70**	**0.92**	**99.55**	**0.78**	**99.38**	**0.89**

Table 4.3 shows that Wildes' approach gives good accuracy for MMUV1 database as it contains less noisy images, but its accuracy degrades for the noisy images of CILV3 and CITHV4. The Wildes' approach gives less iris localization accuracy mainly due to the reflection spots in CITHV4 and too many false edges from the occlusions by eyelids and eyelashes in CILV3. The proposed algorithm is more accurate and faster than Wildes' approach as it uses the optimal edge-maps with very less false edges and the adaptive CHT for iris boundary detection improves it further. The proposed method is faster than Wildes' approach due to significant reduction in the number of false edges in the edge-maps for each of the pupillary and limbic boundaries detection. During accumulator voting process, the CHT requires to draw significantly less number of circles on the edge-points, which reduces the computation time in the proposed method. Although the edge-map generation for pupillary boundary in the proposed method is slower than the edge-map generation in Wildes' approach, as it is based on generating two edge-maps and then combining them using intersection operation, but the overall speed of the proposed algorithm is higher due to significant reduction in the number of false edges in the edge-map for CHT.

Table 4.4. Comparison with published iris localization results

Method	Accuracy (%) & Average time cost per image (sec)	
	CITHV4	CILV3
[Jan et al., 2012]	99.5 & 6.4	98 & 4.93
[Jan et al., 2013]	99.23 & 3.4	99.21 & 3.35
Proposed method	**99.70 & 0.92**	**99.38 & 0.89**

The comparison of the results of the proposed method with the published results is shown in Table 4.4. The published methods included in the comparison are chosen on the basis that they used same databases and similar platform for experimentation that we have taken. Moreover,

[Jan et al., 2012], [Jan et al., 2013] show the highest accuracy among all the iris localization methods available in the literature for CITHV4 and CILV3 databases. The Table 4.4 shows that the proposed method has the highest accuracy and lowest time cost per image, which has happened due to the proposed edge-map and the adaptive CHT used for pupil and limbic boundary detection respectively, as compared to the other methods in the table. In the proposed method, the original iris image is scaled down to half size, which was also done in the Jan et al. methods to speed up the processing. The image resizing by a scaling factor, s = 0.5 not only reduces all the edge pixels to half in number, but also the number of radii taken in a CHT algorithm becomes half.

4.4 Concluding Remarks

The Wildes' approach was implemented for localizing irises in NIR images and its performance evaluation showed that this approach could be applied to constrained images having less occlusions by eyelids and eyelashes, no reflection spots and homogeneous image characteristics, etc.

The proposed iris localization method is tolerant to the non-ideal issues and noises in the iris images such as iris-occlusions by the eyelids and eyelashes, lighting reflections, non-uniform illumination, eyeglasses, low contrast and eyebrow hair. However, the simulation results show that the proposed method also improves iris localization in the images that do not have reflection spots and non-uniform illumination, but have mainly the iris-occlusions by the eyelids and eyelashes.

The comparison of the proposed method with the Wildes' approach demonstrates that the introduction of new edge-maps for pupillary and limbic boundary detection, and adaptive CHT for limbic boundary detection make the proposed iris localization method more accurate and fast. The performance results of the proposed algorithm are much better than the Wildes' approach. The comparison with some recent published results for CASIA databases also shows that the proposed method has improved performance. The proposed algorithm can be used for the accurate iris segmentation in less constrained iris recognition systems.

Chapter 5
Hardware Implementation of Iris Localization for NIR Images

The circular Hough transform (CHT) based method is chosen for hardware implementation of iris localization as we found it more suitable for parallel implementation on field programmable gate array (FPGA) in contrast to the integro-differential operator (IDO) based algorithm. To impart parallelism in the IDO based algorithms, *k* copies of the input image are required to store in the 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.

This chapter first discusses the hardware implementation of CHT algorithm that detects the circle in an image. The CHT algorithm is core of any CHT based iris localization method. However, the edge-map of image that acts as input for CHT algorithm is also important as it decides the overall performance of the iris localization method. The hardware implementation of the edge-map generation techniques has been described first for the edge-map generation of pupillary boundary followed by the edge-map generation of limbic boundary in individual sections.

5.1 CHT Hardware

The voting process in the CHT for a radius is an iterative process, which repeats for every edge pixel (edge-point) in the edge-map as shown in Figure 5.1. The Figure 5.1 shows that the size of 2D accumulator array (A) is same as the image in the direct CHT implementation and *A* stores the count values those are incremented by the generated circle points (x,y). The Figure 5.1 shows the voting process in the direct CHT for a radius (r), but the maximum count (peak) value stored

in *A* is also determined for every radius. The maximum of all the peaks corresponding to different radii gives center and radius of the detected circle.

```
[rows,cols]=size(image);                    // Input iris image
A=zeros(rows,cols);                         // 2D accumulator array
  for all "Edge-points (a,b)" in edge map of image do
      for θ =1 to 360° do
          x = a + r * cosθ ;
          y = b + r * sinθ ;                // Circle point generation
          x = round( x );
          y = round( y );
          if (x,y) is in image bounds do
          A( x, y ) = A( x, y )+1 ;         // Accumulator voting step
          end if
      end for
  end for
```

Figure 5.1. Pseudo code for voting process in general (direct) CHT algorithm.

For an image of P×Q pixels, the sequential implementation of general (direct) CHT requires a single 2D accumulator array of P×Q cells (Figure 5.1) for all *k* number of radii, but for the parallel implementation on FPGA, *k* number of P×Q accumulator arrays are required (One accumulator array per radius). This would result in large memory requirement as it is equivalent to storing *k* images in the memory. Assuming *k* equal to 16 and each image pixel of 16 bits, the total memory requirement in the CHT implementation for a 320×240 image would be $320 * 240 * 16 * 16$ (=19660800) bits. This memory is too large that it exceeds the total size of embedded memory blocks in some FPGAs, such as Xilinx Zynq-7000 or Spartan 6. Moreover, this memory requirement would increase further in the situations, where the image size is beyond 320×240 pixels or *k* is greater than 16, which is very much feasible. Therefore, an alternative CHT architecture has been proposed that will solve this problem. This architecture is described in next subsection.

5.1.1 Proposed CHT Method

For a given radius, the circle detection in the proposed CHT is a two-step process: (1) Coarse circle center detection using one 2D accumulator array; (2) Fine circle center detection using two 1D accumulator arrays. For detecting a circle in an image of P×Q pixels, the proposed CHT uses a 2D accumulator array of (P/m)×(Q/n) cells and two 1D accumulator arrays of P×1 and Q×1 cells as shown in Figure 5.2, where *m* and *n* are integers in power of two form. In 'voting process

in the proposed CHT for a radius (r), the coordinates, *x* and *y* of the generated circle point (x,y) in Figure 5.1 are divided by *n* and *m* respectively before the accumulator voting step to map (fit) the coordinates in the reduced size 2D accumulator array, but voting in the two 1D accumulator arrays is done directly using the coordinates, *x* and *y* without dividing by *n* and *m*. The 2D accumulator array is used to find radius of the circle based on the maximum count value stored in it and the 1D accumulator arrays are used to find center of the circle (Figure 5.2).

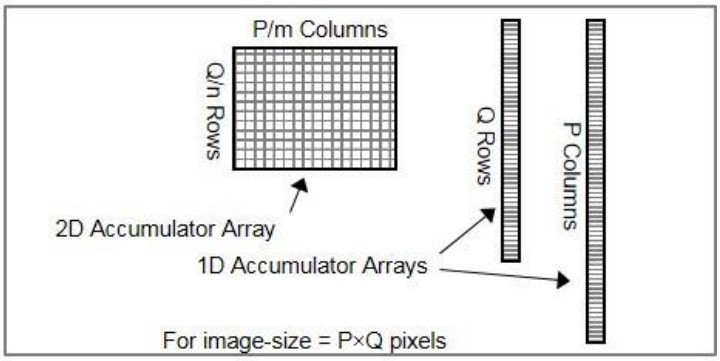


Figure 5.2. Voting space structure in the proposed CHT.

A cell in the 2D array pointed by (row number, column number) stores the count value, which is incremented by 1 every time a circle passes through it. Similarly, the cells in 1D arrays store the count values. The 2D array in Figure 5.2 cannot give the exact value of circle center because its each column comprises *m* consecutive columns of the image and each row comprises *n* consecutive rows of the image. For example, for *m*=2, the columns 0 and 1 of the image are combined to form column 0 in the 2D array, and the columns 2 and 3 of the image constitute column 1 of the 2D array and so on. Similarly, for *n*=4, the rows 0, 1, 2 and 3 of the image are combined to form row 0 in the 2D array, and the rows 4, 5, 6 and 7 of the image constitute row 1 of the 2D array and so on. Therefore, the 2D accumulator array in the proposed CHT provides coarse center of the circle, which is a cell with maximum count value.

To find the exact (fine) circle center from the coarse circle center, the 1D arrays shown in Figure 5.2 are used. The *m* and *n* consecutive cells in the two 1D accumulator arrays are located corresponding to the coordinates of the coarse center and a cell with maximum count value in each group of *m* and *n* consecutive cells gives a coordinate of fine circle center. The coarse to fine detection approach for circle center in the proposed CHT is illustrated in Figure 5.3. The Figure 5.3 shows that the proposed CHT uses one 2D accumulator array of 160×60 cells and two 1D accumulator arrays of 320 and 240 cells to detect a circle in a 320×240 pixel image for a given radius value, where the 2D array gives coarse center (90,29) of the circle (coarse detection)

and the coordinates, 90 and 29 of coarse center are multiplied by *m* (=2) and *n* (=4) respectively to locate the cells, 180 and 116 in the 1D arrays for fine detection process. The cell with maximum value in each of {180,181} and {116,117,118,119} cells gives the column and row coordinates of fine center respectively (fine detection).

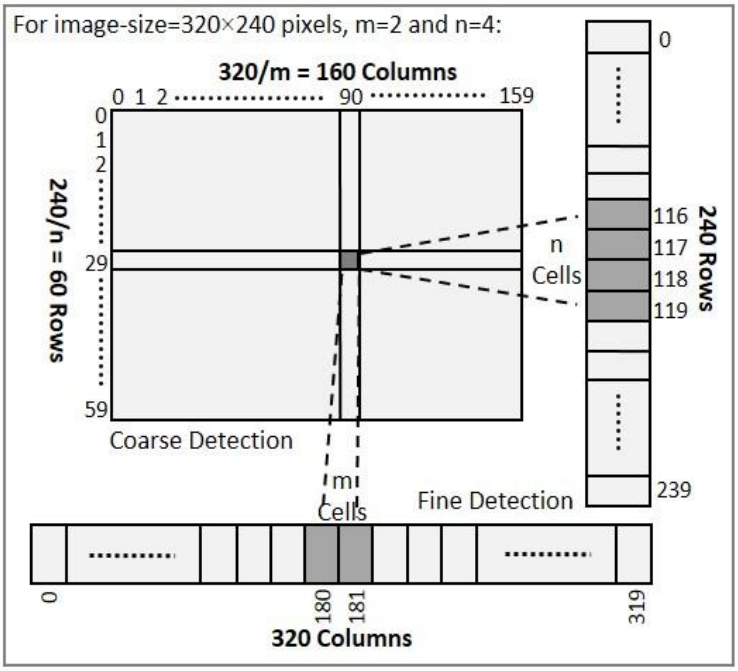


Figure 5.3. Coarse to fine detection strategy for the circle center in the proposed CHT.

Table 5.1. Memory requirements in the proposed CHT

CHT technique	Memory in bits	Memory reduction (%)
Direct CHT	19660800	0
Proposed (m=2, n=2)	5058560	74.27
Proposed (m=2, n=4)	2600960	86.77
Proposed (m=4, n=2)	2600960	86.77
Proposed (m=4, n=4)	1372160	93.02
Proposed (m=4, n=8)	757760	96.14
Proposed (m=8, n=4)	757760	96.14
Proposed (m=8, n=8)	450560	97.70

5.1.2 Memory Requirements

The memory requirements in the proposed CHT architecture for detecting a circle in a 320×240 pixel image are shown in Table 5.1, when each memory location is 16 bits wide and 16 different radii are taken. Therefore, for *m*=2 and *n*=2, the memory required to realize all the accumulator arrays of the proposed CHT is 5058560 (=[160×120+320+240]×16×16) bits, whereas the

memory required in the direct CHT implementation is 19660800 (=320×240×16×16) bits, thereby providing a memory reduction of 74.27% in the proposed CHT as compared to the direct CHT. Since the size of 1D arrays are negligible as compared to the 2D array for smaller values of $m \times n$, so, it can be stated that the memory requirement in the proposed CHT is reduced by a factor of $m \times n$ as compared to the direct CHT implementation.

5.1.3 CHT Hardware Architecture

The proposed CHT hardware architecture is shown in Figure 5.4, which imparts the parallelism by running the CHT modules for different radii (CHT Module 0, CHT Module 1, CHT Module 2 and so on corresponding to radii r0, r1, r2 and so on) in parallel. Each CHT module detects a circle in the image for a given radius value and the best circle among all the detected circles is chosen using 'maximum selector' component and a multiplexer (Figure 5.4).

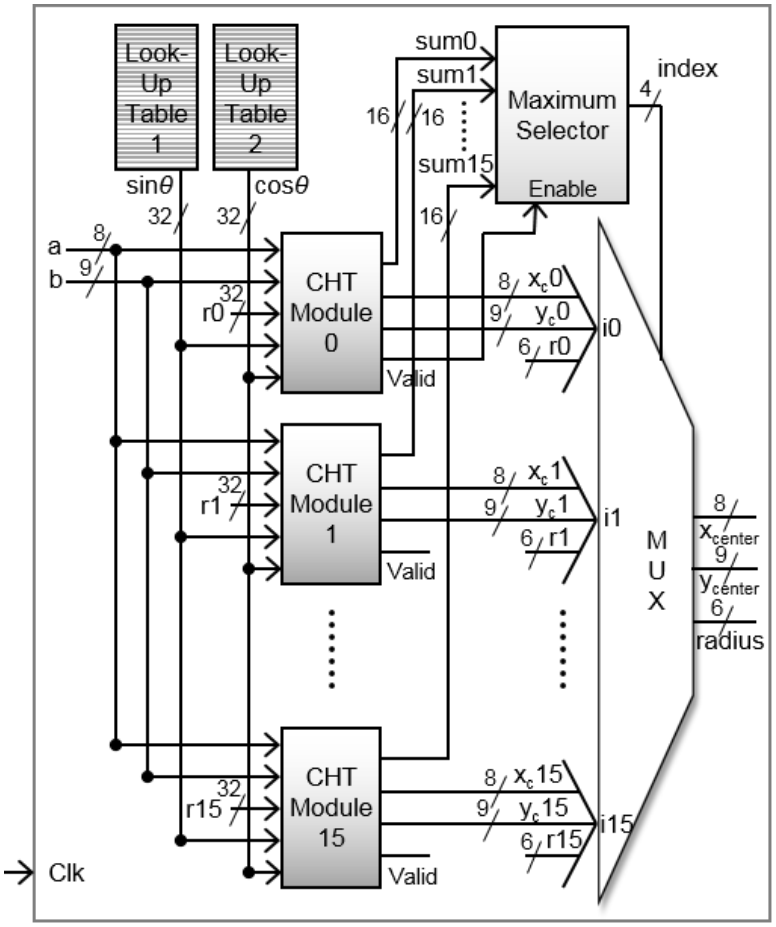


Figure 5.4. The proposed CHT hardware architecture.

For an edge-point (a,b), each of the CHT modules generates the circle points and updates its corresponding accumulator arrays (one 2D and two 1D). After all the edge-points are taken,

each CHT module is also responsible for finding the maximum count (peak) value stored in its 2D accumulator array and detecting the fine circle center. Once fine detection is over, the CHT modules output the center coordinates as (x_c0,y_c0), (x_c1,y_c1), (x_c2, y_c2) and so on, along with the peak values of the 2D accumulator arrays as sum0, sum1, sum2 and so on (Figure 5.4). Now, the 'maximum selector' block finds the maximum among the peak values (sum0, sum1, sum2 and so on) and outputs the index of the maximum peak value. For example, if the maximum is sum9, an index 9 (1001) is the output of the 'maximum selector' block. The index value then selects the best circle among all the circles using the multiplexer.

The number of CHT modules taken is 16, which are decided by the range of radii specified as input to the CHT. For example, if the minimum and maximum radius of the circle to be detected were specified as 15 and 30, the number of CHT modules would be 16. The two user defined look-up tables are used to store sine and cosine values, which are needed for the circle point generation (Figure 5.4). The each look-up table is implemented by configuring block RAM as read only memory (ROM) of size 360×32-bits, which stores the values in 32-bit fixed-point format. For example, $sin240^o$ (= -0.866) is stored as *10000000011011101100110011110111*, where most significant bit (MSB) represents the sign, 8 bits next to the MSB represent integer part and rest of 23 bits are used for fraction part. As shown in Figure 5.4, (a,b) is the edge-point in an image of 320×240 pixels, where *a* is 8-bit number and *b* is 9-bit number.

All the CHT modules (CHT Module 0 to CHT Module 15) in Figure 5.4 are identical construction and the hardware architecture of a single CHT module is discussed next.

A) CHT Module

The main units in a CHT module are circle point generator, address generation for coarse circle detection and accumulator voting with fine detection as shown in Figure 5.5. Each unit and its constituents are described below. For a given radius, the computation of *sum* (the maximum count) stored in 2D accumulator array and corresponding fine circle center (x_c,y_c) is explained in this section.

i) Circle Point Generator

This unit generates the circle points (x,y) taking edge-point (a,b) as center of the circle for a given radius (r), using the equations provided in Figure 5.1. The product terms, $r*cos\theta$ and $r*sin\theta$ *a*re generated using two fixed-point multipliers followed by rounding off operation as shown in

Figure 5.5. The radius value is provided in 32-bit fixed-point format and the values of *cosθ* and *sinθ* are read from the ROMs (user defined look-up tables). In 'rounding and signed integer conversion' component (Figure 5.5), the 32-bit output of the fixed-point multiplier is rounded off to the nearest integer using a 32-bit adder and then applied to a logic circuit to convert it to 8-bit signed integer. The signed integers are added to the edge-point coordinates, *a* and *b* to obtain the circle point coordinates, *x* and *y*.

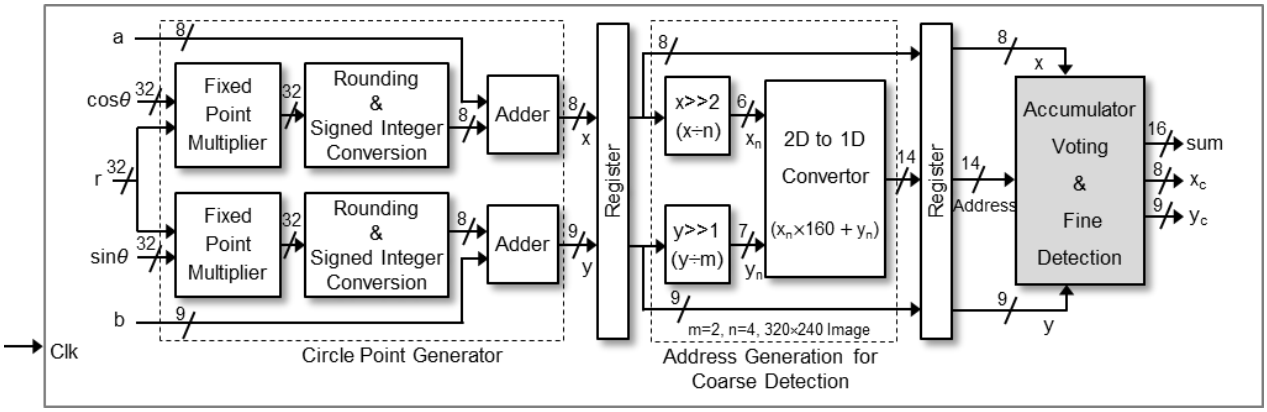


Figure 5.5. Hardware block diagram of a single CHT module.

ii) Address Generation for Coarse Detection

This unit takes the generated circle points (x,y) as input and divides their coordinates by *m* and *n* to map (fit) them in the 2D accumulator array as shown in Figure 5.5. The division is performed using shifters as *m* and *n* are the numbers in power of two form. The 2D accumulator array is first converted to a 1D array and then implemented using block RAMs, which provides minimal wastage of the words of block RAMs as discussed later in subsection 5.1.7. Therefore, the two indices (x_n,y_n) which are used to access cells in a 2D array are converted to a single index denoted by 'Address' in Figure 5.5, to access cells in the 1D array. The 'Address' is generated by a '2D to 1D convertor' for voting in the accumulator space as shown in Figure 5.5 for *m*=2, *n*=4 and 320×240 image. For the generated circle point (x,y), *x* is 8-bit number and *y* is 9-bit number. For *m*=2 and *n*=4, as the rows and columns get reduced, the circle point (x,y) gets converted to 6-bit and 7-bit number respectively for coarse detection and perform the accumulator voting in 2D array, whereas for finer detection, the circle point (x,y) remains 8-bit and 9-bit number respectively and performs the accumulator voting in 1D array. The accumulator voting and fine detection is explained in next section.

iii) Accumulator Voting and Fine Detection

The one 2D accumulator array and two 1D accumulator arrays of the proposed CHT are implemented using block RAMs, which are denoted by BRAM1, BRAM2 and BRAM3 respectively in Figure 5.6. For an image of 320×240 pixels, the size of BRAM1 is 9600×16-bits for *m*=2 and *n*=4, whereas the size of BRAM2 and BRAM3 are 320×16-bits and 240×16-bits respectively. The BRAM1 is used for coarse circle detection, whereas BRAM2 and BRAM3 are used for fine detection.

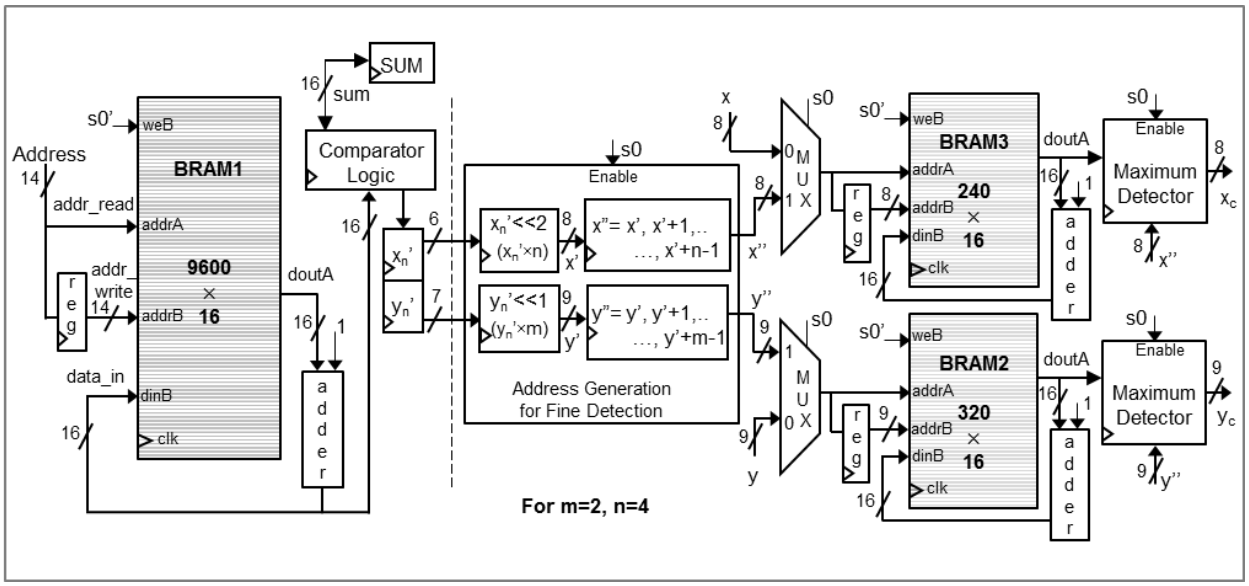


Figure 5.6. Accumulator voting and fine detection unit.

The BRAM1 is updated using the address generated by the 'address generation for coarse detection' unit as shown in Figure 5.5, whereas the addresses for updating BRAM2 and BRAM3 are the coordinates, *x* and *y* of the circle point. The updating of BRAM1, BRAM2 or BRAM3 means reading the contents of a memory location, adding one to the contents and writing the contents back to the same memory location. The dual port block RAMs are used so that data can be read and written in same clock cycle. To update BRAM1, BRAM2 or BRAM3 in a single clock cycle, an additional register is introduced between read and write address as shown in Figure 5.6. In a single clock cycle, the memory for current address is read; and the previously read and incremented memory data is written at the previous address, called as accumulator voting. The process of finding maximum count (peak) value stored in the 2D accumulator (BRAM1), also goes simultaneously with accumulator voting and the peak value is saved in a register named as SUM (Figure 5.6). The SUM is initially zero and while updating BRAM1, the read out data is incremented by 1 and compared with the current value of SUM and if it is

greater, the SUM contents are replaced by incremented data, but finding the peak value in BRAM2 and BRAM3 is not required. The indices x_n and y_n associated with SUM contents are stored in the registers x_n' and y_n', and the coarse center (x',y') of the circle is obtained by multiplying these registers' contents by *m* and *n* as shown in Figure 5.6.

Once voting is over for all edge-points, the process of fine detection for circle center is turned on by a control signal *s0*. The value of *s0* remains 0 during the accumulator voting process and it becomes 1 as soon as the process completes. The *s0* activates all the blocks involved in fine detection process (the portion in the diagram right to dotted line) as shown in Figure 5.6. The writing in capability for BRAM2 and BRAM3 is disabled using *s0* and read address line is also changed using a multiplexer. Now, *m* memory locations in BRAM2 and *n* memory locations in BRAM3 are read, whose addresses are generated by the 'address generation for fine detection' block illustrated in Figure 5.6. A memory location with maximum stored value among each of *m* and *n* memory locations is determined by the 'maximum detector' block, which gives the fine circle center (x_c,y_c).

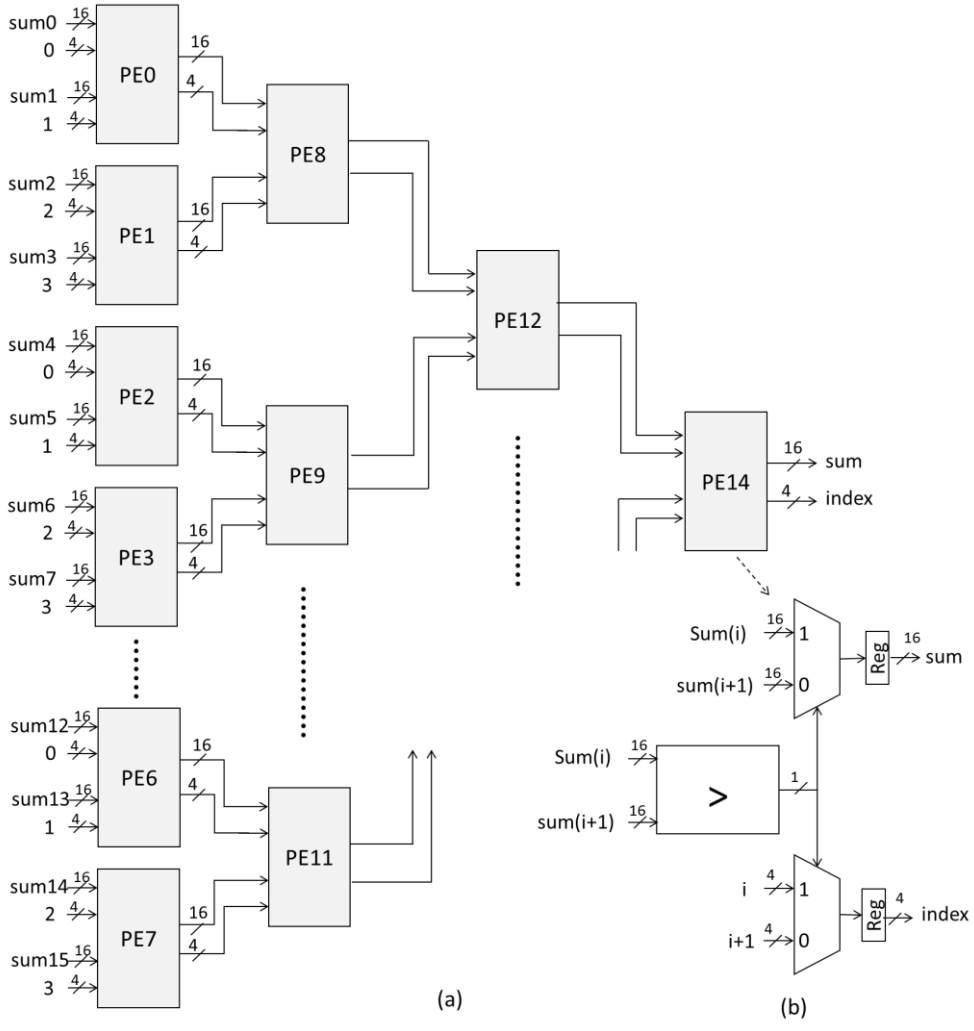


Figure 5.7. Maximum selector: **(a)** Architecture; **(b)** Processing element (PE) used in (a).

B) Maximum Selector

Referring to Figure 5.4 shows that the 'maximum selector' block finds the maximum of the accumulated peak values (sum0, sum1, sum2 and so on) as soon as accumulator voting and fine detection process completes. It not only finds the maximum value but also outputs index of the maximum value that is used to select one of the sets of circle parameters provided by CHT modules. For instance, if sum8 happens to be the maximum value, the output of the 'maximum selector' block would be index 8 (1000 in binary).

The 'maximum selector' module is designed as a comparator tree as shown in Figure 5.7. For 16 input values, the overall architecture is partitioned into 4 stages consisting of 15 interconnected homogeneous processing elements (PEs). The results of the PEs in the previous stage are used as inputs to the PEs in the next stage. A PE compares the two values and the result is used to allow the higher value to pass through to the next stage along with its (higher value's) index. Each PE consists of one comparator, two multiplexers and two registers as shown in Figure 5.7(b).

5.1.4 Hardware Control Flow

The hardware components involved in the proposed CHT architecture are already discussed, but the control of operation of the components is described in Figure 5.8. The major portion of the flowchart in Figure 5.8 describes the control flow inside a CHT module, which applies to all the CHT modules (CHT module 0 to CHT module 15 corresponding to radii r0 to r15 as discussed earlier) running in parallel. The CHT module runs every time it receives a new edge-point (a,b) and stops after all the *N* edge-points in the edge-map of the image are taken.

When a new edge-point (a,b) is received (read), the addresses of the ROMs (look-up tables, which store sine and cosine values) are reset to zero for reading the first memory locations of the ROMs, which is indicated in the flowchart (Figure 5.8) by initializing θ to 0. The 'circle point generator' unit generates a circle point (x,y) every clock cycle by incrementing the addresses of ROMs by 1 and a total 360 clock cycles are taken to generate all the circle points for an edge-point (a,b). The accumulator voting corresponding to the generated circle points also goes in parallel as the pipelined registers are used between different units of the CHT module (Figure 5.5).

For every clock cycle, a circle point (x,y) is generated, but BRAM1, BRAM2 and BRAM3 inside the 'accumulator voting and fine detection' unit are updated only if the generated

circle point is in image bounds. The generated circle point may go out of the image boundaries for the edge-points lying close to the image border. Therefore, the 'address generation for coarse detection' unit takes only those circle points that fall within the image boundaries.

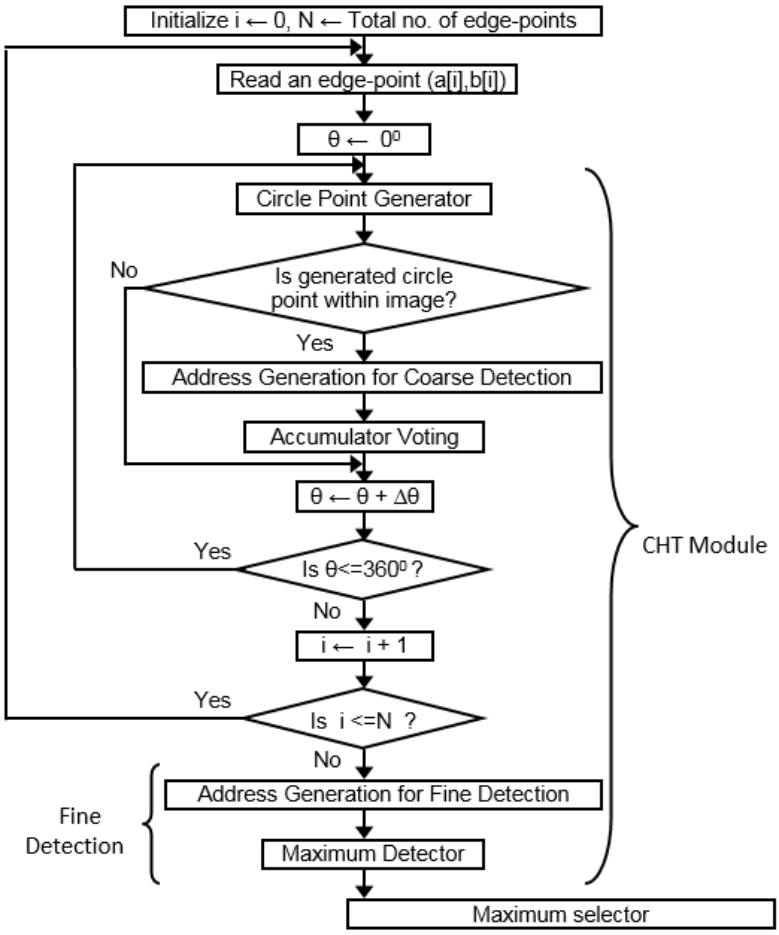


Figure 5.8. Hardware control flow of the proposed CHT architecture.

The fine detection process inside each CHT module is triggered immediately after all the *N* edge-points are taken, followed by best circle selection using 'maximum selector' component. For detecting circle in an image, the processing time of complete CHT architecture is directly proportional to the total number of edge-points, *N* because the time taken by the fine detection process and the best circle selection is insignificant as they execute only once.

5.1.5 Accuracy Evaluation of Proposed CHT Architecture

In this section, the accuracy performance of the proposed CHT method (architecture) is evaluated for iris localization application. We have proposed a new CHT method for hardware implementation, which requires less memory requirements due to reduced size of accumulators. This method was evaluated for its accuracy performance using MATLAB, as the MATLAB

supports accuracy evaluation of image processing algorithms. To evaluate the accuracy performance, the edge-maps for pupillary and limbic boundary detection were generated using MATLAB and the edge-points were applied to CHT as shown in Figure 5.9.

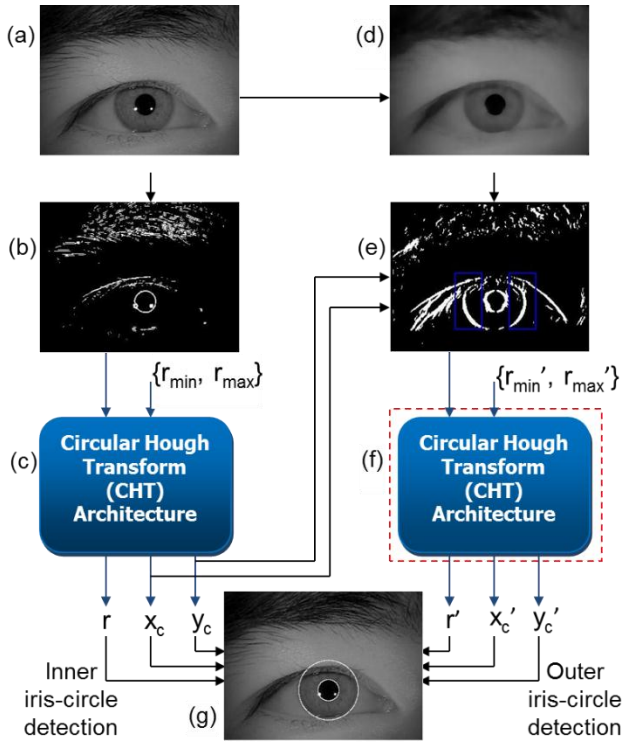


Figure 5.9. Accuracy evaluation of the proposed CHT architecture: **(a)** Input iris image; **(b)** Edge-map for pupillary boundary detection obtained after applying Sobel edge detector without thinning on (a) in both horizontal and vertical directions; **(c)** CHT architecture to detect circle in (b); **(d)** Smoothed image of (a) obtained using median filter; **(e)** Edge-map of (d) obtained using vertical Sobel edge detector without thinning: The edge-points covered by two rectangles (in blue) are used for limbic boundary detection; **(f)** The CHT architecture used for pupillary boundary detection is used again for limbic boundary detection also.

In Figure 5.9, the edge-map generation technique for limbic boundary detection is same as was discussed in the proposed method of chapter 4, but the edge-map generation technique for pupillary boundary detection is different, because here, the purpose is to find the accuracy of the proposed CHT method (architecture) with respect to direct CHT. The CHT hardware runs twice in Figure 5.9, to detect both pupillary boundary (inner iris-circle) and limbic boundary (outer iris-circle). The CHT for detecting outer iris-circle is applied only on the selected edge-points in the edge-map. The edge-points are selected taking the detected center of inner iris-circle as input for selection and using two rectangles to cover vertical outer iris-contours avoiding iris-occlusions by the eyelids as shown in Figure 5.9(e).

The two CASIA iris databases are chosen: a) CASIA-iris-thousand, version 4.0 (CITHV4); b) CASIA-Iris, version 1.0 (CIV1). The CIV1 database images have bigger iris-circles as compared to CITHV4 as shown later in Figure 5.10(b). The 200 images of different subjects are chosen from each database. These images were chosen such that, in all the images, the pupils are accurately localized by the direct CHT using the method of Figure 5.9. This image selection makes easier the visualization of accuracy difference between the proposed CHT architectures and the direct CHT. The accuracy results of inner and outer iris-circle detection are summarized in Table 5.2 and Table 5.3 respectively.

Table 5.2. Inner iris-circle (pupillary boundary) detection results

CHT method	CITHV4 (320×240) Accuracy (%)	CIV1 (320×240) Accuracy (%)
Direct CHT	100	100
Proposed (m=2, n=2)	100	100
Proposed (m=2, n=4)	99	100
Proposed (m=4, n=4)	96.5	98.5

Table 5.3. Outer iris-circle (limbic boundary) detection results

CHT method	CITHV4 (320×240) Accuracy (%)	CIV1 (320×240) Accuracy (%)
Direct CHT	100	94
Proposed (m=2, n=2)	100	94
Proposed (m=2, n=4)	98	94
Proposed (m=4, n=4)	96	93

Table 5.2 shows that the direct CHT detects the inner iris-circles accurately in all the images taken for testing. To find the accuracy of outer iris-circle detection, all input images taken are the images with accurately detected inner iris-circles because inner iris-circle center is used to select the edge-points on which the CHT is applied as described in Figure 5.9. Three different sets of *m* and *n* values are only taken for finding the accuracy of the proposed CHT (Table 5.2, Table 5.3), because for higher *m* and *n* values, the accuracy falls down and does not remain close to the direct CHT. The set {m=4, n=2} is not shown in Table 5.2 and Table 5.3 as its accuracy is same as the set {m=2, n=4}.

Table 5.4. Iris localization (inner + outer iris-circle detection) results

CHT method	Accuracy (%)		Memory reduction (%)
	CITHV4	CIV1	
Direct	100	94	0
Proposed (m=2, n=4)	**96**	**94**	**87**

The iris localization accuracy of the proposed CHT for localizing both iris-boundaries is shown in Table 5.4. We choose the set {m=2, n=4} for the proposed CHT as it gives the accuracy almost same as the direct CHT but memory reduction of 87% is obtained, whereas the sets {m=2, n=2} and {m=4, n=4} provide lesser memory reduction and lesser accuracy respectively. For CITHV4 and CIV1 images, the proposed CHT gives the accuracies of 96% and 100% respectively as compared to the direct CHT.

5.1.6 FPGA Implementation Results

The proposed CHT architecture was implemented with the Verilog HDL to target Xilinx's 7 series FPGA, Zynq xc7z020-1clg484 and tested on Zedboard. Table 5.5 shows the synthesis results of the complete CHT architecture presented in Figure 5.4. The overall frequency obtained is 203.00 MHz. The total number of block RAMs is 98 which implements all the accumulator arrays of 16 CHT modules and two ROMs (user defined look-up tables) used for storing sine and cosine values. For a single CHT module, 6 block RAMs are used which comprises 5 block RAMs for implementing 2D array converted to 1D array of size 9600 and 1 block RAM for both 1D arrays of size 320 and 240. The target FPGA contains 36K size block RAMs and each block RAM can be configured as a dual port RAM of aspect ratio 2K×16-bits. Each block can also be used as two independent 1K×16-bits dual port RAMs. The dual port block RAMs are used to realize accumulators, which are available in Zynq FPGA device. We did not use FPGA fabric to implement these multiport block RAMs.

Table 5.5. Synthesis results of the proposed CHT architecture

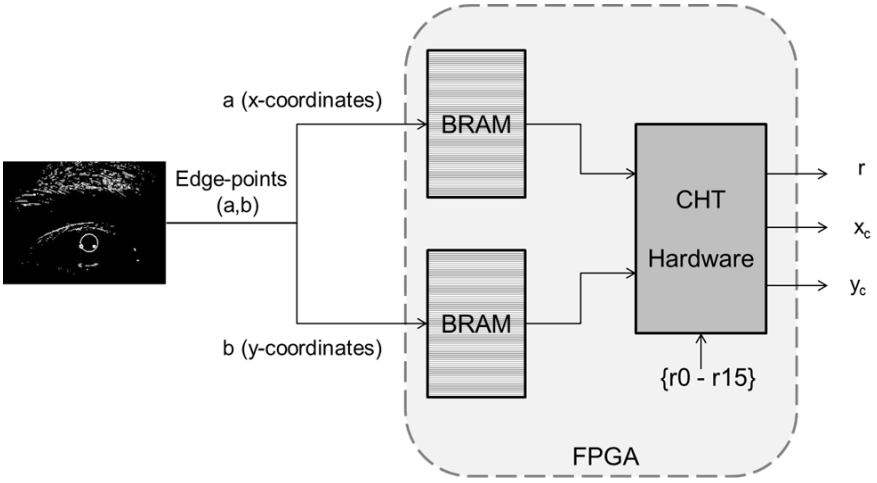
Logic utilization	Proposed CHT (m=2,n=4)
Number of slice LUTs	12886
Number of slice registers	1416
Number of block RAMs	98
Maximum frequency, f_{max} = 203.00 MHz	

Table 5.6 shows that the memory reduction in terms of the number of block RAMs is 84% for

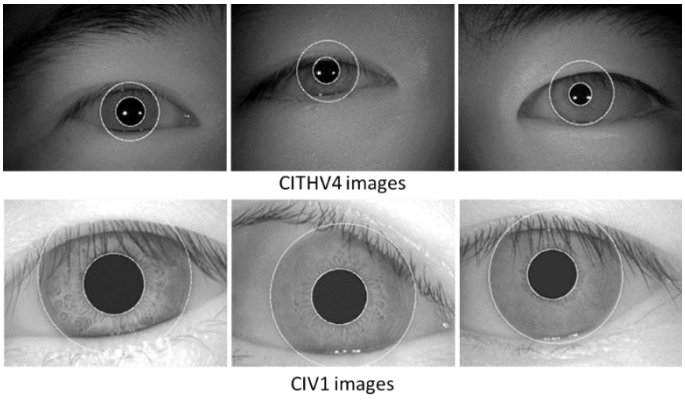
the proposed CHT, whereas theoretical reduction was 87%. This difference is due to the wasted bits (words) in implementing 2D array converted to 1D array of 9600 cells (BRAM1) and two 1D arrays of 320 (BRAM2) and 240 (BRAM3) cells using 36K size block RAMs.

Table 5.6. Utilization of number of block RAMs (synthesis results)

Proposed CHT (m=2,n=4)	Direct CHT	Block RAMs reduction
98	602	84%



(a)



(b)

Figure 5.10. (a) Testing of CHT hardware on FPGA; **(b)** Iris localized images using the proposed CHT architecture.

5.1.7 Performance Results and Discussion

Figure 5.10(a) shows testing of CHT hardware on FPGA. The coordinates of edge-points were stored in two BRAMs, which act as input for CHT hardware. These BRAMs and CHT hardware was ported on FPGA, which gives circle parameters as output after execution. The Zedboard was

used, which contains Xilinx's 7 series FPGA, Zynq xc7z020-1clg484.

To evaluate the time performance, 100 images of different subjects from CITHV4 database were taken. The numbers of edge-points in the edge-maps of iris image are calculated for inner and outer iris-circle detection (Figure 5.9). The time performance results (obtained by multiplying number of clock cycles and clock period) of 5 images with a clock of 200 MHz are listed in Table 5.7, and it was found that processing time for circle detection is directly proportional to the number of edge-points. The average processing time per image would be 6.25 ms, which is calculated from 100 images as shown in Table 5.8.

In subsection 5.1.4, it is mentioned that 360 clock cycles are taken to generate all the circle points for one single edge-point. In addition to this, 21 clock cycles are taken collectively by pipelined registers, fine detection process and maximum selector block. Therefore, for N number of edge-points, processing time is equal to (N*360+21)*(1/200MHz), where, 200MHz is frequency of operation of the architecture. Ignoring the number, 21, as it is negligible as compared to (N*360), the processing time shown in Table 5.8 is obtained as follows:

Average processing time for inner iris-circle detection = 2350*360*(1/200MHz) = 4.23 ms

Average processing time for outer iris-circle detection = 1124*360*(1/200MHz) = 2.02 ms

Thus, average processing time for iris localization = 4.23 + 2.02 = 6.25 ms

The Ngo et al. [7] CHT has the average processing time of 5.15 ms but for outer iris-circle detection only. The sequential execution of the proposed CHT takes average computation time of 1.6 sec per image when runs using MATLAB (version 8.4) installed on a computer with Intel i5 CPU @ 2.40 GHz, 8 GB RAM and Windows 7 operating system. Therefore, the proposed CHT hardware architecture gives a 250× speedup as compared to the sequential execution.

Table 5.7. Processing time of the proposed CHT hardware architecture

CITHV4 image	Inner iris-circle detection		Outer iris-circle detection		Total processing time (ms)
	Number of edge-points	Processing time (ms)	Number of edge-points	Processing time (ms)	
S5004R00.jpg	2065	3.72	1399	2.52	6.24
S5009R07.jpg	2842	5.12	955	1.72	6.84
S5024L00.jpg	4190	7.54	973	1.75	9.29
S5033L00.jpg	3481	6.27	1241	2.23	8.5
S5063L00.jpg	1483	2.67	885	1.59	4.26

Table 5.8. Average processing time

Number of CITHV4 images	Inner iris-circle detection		Outer iris-circle detection		Average processing time per image (ms)
	Average number of edge-points (N)	Average processing time (ms)	Average number of edge-points (N)	Average processing time (ms)	
100	2350	4.23	1124	2.02	6.25

5.1.8 Comparison with Other CHT Architecture

To compare the accuracy of the proposed CHT, we also implemented the Ngo et al. CHT [Ngo et al., 2014] and applied in the iris localization method presented in Figure 5.9. The Ngo et al. CHT differs from the proposed CHT as its voting space structure contains one 2D accumulator array of size 320×(240/n) cells and one 1D accumulator array of size 240×1 cells (compare with Figure 5.2) for detecting circle in a 320×240 image. The 2D array is used to find radius and column coordinate (y) of circle center, whereas 1D array gives the row coordinate (x) of the circle center [Ngo et al., 2014]. For n=4, n=8 and n=16, the obtained memory reduction in Ngo et al. CHT is 74.4%, 86.9% and 93.1% respectively as compared to the direct CHT. The accuracy results of the Ngo et al. CHT are shown in Figure 5.11.

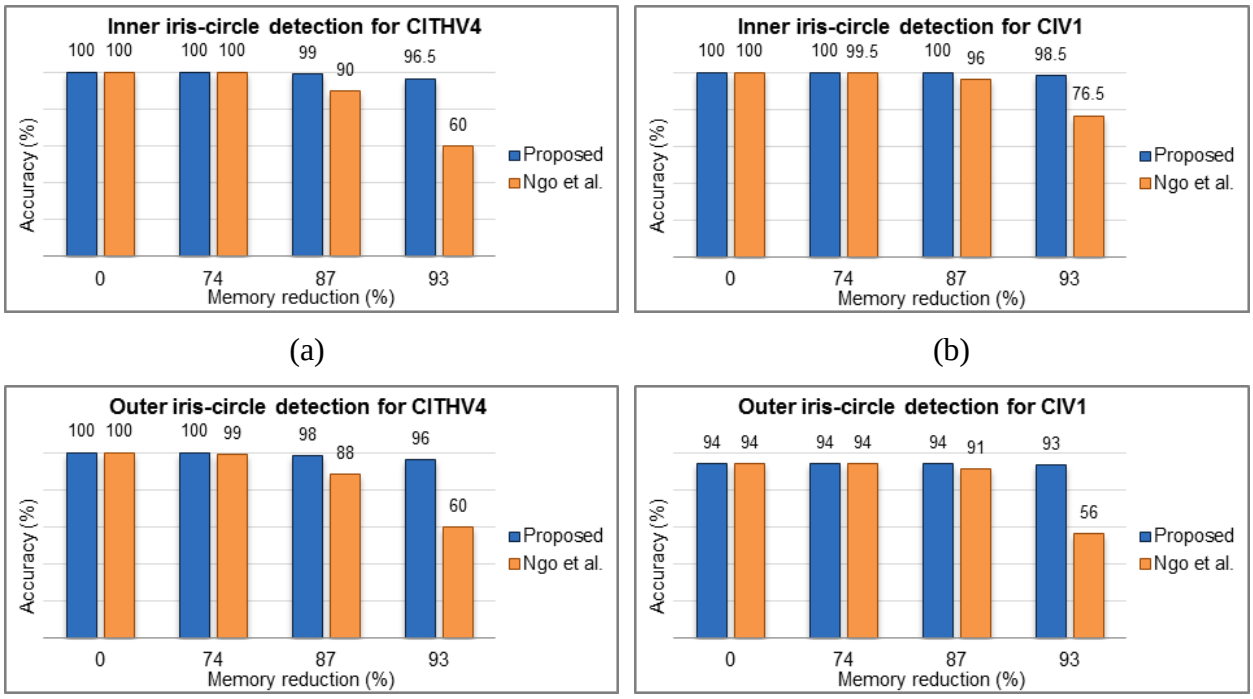


Figure 5.11. Accuracy comparison of the proposed CHT with the [Ngo et al., 2014] CHT for inner and outer iris-circle (iris-boundary) detection in CITHV4 and CIV1 images.

The MATLAB simulation results in Figure 5.11 show that the proposed CHT has better accuracy than the Ngo et al. CHT for both inner and outer iris-circle detection to get same memory reduction. The results are discussed below.

The charts (a) and (b) in Figure 5.11 show that the Ngo et al. CHT is more accurate for CIV1 images than CITHV4 because the circles in CIV1 are bigger, whereas the proposed CHT is accurate for both CITHV4 and CIV1 images. The Ngo et al. CHT combines more number of rows in a group as compared to the proposed CHT to get same memory reduction, which reduces its accuracy for small circles. For example, to get the memory reduction of 93%, the Ngo et al. CHT combines 16 rows of the image in a single row of the 2D accumulator array (i.e. n=16), whereas the proposed CHT combines 4 rows of the image in a single row and 4 columns of the image in a single column of the 2D accumulator array (i.e. n=4 and m=4). It is concluded here, that the Ngo et al. CHT can detect bigger circles accurately but its accuracy degrades when applied to detect smaller circles, whereas the proposed CHT shows better accuracy for both smaller and bigger circles.

The charts (c) and (d) in Figure 5.11 illustrate that although the outer iris-circles are bigger, but the accuracy is not better than the inner iris-circle detection for both the Ngo et al. and the proposed CHTs because full circle is not visible as the iris is obstructed by eyelids and eyelashes. In [Ngo et al., 2014], the accuracy of 92% for outer iris-boundary detection with 93% memory reduction was reported, but remember that it was not for CASIA database (CITHV4 and CIV1) images where the iris is occluded by eyelids and eyelashes.

For complete iris localization also, the proposed CHT architecture is more accurate than the Ngo et al. CHT as shown in Table 5.9.

Table 5.9. Comparison of iris localization (inner + outer iris-circle detection) results

CHT method	Accuracy (%)		Memory reduction (%)
	CITHV4	CIV1	
Direct	100	94	0
Ngo et al. (n=8)	68	88.5	87
Proposed (m=2, n=4)	**96**	**94**	**87**

The synthesis results of the proposed CHT are compared with the synthesis results taken from [Ngo et al., 2014] as shown in Table 5.10. However, these results are for different FPGA devices. This comparison is to show the drastic reduction in number of BRAMs used. The synthesis results reported in [Ngo et al., 2014] are however for n=16, whereas the proposed CHT's synthesis results are for m=2 and n=4, but they can be compared for the resources usage

other than block RAMs because by changing *m* or *n*, only the number of block RAMs changes without any major change in slice LUTs or other resources of FPGA. The [Ngo et al., 2014] used 16 block RAMs for a single CHT module and the complete CHT architecture had required 256 block RAMs for 16 different radii as shown in Table 5.10, but this number of block RAMs would get double almost for *n*=8 as each row buffer uses one block RAM. The target FPGA used in [Ngo et al., 2014] contains M9K memory blocks (block RAMs), where each block is configured as 512×16-bits dual port RAM module.

Table 5.10. Comparison with other CHT implementation

CHT implementation	[Ngo et al., 2014]	Proposed
Image resolution (pixels)	320×240	320×240
Bit resolution (bits)	16	16
Number of slice LUTs	9688	12886
Number of slice registers	12165	1416
Number of block RAMs	256	98
Maximum frequency	214.78 MHz	203.00 MHz

There are two ways to implement memory in FPGA: 1) As distributed RAM realized using FPGA fabric; 2) As block RAM realized using prebuilt embedded memory blocks called BRAMs. There is no wastage of bits if memory is realized as distributed RAM, but there may be wastage of bits if realized as block RAM. For example, if BRAM size is 1K words and we want to implement 1D accumulator array (memory) of 320 cells using BRAM, then (1K-320) are wasted words.

The number of block RAMs and wasted words in implementing the 2D accumulator array is more in Ngo et al. as it uses one block RAM of size 512×16-bits for each row buffer of 320 columns, thus wasting 192 (= 512 - 320) words per block RAM; whereas in the proposed CHT, the 2D accumulator array is first converted to 1D array and then implemented by interconnection of multiple block RAMs using their full words-capacity except last one which could have wasted words. Therefore, the proposed implementation utilizes embedded memory of FPGA more efficiently. The utilization of slice LUTs is more in the proposed CHT because it computes the circle points ($r*cos\theta,\ r*sin\theta$) for 16 different radii from the stored values of cosθ and sinθ, whereas the Ngo et al. CHT stores the pre-computed circle points ($r*cos\theta,\ r*sin\theta$) for all 16 different radii in the look-up tables. Moreover, there are other components in the proposed CHT which are not used in the Ngo et al. CHT such as ‘Rounding and signed integer conversion’, ‘2D to 1D convertor’ and hardware for fine detection.

5.2 Edge-Map Generation Hardware for Pupillary Boundary

The technique of edge-map generation for pupillary boundary detection was discussed in section 4.2 of the previous chapter, which generates the optimal edge-map by combining two different edge-maps using intersection operation. Now, this section describes the hardware implementation of that technique. However, that technique is first optimized here for hardware implementation as shown in Figure 5.12. The optimized technique shows some modification of excluding the morphological operations over the previous technique presented in chapter 4: section 4.2, but both the techniques are based on the same idea of applying intersection operation (logical ANDing) on two different edge-maps of the same image to get the final edge-map. This modification has insignificant effect on the iris localization accuracy, which is shown later in section 5.5 of this chapter.

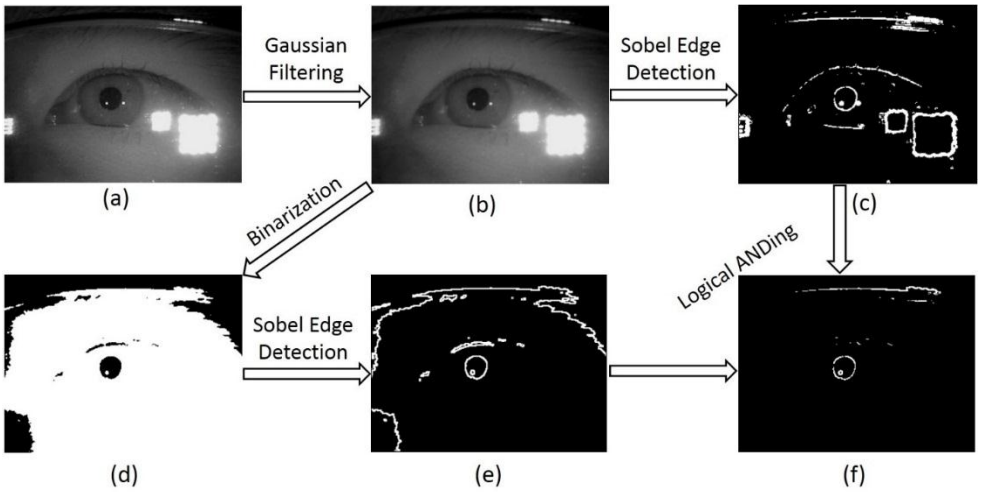


Figure 5.12. Edge-map generation technique for pupillary boundary optimized for hardware implementation: **(a)** Original iris image (320×240) from CITHV4; **(b)** Gaussian smoothed iris image (σ =1.0, k=3); **(c)** Edge image obtained after applying Sobel edge detector without thinning on (b); **(d)** Binary image after applying intensity thresholding on (b); **(e)** Edge image obtained after applying Sobel edge detector without thinning on (d); **(f)** Edge-map obtained by intersection (logical ANDing) operation on (c) and (e).

The proposed technique is suitable for parallel implementation on hardware as both the edge-maps in Figure 5.12(c) and Figure 5.12(e) can be generated in parallel and then combined using AND gates to obtain the optimized edge-map. The proposed technique (Figure 5.12) not only reduces the false edges due to eyelids, eyelashes, eyeglasses and eyebrow hair, but also eliminates the false edges due to reflections and dark illumination caused by the lighting source while capturing the image.

5.2.1 Proposed Hardware Architecture

The proposed hardware architecture is shown in Figure 5.13 for the edge-map generation technique of Figure 5.12. This architecture reads one pixel of the input image every clock cycle and starts outputting one pixel per clock cycle of the final edge-map after an initial latency, thereby giving a throughput of one pixel per clock cycle. The initial latency is majorly caused by the sliding window component. This architecture is a parallel and pipelined design.

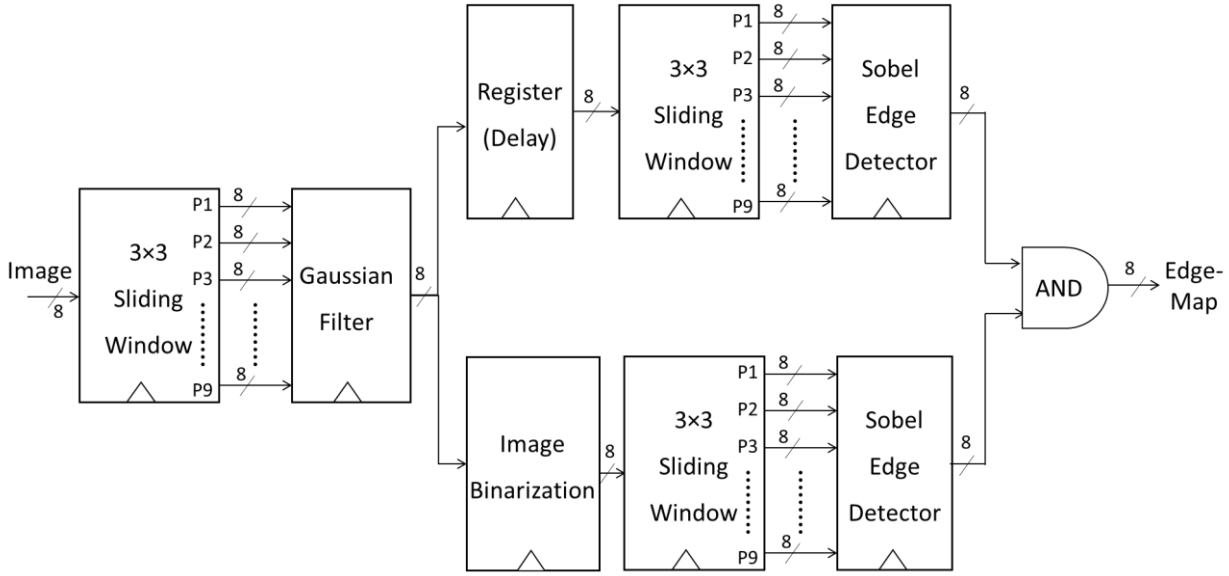


Figure 5.13. Proposed edge-map generation hardware architecture for pupillary boundary.

The pipelining is realized using the sliding windows and the pipelined registers, and parallelism means the two edge-detected images are generated in parallel and combined to give a single edge-map. The sliding windows provide 3×3 pixels (P1, P2, P3… P9) of image to the Gaussian and Sobel filters. Figure 5.13 shows that the pixels of the input (original) image are processed by passing through a number of modules, such as sliding window, Gaussian filter, image binarization, Sobel edge detector and finally through AND gate, to get the final edge-map of the input image. The various hardware modules used in the proposed architecture are described next.

5.2.2 Sliding Window

In window based filtering [Bailey, 2011], [Gonzalez et al., 2009], the pixels of the input image are transformed using their local neighborhood pixels, which are selected using a window of size

W×W as shown in Figure 5.14. This window is generally a square in shape with *W* an odd number. The pixel of the output (filtered image) is obtained by performing some computations on the pixels within the window. The computations performed are determined by the type of filter (filter function, *f*). As the window is scanned through the input image, each possible position of window generates an output pixel according to filter function, *f*. Since the output depends not only on the input pixel but also on its local context, filters can be used for noise removal or reduction, edge detection and feature detection [Bailey, 2011].

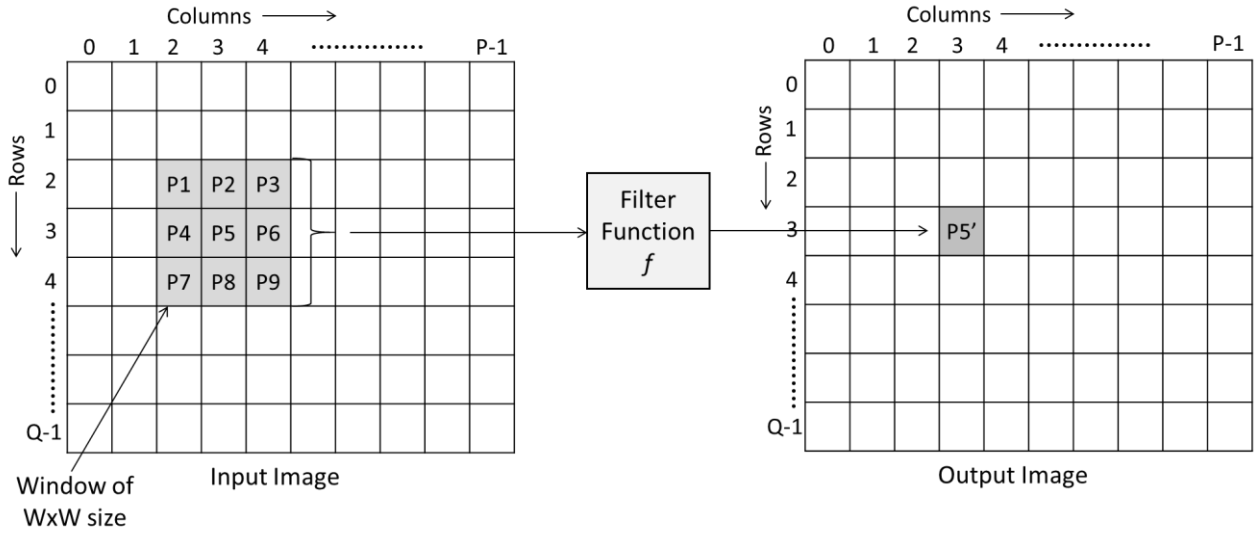


Figure 5.14. Window filter. The shaded pixels represent the input window located at P5 that produces the filtered value, P5' for the corresponding location in the output image. Each possible window position in the input image generates the corresponding pixel value in the output image.

From a purely software perspective, images are 2D arrays that reside in memory. The image filtering is a convolution operation of the input image with filter mask (kernel) defining the filter function. A 3×3 convolution in software pseudo-code would look as shown in Figure 5.15 for a filter kernel, *k* and input image, *I* to get the filtered output image, *O* (= $k * \mathrm{I}$). The pseudo-code shows that image filtering is an iterative process, which repeats for each pixel in the input image. Figure 5.15(c) shows the 3×3 pixel window of the input image, whose pixels keep on changing when the window moves through the whole image from top-left to bottom-right of the image.

$O = k * I$

$$k = \begin{bmatrix} c1 & c2 & c3 \\ c4 & c5 & c6 \\ c7 & c8 & c9 \end{bmatrix}$$

(a)

```
for (x = 1; x < ImageHeight-1; x++) do
  for (y = 1; y < ImageWidth-1; y++) do
    O(x,y) =  c1*I(x-1,y-1) + c2*I(x-1,y) + c3*I(x-1,y+1) +
              c4*I(x,y-1) + c5*I(x,y) + c6*I(x,y+1) +
              c7*I(x+1,y-1) + c8*I(x+1,y) + c9*I(x+1,y+1);
  end for
end for
```

(b)

$$\text{Window} = \begin{bmatrix} I(x-1,y-1) & I(x-1,y) & I(x-1,y+1) \\ I(x,y-1) & I(x,y) & I(x,y+1) \\ I(x+1,y-1) & I(x+1,y) & I(x+1,y+1) \end{bmatrix} = \begin{bmatrix} P1 & P2 & P3 \\ P4 & P5 & P6 \\ P7 & P8 & P9 \end{bmatrix}$$

(c)

Figure 5.15. Window based image filtering: **(a)** Filter kernel; **(b)** Pseudo-code of 3×3 convolution for image filtering using filter kernel of (a); **(c)** 3×3 image window.

For hardware implementation of image filters, the sliding window architecture is used to perform the filtering operation as shown in Figure 5.16 for a 3×3 filter kernel. This architecture takes a pixel stream (one pixel per clock cycle) of the image as input and provides a 3×3 pixel stream (3×3 pixels per clock cycle) of the image as output, after an initial delay that is roughly equal to the time required to fill the line buffers (as described later). The output, 3×3 pixel (P1, P2, P3… P9) stream of the image is used in calculations with the 3×3 filter mask (kernel) of a particular filtering operation. This will have the effect of sliding a 3×3 kernel window in raster scan over the image, that is starting at the top-left and going up to the bottom-right of the image. The sliding window architecture has a single input line and nine output lines.

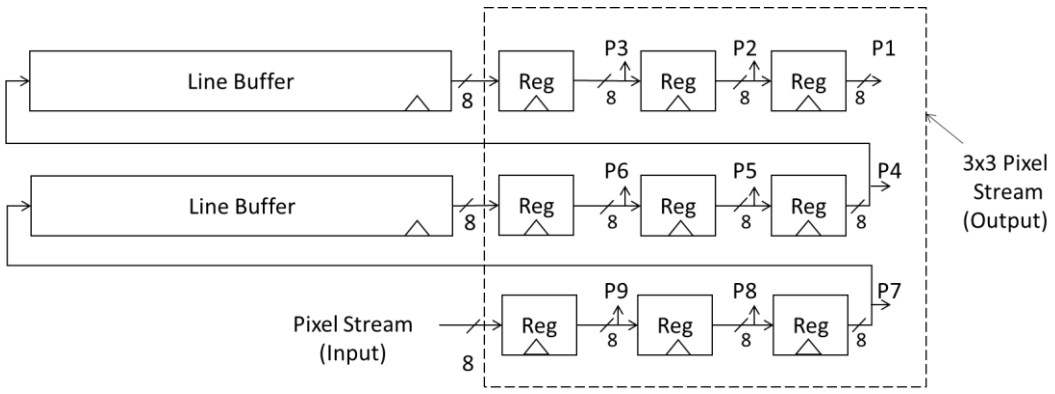


Figure 5.16. 3×3 sliding window architecture.

The architecture uses two line buffers, which are essentially two FIFO (First in, First out) register sets, and nine 8-bit registers. The length of each line buffer is equal to *W*-3 for an image width, *W*. The line buffer is implemented using a dual port block RAM (BRAM) of FPGA, where a pixel is read from and another pixel is written into the BRAM simultaneously at every clock cycle to use it as a FIFO. The read and write addresses of the BRAM are incremented at every clock cycle and they reset to zero when they reach the full length of the line buffer. The read address always advances the write address by one.

This architecture provides an initial delay of 2*W*+3 clock cycles before it gives the first valid 3×3 pixel set as output, but after that it gives 3×3 pixel set at every clock cycle.

5.2.3 Gaussian Filter

The 3×3 filter mask or kernel (*k*) for Gaussian image filtering [Davies, 2012] is shown in Equation (5.1). The Equation (5.2) performs convolution of filter kernel (*k*) with the input image (I), which gives the smoothed image (I_G). The elements of *k* are multiplied with corresponding pixels of image window to get the weighted sum as per Equation (5.3). The hardware architecture of Gaussian filter module implements the Equation (5.3) as shown in Figure 5.17, which computes a pixel *P5'* of I_G corresponding to a pixel *P5* of *I* by placing the kernel (*k*) at *P5* (center of *k* lies on *P5*). The input pixel values (P1, P2, P3… P9) change every clock cycle that are provided by sliding window architecture as discussed before in Figure 5.15. The blocks in Figure 5.17 indicate the operations that are done on the 3×3-pixel stream of the input image to get the pixel stream of the filtered output image. Apart from the adders, the architecture uses the shifters to carry out multiplication and division operations and shifting by more than one bit is obtained in a single clock cycle. The pipelined registers are introduced in the architecture to get the throughput of one output pixel per clock cycle.

$$\mathrm{k} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.1}$$

$$\mathrm{I_G} = \mathrm{k} * \mathrm{I} \tag{5.2}$$

$$\mathrm{P5'} = (\mathrm{P1} + 2 * \mathrm{P2} + \mathrm{P3} + 2 * \mathrm{P4} + 4 * \mathrm{P5} + 2 * \mathrm{P6} + \mathrm{P7} + 2 * \mathrm{P8} + \mathrm{P9})/16 \tag{5.3}$$

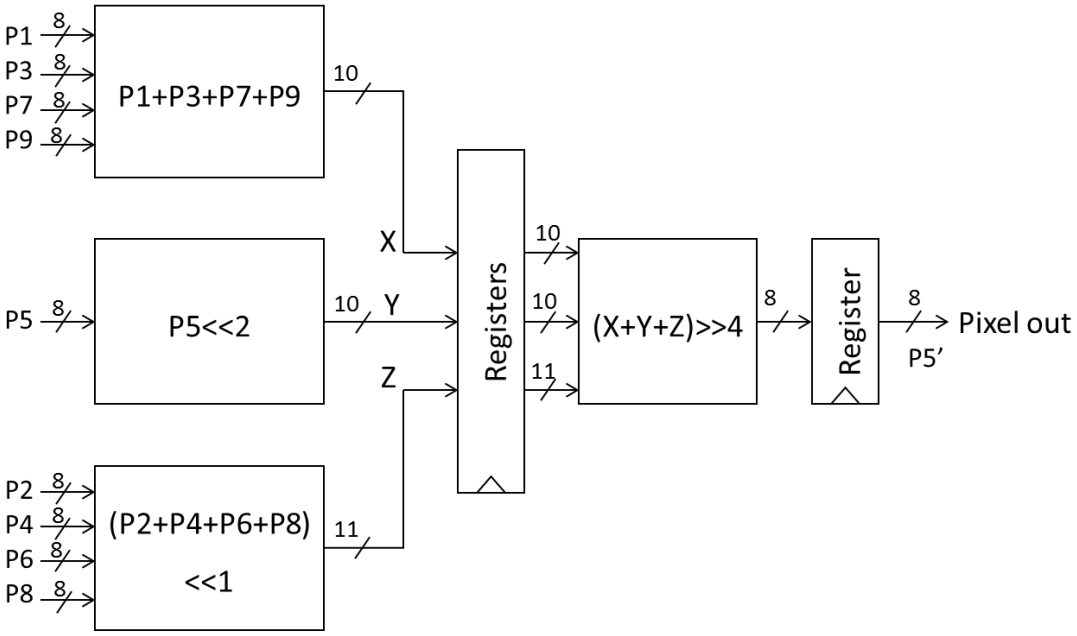


Figure 5.17. Gaussian filter architecture.

5.2.4 Sobel Edge Detector

The Sobel edge detector [Davies, 2012] utilizes two 3×3 gradient filter masks, which are convolved with the input image (I) to compute x-derivative and y-derivative of the image gradients that is G_x and G_y using Equation (5.4) and Equation (5.5) respectively.

$$\mathrm{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathrm{I} \tag{5.4}$$

$$\mathrm{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \mathrm{I} \tag{5.5}$$

The local edge strength is defined as the gradient magnitude (GM) and it is computed using Equation (5.6). It is expensive to design a hardware that will compute square and square root operations for every pixel. Thus, it is suitable to approximate the operations by absolute values as shown in Equation (5.7).

$$\mathrm{GM} = \sqrt{\mathrm{G_x^2} + \mathrm{G_y^2}} \tag{5.6}$$

$$\mathrm{GM} = |\mathrm{G_x}| + |\mathrm{G_y}| \tag{5.7}$$

$$\mathrm{G_x} = \mathrm{P3} + 2 * \mathrm{P6} + \mathrm{P9} - \mathrm{P1} - 2 * \mathrm{P4} - \mathrm{P7} \tag{5.8}$$

$$\mathrm{G_y} = \mathrm{P7} + 2 * \mathrm{P8} + \mathrm{P9} - \mathrm{P1} - 2 * \mathrm{P2} - \mathrm{P3} \tag{5.9}$$

The Sobel edge detection architecture is shown in Figure 5.18, which computes image gradient components, G_x and G_y as per Equations (5.8) and Equation (5.9). These equations are obtained using image window and filter kernels given in Equation (5.4) and Equation (5.5), where *P1, P2, P3… P9* are pixels of image window and these pixel values change as the window moves through the entire image as discussed previously in Figure 5.15. The Sobel edge detection architecture computes G_x and G_y in parallel and uses the pipelined registers for want of achieving throughput of one output pixel per clock cycle. After computing gradient magnitude using Equation (5.6), the edge pixel is chosen based on a threshold value as shown in Figure 5.18. The threshold was set at 85. The architecture involves addition, subtraction and multiplication operations and since the multiplication is costly in hardware, it is performed by left shift operation.

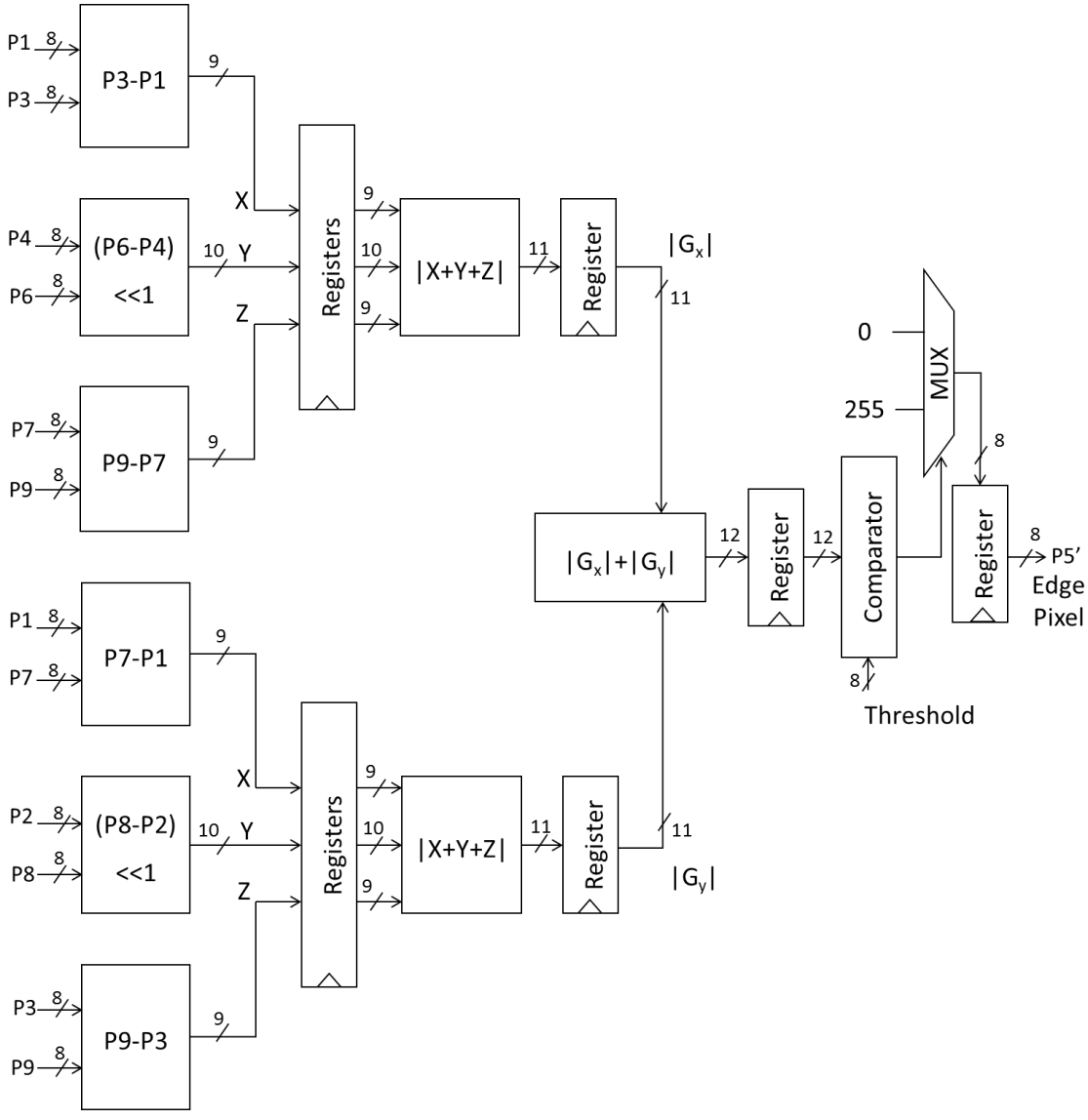


Figure 5.18. Sobel edge detection architecture.

5.2.5 Image Binarization

Image binarization also known as image thresholding is a region segmentation technique in the image based on intensity values. The global thresholding [Davies, 2012], [Gonzalez et al., 2009] with threshold value (T) is applied on the intensity image, *f(x,y)* to get the binarized-image (binary image). The binary image, *g(x,y)* is obtained using Equation (5.10).

$$g(x,y) = \begin{cases} 255 \leftarrow f(x,y) \geq T \\ 0 \quad \leftarrow f(x,y) < T \end{cases} \tag{5.10}$$

The hardware module with registered output for image binarization is shown in Figure 5.19, which is implementing Equation (5.10) with $T = 40$.

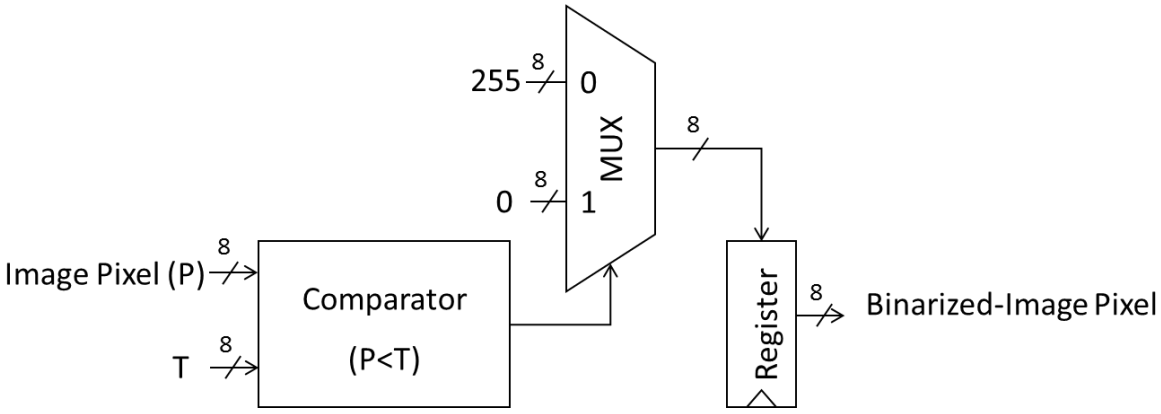


Figure 5.19. Image binarization hardware.

5.2.6 FPGA Implementation Results

The edge-map generation architecture for pupillary boundary detection (Figure 5.13) was implemented with the Verilog HDL to target Xilinx's 7 series FPGA, Zynq xc7z020-1clg484 and tested on Zedboard. Table 5.11 shows the synthesis results of the complete architecture presented in Figure 5.13. The overall frequency obtained is 276.625 MHz. The total number of block RAMs is 3, which were used to realize line buffers of sliding window architecture.

Table 5.11. Synthesis results of proposed edge-map generation hardware architecture for pupillary boundary

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	396	106400	0%
Number of Slice LUTs	462	53200	0%
Number of fully used LUT-FF pairs	315	543	58%
Number of Block RAM	3	140	2%
Maximum frequency, f_{max} = 276.625 MHz			

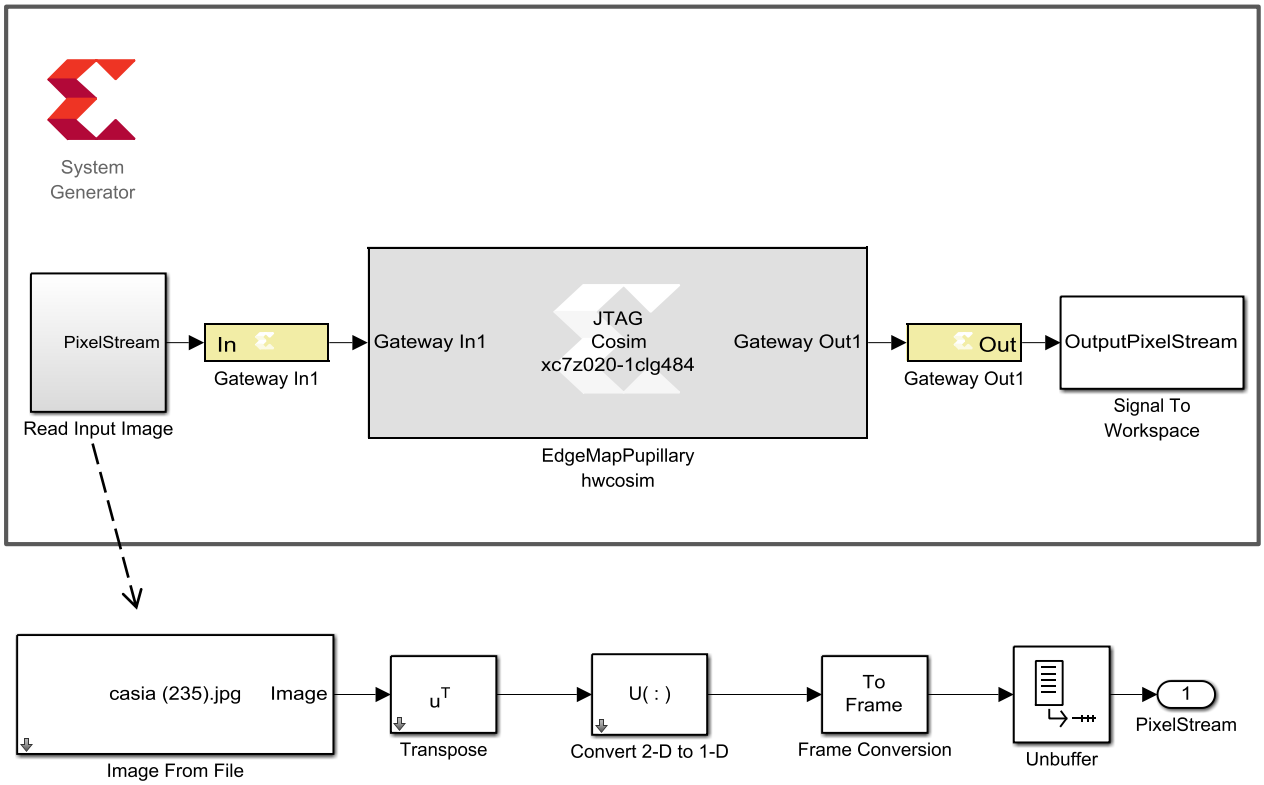


Figure 5.20. Set-up to test the edge-map generation hardware for pupillary boundary on FPGA.

5.2.7 Performance Results and Discussion

A) Accuracy Test of the Proposed Edge-Map Generation Hardware for Pupillary Boundary

For testing hardware of edge-map generation on FPGA, the Xilinx's 'system generator for DSP' tool of VIVADO v2014.4 was used. A Simulink model was created, which uses Simulink

blocksets for reading image files from computer's hard disk and saving the output image into MATLAB's workspace. A snapshot of Simulink model that was used for testing the images on ZedBoard (Zynq FPGA board) is shown in Figure 5.20.

The 'Read Input Image' is a subsystem that we built using Simulink blocks to provide input pixel stream to the hardware under test as shown in Figure 5.20. The central block in this Simulink model is Xilinx's JTAG hardware co-simulation, which contains the bitstream file (.bit) of our edge-map generation hardware for configuration of Zynq xc7z020-1clg484 FPGA on ZedBoard. The Zedboard connects to the computer using JTAG cable only. When the model runs first time, the FPGA device on ZedBoard is programmed, and then it 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 is saved into the computer as ID array of pixels and it is viewed as image using MATLAB. Now, to test next image, the new image path is specified in the model and the edge-map generation hardware in the FPGA runs again to generate edge-map of the image. This time model is run with skipping the FPGA configuration as FPGA is already programmed.

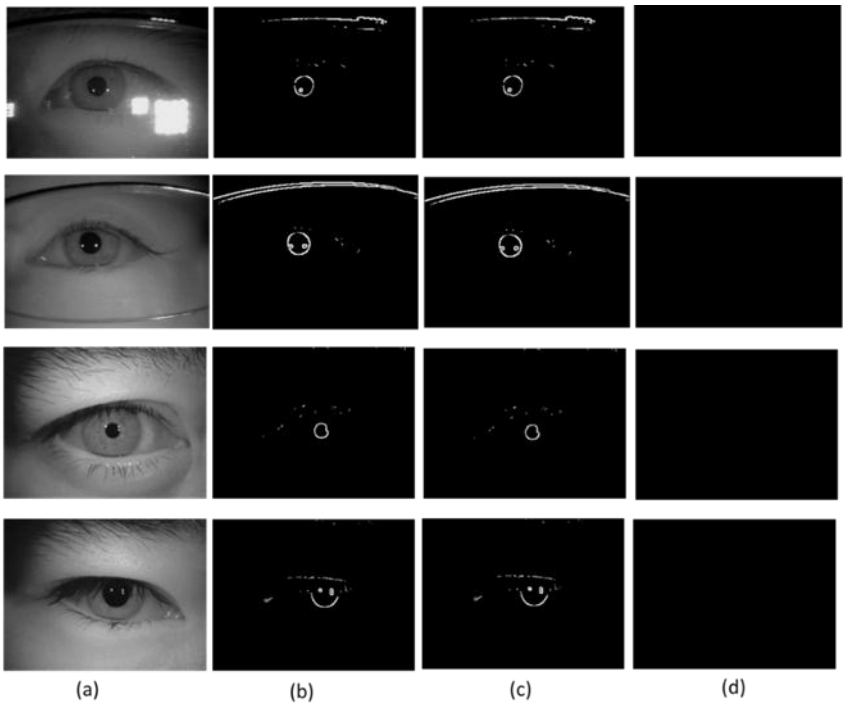


Figure 5.21. Accuracy-test of edge-map generation hardware for pupillary boundary: **(a)** Test image; **(b)** Edge-map generated using the edge-map generation hardware executing on FPGA; **(c)** Edge-map generated using equivalent MATLAB code of edge-map generation hardware; **(d)** Difference image of (b) and (c). The images (320×240) are taken from CITHV4 and CILV3 database.

The edge-maps generated by the FPGA implementation of edge-map generation for pupillary boundary detection are shown in Figure 5.21. The edge-map generation technique described in Figure 5.12 was also implemented in MATLAB, but without using built-in functions of MATLAB. Figure 5.21 shows that edge-map obtained from hardware implementation on FPGA exactly matches the edge-map obtained using MATLAB implementation. Therefore, the proposed hardware implementation is 100% accurate.

B) Processing Time of the Proposed Edge-Map Generation Hardware for Pupillary Boundary

The processing time of edge-map generation hardware was determined by counting the total clock cycles taken in completing the task of edge-map generation. The clock cycle latency of edge-map architecture and the latencies of its modules are given in Table 5.12.

Table 5.12. Clock cycle latency of the proposed edge-map generation hardware for pupillary boundary

Module	Clock cycle latency
3×3 sliding window (for 320×240 image, W=320)	643 (=2W+3)
Gaussian filter	2
Sobel edge detector	4
Image binarization	1
Complete architecture	643×2+7=1293

Table 5.13. Processing time per image of the proposed edge-map generation hardware for pupillary boundary

Image size	Number of clock cycles	Hardware simulation @200 MHz	Sequential execution CPU@2.40GHz	Speedup
320×240 pixels	1293+320×240	390.46 µsec	0.82 sec	2100

The total number of clock cycles taken to generate the edge-map of an image of size 320×240 pixels is given in Table 5.13. The maximum frequency of operation of the proposed edge-map generation hardware is 276.625 MHz, but we calculated processing time for a clock of 200 MHz because maximum operating frequency of CHT was 203 MHz. The overall operating frequency is decided by the block having minimum frequency of operation, which is CHT in this case. The processing time is 390.46 µsec, which was obtained by multiplying number of clock

cycles with clock period as shown in Table 5.13. The equivalent MATLAB code of edge-map generation hardware takes average computation time of 0.82 sec per image when runs using MATLAB (version 8.4) installed on a computer with Intel i5 CPU @ 2.40 GHz, 8 GB RAM and Windows 7 operating system. However, the edge-map generation code written in MATLAB does not use built-in function of MATLAB, such as 'edge ()' function for edge detection. We wrote the MATLAB code for image filtering operations the way it is described in pseudo code of Figure 5.15. The MATLAB's timer functions 'tic' and 'toc' were used to find the execution time of the MATLAB code. The proposed parallel and pipelined edge-map generation hardware architecture gives a 2100× speedup as compared to its equivalent MATLAB code executing sequentially using high speed CPU mentioned above.

5.3 Edge-Map Generation Hardware for Limbic Boundary

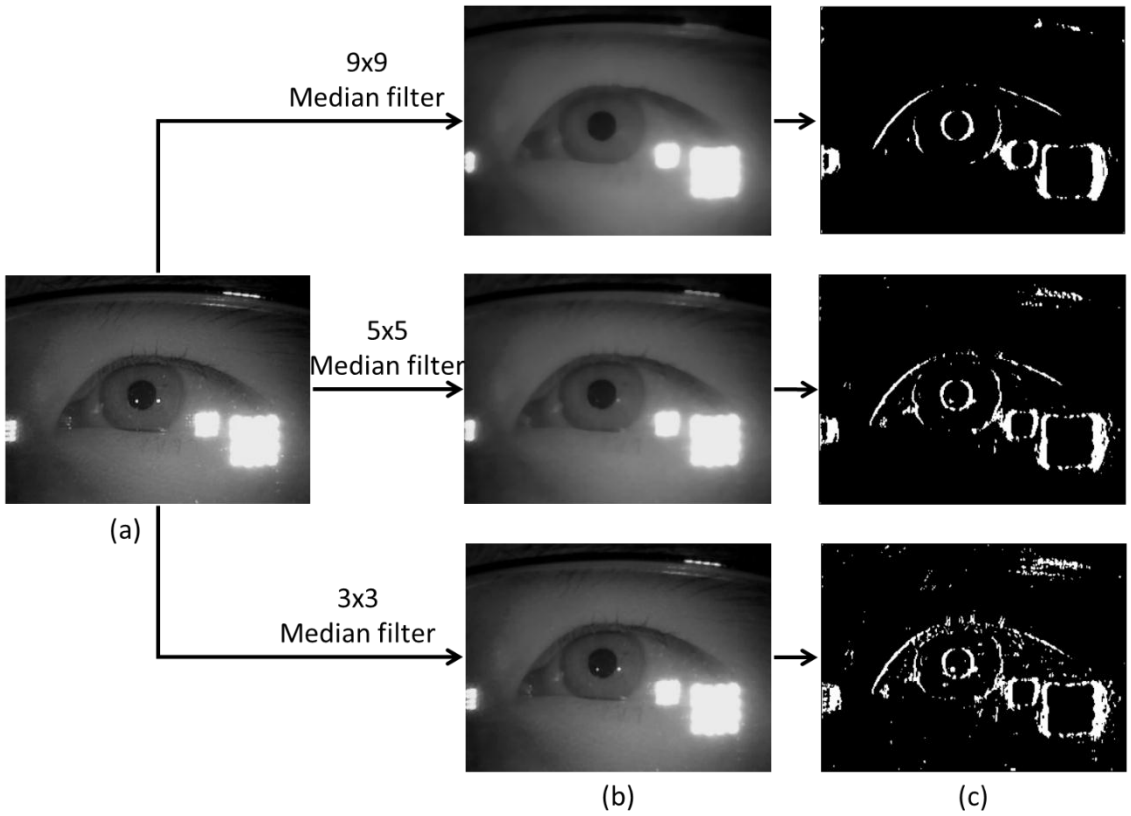


Figure 5.22. Edge-map generation for limbic boundary: **(a)** Input image; **(b)** Median filtered image; **(c)** Edge-map obtained using vertical Sobel edge detection.

The edge-map for limbic boundary is generated by applying median filter on the input image followed by Sobel edge detection tuned to detect vertical edges in the filtered image. All the

edge-points in this edge-map are not used for circle detection using CHT rather only a selected number of edge-points on left and right sides of the pupil are used, as was shown in Figure 5.9 and discussed in detail in the proposed method in chapter 4. The proposed method in chapter 4 uses 9×9 median filter, but we have chosen a 5×5 median filter for hardware implementation to save the hardware resources. However, this filter size reduction has marginal effect on circle detection accuracy as it has been demonstrated later in section 5.5. The effect of size of median filter on edge-map generation is shown in Figure 5.22. The median filter is a nonlinear filter that is used to remove noise from images while preserving edges. It is particularly effective at removing 'salt and pepper' type noise.

5.3.1 Proposed Hardware Architecture

The edge-map generation hardware architecture for limbic boundary is shown in Figure 5.23. A general n×n median filter is shown in this figure, which means that depending on input data (image), a suitable size can be chosen. The larger size of the median filter hides more the small details of the image, but without damaging the edge structure in the image. To implement n×n median filter, an n×n sliding window architecture is required to provide the inputs to the filter. The vertical Sobel edge detection is obtained using a 3×3 filter. This differs from the Sobel filter discussed before in subsection 5.2.4, in a way that it computes vertical gradients only, hence, uses a single kernel. The image is applied as a pixel stream to this architecture and it produces an output pixel stream of the edge-map.

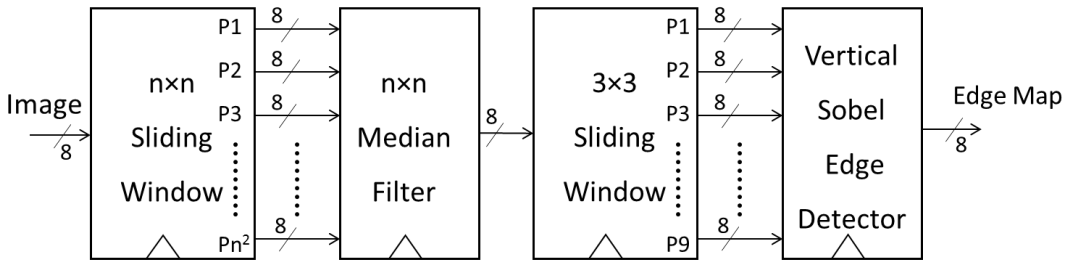


Figure 5.23. Proposed edge-map generation hardware architecture for limbic boundary.

We have implemented hardware of 5×5 median filter. The parallelism and pipelining techniques have been used in the implementation of median and Sobel filters, so that a new output pixel is obtained at every clock cycle after an initial latency. For implementing the 5×5 median filtering, a 5×5 sliding window architecture is required, which is described next.

A) 5×5 Sliding Window

The 3×3 sliding window architecture discussed in subsection 5.2.2 is extended here, to a 5×5 sliding window architecture shown in Figure 5.24 that provides a 5×5 pixel stream of the input image to the median filter core. This window buffer gives the first valid output after a delay of (4W+5) clock cycles for an image width, W and after that it gives valid output every clock cycle. It uses 4 line buffers and 25 shift registers. This module has a single input line and 25 output lines.

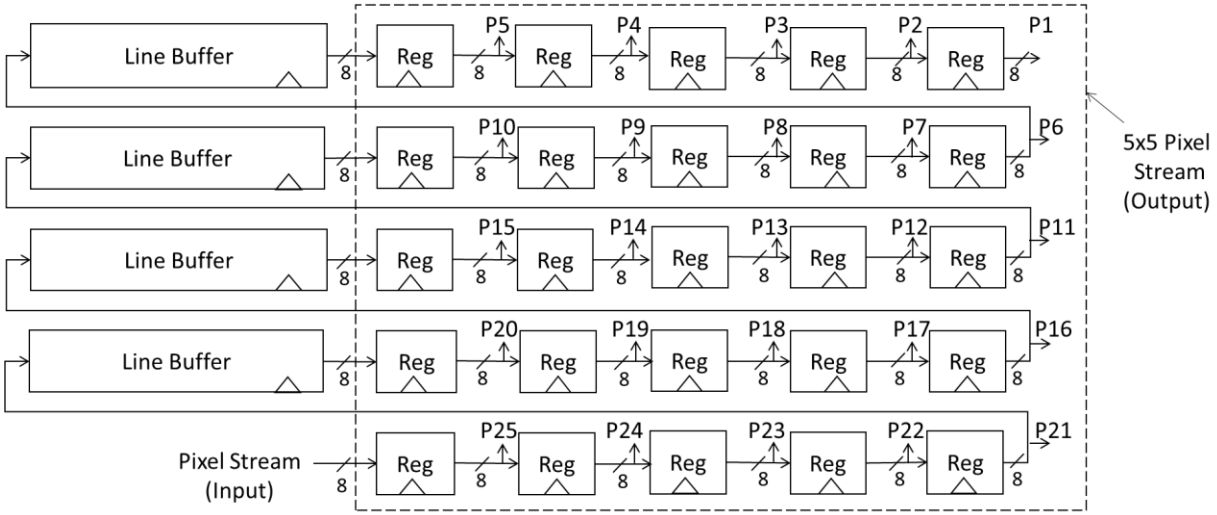


Figure 5.24. 5×5 sliding window architecture.

5.3.2 Median Filter Architecture

The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighboring pixels. The pattern of neighbors is called the 'window', which slides, pixel by pixel over the entire image. The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value.

For a 5×5 window centered at the pixel to be processed in the input image, first all the 25 pixels are arranged in ascending order and the median is taken that assigns to the processing pixel in the output image as shown in Figure 5.25. This 5×5 window then will move to the next pixel to be processed and then performs the same operation. This process continues, until all pixels of the input image are covered.

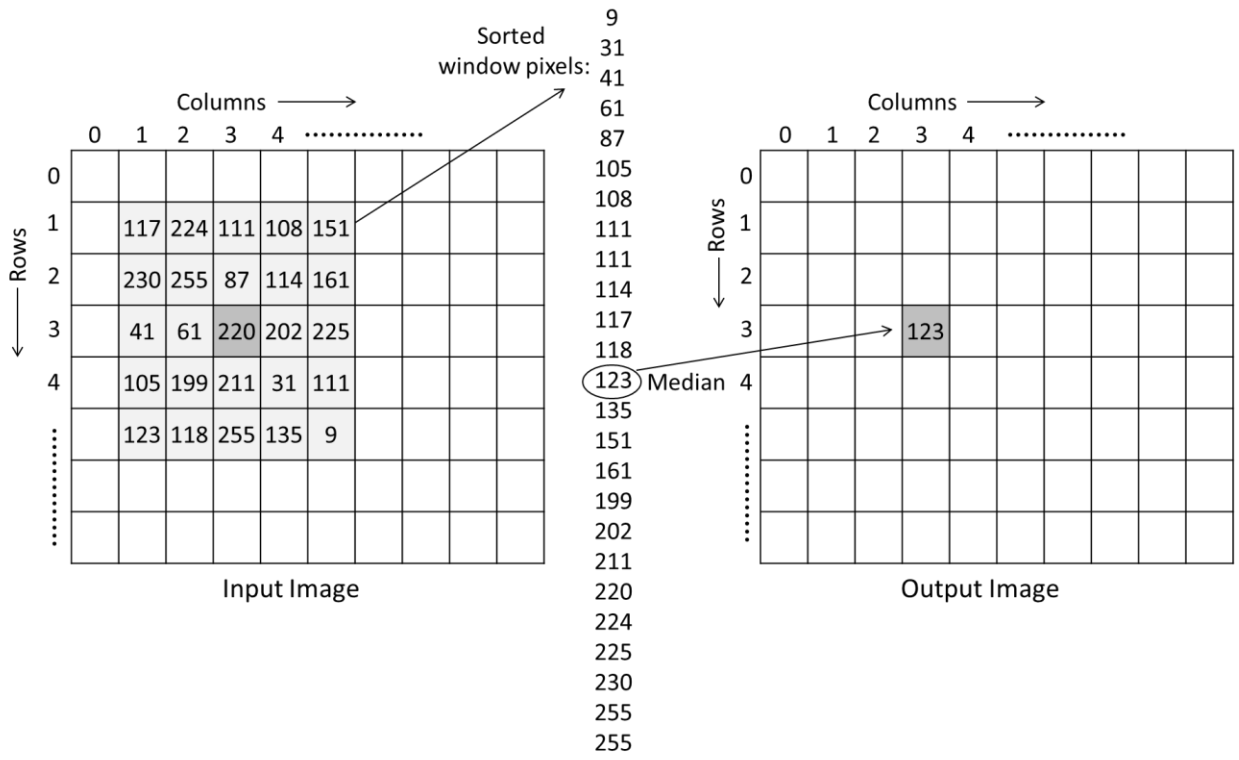


Figure 5.25. Working of 5×5 median filtering.

A) Algorithm for Finding Median

The algorithm to find median of 25 values is taken from [Kolte and Smith, 2000] and is shown in Figure 5.26. This algorithm was used for hardware implementation of 5×5 median filter. All the 25 values are arranged as a 5×5 matrix without worrying about their order (position) in the matrix. In first stage (level) of the algorithm, all the 5 rows of the matrix are sorted in ascending order and sorted values are placed in new matrix as shown in Figure 5.26 (b). In second level, all the 5 columns of the new matrix are sorted in ascending order and three [1,1] middle diagonals of the sorted matrix are used for further processing. These 3 diagonals are shown in gray in Figure 5.26 (d), which are also sorted and a single [2,1] diagonal is selected from these three [1,1] diagonals after sorting as shown in Figure 5.26(e). Finally, this [2,1] diagonal is sorted and the middle value is the median. The white cells in Figure 5.26(d) and Figure 5.26(e) are not involved in the computations.

This algorithm was chosen for hardware implementation of median filter because it involves independent sorting operations that can be carried out in parallel. For example, sorting of all rows or columns or diagonals can be done in parallel.

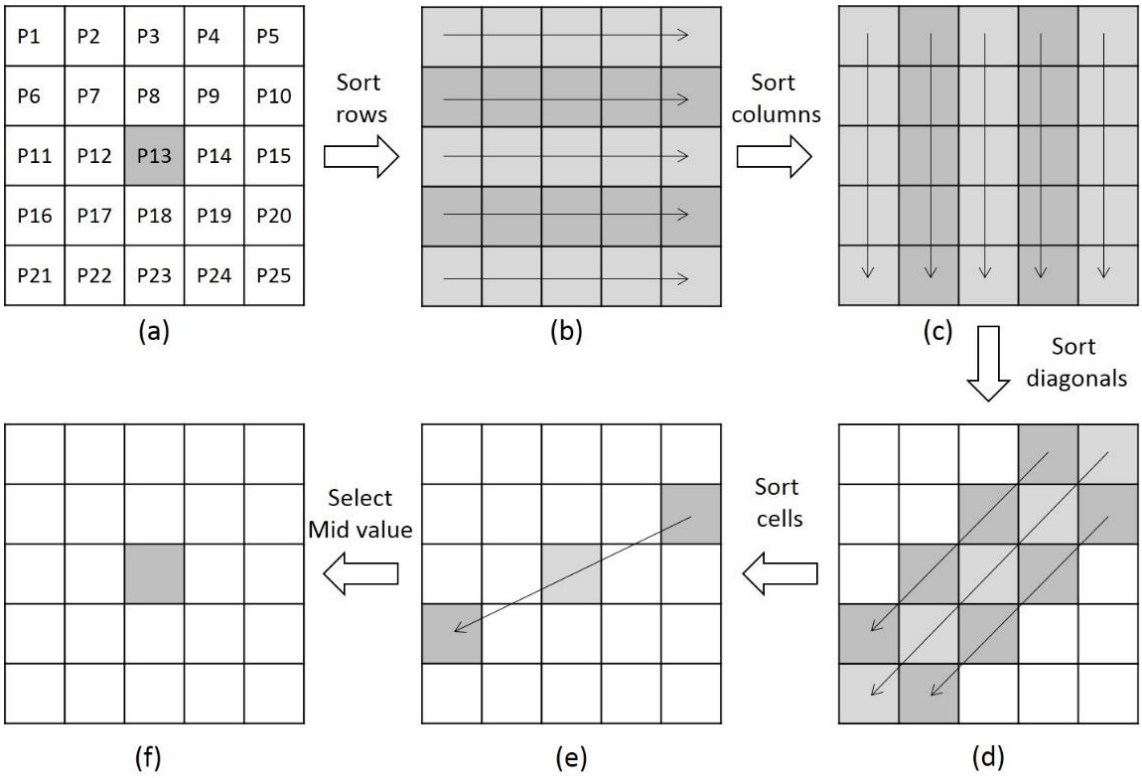


Figure 5.26. Algorithm for finding median of 5×5 values.

B) Proposed Hardware Architecture for 5×5 Median Filter

The proposed hardware architecture for 5×5 median filter is shown in Figure 5.27. This architecture implements the algorithm of Figure 5.26, which can be seen as four different stages: (1) Row sort; (2) Column sort; (3) Three [1,1] diagonal sort; (4) One [2,1] diagonal sort. In stage 1 and stage 2 of the architecture, sorting of a row or a column means sorting 5 values, whereas in stage 3, two diagonals consist of 4 values each and one diagonal contain 5 values, that require sorting of 4 and 5 values respectively. Finally, a three value sorter is used for a single diagonal in stage 4, which gives the output of the architecture. All the rows, columns, or diagonals in a stage are sorted in parallel and the values are sorted in ascending order. In order to get throughput of one pixel per clock cycle, the pipelined registers are introduced in between different stages of the architecture. This parallel and pipelined architecture makes the median filter hardware fast. The sorting in each stage completes in a single clock cycle and the sorted values are stored in the registers that act as input for next stage. The sorting of five values or four values is carried out using a three value sorter as explained in next section.

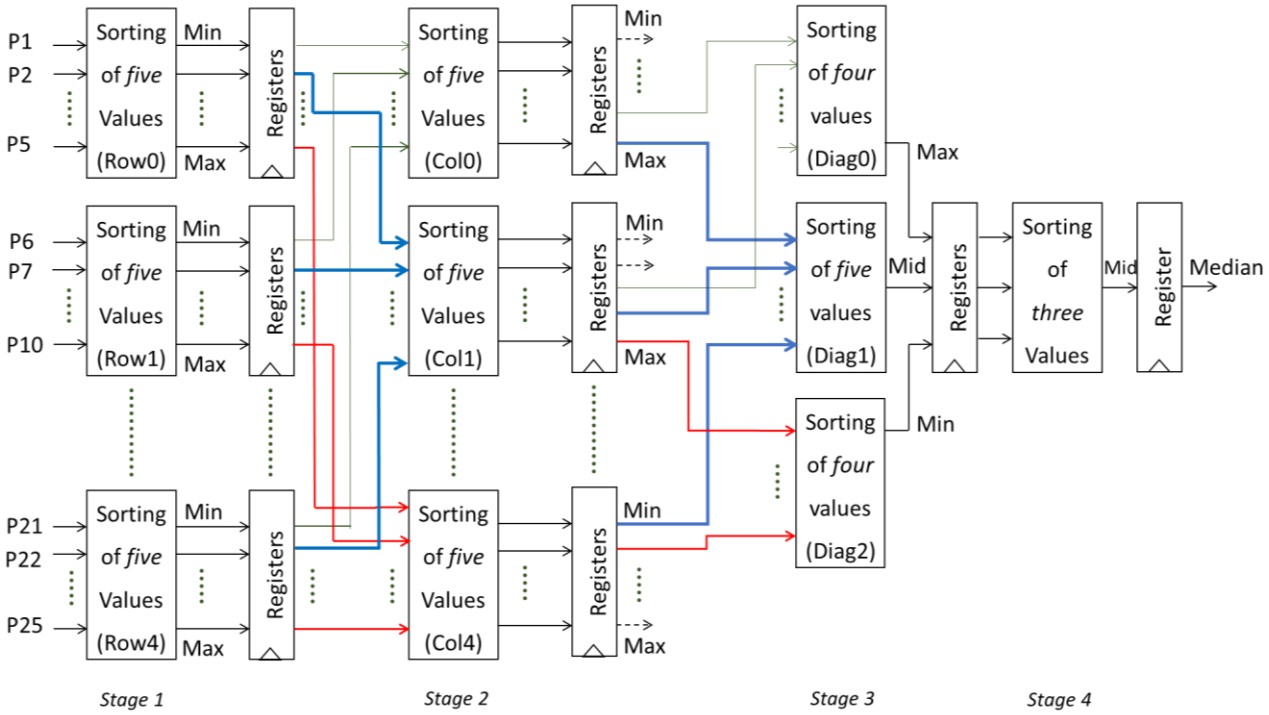


Figure 5.27. The proposed 5×5 median filter hardware architecture.

C) Sorting of Five Values

The procedure for sorting five values using sorting of three values (three-value sorter) at a time is shown in Figure 5.28. This sorting completes in four steps. For sorting *n* values using a three-value sorter, it will take *n-1* steps (iterations) to complete.

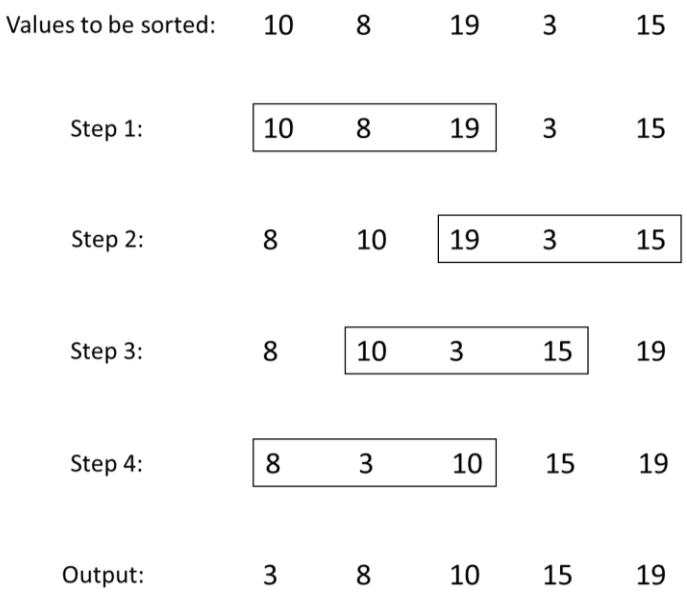


Figure 5.28. Example of sorting of five values using a three-value sorter.

The combinational circuit for sorting of five values is shown in Figure 5.29. The circuit takes five 8-bit values as input, which are a, b, c, d and e and provides these values in ascending order as v, w, x, y and z. The inputs are applied to three-value sorter in a specific order as per procedure described in Figure 5.28. The proposed hardware architecture in Figure 5.27 also uses sorting module for four values, which requires 3 three-value sorters, as sorting of four numbers takes three steps to complete as per procedure explained in Figure 5.28.

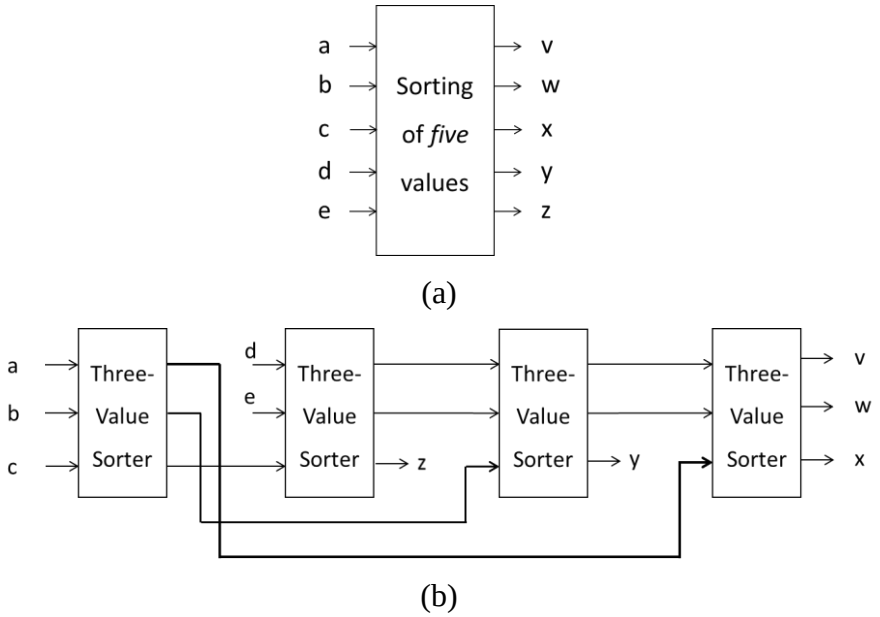


Figure 5.29. Sorting of five values: **(a)** Sorting module; **(b)** Circuit that uses three-value sorter.

D) Three-Value Sorter

The three-value sorter is fundamental component in the proposed median filter architecture and all the other sorting modules have been implemented using it. The conventional three-value sorter is shown in Figure 5.30, which is implemented using 3 two-value sorter [Vasanth et al., 2010]. This is a three level serial implementation, where output of a two-value sorter in one level is used as input for other two-value sorter in next level.

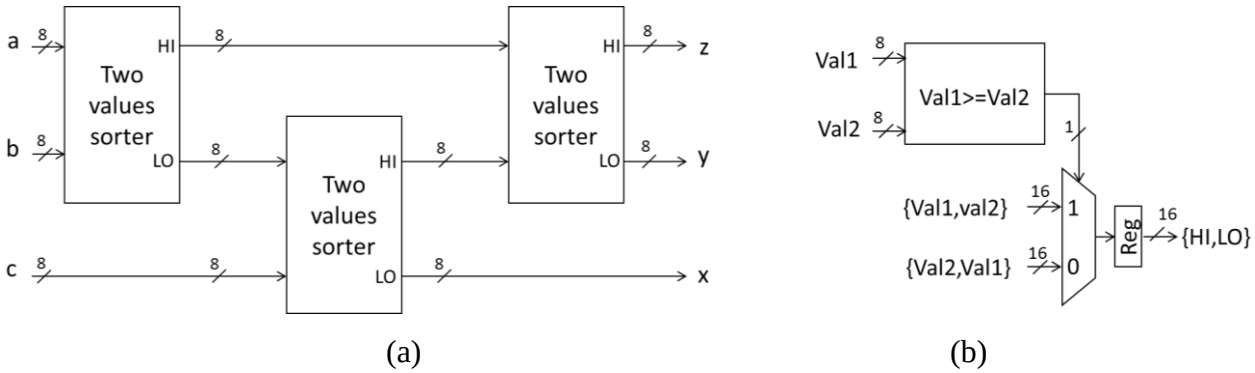


Figure 5.30. Three-value sorter: **(a)** Conventional sorter; **(b)** Two-value sorter used in (a).

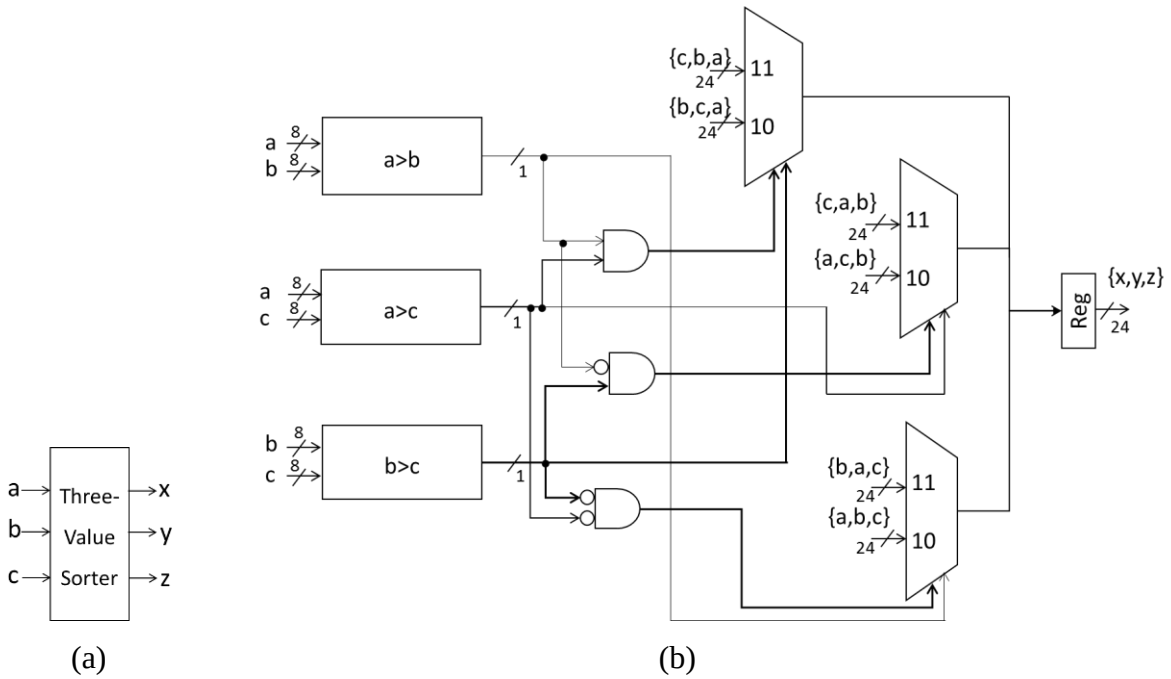


Figure 5.31. Proposed three-value sorter: **(a)** Sorting module; **(b)** Circuit diagram.

The three-value sorter is very important component in the proposed median filter architecture (Figure 5.27) as it decides the overall speed and resource utilization of the architecture. Sorting of five values and sorting of four values is achieved by cascading 4 and 3 three-value sorters respectively as shown in Figure 5.29. The speed of all the sorting modules and hence, speed of overall architecture depends on the propagation delay of this combinational circuit. The proposed three-value sorter is designed using 3 two-value comparators and 3 multiplexers. This proposed circuit shown in Figure 5.31 is faster than conventional three-value sorter and requires lesser hardware resources.

5.3.3 Vertical Sobel Edge Detector

The vertical Sobel edge detector computes only x-derivative component of image gradients to detect vertical edges in the image. It utilizes a 3×3 filter mask, which is convolved with the input image (I) to compute the x-derivative component, G_x using Equation (5.11). The local edge strength is defined as the gradient magnitude (GM) shown in Equation (5.12).

$$\mathrm{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathrm{I} \tag{5.11}$$

$$\mathrm{GM} = |\mathrm{G_x}| \tag{5.12}$$

$$\mathrm{G}_x = \mathrm{P3} + 2 * \mathrm{P6} + \mathrm{P9} - \mathrm{P1} - 2 * \mathrm{P4} - \mathrm{P7} \tag{5.13}$$

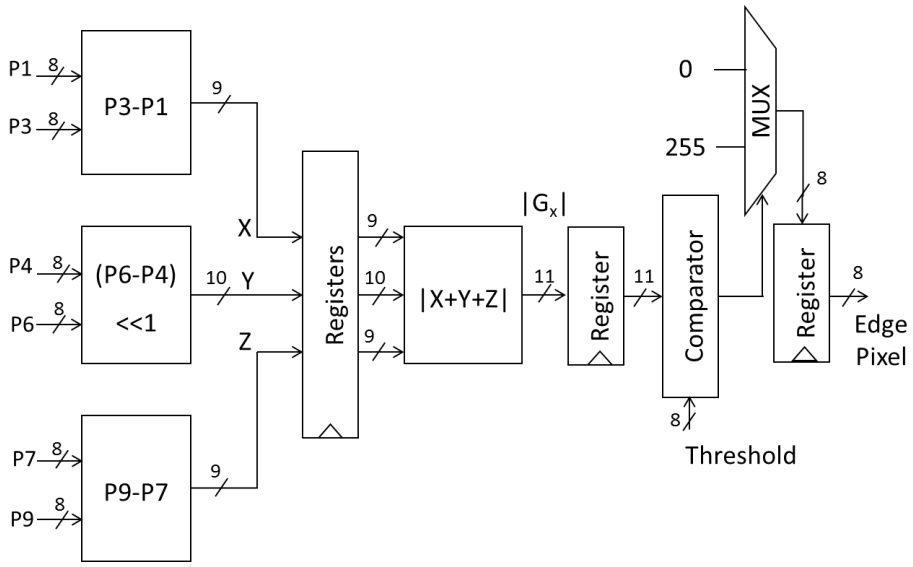


Figure 5.32. Vertical Sobel edge detector

The vertical Sobel edge detector architecture computes image gradient using Equation (5.13) and after taking its absolute value, the edge pixel is chosen based on a threshold value as shown in Figure 5.32. The threshold was set at 25.

5.3.4 FPGA Implementation Results

The edge-map creation architecture for limbic boundary detection (Figure 5.23) was implemented with the Verilog HDL to target Xilinx's 7 series FPGA, Zynq xc7z020-1clg484 and tested on Zedboard. Table 5.14 shows the synthesis results of the complete architecture given in Figure 5.23. The overall frequency obtained is 394.633 MHz. Four block RAMs were used to realize line buffers of 5×5 and 3×3 sliding window architectures.

Table 5.14. Synthesis results of the proposed edge-map generation hardware for limbic boundary using 5×5 median filter

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1681	106400	1%
Number of Slice LUTs	1841	53200	3%
Number of fully used LUT-FF pairs	1368	2154	63%
Number of Block RAM	3	140	2%
Maximum frequency, f_{max} = 394.633 MHz			

We also implemented 3×3 median filter for comparing it with the 5×5 median filter, and it was used in the edge-map generation architecture of Figure 5.23. The algorithm for finding median of 3×3 values completes in three steps as per algorithm shown in Figure 5.26. These three steps are: (1) Sorting of three rows; (2) Sorting of three columns; and finally (3) Sorting of three elements of the [1,1] diagonal [Kolte and Smith, 2000]. The mid value of these three elements is the median of 3×3 matrix. The synthesis results in Table 5.15 show that use of 3×3 median filter instead of 5×5 median filter reduces the logic utilization significantly.

Table 5.15. Synthesis results of the proposed edge-map generation hardware for limbic boundary using 3×3 median filter

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	373	106400	0%
Number of Slice LUTs	314	53200	0%
Number of fully used LUT-FF pairs	246	441	55%
Number of Block RAM/FIFO	2	140	1%
Maximum frequency, f_{max} = 394.633 MHz			

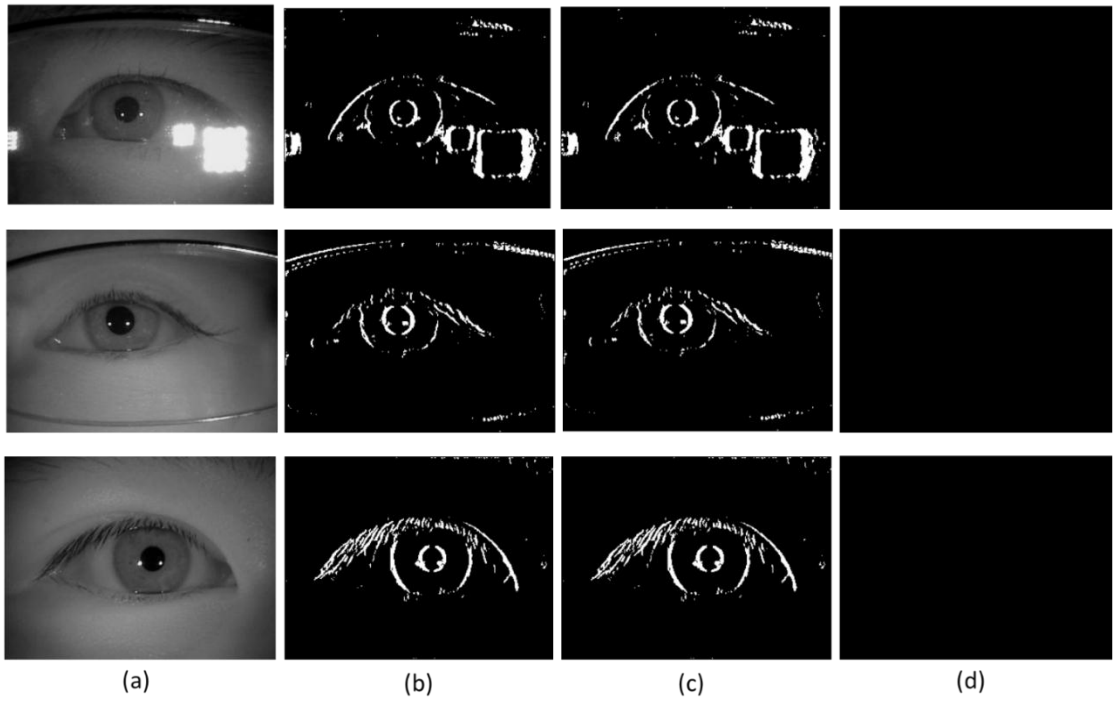


Figure 5.33. Accuracy-test of edge-map generation hardware for limbic boundary: **(a)** Test image; **(b)** Edge-map generated using the edge-map generation hardware running on FPGA; **(c)** Edge-map generated using equivalent MATLAB code of edge-map generation hardware; **(d)** Difference image of (b) and (c).

5.3.5 Performance Results and Discussion

A) Accuracy of Proposed Edge-Map Generation Hardware for Limbic Boundary

To test the accuracy of edge-map generation hardware with respect to its equivalent MATLAB implementation, the previously discussed set-up of Figure 5.20 was used. The test images in Figure 5.33 show that the edge-maps obtained using the hardware (FPGA) and software (MATLAB) implementations are 100% matching.

B) Processing Time of Proposed Edge-Map Generation Hardware for Limbic Boundary

The clock cycles latency of the proposed architecture is shown in Table 5.16 and the total number of clock cycles the architecture takes for edge-map generation of 320×240 image is shown in Table 5.17. These numbers of clock cycles are practically verified. The input was fed to the hardware under test as shown in Figure 5.20 and output was received from FPGA board (Zedboard) via JTAG cable using 'system generator for DSP' tool.

Table 5.16. Clock cycle latency of the proposed edge-map generation hardware for limbic boundary

Module	Clock cycle latency (for 320×240 image, W=320)
n×n sliding window (for n=5)	1285 (=4W+5)
n×n median filter (for n=5)	4
3×3 sliding window	643 (=2W+3)
Vertical Sobel edge detector	3
Complete architecture	1935

Table 5.17. Processing time/ image of the proposed edge-map generation hardware for limbic boundary

Image size	Number of clock cycles	Hardware simulation @200 MHz	Sequential execution using CPU@2.40GHz	Speedup
320×240 pixels	1935+320×240	393.675 µsec	5.53 sec	14047

The proposed architecture gives a processing time (obtained by multiplying no. of clock cycles and clock period) of 393.675 µsec using a clock of 200 MHz as shown in Table 5.17. However, the maximum frequency of operation of the proposed architecture is 394.633 MHz.

The performance is evaluated at 200 MHz to keep minimum operating speed for all the blocks. The equivalent MATLAB code of edge-map generation hardware for limbic boundary detection takes execution time of 5.53 sec when allowed to run sequentially using MATLAB (version 8.4) on a computer with Intel i5 CPU @ 2.40 GHz, 8 GB RAM and Windows 7 operating system. This code does not use built-in MATLAB functions and the code for median filter was written as per algorithm described in subsection 5.3.2 that was used for hardware implementation of median filter. We did not use MATLAB function 'sort ( )' for sorting of values in the median filter algorithm, rather we coded a three-value sorter and it was used to sort the higher number of values like we did in the hardware implementation of median filter. The proposed edge-map generation hardware architecture gives a huge 14047× speedup as compared to its equivalent MATLAB code execution on a PC platform mentioned above.

Table 5.18. 5×5 window median filter architectures

Architecture	Clock cycle latency	Maximum frequency of operation (f_{max}) in MHz	FPGA Resources (%)
[Prokin and Prokin, 2010]	8	318	<1 (Altera Stratix II)
Batcher sort*	15	215	2 (Altera Stratix II)
Systolic sort*	27	310	4.2 (Altera Stratix II)
Radix sort*	30	280	2.4 (Altera Stratix II)
Merge sort*	70	140	11 (Altera Stratix II)
Proposed	**4**	**394.633**	**(Xilinx Zynq) Slice registers=1 Slice LUTs =3**

*Results are taken from [Prokin and Prokin, 2010].

5.3.6 Comparison of Median Filter Architectures

The proposed 5×5 median filter architecture (Figure 5.27) provides least latency for computing median of 25 values as compared to the existing architectures [Prokin and Prokin, 2010]. The proposed architecture has a clock cycle latency of 4, whereas the other architectures based on Batcher sort [Batcher, 1968], systolic sort [Thompson, 1983], radix sort [Danielsson, 1981] and merge sort [Knuth, 1998] have clock cycle latency of 15, 27, 30 and 70 respectively (Table 5.18). These architectures are described in [Scott et al., 2008]. The most recent architecture proposed by [Prokin and Prokin, 2010] has a latency of 8 clock cycles, which is still double of the proposed architecture. The maximum frequency of operation of the proposed architecture is 394.633 MHz

as discussed later, which is higher than all the above-mentioned architectures. The proposed architecture achieves this improvement in the latency at the cost of increased hardware resources as compared to [Prokin and Prokin, 2010]. However, our 3×3 median filter implemented using same method is more hardware efficient.

5.4 Adaptive CHT for Limbic Boundary Detection

CHT hardware discussed in section 5.1 can be used as adaptive CHT hardware for limbic boundary detection. While using the CHT hardware of Figure 5.4 as adaptive CHT, the circle point generator unit will have to read 180 values from each look-up table of $\sin\theta$ and $\cos\theta$ instead of all the 360 values stored in the look-up tables, because the adaptive CHT detects a structure of two circular arcs defined by -45:45 and 135:225 degree as was discussed in chapter 4.

Table 5.19. Difference between the method used for iris localization hardware implementation and the proposed method described in chapter 4

Stages	Proposed method described in chapter 4	Iris localization hardware method
Edge-map generation for pupillary boundary detection	• 5×5 Gaussian filtering • Thresholding • Image opening and hole filling • Sobel edge detection without thinning • Intersection of images	• 3×3 Gaussian filtering • Thresholding • Sobel edge detection without thinning • Intersection of images
Edge-map generation for limbic boundary detection	• 9×9 median filtering • Vertical Sobel edge detection without thinning	• 5×5 median filtering • Vertical Sobel edge detection without thinning
CHT	• General CHT for pupillary circle detection • Adaptive CHT for limbic circle detection	• CHT (m=2, n=4) for pupillary circle detection • CHT (m=2, n=4) or adaptive CHT (m=2,n=4) for limbic circle detection

5.5 Accuracy Evaluation of Iris Localization Hardware

The iris localization hardware would consist of the edge-map generation hardware (for pupillary and limbic boundary) and CHT hardware. The edge-map generation hardware modules for

pupillary and limbic boundary detection can run in parallel, but CHT hardware has to run twice; first to detect pupillary boundary and then limbic boundary, as limbic boundary detection requires pupil circle parameters as input as shown in Figure 5.9. As it has been discussed in the previous sections that the CHT technique and the edge-map generation techniques for pupillary and limbic boundary detection have been modified (or optimized) in favor of their hardware implementation and they are not exactly same as the techniques described in the proposed method in previous chapter. Optimization for hardware implementation means, for example, CHT was modified by reducing its accumulator voting space in order to achieve memory reduction and median filter size was reduced to save hardware resources. However, these optimizations have resulted in marginal loss of accuracy. The differences between the proposed method of chapter 4 and its new version that was implemented in hardware in this chapter are show in Table 5.19. These modified techniques were evaluated to know their impact on iris localization accuracy. The evaluation was done using MATLAB software, because the edge-maps obtained using edge-map generation hardware are exactly same as the edge-maps obtained using the equivalent MATLAB code for both pupillary and limbic boundary detection as discussed in the previous sections.

5.5.1 Datasets Used

The following two datasets were used in the experiments to find accuracy of the iris localization hardware method:

1. CITHV4: First 1000 images of this database were taken.
2. CASIA-iris-lamp, version 3.0 (CILV3): A total 811 images were chosen from this database selecting first left and first right eye image of each subject.

Table 5.20. Accuracy results of iris localization methods

S.No.	Method	Accuracy (%)	
		CITHV4	CILV3
1	Proposed method described in chapter 4	99.7	99.38
2	**Iris localization hardware method with CHT (m=2, n=4)**	**98.7**	**96.9**
3	**Iris localization hardware method with CHT (m=2, n=4) for pupillary boundary and adaptive CHT (m=2, n=4) for limbic boundary detection**	**98.7**	**97.4**

The accuracy results are shown in Table 5.20, which shows that the accuracy of iris localization hardware reduces a bit as compared to the proposed method described in chapter 4. This reduction in accuracy is the impact of modifying CHT and edge-map generation techniques for hardware implementation of iris localization. The Table 5.20 shows that use of adaptive CHT for limbic boundary detection improves the accuracy of iris localization hardware for CILV3 database.

5.6 Concluding Remarks

A) CHT Architecture

The proposed CHT architecture can provide a large memory reduction but with little or no accuracy degradation; therefore, depending on the application, the priority can be given to either accuracy or memory resources usage. The proposed CHT hardware architecture gives a 250× speedup as compared to its equivalent MATLAB code execution on today's high speed general purpose CPUs. Hence, the proposed CHT can be used as a hardware accelerator for 'iris localization' in the iris recognition' systems to get real-time performance in the embedded applications where low speed CPUs are used and the resources are limited. The comparison with previous [Ngo et al., 2014] work on FPGA based iris-boundary detection shows that the proposed architecture is not only more accurate, but also utilizes less FPGA's embedded memory (block RAMs). Moreover, the previous work can detect outer iris-boundary only, whereas the proposed CHT hardware detects both inner and outer iris-boundaries.

B) Edge-Map Generation Hardware for Pupillary Boundary

The technique used for edge-map generation reduces the false edge-points significantly in the final edge-map due to logical AND operation on two different edge-maps. The two different edge-maps are generated in parallel and are combined into a single (final) edge-map simultaneously, which makes the proposed hardware architecture faster. The complete edge-map generation hardware is made up of sliding window, Gaussian filter, Sobel edge detection and image binarization modules. This is a parallel and pipelined architecture, which produces the output image pixels at a rate of one pixel per clock cycle while the input image pixels are also coming at the rate of one pixel per clock cycle. The edge-map generation hardware takes 390.465 µsec to generate the final edge-map for an input image of size 320×240 pixels. The edge-map

generation hardware for pupillary boundary gives a 2100× speedup as compared to its equivalent MATLAB code execution on today's high speed general purpose CPUs. The edge-map generation hardware is the resource efficient FPGA implementation as it takes negligible slices (almost 0% resource utilization) of Xilinx's Zynq FPGA device. The edge-map generated using the proposed edge-map generation hardware is exactly same as the edge-map generated by its equivalent MATLAB implementation with or without using built-in functions of MATLAB.

C) Edge-Map Generation Hardware for Limbic Boundary

The edge-map is generated using median filtering followed by vertical Sobel edge detection. The edge-map generation hardware consists of a 5×5 median filter and vertical Sobel edge detection modules. The 5×5 and 3×3 sliding window architectures were implemented for median and Sobel edge detection filters respectively. This is a parallel and pipelined architecture, which produces the output image pixels at a rate of one pixel per clock cycle while the input image pixels are also coming at the rate of one pixel per clock cycle. The edge-map generation hardware takes 393.675 µsec to generate the final edge-map for an input image of size 320×240 pixels. This gives a huge 14047× speedup for edge-map generation as compared to its equivalent MATLAB code execution on today's high speed general purpose CPUs. The edge-map generation hardware is the resource efficient FPGA implementation as it takes negligible slices (almost 4% resource utilization) of Xilinx's Zynq FPGA device. The edge-maps of iris image generated using the proposed edge-map generation hardware and its equivalent MATLAB implementation are exactly same.

The 5×5 median filter is the main module of the edge-map generation hardware for limbic boundary. The proposed median filter architecture has clock cycle latency of 4, which is much better than the previous median filter architectures in the literature. The maximum frequency of operation is also improved as compared to previous architectures. We also implemented 3×3 median filter, which has a clock cycle latency of 3.

Chapter 6
Preliminary Work towards Hardware Implementation of Iris Localization for VW Images

The previous chapters have discussed about iris localization algorithms for near infrared (NIR) images and the hardware implementation of a CHT based method. The hardware implementation discusses the implementation of individual stages of iris localization that is circular Hough transform (CHT) and edge-map generation techniques for pupillary and limbic boundaries on field programmable gate array (FPGA). The previous chapters do not discuss iris localization for visible wavelength (VW) images, which is also important as iris recognition with VW images is also becoming popular nowadays as discussed in subsection 2.1.3. In this chapter, we propose an accurate iris localization algorithm for VW images and characteristic of this algorithm is that it is suitable for hardware implementation. The proposed algorithm uses the same CHT that was used for NIR images, but difference lies in the image preprocessing techniques. This proposed algorithm for VW images is described in this chapter and its hardware implementation will require mostly the hardware modules that have already been discussed in chapter 5 for NIR images.

6.1 Proposed Iris Localization Method for VW Images

The proposed method is based on CHT. The CHT based method is chosen for VW images as it is more suitable for parallel implementation on FPGA as discussed in the previous chapter. Moreover, we already have developed CHT and edge detection hardware modules in the previous chapter for NIR images, which can be used for VW image also. In the iris localization methods of VW images, the limbic boundary is detected prior to the pupillary boundary, unlike the iris localization methods of NIR images, because the gray difference between iris and sclera is much

higher than the gray difference between pupil and iris in the VW images [Radman et al., 2012], [Proenca, 2010]. The proposed algorithm uses the same CHT that was used for NIR images, but difference lies in the edge-map generation techniques.

6.1.1 Limbic Boundary Detection

The process of limbic boundary detection is illustrated in Figure 6.1. The edge-map of iris image is generated using median filtering and Sobel edge detection. The image preprocessing using the median filter is done to remove the eyelashes and eyebrow hair threads, which are rapid gray-level variations in the image. The median filter [Gonzalez et al., 2009] also smoothens the image by reducing the minute details in the image, but preserves the edge information in the image, which reduces the false edges in the edge-map of the image. The size of the median filter kernel taken is [5×5], which was determined experimentally.

Next, the Sobel edge detection is applied on the filtered image to find the vertical edges as the horizontal iris edges may not be visible in the image due to occlusions by eyelids and eyelashes. The Sobel kernel used for vertical edge detection is given in [Gonzalez et al., 2009], which computes image gradient magnitude using x-derivative component only. For Sobel edge detection method, a threshold value is used to threshold the computed gradient magnitude for an edge pixel. The initial value of threshold was determined by 'edge ( )' function of MATLAB, which was then adjusted by experiments on database images. The higher threshold gives a fewer edge pixels and lower threshold gives more edge pixels. The threshold value was set at 0.035 in the 'edge ( )' function.

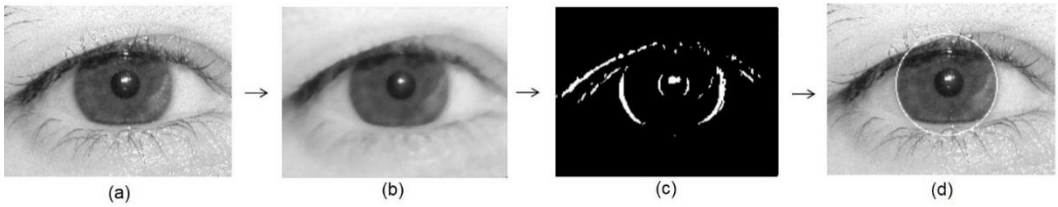


Figure 6.1. Limbic boundary detection for VW images: **(a)** Original Image; **(b)** Smoothed image of (a) obtained using 5×5 median filter; **(c)** Edge-map for limbic boundary detection, obtained by applying vertical Sobel edge detector on (b) without thinning operation; **(d)** Limbic boundary detected image obtained on applying adaptive CHT on (c).

The adaptive CHT is applied to detect limbic boundary in the edge-map, which essentially looks for a set of two vertical arcs in the edge-map instead of a full circle as discussed previously in chapter 4.

6.1.2 Pupillary Boundary Detection

The center of limbic boundary circle is used for detecting the pupil boundary. Using the circle center, a subimage of W×W pixels is extracted from the iris image, where W is diameter of biggest pupil in the image database chosen by manual inspection of the database images. This subimage centered at the limbic boundary center in the image, is used for the pupillary boundary detection as shown in Figure 6.2. The process of pupillary boundary detection involves an image preprocessing step of reflection removal in the subimage. The light source reflections in the pupil region causes problem in pupillary boundary detection because they appear as thick and strong white edges in the edge-map obtained after applying Sobel edge detection without thinning operation. The reflection has to be removed before the edge detection step as shown in Figure 6.2.

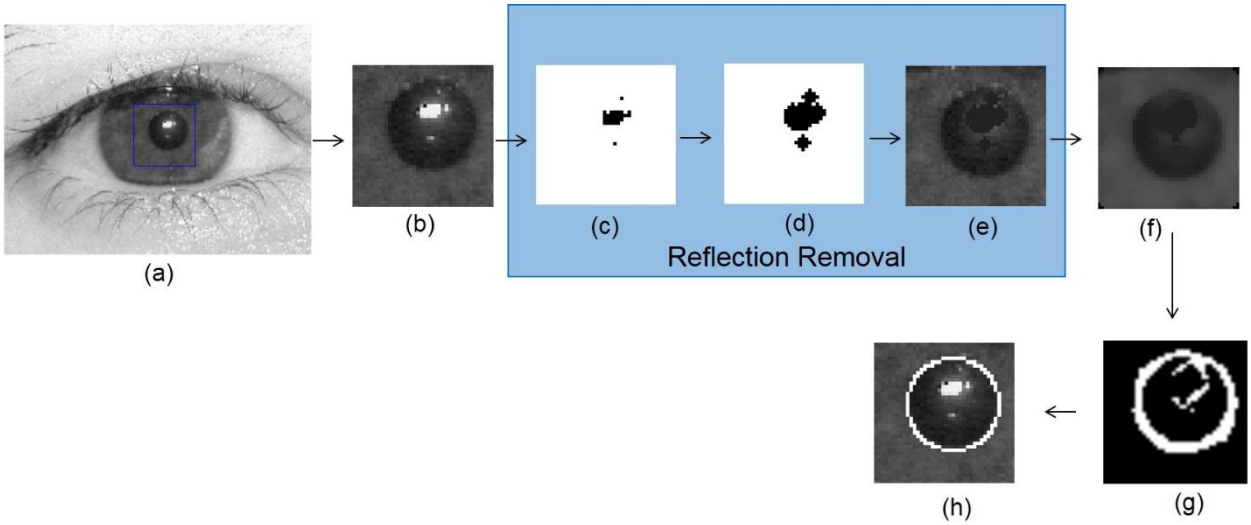


Figure 6.2. Pupillary boundary detection for VW images: **(a)** Original image after limbic boundary detection. The rectangle in blue contains the subimage to be processed for pupil boundary detection; **(b)** The subimage extracted from (a); **(c)** Reflection map of subimage; **(d)** Enhanced-reflection map, obtained using dilation operation on (c) with circular structuring element of radius two; **(e)** Reflections free subimage; **(f)** Smoothed image after median filtering on (e); **(g)** Edge-map, obtained by applying both horizontal and vertical Sobel edge detection on (f); **(h)** Pupillary boundary detected image obtained after applying CHT on (g).

The reflections are first localized using thresholding operation on grayscale images [Gonzalez et al., 2009]. The intensity threshold value used in the operation is 130, which was chosen based on the histogram of a few images and value once chosen after experiments remains

same for whole database images. The localized reflection pixels are shown as black pixels in the binary image. The obtained reflection region (map) in the binary image is enhanced through morphological dilation operation for black objects. The dilation operation increases the areas of localized reflections, so that the reflection map contains the reflections together with glows around them. The glows are formed by pixels with intensities intermediate between the high reflection intensities and the intensities of surrounding pixels not affected by the reflection. The dilation operation uses 'disk' type structuring element of radius two. The coordinates of reflections in the enhanced reflection map (Figure 6.2(d)) are found and the pixel values at these coordinates in the original intensity subimage are replaced by a low pixel value of I_L. The intensity value, I_L should be close to the average intensity of the pupil area not affected by the reflections, which we chose heuristically using the histograms of a number of subimages. The intensity value, I_L chosen is 25, which was kept same for all the database images.

After reflection removal, the image is passed through a median filter of window size [5×5] to smooth the image as shown in Figure 6.2(f). The resulting image is called the preprocessed image that is used as input for edge detection step. Both vertical and horizontal Sobel edge detection is used to find the pupil boundary edges, as the pupil is mostly available as full circle in the images. The threshold value used in the detection remains constant for the whole database images. Next, CHT is used to detect the pupil circle in the edge-map of the subimage. The CHT implementation is same as discussed in chapter 4.

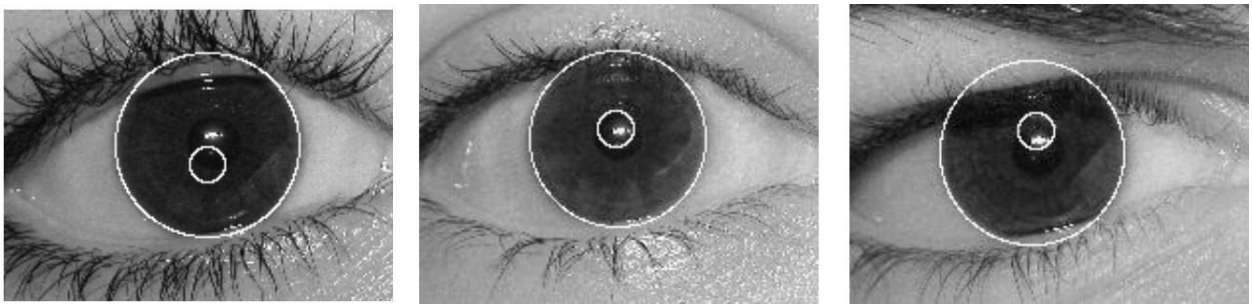


Figure 6.3. Images with wrong pupillary boundary detection.

Figure 6.3 shows a few images in which the limbic boundary was detected correctly, but the pupil boundary wrongly. This has happened because of the low intensity difference between the pupil and iris regions, the edge detector fails to detect the pupil edges. To resolve this issue and improve the accuracy of pupil boundary detection, we used an extra step of image contrast enhancement before the reflection removal step as shown in Figure 6.4. Path 1 in Figure 6.4 shows how the pupil localization fails in the images with less gray difference between pupil and

iris and Path 2 shows how it was corrected. To enhance the contrast of the image, the MATLAB function, 'imadjust (I)' was used, which maps the intensity values in grayscale image *I* to new values in *J* such that 1% of data is saturated at low and high intensities of *I*. This increases the contrast of the output image *J*.

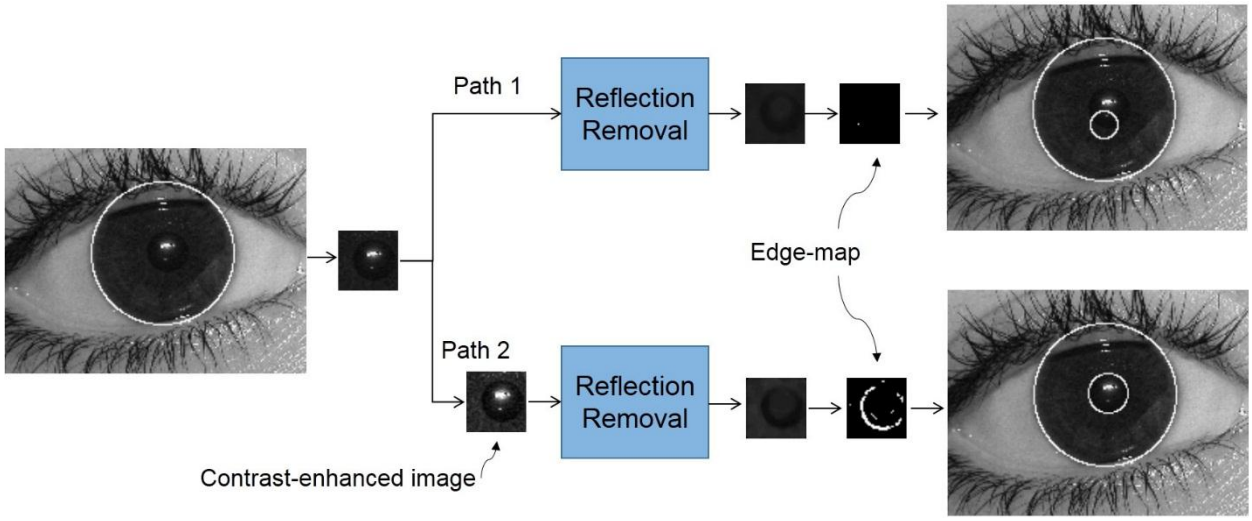


Figure 6.4. Pupillary boundary detection corrected. Path 1 results in wrong detection of pupillary boundary; Path 2 results in correct detection of pupillary boundary by introducing a contrast-enhanced image.

6.1.3 Performance Evaluation

The datasets used for evaluating the proposed method were taken from UBIRIS.v1 database [Proença and Alexandre, 2005] that contains the visible spectral range iris images. This database contains the images that were collected in two separate sessions. The first session contains 1214 images, whereas the second session contains 662 images. We did experimentations with all the images of session-1, which contains less unconstrained images as compared to session-2. However, the session-1 images contain non-ideal issues such as reflection, low contrast between pupil and iris, poor focus and occlusion by eyelids and eyelashes. The example images of the correct iris localization are shown in Figure 6.5 and Figure 6.6 by the proposed method along with the edge-maps for limbic and pupillary boundary detection. The example images with wrong iris localization by the proposed method are shown in Figure 6.7.

The experiments on the datasets were done using a computer with Intel i5 CPU @ 2.40 GHz, 8 GB RAM and Windows 7 operating system. The proposed algorithm is implemented and tested with MATLAB (version 8.4) tool.

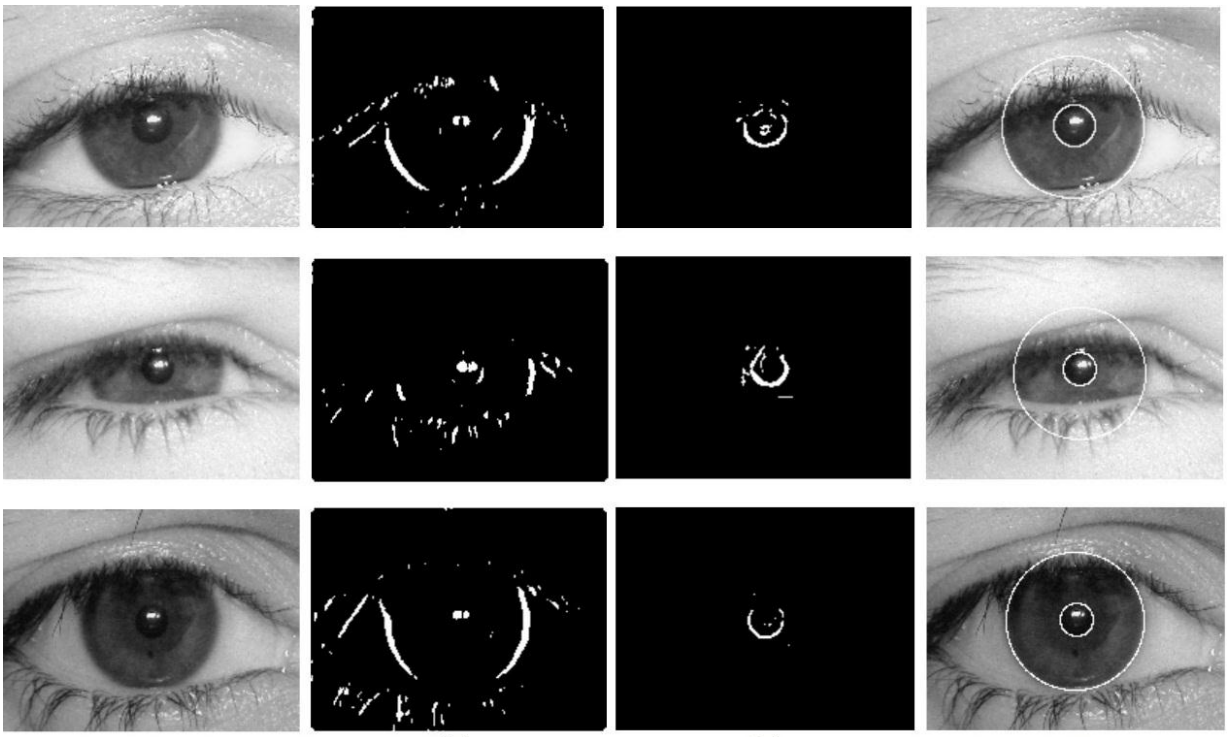


Figure 6.5. Iris localization in 200×150 pixel images from UBIRIS.v1: **(a)** Original image; **(b)** Edge-map for limbic boundary detection; **(c)** Edge-map for pupillary boundary detection; **(d)** Iris localized image.

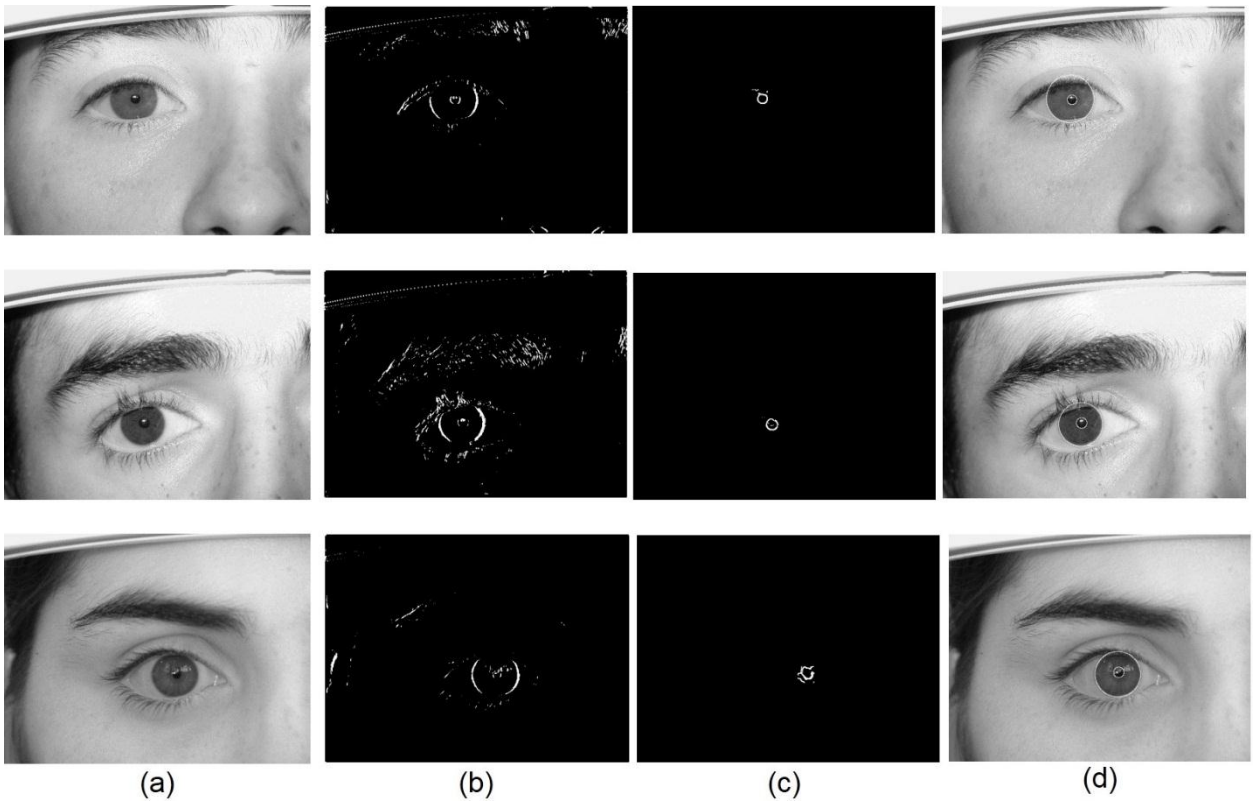


Figure 6.6. Iris localization in 640×480 pixel images from UBIRIS.v1: **(a)** Original image; **(b)** Edge-map for limbic boundary detection; **(c)** Edge-map for pupillary boundary detection; **(d)** Iris localized image.

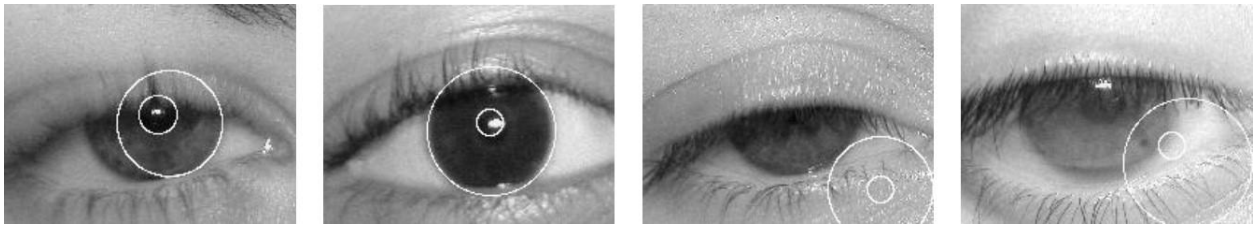


Figure 6.7. Images with wrong iris localization by the proposed method.

For limbic boundary detection, the proposed method uses vertical (V) Sobel edge detection plus adaptive CHT after having preprocessed the image with median filter; but we also tried other alternatives before reaching to the proposed method. For example, we observed a methodology that uses both vertical and horizontal (H) Sobel edge detection for edge-map generation; and uses CHT for circle detection in the edge-map. This method resulted in more computation time and degradation in the accuracy of limbic boundary detection. The reasons for this reduced performance are more number of false edges and full circle lengths drawn in the CHT. The different combinations of edge detection and CHT techniques for limbic boundary detection are mentioned in Table 6.1. The input image to all these combinations (methods) is a preprocessed image obtained after applying median filter on the original image. The Table 6.1 shows that the 4th method (the proposed method) in the table gives the best accuracy and time performance.

Table 6.1. Variants of edge detection plus CHT based iris localization

Method*	Limbic boundary detection		Pupillary boundary detection		Iris localization results	
	Sobel edge detection	Circle detection	Sobel edge detection	Circle detection	Accuracy (%)	Av. time per image (sec)
1	V+H	CHT	V+H	CHT	95	2.8
2	V	CHT	V+H	CHT	97.5	2.1
3	V	Adaptive CHT	V+H	CHT	98.4	0.83
4	**V**	**Adaptive CHT**	**Contrast enhancement plus (V+H)**	**CHT**	**99.17**	**0.87**

*All methods use image preprocessing: Median filter for limbic boundary; Reflection removal for pupillary boundary.

For pupillary boundary detection, enhancing the contrast of image results in improved accuracy as shown in the Table 6.1. The Sobel edge detection for pupillary boundary detection is applied on the reflection free preprocessed image in all the methods of Table 6.1.

Table 6.1 shows that use of adaptive CHT instead of CHT for limbic boundary detection gives more time benefit than accuracy benefit in iris localization.

6.1.4 Comparison with Other Methods

The proposed method is compared with the existing methods to verify its effectiveness. Table 6.2 shows that the accuracy of the proposed method is better than some of the methods in the table. The accuracy results of [Daugman, 1993] and [Wildes, 1997] methods listed in the table are taken from [Radman et al., 2012], but [Radman et al., 2012] does not provide time cost results, which are, therefore, taken from [Proença and Alexandre, 2006]. In [Proença and Alexandre, 2006], all the algorithms were implemented in C++ following an object-oriented paradigm and running in an image processing framework. However, the computation time cannot be compared accurately as the algorithms may be running on different machines or computing platforms, but [Jan et al., 2012] method is implemented in MATLAB and uses similar PC platform that we have used for executing the proposed method. The computation time per image is lowest for the proposed method among all the methods in the table.

Table 6.2. Comparison with published results for UBIRIS.v1 database

Iris localization method	Accuracy (%)	Time per image (sec)
[Daugman, 1993]	95.22	2.73
[Wildes, 1997]	98.68	1.95
[Proença and Alexandre, 2006]	98.02	2.30
[Jan et al., 2012]	93.5	1.14
Proposed	**99.17**	**0.87**

6.2 Concluding Remarks

In this chapter, we have presented an iris localization method for VW images, which can be implemented on FPGA using the hardware implementation techniques discussed in chapter 5. The proposed iris localization method is tolerant to some of the non-ideal issues and noises in the images, such as iris-occlusions by the eyelids and eyelashes, lighting reflections, and low contrast etc. The comparison with published results shows that the proposed method has improved time performance than the previous methods, but similar accuracy performance.

The preliminary work of algorithm optimization towards the hardware implementation of iris localization for VW images is over. This algorithm will require for its hardware implementation the same CHT architecture and same edge-map generation hardware for limbic boundary detection those were used for NIR images in the previous chapter. The edge-map generation hardware for pupillary boundary detection will require two additional filtering modules: image dilation for reflection removal and image enhancement.

Chapter 7
Conclusion and Future Directions

We have discussed two algorithms to make iris localization task faster and accurate: (1) the integro-differential operator (IDO) based method; and (2) the circular Hough transform (CHT) based method. The CHT based algorithm was chosen for hardware implementation of iris localization. This iris localization hardware can be useful to embedded iris recognition systems to meet real time performance. Our main contributions in this thesis are summarized below.

7.1 Summary of Contributions

A) Improved Iris Localization Based on Daugman's IDO for Near Infrared (NIR) Images

The proposed method proposes a technique based on intensity thresholding and morphological operation on binary image to reduce the number of pixels on which the IDO is applied for detecting pupil, which improves the time and accuracy performance. This technique reduces the number of false candidate pixels, which cannot be potential centers of pupil circle. The use of adaptive IDO for limbic boundary detection further improves the performance. The proposed method was tested with CASIA-iris-thousand, version 4.0 (CITHV4) and Multimedia University, version 1.0 (MMUV1) database images. This method is tolerant to the image noises such as reflections, low contrast and occlusions by the eyelids, eyelashes and eyebrow hair etc. (Chapter 3).

B) An Accurate and Fast Iris Localization Method Using Edge-Map Generation and Adaptive CHT for Less Constrained NIR Images

The two main contributions in this method are an edge-map generation technique for pupillary boundary detection and an adaptive CHT algorithm for limbic boundary detection, which make the iris localization more accurate and fast. The proposed method is tolerant to the noises such as

iris-occlusions by the eyelids and eyelashes, lighting reflections, non-uniform illumination, eyeglasses, low contrast and eyebrow hair. However, the method also improves iris localization in the images that do not have reflection spots and non-uniform illumination, but have mainly the iris-occlusions by the eyelids and eyelashes. The proposed method was tested with CITHV4, CASIA-iris-lamp, version 3.0 (CILV3) and MMUV1 database images (Chapter 4).

C) Design and Hardware Implementation of a Memory-Efficient CHT Architecture on Field Programmable Gate Array (FPGA)

The proposed CHT architecture can provide a large memory reduction between 74% and 93%, but with little or no accuracy degradation. The average iris localization time obtained is 6.25 ms per image for an image of size 320×240 pixels. The proposed CHT can be used as a hardware accelerator for 'iris localization' in the iris recognition' systems to get real-time performance in the embedded applications where low speed central processing units (CPUs) are used and the resources are limited (Chapter 5).

D) Hardware Implementation of a Novel Edge-Map Generation Technique for Pupillary Boundary Detection in NIR Images

This is a parallel and pipelined implementation, which produces the output image pixels at a rate of one pixel per clock cycle while the input image pixels are also coming at the same rate of one pixel per clock cycle, after a small initial latency. The edge-map generation hardware takes 390 μsec to generate the edge-map for an input image of size 320×240 pixels, when it is clocked at 200 MHz. This gives a 2100× speedup for edge-map generation as compared to its equivalent MATLAB implementation executing sequentially on today's high speed general purpose CPUs (Chapter 5).

E) Hardware Implementation of Edge-Map Generation for Limbic Boundary Detection in NIR Images

This parallel and pipelined implementation produces the output pixels at a rate of one pixel per clock cycle for an input pixel rate of one pixel per clock cycle, after a small initial latency. The edge-map generation hardware takes 393.675 μsec to generate the final edge-map for an input image of size 320×240 pixels, when it is clocked at 200 MHz. This gives a huge 14047× speedup

for edge-map generation as compared to its equivalent MATLAB code execution on today's high speed general purpose processors (Chapter 5).

F) Architecture Design and Hardware Implementation of 5×5 Median Filter

The proposed median filter architecture was used in the edge-map generation hardware for limbic boundary detection in NIR images. However, the median filter is also used for visible wavelength (VW) images to generate edge-map for limbic boundary detection (Chapter 6), and its another application in iris recognition is the iris feature extraction based on median filter [Zuo et al., 2008]. The proposed median filter architecture has clock cycle latency of 4, which is much better than the previous median filter architectures in the literature. The maximum frequency of operation is also improved as compared to previous architectures. The proposed architecture can be extended to the larger size median filter also (Chapter 5).

G) Iris Localization Algorithm for VW Images Optimized for Hardware Implementation

The CHT based algorithm is proposed for iris localization in VW images, which is fast, accurate and suitable for hardware implementation. This algorithm will require for its hardware implementation the same CHT architecture and same edge-map generation hardware for limbic boundary detection that were used for NIR images in this thesis. The edge-map generation hardware for pupillary boundary detection will require two additional filtering modules: image dilation for reflection removal and image enhancement. The edge-map generation hardware for pupillary boundary detection is only the difference between hardware implementations of iris localization for VW and NIR images, and rests all the other hardware modules are same (Chapter 6).

7.2 Future Directions

The work presented in this thesis is useful for the following systems/ applications:

A) Real Time Iris Localization in Video Based Biometric System

The iris region can be localized in video for identifying persons using iris biometrics for security applications. For such systems, FPGA based iris localization can be very useful to meet real time performance for a video of high frame rate. Face and iris recognition on the move are advanced

biometric systems, which require video input instead of the pictures captured with user's co-operation.

B) FPGA Based Iris Recognition System

A prototype of the complete iris recognition system can be developed using FPGA platform. Today's FPGA chip contains both the CPU core(s) and configurable logic (fabric). The iris localization can be implemented on FPGA fabric, as it is slower process when executes on serial processor as compared to feature extraction and matching stages.

C) Application Specific Instruction-Set Processor (ASIP)

This kind of processor is developed for a particular application and it contains dedicated hardware blocks for executing some instructions or functions. For example, to perform edge detection in MATLAB, we use 'edge ( )' function, but ASIP will use dedicated hardware to perform edge detection. The various hardware modules discussed in this thesis can be used in the development of ASIPs for image processing or iris recognition application.

References

[Bailey, 2011]
D. G. Bailey, “Design process” in “Design for embedded image processing on FPGAs,” John Wiley & Sons (Asia) Pvt. Ltd., Singapore, 2011.

[Basit and Javed, 2007]
A. Basit and M. Y. Javed, “Localization of iris in gray scale images using intensity gradient,” Opt. Lasers Eng., vol. 45, no. 12, pp. 1107–1114, Dec. 2007.

[Batcher, 1968]
K. E. Batcher, “Sorting networks and their applications,” Proc. Jt. Comput. Conf. - AFIPS ’68, p. 307, April 30-May 2, 1968.

[Berger, 2001]
A.S. Berger, “Embedded systems design: An introduction to processes, tools & techniques”. Ed. CMP Books, 2001.

[Bowyer et al., 2008]
K. W. Bowyer, K. Hollingsworth, and P. J. Flynn, “Image understanding for iris biometrics: A survey,” Comput. Vis. Image Underst., vol. 110, no. 2, pp. 281–307, May 2008.

[CASIA Iris Database]
Available online at: http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp (accessed on 28 April 2016)

[Chen et al., 2010]
Y. Chen, M. Adjouadi, C. Han, J. Wang, A. Barreto, N. Rishe, and J. Andrian, “A highly accurate and computationally efficient approach for unconstrained iris segmentation,” Image Vis. Comput., vol. 28, no. 2, pp. 261–269, Feb. 2010.

[Chen et al., 2012]
Z-H. Chen, A. W. Y. Su, and M-T. Sun, “Resource-efficient FPGA architecture and implementation of Hough transform,” IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol. 20, no. 8, pp. 1419-1428, August 2012.

[Danielsson, 1981]
P. E. Danielsson, “Getting the median faster,” Comput. Graph. Image Process., vol. 17, no. 1, pp. 71–78, 1981.

[Davies, 2012]
E. R. Davies, “Computer and machine vision: Theory, algorithms, practicalities,” Academic Press, 2012.

[Daugman, 1993]
J. Daugman, “High confidence visual recognition of persons by a test of statistical independence,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 15, no. 11, pp. 1148–1161, 1993.

[Daugman, 2004]
J. Daugman, “How iris recognition works,” IEEE Transactions on Circuits and Systems for Video Technology. Vol. 14, pp. 21-30, 2004.

[Daugman, 2007]
J. Daugman, “New methods in iris recognition,” IEEE Trans. Syst. Man Cybern. B, Cybern., vol. 37, no. 5, pp. 1167–1175, 2007.

[Daugman, 2006]
J. Daugman, “Probing the uniqueness and randomness of iris codes: Results from 200 billion iris pair comparisons,” Proc. IEEE, vol. 94, no. 11, pp. 1927–1935, Nov. 2006.

[Dorairaj et al., 2005]
Vivekanand Dorairaj, Natalia A. Schmid, and Gamal Fahmy, “Performance evaluation of iris based recognition system implementing PCA and ICA encoding techniques,” in SPIE 5779: Biometric Technology for Human Identification II, vol. 5779, pp. 51–58, 2005.

[Elhossini and Moussa, 2012]
A. Elhossini and M. Moussa, “A memory efficient FPGA implementation of Hough transform for line and circle detection”, in Proc. of the 25th Canadian Conference on Electrical and Computer Engineering, Montreal, QC, Canada, 29 April–2 May, 2012.

[Faundez-Zanuy, 2006]
M. Faundez-Zanuy, “Biometric security technology,” IEEE Aerosp. Electron. Syst. Mag., vol. 21, no. 6, pp. 15–26, Jun. 2006.

[Grabowski and Napieralski, 2011]
K. Grabowski and A. Napieralski, “Hardware architecture optimized for iris recognition,” IEEE Trans. Circuits Syst. Video Technol., vol. 21, no. 9, pp. 1293–1303, 2011.

[Grabowski et al., 2006]
K. Grabowski, W. Sankowski, M. Napieralska, M. Zubert, and A. Napieralski, “Iris recognition algorithm optimized for hardware implementation,” in IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, no. 1, pp. 1–5, 2006.

[Gonzalez et al., 2009]
R.C. Gonzalez, R.E. Woods, and S.L. Eddins, “Digital image processing using Matlab,” Gatesmark Publishing, 2009.

[Hasan and Amin, 2014]
K. M. I. Hasan and M. A. Amin, “Dual iris matching for biometric identification,” Signal, Image and Video Processing, vol. 8, no. 8, pp. 1605–1611, 2014.

[He et al., 2009]
Zhaofeng He, Tieniu Tan, Zhenan Sun, and Xianchao Qiu, “Toward accurate and fast iris segmentation for iris biometrics,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 9, pp. 1670–1684, 2009.

[Hentati et al., 2010]
R. Hentati, M. Bousselmi, and M. Abid, “An embedded system for iris recognition,” In 5th International Conference on Design & Technology of Integrated Systems in Nanoscale Era. IEEE, pp. 1–5, 2010.

[Hentati et al., 2011]
R. Hentati, M. Abid, and B. Dorizzi, “Software implementation of the OSIRIS iris recognition algorithm in FPGA”. In ICM 2011 Proceeding. IEEE, pp. 1–5, 2011.

[Ibrahim et al., 2012]
M. T. Ibrahim, T. M. Khan, S. A. Khan, M. A. Khan, and L. Guan, "Iris localization using local histogram and other image statistics," Opt. Lasers Eng., vol. 50, no. 5, pp. 645–654, 2012.

[Jain and Kumar, 2012]
A. K. Jain and A. Kumar, "Biometric recognition: An overview," Second Gener. Biometrics Ethical, Leg. Soc. Context, pp. 49–79, 2012.

[Jain et al., 1997]
A. K. Jain, L. Hong, S. Pankanti, and R. Bolle, "An identity-authentication system using fingerprints," Proc. IEEE, vol. 85, no. 9, pp. 1365–1388, 1997.

[Jan et al., 2012]
F. Jan, I. Usman, and S. Agha, "Iris localization in frontal eye images for less constrained iris recognition systems," Digit. Signal Process. A Rev. J., vol. 22, no. 6, pp. 971–986, 2012.

[Jan et al., 2013]
F. Jan, I. Usman, and S. Agha, "Reliable iris localization using Hough transform, histogram-bisection, and eccentricity," Signal Processing, vol. 93, no. 1, pp. 230–241, 2013.

[Jan et al., 2014]
F. Jan, I. Usman, S. A. Khan, and S. A. Malik, "A dynamic non-circular iris localization technique for non-ideal data," Computers and Electrical Engineering, vol. 40, no. 8, pp. 215–226, 2014.

[Khalighi et al., 2015]
S. Khalighi, F. Pak, P. Tirdad, and U. Nunes, "Iris recognition using robust localization and nonsubsampled contourlet based features," J. Sign. Process. Syst., vol. 81, no. 1, pp. 111-128, 2015.

[Knuth, 1998]
D. E. Knuth, "The art of computer programming volume 3: Sorting and searching," Addison Wesley, vol. 3, p. 829, 1998.

[Kolte and Smith, 2000]
P. Kolte and R. Smith, "SIMD computation of rank based filters for M×N grids," US Patent 6058405 A, 2000. Available from: USPTO Patent Grants (viewed: 31 March 2016).

[Li et al., 2010]
P. Li, X. Liu, L. Xiao, and Q. Song, "Robust and accurate iris segmentation in very noisy iris images," Image Vis. Comput., vol. 28, no. 2, pp. 246–253, 2010.

[Liu et al., 2005]
X. Liu, K. W. Bowyer, and P. J. Flynn, "Experimental evaluation of iris recognition," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, pp. 1–8, 2005.

[Liu-Jimenez et al., 2005]
Judith Liu-Jimenez, Raul Sanchez-Reillo, and Carmen Sanchez-Avila, "Full hardware solution for processing iris biometrics," IEEE International Carnahan Conference on Security Technology (CCST 2005), pp. 157-163, 2005.

[Liu-Jimenez et al., 2006]
Judith Liu-Jimenez, Raul Sanchez-Reillo, Almudena Lindoso, and Oscar Miguel-Hurtado, "FPGA implementation for an iris biometric processor," IEEE International Conference on Field-Programmable Technology (FPT 2006), pp. 265-268, 2006.

[Liu-Jimenez et al., 2007]
Judith Liu-Jimenez, Oscar Miguel-Hurtado, Almudena Lindoso, and Belen Fernandez-Saavedra, "Hardware/software codesign for an iris biometric search engine," EUROCON, The International Conference on "Computer as a Tool", pp. 642-648, 2007.

[Liu-Jimenez et al., 2011]
J. Liu-Jimenez, R. Sanchez-Reillo, and B. Fernandez-Saavedra, "Iris biometrics for embedded systems," IEEE Trans. Very Large Scale Integr. Syst., vol. 19, no. 2, pp. 274–282, Feb. 2011.

[López et al., 2011]
M. López, J. Daugman, and E. Cantó, "Hardware–software co-design of an iris recognition algorithm," IET Inf. Secur., vol. 5, no. 1, p. 60, 2011.

[Lu and Lu, 2005]
Chenhong Lu and Zhaoyang Lu, "Efficient iris recognition by computing discriminable textons," in International Conference on Neural Networks and Brain, vol. 2, pp. 1164–1167, October 2005.

[Ma et al., 2003]
L. Ma, T. Tan, Y. Wang, and D. Zhang, "Personal identification based on iris texture analysis," Pattern Anal. Mach. Intell. IEEE Trans., vol. 25, no. 12, pp. 1519–1533, 2003.

[Ma et al., 2004]
L. Ma, T. Tan, Y. Wang, and D. Zhang, "Efficient iris recognition by characterizing key local variations," IEEE Trans. Image Process., vol. 13, no. 6, pp. 739–750, Jun. 2004.

[Marciniak et al., 2014]
T. Marciniak, A. Dabrowski, A. Chmielewska, and A. A. Krzykowska, "Selection of parameters in iris recognition system," Multimed. Tools Appl., vol. 68, no. 1, pp. 193–208, 2014.

[Masek and Kovesi, 2003]
L. Masek and P. Kovesi, "MATLAB source code for a biometric identification system based on iris patterns," The School of Computer Science and Software Engineering, the University of Western Australia, 2003.

[Matey et al., 2006]
James R. Matey, Oleg Naroditsky, Keith Hanna, Ray Kolczynski, Dominick J. LoIacono, Shakuntala Mangru, Michael Tinker, Thomas M. Zappia, and Wenyi Y. Zhao, "Iris on the move: Acquisition of images for iris recognition in less constrained environments," Proceedings of the IEEE, vol. 94, no. 11, pp. 1936–1947, 2006.

[Mehrotra et al., 2013]
H. Mehrotra, P. K. Sa, and B. Majhi, "Fast segmentation and adaptive SURF descriptor for iris recognition," Math. Comput. Model, vol. 58, no. 1–2, pp. 132–146, 2013.

[Monro et al., 2007]
D. M. Monro, S. Rakshit, and S. Member, "DCT-based iris recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 4, pp. 586–595, 2007.

[Muroň and Pospišil, 2000]
A. Muroň and J. Pospišil, "The human iris structure and its usages," Physica, vol. 39, pp. 87–95, 2000.

[Ngo et al., 2012]
H. T. Ngo, J. Shafer, R. W. Ives, R. N. Rakvic, and R. P. Broussard, "Real time iris segmentation on FPGA", in Proc. the 23rd IEEE International Conference on Application-specific Systems, Architectures and Processors, Delft, The Netherlands, pp. 1-7, July 2012.

[Ngo et al., 2014]
H. Ngo, R. Rakvic, R. Broussard, and R. Ives, “Resource-aware architecture design and implementation of hough transform for a real-time iris boundary detection system,” IEEE Trans. Consum. Electron., vol. 60, no. 3, pp. 485–492, 2014.

[Noegaard, 2005]
Tammy Noegaard, “Embedded system architecture: a comprehensive guide for engineers and programmers (embedded technology),” Ed. Newness, 2005.

[Pedersen, 2007]
S.J.K. Pedersen, “Circular Hough transform”, Aalborg University, Vision, Graphics and Interactive Systems, 2007.

[Proenca, 2010]
H. Proença, “Iris recognition: On the segmentation of degraded images acquired in the visible wavelength,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 8, pp. 1502–1516, 2010.

[Proença and Alexandre, 2005]
H. Proença and L.A. Alexandre, “UBIRIS: A noisy iris image database,” In Proceedings of 13th International Conference on Image Analysis and Processing, Cagliari, Italy, pp. 970-977, 2005.

[Proença and Alexandre, 2006]
H. Proença and L.A. Alexandre, “Iris segmentation methodology for non-cooperative recognition,” IEE Proceedings of Vision, Image and Signal Processing, vol. 153, no. 2, pp. 199-205, 2006.

[Proenca et al., 2010]
H. Proença, S. Filipe, R. Santos, J. Oliveira, and L.A. Alexandre, “The UBIRIS.v2: A database of visible wavelength iris images captured on-the-move and at-a-distance,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(8), pp.1529–1535, 2010.

[Prokin and Prokin, 2010]
D. Prokin and M. Prokin, “Low hardware complexity pipelined rank filter,” IEEE Trans. Circuits Syst. II Express Briefs, vol. 57, no. 6, pp. 446–450, 2010.

[Puhan et al., 2011]
N. B. Puhan, N. Sudha, and A. Sivaraman Kaushalram, “Efficient segmentation technique for noisy frontal view iris images using Fourier spectral density,” Signal, Image Video Process., vol. 5, no. 1, pp. 105–119, Mar. 2011.

[Radman, 2013]
Abduljalil Radman, “Fast and reliable iris segmentation algorithm,” IET Image Processing, vol. 7, no. 1, pp. 42-49, 2013.

[Radman et al., 2012]
A. Radman, K. Jumari, and N. Zainal, “Iris segmentation in visible wavelength environment,” Procedia Engineering 41 pp. 743-748, 2012.

[Rakvic et al., 2009]
R. N. Rakvic, B. J. Ulis, R. P. Broussard, R. W. Ives, and N. Steiner, “Parallelizing iris recognition,” IEEE Trans. Inf. Forensics Secur., vol. 4, no. 4, pp. 812–823, 2009.

[Ross, 2010]
A. Ross, “Iris recognition: The path forward,” Computer (Long. Beach. Calif.), vol. 43, no. 2, pp. 30–35, 2010.

[Roy et al., 2011]
K. Roy, P. Bhattacharya, and C. Y. Suen, “Iris segmentation using variational level set method,” Opt. Lasers Eng., vol. 49, no. 4, pp. 578–588, Apr. 2011.

[Sahmoud and Abuhaiba, 2013]
S. A. Sahmoud and I. S. Abuhaiba, “Efficient iris segmentation method in unconstrained environments,” Pattern Recognit., vol. 46, no. 12, pp. 3174–3185, 2013.

[Sankowski et al., 2010]
W. Sankowski, K. Grabowski, M. Napieralska, M. Zubert, and A. Napieralski, “Reliable algorithm for iris segmentation in eye image,” Image Vis. Comput., vol. 28, no. 2, pp. 231–237, Feb. 2010.

[Schuckers et al., 2007]
S. A. C. Schuckers, N. A. Schmid, A. Abhyankar, V. Dorairaj, C. K. Boyce, and L. A. Hornak, “On techniques for angle compensation in nonideal iris recognition,” IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 37, no. 5, pp. 1176-1190, 2007.

[Scott et al., 2008]
J. Scott, M. Pusateri, and M. U. Mushtaq, “Comparison of 2D median filter hardware implementations for real-time stereo video,” 2008 37th IEEE Appl. Imag. Pattern Recognit. Work., pp. 176–181, 2008.

[Shah and Ross, 2009]
S. Shah and A. Ross, “Iris segmentation using geodesic active contours,” IEEE Trans. Inf. Forensics Secur., vol. 4, no. 4, pp. 824–836, 2009.

[Tan et al., 2010]
T. Tan, Z. He, and Z. Sun, “Efficient and robust segmentation of noisy iris images for non-cooperative iris recognition,” Image Vis. Comput., vol. 28, no. 2, pp. 223–230, Feb. 2010.

[Thompson, 1983]
C. D. Thompson, “The VLSI complexity of sorting,” IEEE Trans. Comput., vol. C–32, no. 12, pp. 1171–1184, 1983.

[Thoonsaengngam et al., 2006]
Peeranat Thoonsaengngam, Kittipol Horapong, Somying Thainimit, and Vutipong Areekul, “Efficient iris recognition using adaptive quotient thresholding,” in Proceedings of International Conference on Biometrics , pp. 472–478, January 2006.

[UBIRIS Image Database]
Available online at: http://iris.di.ubi.pt/ (accessed on 15 May 2016)

[Vasanth et al., 2010]
K. Vasanth, S. Karthik, S. Nirmal Raj, and P. Preetha Mol, “FPGA implementation of optimized sorting network algorithm for median filters,” in International Conference on “Emerging Trends in Robotics and Communication Technologies”, INTERACT, pp. 224–229, 2010.

[Wang et al., 2014]
N. Wang, Q. Li, A. A. A. El-Latif, T. Zhang, and X. Niu, “Toward accurate localization and high recognition performance for noisy iris images,” Multimedia Tools Appl., vol. 71, no. 3, pp. 1411-1430, 2014.

[Wildes, 1997]
R. P. Wildes, “Iris recognition: an emerging biometric technology,” Proc. IEEE, vol. 85, no. 9, pp. 1348–1363, 1997.

[Yao et al., 2006]
Peng Yao, Jun Li, Xueyi Ye, Zhenquan Zhuang, and Bin Li, “Iris recognition algorithm using modified log-gabor filters,” in Proceedings of the 18th International Conference on Pattern Recognition, Washington, DC, USA, vol. 4, pp.461–464, August 2006.

[Yuen et al., 1990]
H.K. Yuen, J. Princen, J. Illingworth, and J. Kittler, “Comparative study of Hough Transform methods for circle finding”, Image and Vision Computing, vol. 8, issue 1, pp. 71-77, Feb. 1990.

[Zhou and Xie, 2010]
H. Zhou and M. Xie, “Iris biometric processor enhanced module FPGA-based design,” in Second International Conference on Computer Modeling and Simulation, vol. 2, pp. 259–262, 2010.

[Zhou et al., 2013]
X. Zhou, N. Tomagou, Y. Ito, and K. Nakano, “Efficient Hough transform on the FPGA using DSP slices and block RAMs”, in Proc. of IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum, Cambridge, MA, pp. 771 - 778, May 2013.

[Zuo and Schmid, 2010]
J. Zuo and N. A. Schmid, “On a methodology for robust segmentation of nonideal iris images,” IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 40, no. 3, pp. 703–718, 2010.

[Zuo et al., 2008]
J. Zuo, N. K. Ratha, and J. H. Connell, “Median filter based iris encoding technique,” in IEEE 2nd International Conference on Biometrics: Theory, Applications and Systems, Arlington, VA, pp. 1-6, Sept. 29 - Oct. 1, 2008.

List of Publications

International Journals

1. Vineet Kumar, Abhijit Asati, Anu Gupta, "A Novel Edge-Map Creation Approach for Highly Accurate Pupil Localization in Unconstrained Infrared Iris Images," Journal of Electrical and Computer Engineering, vol. 2016, Article ID 4709876, 10 pages, 2016. (Hindawi publication-Scopus-indexed).
2. Vineet Kumar, Abhijit Asati, Anu Gupta, "Accurate iris localization using edge-map generation and adaptive circular Hough transform for less constrained infrared iris images", International Journal of Electrical and Computer Engineering, vol. 6, no. 4, pp. 1637-1646. (IAES publication- Scopus-indexed).
3. Vineet Kumar, Abhijit Asati, Anu Gupta, "Hardware Implementation of a Novel Edge-Map Generation Technique for Pupil Detection in NIR Images", Engineering Science and Technology, an International Journal, 2016 (Elsevier).
   http://dx.doi.org/10.1016/j.jestch.2016.11.001

International Conferences

1. Vineet Kumar, Abhijit Asati, Anu Gupta, "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.
2. Vineet Kumar, Abhijit Asati, Anu Gupta, "Iris Localization Based on Integro-Differential Operator for Unconstrained Infrared Iris Images", In Proceedings of International Conference on Signal Processing, Computing and Control, JUIT, Waknaghat, India, pp. 277-281, Sept 2015.

Brief Biography of Candidate

Vineet Kumar received M.Tech. (Microelectronics and VLSI Design) degree (in 2007) from University Institute of Engineering and Technology (UIET), Kurukshetra University (KU) Kurukshetra and the M.Sc (Electronic science) degree (in 2005) from Department of Electronic Science, KU Kurukshetra. Prior to this, he completed the B.Sc. (Electronics and Computer Science) degree from Dayanand (DN) College Hisar in year 2003. In 2007, he joined the Mody Institute of Technology and Science (MITS) Lakshmangarh as Lecturer. In 2008, he joined Birla Institute of Technology and Science (BITS) Pilani as Assistant Lecture and presently he is working as Lecturer at BITS Pilani. From 2012 onwards, he is pursuing his doctoral research in the area of image processing algorithms and their hardware implementation for iris recognition application.

Brief Biography of Supervisor

Dr. Abhijit Asati completed Ph.D. degree (in 2010) and M.E. degree (in 2002) from BITS Pilani. Prior to this, he completed B.E. degree in electrical engineering from Amravati University in the year 1996. He has more than 15 years of experience in academia and currently working as 'Assistant Professor' in the 'Electrical and Electronics Engineering Department' at BITS Pilani. He also served as a faculty member at the Visveswarya National Institute of Technology, Nagpur from 1997 to 1999. He taught several courses such as Microelectronic Circuits, Analog and digital VLSI design, CAD for IC Design, VLSI test and testability. Dr. Asati also supervised around 15 B.E. and M.E. thesis and currently supervising 2 research scholars. He has contributed more than 20 journal papers and more than 25 conference papers in the area of microelectronics and VLSI design. His research contributions are in the area of high performance VLSI data path design, microprocessor design, NBTI degradation issues and biometric identification. He also visited "Cypress Semiconductor Limited, Bangalore", from May to July 2014 as a part of Industry Immersion Program of BITS, Pilani.

Brief Biography of Co-Supervisor

Dr. Anu Gupta completed Ph.D. degree (in 2003) and M.E. degree (in 1995) from Birla Institute of Technology and Science (BITS) Pilani. In 1995, she joined BITS Pilani as Assistant Lecturer, became Assistant Professor in 2003 and Associate Professor in 2010. She is presently working as Associate Professor and Head of Electrical and Electronics Engineering Department at BITS Pilani. Her research interest includes low power, high performance analog/ digital/ mixed signal design for FPGA/ ASIC applications.