Detection and Cognitive Assessment of Pupil Dilation using High Performance Computing

AMNAH NASIM

RESEARCH CENTRE FOR MODELING & SIMULATION

2014

DETECTION AND COGNITIVE ASSESSMENT OF PUPIL DILATION USING HIGH PERFORMANCE COMPUTING

AMNAH NASIM

Research Centre for Modeling & Simulation

A thesis submitted to the

National University of Sciences & Technology

In partial fulfillment of the requirement for the degree of

Masters of Science

2014

STATEMENT OF ORIGINALITY

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

Date	AmnahNasim

Acknowledgements

"Recite in the name of your Lord who created - Created man from a clinging substance. Recite, and your Lord is the most Generous - Who taught by the pen - Taught man that which he knew not."

The QURAN, Sura Al-Alaq 96/1-5

First and foremost I praise and acknowledge Allah, my sole belief in Whose mercy held me strong through the toughest of times. My humblest gratitude to the Holy Prophet Muhammad (PBUH), whose way of life has been a continuous guidance for me.

I express a deep sense of appreciation for my supervisor, Dr. Adnan Maqsood, for his valuable and constructive suggestions during the formation and development of this research work. His calm nature and positive criticism motivated me to thrive for the best. Secondly, I would like to thank the esteemed members of my G.E.C committee Engr. Sikander Hayat Mirza, Assistant Professor Tariq Saeed and Dr. Jamil Ahmad for their important remarks about my thesis.

I would also like to thank my senior Junaid Ahmad, for getting me started with MATLAB, and for always being helpful whenever I took a problem to him. I would like to thank all the lab staff, Engr. Muhammad Usman, Hasnain Jaffery, Asim Younis, Muhammad Aslam for providing all the software installations and configurations for my system setup.

I am grateful to my very close friends Farzana and Mubashra who constantly encouraged me and handled all my good, bad and seriously bad ☺ moods throughout. Last but not least I would like to acknowledge my parents and sister for their constant support, prayers and best wishes. They uplifted my morale whenever I needed. Finally I would say thanks to all who have been supportive and well-wishers to me in this period of life.

Abstract

Pupil dilations are sensitive to human affective responses. The assessment of pupillary responses has found multi-purpose applications such as lie detection in law enforcement agencies, providing an innovative solution for human-computer interaction, minimizing dangers in air traffic displays, communicating with people suffering from ailments such as autism, developing interactive video games, eye gaze correction for videoconferencing, marketing and consumer research.

In this work, the detection and assessment of pupil diameter is done by developing a low-cost eye tracking system. The scope of the system development is to accurately detect pupillary responses over a period of time i.e. subtle changes in pupil size that indicate cognitive load. The features of the proposed framework are eye and pupil-localization, pupil area and diameter calculation in a recorded video sequence and investigation of pupil size variation during and after auditory stimulation. In this case human eye images were acquired by a webcam and processed offline.

The hardware includes a micro-lens webcam and head mounted structure with a USB data transfer capability to computer. The framework for the algorithm development includes iris and pupil localization, pupil area and diameter extraction and calibration that are implemented in MATLAB. Subsequently, the algorithm is implemented on video sequences from the

webcam. Pattern Recognition techniques are used to transform the images from RGB to binary and Hough Transform is used to detect pupil area from the rest of the eye image. Moreover, a high performance implementation of the proposed algorithm is done using MATLAB® parallel processing toolbox and Graphical Processor Unit (GPU). A comparison of speedups achieved by implementing the above techniques is analyzed. Finally, experiments are conducted on two subjects to measure human cognition and the algorithmic efficiency of eye pupil location and variation.

Contents

CHAPTER 1 11

INTRODUCTION 11

1.1 INTRODUCTION 11

1.2 HUMAN AUTONOMIC NERVOUS SYSTEM 12

1.2.1 Sympathetic and Parasympathetic responses 12

1.2.2 Skin Conductance Response (SCR) 13

1.2.3 Peripheral Arterial Tone (PAT) 13

1.2.4 Pupil Diameter (PD) 14

1.3 ANATOMY OF PUPIL 14

1.3.1 Task-Evoked Pupillary Responses 16

1.4 PARALLEL PROGRAMMING 17

1.5 RESEARCH OBJECTIVES 17

1.6 METHODOLOGY 18

1.7 CONTRIBUTIONS 19

1.8 ORGANIZATION OF THE THESIS 20

CHAPTER 2 22

LITERATURE REVIEW 22

2.1 BACKGROUND 22

2.2 SURVEY OF RESEARCH IN PUPIL DILATION RESPONSES AND HUMAN COGNITION 22

2.3 HIGH PERFORMANCE IMPLEMENTATION REQUIREMENTS 26

2.4 PARALLELIZING HOUGH TRANSFORM 26

2.5 MISSING LINKS IN LITERATURE 28

CHAPTER 3 30

PUPIL SEGMENTATION 30

3.1 BACKGROUND 30

3.2 REGION OF INTEREST CROPPING 31

3.3 IMAGE SHARPENING AND SMOOTHENING 32
3.4 CANNY EDGE DETECTION 33
3.5 HOUGH TRANSFORM 34
3.5.1 Parabolic Hough Transform 35
3.5.2 Elliptic Hough Transform 37
3.6 PUPIL AREA AND DIAMETER CALCULATION 39
3.6.1 Pixel to mm Conversion 40
CHAPTER 4 42
HIGH PERFORMANCE COMPUTATION 42
4.1 HIGH PERFORMANCE COMPUTER ARCHITECTURE 42
4.2 PARALLEL PROGRAMMING METHODS IN MATLAB 42
4.2.1 The Par-for Loop 42
4.2.2 GPU Computing in MATLAB 43
4.3 PARALLELIZING HOUGH TRANSFORM 44
4.4 SPEED-UP RESULTS 45
4.4.1 MATLAB Parallel-for workers 45
4.4.2 GPU-based Implementation 49
4.4.3 Parallelized Pupil Detection on a single eye frame 50
CHAPTER 5 54
COGNITIVE LOAD ASSESSMENT 54
5.1 BACKGROUND 54
5.1.1 Participants 54
5.1.2 Apparatus 54
5.1.3 Procedure 55
5.2 EXPERIMENT RUNS AND ANALYSIS 55
5.2.1 Experiment I 56
5.2.2 Analysis Experiment I 57
5.2.3 Experiment II 57
5.2.4 Analysis Experiment II 59

CHAPTER 6 60

CONCLUSIONS & FUTURE WORK....60

6.1 CONCLUSIONS....60

6.1.1 Conclusions Related to Eye Pattern Segmentation....60

6.1.2 Conclusions Related to High Performance Computational Techniques....61

6.1.3 Conclusions Related to Human Cognition....62

6.2 FUTURE WORK....62

BIBLIOGRAPHY 64

APPENDIX A MATLAB PROGRAM CODES....66

List of Figures

Figure 1: Muscles of the iris.Two opposing muscle groups within the iris of the human eye determine the aperture of the pupil. The sphincter muscles of the iris constrict the pupil when they contract, whereas constriction of the dilator muscles increases pupillary diameter[4]Pupillary Reflex Dilations 16

Figure 2: Methodology Flowchart 18

Figure 3: Segmentation steps followed through the program 31

Figure 4: Image cropped from the original image obtained from the camera 32

Figure 5: Input image (a), Image sharpened once (b), Image sharpened twice (c) 32

Figure 6: Edge map obtained after applying Canny edge detector and thresholding 34

Figure 7: Detected parabola (a) and filtered extra edges (b) 36

Figure 8: Detected eyelash parabola on the grayscale image 37

Figure 9: Detected iris on the edge-map (left) and on the original grayscale image (right) 38

Figure 10: Detected pupil region and fitted ellipse through Hough ellipse transform.....39

Figure 11: Complete eye pattern detected through the proposed methodology.....39

Figure 12: Segmentation results of some samples in the CASIA iris dataset 41

Figure 13: Working of GPU acceleration on a piece of code 44

Figure 14: Parallel Runtime of Parabolic Hough Transform on an 8-core Machine 46

Figure 15: Parallel Speedup of Parabolic Hough Transform on 8-core Machine 47

Figure 16: Parallel Runtime of Elliptic Hough Transform on an 8-core Machine.....48

Figure 17: Parallel Speedup of Elliptlic Hough Transform on 8-core Machine 48

Figure 18: GPU runtime for Parabolic Hough Transform on NVIDIA Tesla T10 processor.....49

Figure 19: GPU runtime for Elliptic Hough Transform on NVIDIA Tesla T10 processor 50

Figure 20: Parallel Runtime of proposed method on an 8-core Machine 51

Figure 21: Parallel Runtime of proposed method on an 8-core Machine 52

Figure 22: A comparison of CPU, CPU-8 core and GPU runtimes for proposed algorithm for a single eye frame on NVIDIA Tesla T10 processor.....53

Figure 23: Experimental Setup 55

Figure 24: Pupillary response during the mental multiplication task on the major axis ‘a’ of the ellipse56

Figure 25: Pupillary response during the mental multiplication task on the minor axis ‘b’ of the ellipse57

Figure 26: Pupillary Area Variation during the mental multiplication task 57

Figure 27: Pupillary response during the aural vigilance task on the major axis ‘a’ of the ellipse 58

Figure 28: Pupillary response during the aural vigilance task on the minor axis ‘b’ of the ellipse 58

Figure 30 : Pupillary Area Variation during the aural vigilance task 58

List of Tables

Table 1: Serial and parallel execution times for a single frame on a multicore machine.....46

Table 2: Serial and parallel execution times for a single frame on a multicore machine.....48

Table 3: Execution times and Speedups for CPU, CPU (8-cores) and GPU for parabolic Hough transform 49

Table 4: Execution times and Speedups for CPU, CPU (8-cores) and GPU for elliptic Hough transform50

Table 5: Execution times and speedups achieved by the proposed pupil detection methodology applied on a single eye image.....51

Table 6: Execution times and speedups achieved for CPU, CPU-8 cores and GPU-based implementation for a single eye frame on NVIDIA Tesla T10 processor 52

CHAPTER 1

Introduction

1.1 Introduction

The number of applications in which a user's facial expression is tracked by a video camera and analyzed is growing exponentially these days. Cameras, webcams and cellphones are constantly capturing facial images. Using facial information as an authentic indicator of the current state of the user's mind, several car companies in Japan and US are fitting cameras in the dashboard to detect drowsiness in drivers. Likewise, advertisers on web portals use facial information to determine the influence of their billboards and logos, with the intention of changing the look of a website in response to user's response about the advertisements. Moreover, video game companies and human-computer interaction experts are interested in assessing the player's emotions during game play to help measure the success of their products.

There are two goals in the development of real-time algorithms for eye detection and assessment. The first is the assistive techniques used in Human–Computer Interaction (HCI) applications and the second is to advance our theoretical understanding of emotions and associated facial expression. By using knowledge algorithms, we can link rich datasets of facial expression points and physiological reactions to emotional responses. So we can design precise models

of how emotions stated in reply to suggestive stimuli are captured through facial expressions.

1.2 Human Autonomic Nervous System

The Human Autonomic Nervous System constitutes a part of the peripheral nervous system. Its primary function is to transmit impulses from the central nervous system to peripheral organ system. The main purpose of the Autonomic Nervous System is to maintain homeostasis (managing highly complex internal human body interactions to maintain stability and return the body systems to function within a normal range) in the body. The **autonomic nervous system** controls the pulse rate, constriction and dilation of blood vessels, contraction and relaxation of smooth muscles in various organs, pupillary size and the action of sweat glands. Autonomic Nervous System also controls the coordination and regulation of many emotional responses produced in reaction to environmental stimuli and usually these reactions are automatic or impulsive in nature.

1.2.1 Sympathetic and Parasympathetic responses

Based on anatomy and functionality differences, the autonomic nervous system is subdivided into two distinct parts called the Sympathetic and Parasympathetic nervous systems. The sympathetic nervous system enables the body to be prepared for emotionally charged stimuli, by mobilizing body resources to allow the organism to utilize a large amount of energy. Sympathetic responses include an increase in heart rate, blood pressure, pupil size, and sweat gland activity[1]

and a decrease in peripheral blood vessel diameters (i.e. vasodilatation).In contrast, the parasympathetic nervous system is concerned with conservation and restoration of energy by restoring bodily resources into their initial state. Parasympathetic responses include a decrease in heart rate, blood pressure, pupil size, sweat gland activity, and increase in peripheral blood vessel diameters (i.e. vasodilatation). The basic responses used to measure the cognitive load [2]of the human nervous system are explained ahead.

1.2.2 Skin Conductance Response (SCR)

The skin conductance response (SCR) also known as electro-dermal activity (EDA) is used to describe a method of measuring changes in the electrical properties of the skin in response to environmental stimuli. A change in skin conductance reflects sweat glands activity which is controlled by the Autonomic Nervous System. SCR has been found to be related to various behavioral phenomena and may serve as an effective measure of intuitive processes of decision making.

1.2.3 Peripheral Arterial Tone (PAT)

The term Peripheral Arterial Tone (PAT) is used to describe a method for measuring variations in the vascular size (i.e., diameter of the blood vessels) at the human fingertip. The level of sympathetic activation is indicated by a decrease in the transparency of the blood, as a function of its pulsatile volume i.e. blood pressure. Iani et al. [3]in 2007 used PAT to investigate mental effort during a simulated flight task. Their result demonstrated increased arousal as

the cognitive requirements of the task increased. Thus, although not directly related to emotional or intuitive processes, it seems that PAT may serve as an effective measure of intuitive processes in decision making.

1.2.4 Pupil Diameter (PD)

The term Pupil Diameter (PD) response refers to a method of measuring changes in the diameter of the pupillary aperture of the eye. 'Pupil' is the sphere that is found at the center of the iris of the eye. One of its major functions is to control the amount of light entering the eye. Andreassi [4] shows that psychologically-invoked pupillary dilations can override physiologically-invoked responses, such as intense light

Typical pupil diameter measuring equipment includes an infrared camera and an infrared light source. The infrared light is required to enable clear capturing of pupil in bad-lighting environments. The pupil diameter of the human eye is measured in millimeters and it ranges from 1.5 mm to 9 mm. In addition, pupils may dilate in response to stimuli in as little as 0.2 seconds, and peak in about 0.5 or 1.0 seconds. Thus, unlike other autonomic system measures, the pupillary responses require high temporal resolution[2].

1.3 Anatomy of Pupil

Pupillary movements [5], as shown in Figure 1, are determined by the state of the iridic musculature (muscles controlling the iris) under the direct control of both the sympathetic and parasympathetic branches of the autonomic nervous system. The coupling of pupillary movements to cognitive processes,

however, must occur at much higher levels within the human nervous system. The amount of light entering the eye is restricted by the aperture in the iris, the pupil.

When a person is in a dark room his pupil is large, perhaps eight millimeters (0.3 inch) in diameter, or more. When the room is lighted there is an immediate constriction of the pupil, the light reflex; this is bilateral, so that even if only one eye is exposed to the light both pupils contract to nearly the same extent. After some time the pupils expand even though the bright light is maintained, but the expansion is not large. The final state is determined by the actual degree of illumination; if this is high, then the final state may be a diameter of only about three to four millimeters (about 0.15 inch); if it is not so high, then the initial constriction may be nearly the same, but the final state may be with a pupil of four to five millimeters (about 0.18 inch). During this steady condition, the pupils do not remain at exactly constant size; there is a characteristic oscillation in size that, if exaggerated, is called hippus - the rhythmic but irregular (usually < 0.04 Hz) constrictions and dilations [5] of the pupil that occur independent of eye movements or changes in illumination. Dilation of the pupil occurs as a result of strong psychical stimuli and also when any sensory nerve is stimulated; dilation thus occurs in extreme fear and in pain. The dilator muscle of the iris is activated by sympathetic nerve fibers. Stimulation of the sympathetic nerve in the neck causes a powerful dilation of the iris; again, the influx of adrenalin into the blood from the adrenal glands during extreme excitement results in pupillary dilation.

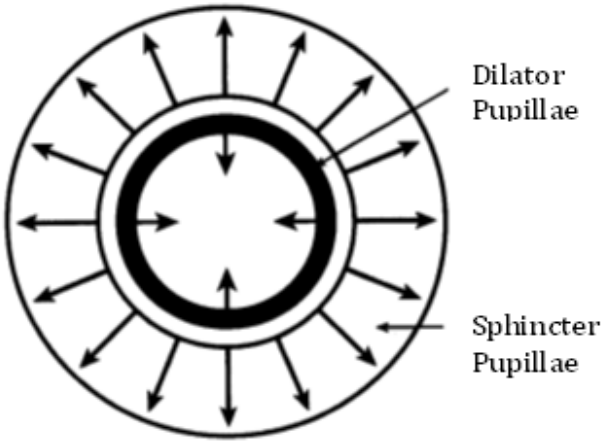


Figure 1: Muscles of the iris.Two opposing muscle groups within the iris of the human eye determine the aperture of the pupil. The sphincter muscles of the iris constrict the pupil when they contract, whereas constriction of the dilator muscles increases pupillary diameter[4]Pupillary Reflex Dilations

Any sensory occurrence [6] – whether tactile (sense of touch), auditory, gustatory (sense of taste), olfactory (sense of smell), or noxious (constituting a harmful influence on mind or behavior) – evokes a pupillary reflex dilation. Exceptions to this are light stimuli and accommodation to near visual stimuli, both of which produce pupil constrictions. However, one should not assume that pupillary reflex dilations occur only to external sensory events, because emotions, mental processes, increases in intentional efforts, and motor output also produce systematic changes in pupillary diameter. Several factors determine the magnitude of these pupillary dilations – for example, the individual's tonic state of arousal, the emotional effect of the stimulus, and luminance levels.

1.3.1 Task-Evoked Pupillary Responses

Non-reflexive phasic pupillary movements are of interest in cognitive psychophysiology because they function as empirically based reporter indicators for brain processes that underlie the dynamic, intensive aspects of human cognition. In order to understand the use of task-evoked pupillary responses in

cognitive psychophysiology, it is necessary to review the methods by which they are measured and analyzed.

1.4 Parallel Programming

Traditionally, software architectures are based on serial computations. A problem is broken into a discrete series of instructions. Instructions are executed sequentially one after another on a single processor. Only one instruction can be executed at any moment in time. With the evolution of simulation algorithms and extensively increasing data spaces and the limitations imposed at transistor level on CPU processing speeds, parallel computing has emerged as a prevalent method to provide good performance metrics, especially for graphical processing applications. In the simplest sense, parallel computing is the simultaneous use of multiple computing resources to solve a computational problem. A problem is broken into discrete parts that can be solved concurrently. Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different processors. An overall control/coordination mechanism is employed.

1.5 Research Objectives

Eye tracking devices are a powerful alternative for individuals with no control, or only limited control, over their physical movements. The device follows the movement of the eyes and detects behaviors with only pupil dilation. The target objectives for this research are summed-up as follows:

- Setting-up an ergonomic Eye Tracker apparatus

- Designing an efficient eye pupil-tracker algorithm
- Parallel implementation of the algorithm
- Precise measurement of affective human response detection through human experiments

1.6 Methodology

Figure 2 shows the methodology followed throughout the thesis project phase:

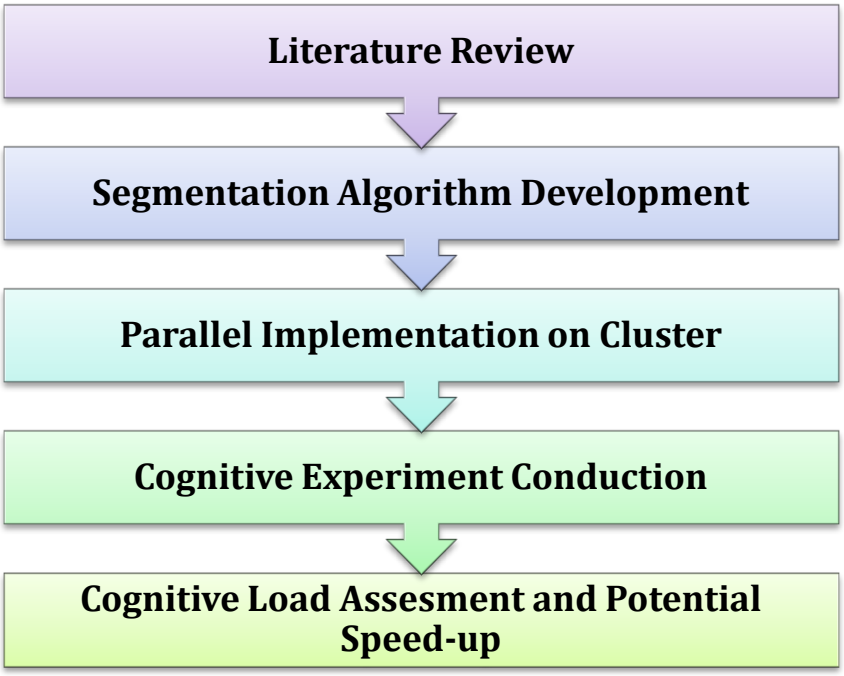


Figure 2: Methodology Flowchart

Initially a detailed literature study was done to review the research carried out previously to prove pupil dilation as an authentic measure of human cognitive load. Different image segmentation techniques were studied to select one method for detection and measurement of pupil diameter from the input eye image to meet the required criteria that the segmentation should be invariant to

rotation and translation. Then previously applied speed-up methods for Hough transform were summarized to follow one criterion for our parallel programming implementation. In the second phase, a pupil detection algorithm was developed on MATLAB® 2012a using Hough transform for parabola and ellipse and the diameter of pupil was measured. Observing the speed tradeoffs that were made at the expense of accuracy of the algorithm, the program was parallelized on 8 CPU cores and GPU. Then, human experiments were conducted by setting up a camera apparatus and three aural experiments were performed to show pupil diameter variation as a reflection of human cognitive load. Finally, the results obtained through cognition experiments and high performance techniques are analyzed to draw conclusions.

1.7 Contributions

Eye tracking and eye movement-based interaction using computer vision techniques have the potential to become an important component in future perceptual user interfaces. Instead of measuring a person's emotional reaction, pupil-dilation provides a tangible gauge to assess the cognitive reactions. The main contributions of the current research work include:

1. Development of low-cost assessment procedure
2. Development of parallel processing algorithm
3. Testing the algorithm on human subjects
4. One journal manuscript preparation

1.8 Organization of the Thesis

This thesis comprises of six chapters. The brief outline of each chapter is discussed as follows:

Chapter 1 --- Introduction

The first chapter gives an introduction to pupillary actions, some basic methods of measurement of pupil dilation and the motivation about what are the task-evoked pupillary responses and why is there a need to measure and assess them in cognitive psychophysiology. The objectives, scopes and methodology used in the research are documented. A systematic framework of the report is given at the end of the chapter.

Chapter 2 --- Literature Review

A detailed review of related literature is given in this chapter. Identification of challenges from the literature survey is also presented as a driver for the research.

Chapter 3 --- Segmentation Techniques

In this chapter, some basic computer vision methods used for pattern recognition of the human eye are discussed.

Chapter 4 --- Parallel Implementation for High Performance

In this chapter, three different parallel computing methods were implemented and some integrative combinations of these techniques were executed to make a comparison for potential speedups.

Chapter 5 --- Cognitive Experiments

In this chapter two cognitive experiments are designed and performed to record and analyze human cognitive load as reflected in pupil size variation.

Chapter 6 --- Results, Conclusions and Future Work

Results from the segmentation algorithm designed to detect and measure pupil diameter, high performance techniques and configurations implemented and potential speedups achieved are reported in this chapter. The outcomes from the cognitive experiments are also interpreted in this chapter. Finally, conclusions drawn from the research presented in dissertation are summarized and recommendations for future line of action are laid down in this chapter.

CHAPTER 2

Literature Review

2.1 Background

A review of existing literature was performed to support the methodology undertaken in this thesis. A general survey was first performed to account for the past research efforts done in developing a connection between pupillometry and human cognition. Next, the potential applications and advantages of using pupil dilation as an authentic and newly emerging biometric technology versus fingerprint, facial and iris recognition technologies are discussed. Lastly, a discussion of literature study associated with eye pattern recognition techniques is performed along with a discussion of high performance computing issues.

2.2 Survey of research in pupil dilation responses and human cognition

Partalaa et al. [7] investigated pupil size variation in the course of auditory emotional stimulation. Pupil responses from thirty subjects' (15 female and 15 male) were measured while they listened to 30 kinds of highly arousing sounds, 10 negative and 10 positive e.g. voice of a baby laughing and crying, and 10 neutral sounds e.g. regular office noise. The results depicted that the size of pupil was considerably larger during both emotionally negative and positive

stimuli as compared to neutral stimuli. The results were simulated for the time period of 2 seconds after the stimulus offset. Their findings suggest that the human autonomic nervous system is sensitive to highly arousing emotional stimulation. The subjective ratings proved that the stimuli influenced the subjects' emotional experiences as expected. Further analysis on the subject's gender showed that female subjects had significantly larger pupil responses than male subjects during both neutral and auditory stimulation. The results revealed that stimuli chosen in a systematic way significantly effect subject's physiological responses and subjective experiences. Hence, pupil size variation can be used as a potential input signal in affective computing. Furthermore, audio related emotion stimuli could also be used to control a person's emotional reactions.

Klingner et al. [8] in 2008 performed several experiments to measure cognitive load known as task-evoked pupillary response (TEPR) using a remote video eye tracker and suggested two extensions to the previous research in this area. First, they showed that cognitive pupillometry can be improved from head-mounted to remote eye-tracking systems. Second, they showed the feasibility of a more fine-grained approach to the assessment of pupil diameter variation captured with the help of an eye tracker, which provides further detail about the timing and magnitude of short-term pupillary responses showing cognitive load, instead of using a simple aggregate mean measurement over a long time period. They performed three experiments on 8 subjects. First they performed a mental multiplication task in which the subjects heard a multiplicand and a multiplier and performed calculation orally. Their second experiment was a short term memory task in which the subjects were asked to memorize numbers in the

exact sequence as heard. In their third task, the subjects consciously listened to a number sequence and picked three errors in it as they were made. In 2010, Klingner [9] studied the effect of aural versus visual task presentation on pupil dilation. They performed three tasks spanning a range of cognitive activities: mental multiplication, digit sequence recall, and vigilance. The patterns of dilation were similar for both aural and visual presentation for all three tasks, but the magnitudes of pupil response were greater for aural presentation. Accuracy was higher for visual presentation for mental arithmetic and digit recall. The findings can be accounted for in terms of dual codes in working memory and suggest that cognitive load is lower for visual than for aural presentation.

Wang et al. [10] in 2010 reported experiments on sender-receiver games with a reason for senders to exaggerate. Their experimental results indicate that subjects "overcommunicate" – messages are more informative of the true state than they should be, in equilibrium. Senders' pupils also dilate when they send deceptive messages (MS) and dilate more when the deception is larger in magnitude. They demonstrated that combining sender messages and proposed look-up patterns, one can predict the true state and lower the miss rate of subjects by one half. These predictions are shown to increase receiver payoffs up to 16-20 percent. Within this paradigm, eye-tracking receivers can be useful for launching their degree of strategic superiority in making inferences from messages.

Jianga et al. [11]aimed to explore how TEPR reflects mental workload changes in a surgical environment. They conducted a simulated surgical task that

has 3 different subtasks with different mental workload requirements. They found a significant effect among these different subtask groups by measuring pupil diameter change rate. Their findings may improve patient safety in a real operating room by non-intrusively monitoring the surgeon's mental workload while performing a surgery using an eye-tracking system.

Palinko et al. [12]showed that pupil diameter can be used as a physiological measure of cognitive load in driving simulator studies. They found that it is possible to dissociate the effects of light and cognitive load for subjects scanning images from a driving simulator. Kun et al. [13] in a recently published study confirmed that it is feasible to use pupil diameter to differentiate between parts of the dialogue that increase the cognitive load of the driver, and those that decrease it during a spoken dialogue with a remote conversant. Their long term goal is to build a spoken dialogue system that can adapt its behavior when the driver is under high cognitive load, whether from the driving task or the dialogue task.

As technology and services have developed, humans regularly engage in transactions in which rapid and reliable personal identification is required. Examples include [14] passport control, computer login control, automatic teller machines and other financial transaction authorizations, premises access control, and security systems for meeting the critical needs of homeland security. Speedy, reliable, automated identification procedures are especially necessary for law and counterterrorism enforcement.

2.3 High performance implementation requirements

The high performance implementation framework is developed on two requirements, time and memory issues. In this case, parallel computing is used to reduce the computation time. Memory issues are of less importance in this case. MATLAB® [15] provides several built-in tools that can help determine not only the amount of time and memory a program is taking but also which specific parts are taking the most time and memory. In the current pupil detection algorithm Hough transform for both parabola (eyelash) and ellipse detection (iris and pupil) is used. The most time consuming part of the program is the calculation for the Hough voting array[16] for both Hough parabola and ellipse.

2.4 Parallelizing Hough Transform

Chen et al. [13] proposed a new Hough Transform implementation in 2008, after analyzing the performance bottlenecks of original Hough Transform on multi-core processors. Initially, a coarse-grain and a fine-grain parallelization of a straightforward Hough Transform implementation on an 8- core machine was studied. It was found that due to parallelization overheads and memory requirements, these schemes do not fully utilize computation power. After that, they proposed a new Hough Transform implementation for parallelization. Their data showed that the new Hough Transform exposes a significant amount of concurrency and improved data locality. On the 8-core machine, the new implementation showed 25% better performance than the old ones.

Experiments have been performed on one platform with sequential program and parallel program so as to testify the efficiency of parallel algorithm. Wu et al. [16] presented another two parallel methods based on block of threads known as Thread Building Block (TBB) and CUDA. Results show that algorithms of circle detection are extremely good in consideration with the amount of time the processing takes.

Braak et al.[17]introduced two novel methods for the Hough transform on a Graphical Processing Unit, one is a fast method and other is an input-data independent method. The Cartesian or polar parameterization used for lines in images does not increase the execution speedup of the Hough transform considerably. Improving the GPU-based program for speed does result in a considerable improvement. Other way to improve the GPU code is to make it input-data independent. Results show that the fast-implementation is the better of both. An implementation that is independent of input data has the similar processing speed for each image and is quicker if the number of edge pixels go above a certain threshold.

The code for the input-data independent execution is difficult that it is very hard to make any alterations to parameters for e.g. image size. However the fast implementation does not suffer from this disadvantage. Therefore the fast implementation of the Hough transform should be used in every case where the processing time is not fixed. GPU used here already supports storing typical data types into vectors. Packing will cost a little extra hardware costs. The equivalent Hough space is much larger than the Hough space for lines, since it has three dimensions instead of two. This will create a new tradeoff between the fast and

the input-data independent approach. For the input-data independent execution, the image no longer has to be revolved, which would save over half of the processing time in the Hough transform for lines. But the Hough [16]space is much larger when detecting circles, ellipses or parabolas which will restrict the number of sub-Hough spaces to be generated and make them more inflated to add together.

2.5 Missing Links in Literature

Eye tracking and eye movement-based interaction using computer vision techniques have the potential to become an important component in future perceptual user interfaces. Instead of measuring a person's emotional reaction, pupil-dilation provides a tangible gauge to assess the cognitive reactions. To do so, two experiments will be conducted in such a fashion that a subject is answering a series of true-and-false questions and the measurements including pupil dilation, response time and errors are recorded.

As has been noted above, most of the biometric identification systems[18] are based on facial, fingerprint or iris recognition methods. Pupil variation is the new emerging technique which has recently gained a very significant and authentic effect in biometrics and other related fields.

If we consider the current market scenario then the cost of these systems is quite high (due to the high resolution IR cameras used) and efforts are being made to make this system economical. This research builds on the foundation of designing an algorithm that is compatible with a low cost eye-tracking camera.

Most of the filters are implemented computationally that make the input usable producing a high pupil detection rate.

Another distinct problem encountered was the large datasets and requirement of high processing rates for the intended applications. Hence, the second problem addressed in this thesis is to implement the designed segmentation program using high performance computing techniques. The most time consuming part of the program was identified to be the calculation of Hough voting array for both Hough parabola and ellipse. The parallel Hough transform model that was determined through literature review was characterized as follows:

- Fine Grain New Hough transform
- Cartesian parameterization,
- No shared memory and
- No use of locks

CHAPTER 3

PUPIL SEGMENTATION

3.1 Background

According to one definition given by R. Haralick and L. Shapiro[19] in 1992, "Image segmentation is the partition of an image into a set of non-overlapping regions whose union is the entire image. The purpose of segmentation is to decompose the image into parts that are meaningful with respect to a particular application."

Segmentation is defined as subdivision of an image into its component regions or objects. Segmentation stops when the regions of interest (ROI) in an application have been isolated or identified. The series of algorithmic steps applied for the segmentation of pupil from the input eye image are shown in Figure 3 and explained in this chapter in detail. Images are cropped according to the region of interest (ROI). Thresholding is a fundamental approach of segmentation that is applied as a basic step to separate regions with high intensity i.e. pupil, iris and eyelashes from regions of low intensity such as skin. Image smoothening is done next to smoothen the abrupt noisy jumps in the image called 'dots' or 'speckles'. Subsequently, image sharpening is done to fine-tune the edges. Edge-based segmentation is applied afterwards using canny edge detector to obtain a good edge pattern of the eye. Finally Region-based

segmentation is done through Hough parabola detection for the eye-lashes and Hough ellipse detection for the iris and pupil of human eye.

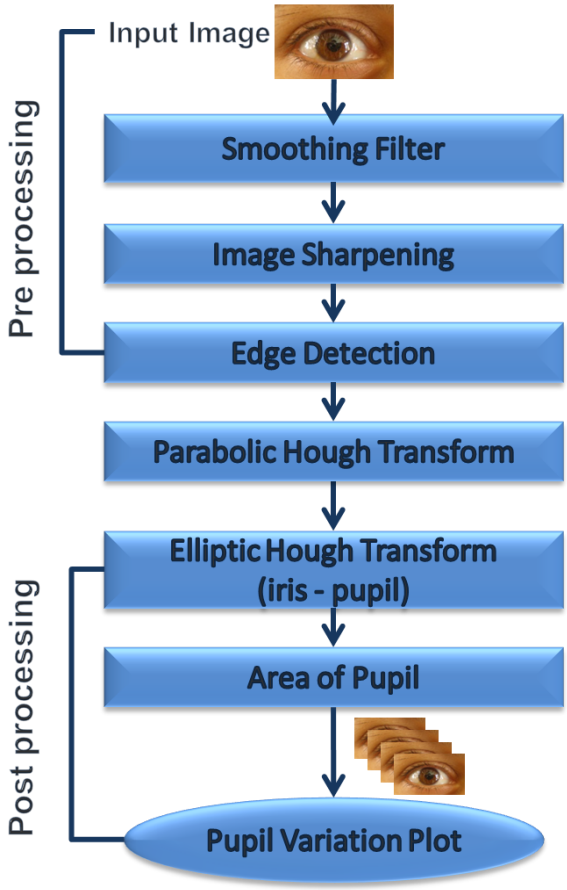


Figure 3: Segmentation steps followed through the program

3.2 Region of Interest Cropping

A Region of Interest (ROI) is a region of the image that deals with operation on image matrix of specific area. In this case, ROI is defined by rectangle-cropping of the input image frame taken from the input video. Figure 4 shows the reselt obtained after cropping the input image:

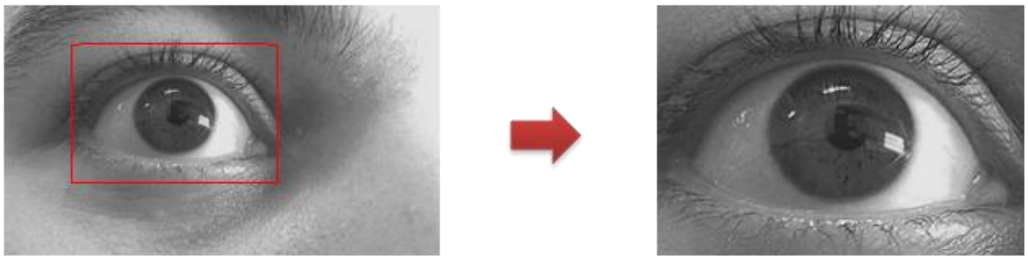


Figure 4: Image cropped from the original image obtained from the camera

3.3 Image Sharpening and Smoothening

Median filter is used to perform image smoothening as presented in Figure 5 to attenuate abrupt noise also known as 'salt and pepper' noise. Median filter is used here instead of convolution because the goal is to simultaneously reduce noise and preserve edges at the same time.

Figure 5: Input image (a), Image sharpened once (b), Image sharpened twice (c)

The main function of the median filter is to run through the image pixel by pixel, replacing each entry with the median of neighboring entries. The pattern of neighbors is called the "window", which slides, point by point, over the entire image array. The window pattern used here is the "box" pattern and the size of the window is 3 by 3. Since, the window has an odd number of entries i.e. 9; the median is simply the middle value after all the entries in the window are sorted numerically.

3.4 Canny Edge Detection

The Canny edge detector was developed in 1986 by John F. Canny. It is an edge detection function that uses a multi-stage algorithm to detect a widely specified range of edges in an image. The Canny algorithm contains a number of adjustable parameters, which can affect the computation time and effectiveness of the algorithm.

- **Size of the Median filter:** The size of median filter used in the first stage for image smoothening directly affects the results of the Canny Edge detector. A larger filter causes more blurring which can result in the smearing out of the value of a given pixel over a larger area of the image. Small filters cause less blurring and allow detection of small, sharp lines. In this case, a median filter of 3x3 dimensions is used for image smoothening in the first stage.
- **Thresholds:** We used two thresholds with hysteresis to allow more flexibility than a single-threshold approach. A threshold set too high can skip important information. Alternatively, a threshold set too low will identify false and irrelevant edges (such as noise) as important. In this case, the threshold values that produce most accurate results are set by iterative approximations on adhoc basis.

Figure 6 shows the image obtained after applying Canny edge detection and thresholding:

Figure 6: Edge map obtained after applying Canny edge detector and thresholding

3.5 Hough Transform

Hough transform[20] is a technique which is used to determine and isolate features of a particular geometric shape within an image. The conventional Hough transform requires that the desired shapes be specified in some parametric (algebraic or polar) form. It is commonly used for the detection of simple curves such as lines, circles, and ellipses within a given image.

Computationally, the Hough algorithm is implemented according to the following steps:

1) An edge detected image is taken as input and highlight all possible discontinuities in the image
2) The parameter region i.e. input image region is discretized into suitable intervals to state the size of the Hough accumulator array
3) A histogram is constructed, representing an accumulator function defined on the (discretized) parameter space: for each pixel in the parameter space, the value of the accumulator corresponds to the number of transformed curves passing through that pixel. The highest number of votes corresponds to the most voted curve, which is then identified to be the required curve.

The main advantages provided by Hough Transform are that:

- It is tolerant of occlusions i.e. gaps in the edges.
- It is reasonably unaffected by noise.
- It is unaffected by rotation and translation of the input images as the voting procedure depends on the highest number of votes and not the order of edges.

3.5.1 Parabolic Hough Transform

For eye lash detection, Parabolic Hough Transform is applied to the upper half of the input eye image using (x, y, r) points in the input 2-D image to generate and (x_0, y_0) points in the Hough transform space for parabolas. The algorithm uses the Cartesian equation for the parabola in vertex form:

$$y = r(x - x_0)^2 + y_0 \qquad (1)$$

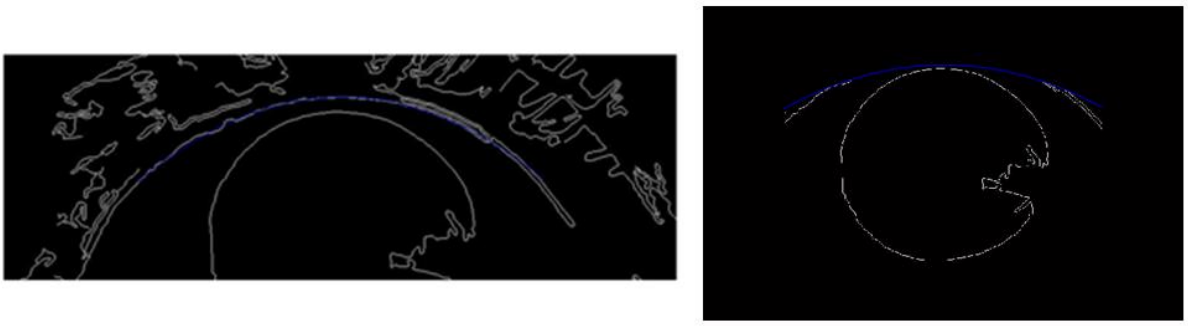
where

- 'x_0' and 'y_0' are the coordinates of the parabola vertex. The upper left corner of image is considered as the origin of coordinate system.
- 'r' is the factor that defines the shape and extent of stretching of the parabola. If 'r' is positive then parabola opens upwards like a regular 'U'. If 'r' is negative then parabola opens downwards like an upside down 'U'.If $|r| < 1$, the graph of the parabola widens. If $|r| > 1$, the graph of the parabola becomes narrow.

- Size for ‘x’ is 900 pixels, size for ‘y’ is 600 pixels and size for ‘r’ is 15. So, the Hough-array becomes of the size 600x900x15.

3.5.1.1 Explanation

The function takes edge image as an input and returns detected parabola points as output. In this case 15 parabolas are found. Figure 7 obtained after detecting the parabola and removing the extra edges is as follows:



(a) (b)

Figure 7: Detected parabola (a) and filtered extra edges (b)

Hence, Hough array for the entire image is generated a separate time to find each parabola. For each (x, y) edge point in the original image, where x and y both not zero, the algorithm calculates the first dimension value (x_0,y_0) and votes in the Accumulator as A(y_0,x_0,r)++. The algorithm then finds all the possible parabolas in the given range, votes for them in the Hough-space, and finds out the one parabola containing the largest number of votes or points. Then it finds the peak point in the (y_0,x_0,r) Hough transform space. The peak point identifies the

required parabola segment in the original image space. Figure 8 shows the detected eyelash through parabolis Hough transform:

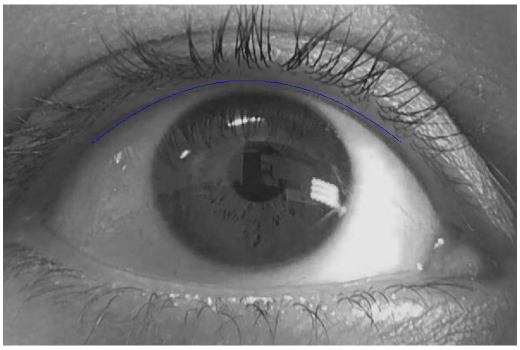


Figure 8: Detected eyelash parabola on the grayscale image

3.5.2 Elliptic Hough Transform

For iris and pupil detection, Elliptic Hough Transform is applied to the rest of the edge eye image using (x_0, y_0, a, b) parameters from the input 2-D image to generate and (x, y) points in the Hough transform space for all possible ellipses. The algorithm uses the Cartesian equation for the parabola in vertex form:

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1 \qquad \text{……………………………………………… (2)}$$

where,

- 'x_0' and 'y_0' are the coordinates of the ellipse center. The upper left corner of image is considered as the origin of coordinate system.
- 'a' and 'b' respectively represent the length of the major and minor axis of the ellipse.If $a < b$, then 'a' is considered as the major axis and if $a > b$, then 'b' is considered as the major axis.

- Size for ‘x’ is 900 pixels, size for ‘y’ is 600 pixels and for ‘a’ and ‘b’, the size of the window is 20. So, the Accumulator-array becomes of the size 600x900x400.

3.5.2.1 Explanation

The function takes edge image as an input and returns detected ellipse points as output. In our case we are checking for 400 ellipses i.e. 400 combinations of ‘a’ and ‘b’. For each (x, y) edge point in the original image, where x and y both not zero, the algorithm calculates the first dimension value (x_0,y_0) and votes in the Accumulator as A(y_0,x_0,r)++. The algorithm then finds all the possible combination of ellipses in the given range, votes for them in the Hough-space, and finds out the one ellipse containing the largest number of votes or points. Then it finds the peak point in the (y_0,x_0,r) Hough transform space. The peak point identifies the required ellipse segment in the original image space and takes it to be the true iris. The detected iris from the program is shown in Figure-9 below:

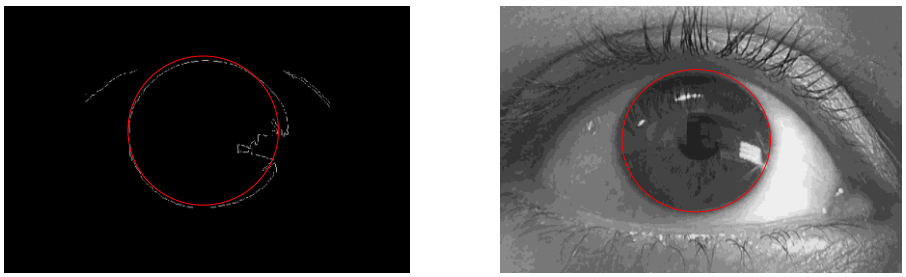


Figure 9: Detected iris on the edge-map (left) and on the original grayscale image (right)

Now we can isolate the pupil from the rest of the image from the known information that it has to be inside iris. Hence all points on and outside the

perimeters of the iris are deleted and the remaining edge image is tested for pupil. This whole process is repeated for a small value of 'a' and 'b' for the detection of pupil inside the iris. Figure 10 presents the detected pupil on the input eye image and a zoomed version for clarity:

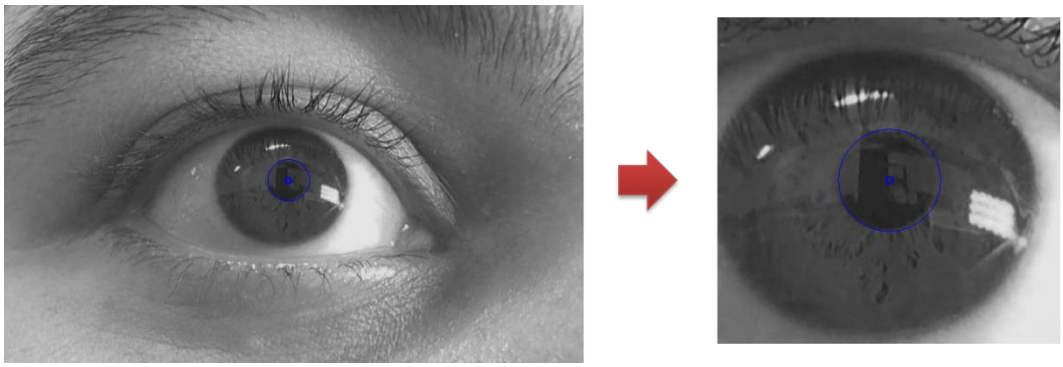


Figure 10: Detected pupil region and fitted ellipse through Hough ellipse transform

The complete eye pattern detected through the proposed methodology is shown in Figure 11 below:

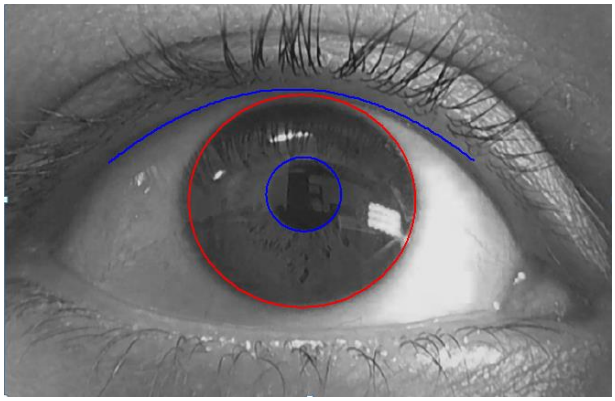


Figure 11: Complete eye pattern detected through the proposed methodology

3.6 Pupil Area and Diameter Calculation

Finally, the area of eye's pupil is calculated to plot the pupil responses under a series of cognitive stimuli in a digital video. First, each video frame is

read and processed using MATLAB's Image Acquisition feature; then, the eye's position is found on the initial frame. Later, in order to locate the boundary pixels of the pupil, above-mentioned image segmentation techniques are used to the eye's cropped image. Finally, the ellipse is fitted to the pupil using a direct Hough algorithm, and the pupil area is approximated by using ellipse area equation.

3.6.1 Pixel to mm Conversion

In digital image processing, a pixel is a physical point in an image, or we can say the smallest constituent in a displaying screen. The pixel address relates to its physical coordinates on the screen. The X coordinate is the horizontal address of any pixel or addressable point on a display screen. Similarly the Y coordinate is the vertical address of any pixel or addressable point on a display screen.

1 pixel (X or Y) = 0.264583333333334 mm

Out of the 500 images processed from 250 subjects 436 showed correctly segmented pupils, which gives the proposed algorithm 87.2% pupil detection accuracy. Figure 12 shows the segmentation results of some samples in the CASIA iris dataset.

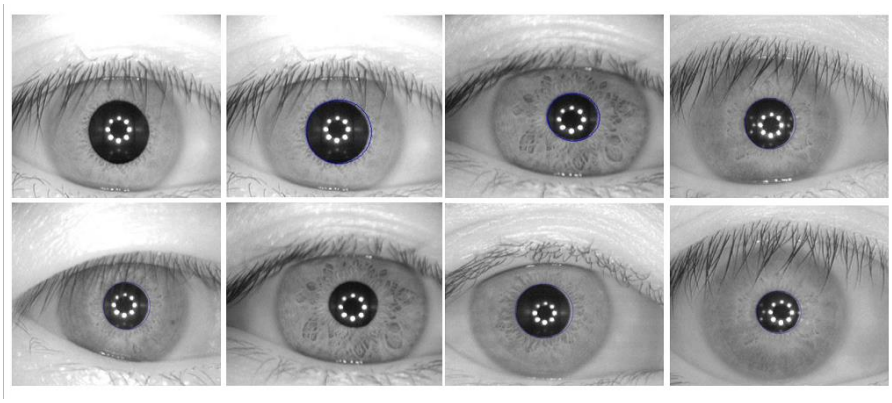


Figure 12: Segmentation results of some samples in the CASIA iris dataset

CHAPTER 4

HIGH PERFORMANCE COMPUTATION

4.1 High Performance Computer Architecture

High performance computing includes computers, networks, algorithms and environments to make the systems ranging from multicore PCs to fastest supercomputers usable. In this chapter we will analyze the parallel programming methods that can be implemented in MATLAB, the supported hardware, and the analysis procedures including speedup and efficiency.

4.2 Parallel Programming Methods in MATLAB

MATLAB® Parallel Computing Toolbox[21]can solve computationally and data-intensive problems using multicore processors, GPUs and computer clusters. High-level constructs – parallel for-loops, special array types, and well-developed GPU and MPI integrated libraries are available that allow us to parallelize MATLAB applications. In this section we will look at how the different parallel methods executed in MATLAB influence the speedup and efficiency of our segmentation code.

4.2.1 The Par-for Loop

A parallel for-loop[15] is an easy way to divide independent loop iterations of intensive computation among different workers. A par-for loop is useful in

situations where many loop iterations are required to perform simple calculations e.g. Hough Transform and Monte Carlo simulation. Par-for loops are also useful when we have loop iterations that take a long time to execute, because the workers can execute those iterations simultaneously. There are some rules of executing a par-for loop:

- A par-for loop cannot be used when iteration in our loop depends on the results of other iterations. In other words, all iterations must be independent of each other.
- Since there is a communication cost involved in a par-for loop, there might be no advantage in using it for a small number of iterations.

4.2.2 GPU Computing in MATLAB

GPU computing offers a unique application performance by offloading compute-intensive portions of the application to the GPU, while the remainder of the code still runs on the CPU. From a user's perspective, applications simply run significantly faster.

4.2.2.1 How GPU Acceleration Works ?

GPU is used to process the computationally intensive in a massive-parallel manner, hence the speed of processing is increased. GPU[22] systems use a combination of CPU and GPU to produce a co-processing model. Computationally intensive parts which need to be processed in a massively parallel manner are accelerated by the GPU in order to benefit from their high computing

performance while the CPU works on sequential algorithms. Overall, the application runs faster and the sharing of tasks makes handling of computationally-intensive calculations very efficient. The performance advantage provided by the use of graphics processing units makes this technology particularly fascinating for scientific applications. Figure 13 shows the method for using GPU acceleration in programming applications:

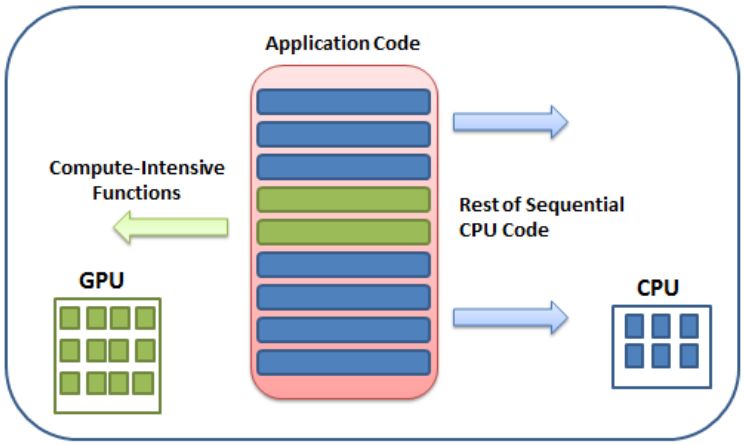


Figure 13: Working of GPU acceleration on a piece of code

4.3 Parallelizing Hough Transform

Hough Transform is accepted as a powerful technique in computer vision and pattern recognition for shape and motion analysis in images having noisy, missing, and extraneous data but it has not been adopted as much due to its high computational cost and storage complexity. After considering the performance blockages of the Hough transform on a single-core processor, a multi-core, multi-node and GPU-based Hough transform implementation for both parabolic Hough transform and elliptic Hough transform was done and results were analyzed. The

performance of Hough transform improved significantly because of the introduction of parallel computing techniques.

4.4 Speed-Up Results

A comparison and analysis of the potential speedups achieved between CPU and GPU platforms and different configurations implemented using abovementioned techniques is given below:

4.4.1 MATLAB Parallel-for workers

The easiest code to parallelize in MATLAB using all cores of a CPU is through the 'par-for' loops. Within the par-for loop each iteration is independent of all others, and the MATLAB built in scheduler, portions-out each iteration to a worker for computation. The results are then collected and returned appropriately by the scheduler. The full listings of the codes can be found in appendix [A].

4.4.1.1 Parabolic Hough Transform

For our purpose, we need to run the voting procedure for the Hough transform parallel because it takes 96% of the total computational runtime. Hence, the voting part is executed in parallel on MATLAB workers working together as a parallel pool. The input data on which parfor operates is sent from the client to workers, where most of the computation happens, and the results are sent back to the client and patched together. Table 1 shows the algorithm execution time for the detection of eyelash through sequential and parallel versions of parabolic Hough transform:

	Sequential Implementation	Number of MATLAB workers			
		2	4	6	8
CPU Runtime (s)	84.233097	70.877573	56.016808	45.05133	42.171589
CPU Speedup (x)	1x	1.1884x	1.5037x	1.8697x	1.9974x

Table 1: Serial and parallel execution times for a single frame on a multicore machine

The reduced computational runtime and speedup achieved by distributing the voting calculation on 2, 4, 6 and 8 processors of a single CPU is presented below:

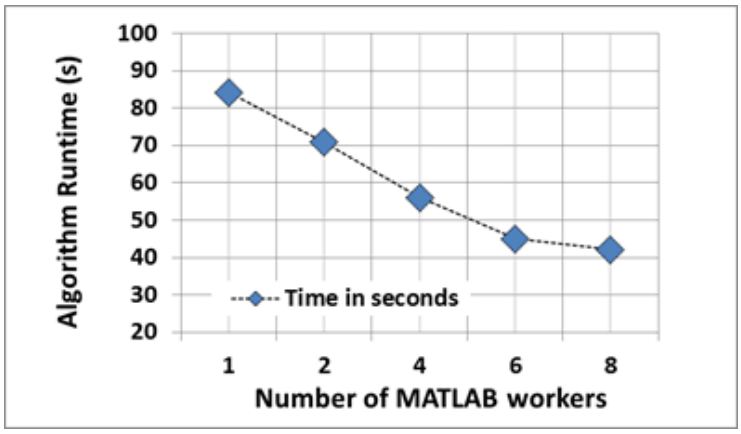


Figure 14: Parallel Runtime of Parabolic Hough Transform on an 8-core Machine

Figure 14 shows parallel runtime of parabolic Hough transform on an 8-core machine. It is clear that the runtime is reduced by increasing the number of processors.

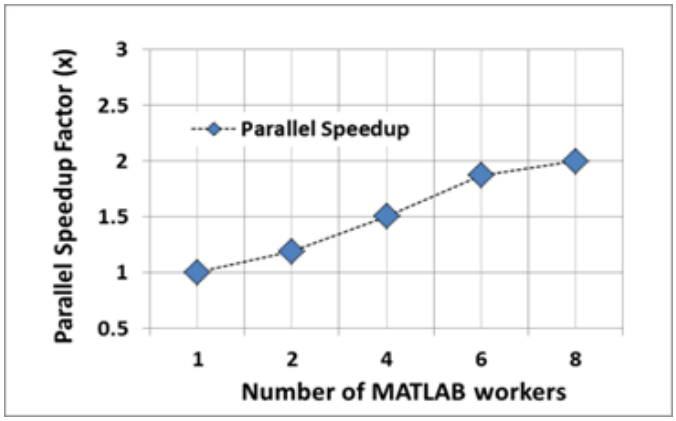


Figure 15: Parallel Speedup of Parabolic Hough Transform on 8-core Machine

Our parallel parabolic Hough Transform (PPHT) reduces the runtime of the program to half the sequential one on an 8-core machine. However, the speedup of PHT as presented in Figure 15 is 200% higher than that of PPHT, a factor of 2. Thus, our program enables data distribution on 8 cores with no performance loss. In PPHT, 8 processors take the responsibility for the maximum of 96 threads, so that the maximum of 84.23309 seconds taken for the sequential Hough Array Computation reduces to 42.17158 seconds using 8 processors which give more than 50% increase in computational speedup.

4.4.1.2 Elliptic Hough Transform

In EPHT, all 8 processors on the machine are used to calculate 400 threads for the ellipse combinations to detect the iris. Table 2 shows the runtimes and speedups achieved for 1 to 8 processors.

	Sequential Implementation	Number of MATLAB workers			
		2	4	6	8
CPU Runtime (s)	66.17994	62.780388	38.412232	31.842755	28.661424
CPU Speedup (x)	1x	1.0541x	1.7229x	2.0783x	2.309x

Table 2: Serial and parallel execution times for a single frame on a multicore machine

For example, our data distribution technique enables a single processor to distribute its workload to eight processors.

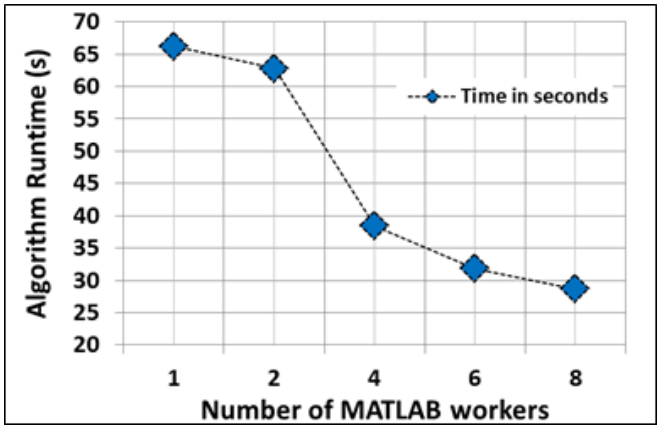


Figure 16: Parallel Runtime of Elliptic Hough Transform on an 8-core Machine

Following graph shows the increase in speedup achieved on an 8-core machine:

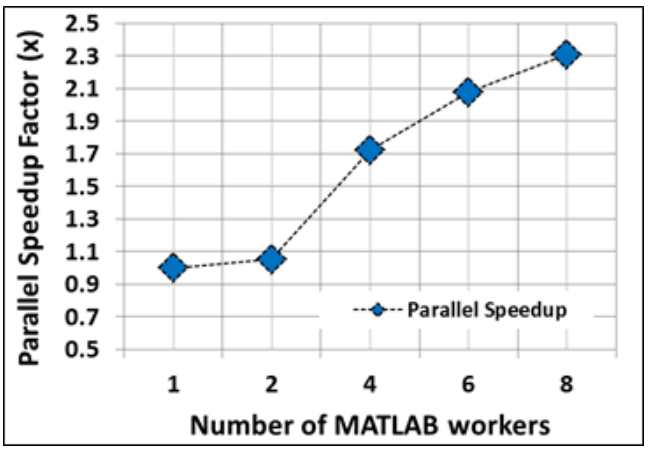


Figure 17: Parallel Speedup of Elliptlic Hough Transform on 8-core Machine

The maximum time for Hough Array Computation in EPHT is reduced from 66.17994 to 28.661424 seconds which indicates almost 2.30 times increase in computational speedup.

4.4.2 GPU-based Implementation

For the purpose of implementation of the parallel algorithm, a low end graphics processing is used which still provides good results to the speedup of the program. The GPU used in our setup is TESLA T10 448 CUDA cores running at 1.2 GHz and has 1280 MB of off-chip global memory.

4.4.2.1 Parabolic Hough Transform

Table 3 shows the algorithm execution times for the detection of eyelash through parabolic Hough transform:

CPU Runtime	CPU Speedup	CPU Runtime 8-core	CPU Speedup 8-cores	GPU Runtime	GPU Speedup
84.233097	1x	42.171589	1.9974x	32.462755	2.5948x

Table 3: Execution times and Speedups for CPU, CPU (8-cores) and GPU for parabolic Hough transform

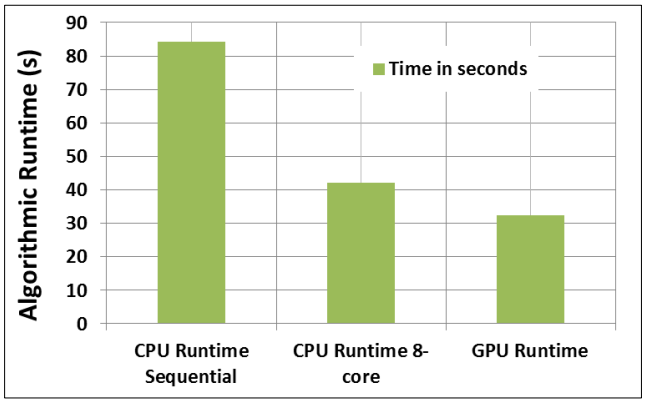


Figure 18: GPU runtime for Parabolic Hough Transform on NVIDIA Tesla T10 processor

The speedup obtained on the GPU is 2.5948 times the sequential CPU runtime and 1.2991 times the parallel 8-core runtime for eyelash detection in a single frame. This is a considerable improvement, given the small sample size of the parallelizing data for parabola detection.

4.4.2.2 Elliptic Hough Transform

Table 4 shows the algorithm execution times for the detection of eyelash through elliptic Hough transform:

CPU Runtime (s)	CPU Speedup	CPU Runtime 8-core (s)	CPU Speedup 8-cores	GPU Runtime (s)	GPU Speedup
66.17994	1x	28.661424	2.309x	26.518284	2.4956x

Table 4: Execution times and Speedups for CPU, CPU (8-cores) and GPU for elliptic Hough transform

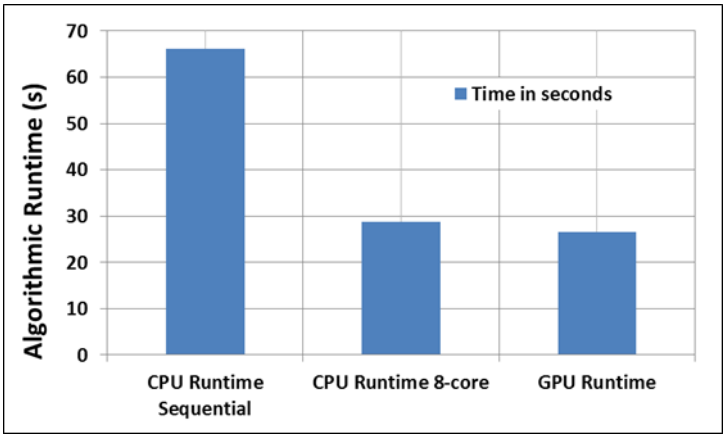


Figure 19: GPU runtime for Elliptic Hough Transform on NVIDIA Tesla T10 processor

The speedup obtained on the GPU is 2.5 times the sequential CPU runtime and 1.0336 times the parallel 8-core runtime for eyelash detection in a single frame. This is a considerable improvement, given the small sample size of the parallelizing data for parabola detection.

4.4.3 Parallelized Pupil Detection on a single eye frame

The effect of parallelizing elliptic and parabolic Hough codes on the whole pupil detection program is given for a single frame is reported in the following section.

Table 5 gives the execution times and speedups achieved with both multicore and GPU-based implementations:

	Sequential Implementation	Number of MATLAB workers			
		2	4	6	8
CPU Runtime (s)	151.948562	132.853126	90.038256	83.542407	81.43823
CPU Speedup (x)	1x	1.1437x	1.6876x	1.8188x	1.8658x

Table 5: Execution times and speedups achieved by the proposed pupil detection methodology applied on a single eye image

Our multicore implementation of the proposed program reduces the runtime of the program to almost half the sequential one on an 8-core machine. However, the speedup achieved is 86.6% higher than that of sequential program, a factor of 1.8658.

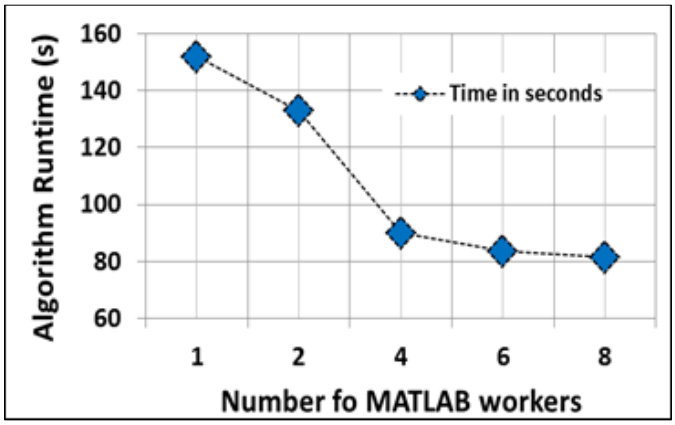


Figure 20: Parallel Runtime of proposed method on an 8-core Machine

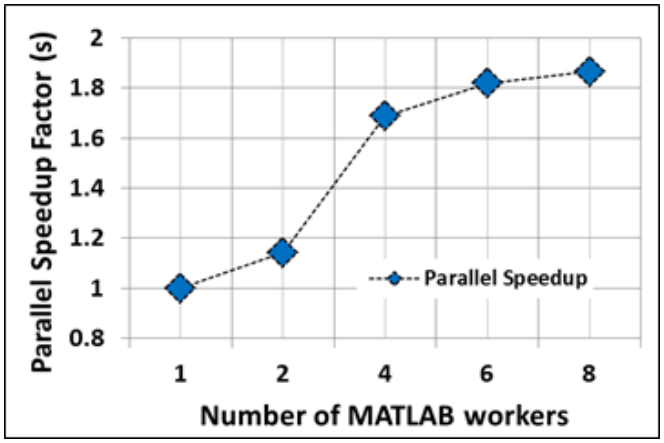


Figure 21: Parallel Runtime of proposed method on an 8-core Machine

The GPU-based implementation made the algorithm performance even better. The following table shows a comparison of runtimes and speedups achieved with sequential, multicore and GPU-based executions for the proposed program.

CPU Runtime (s)	CPU Speedup (x)	CPU Runtime 8-core (s)	CPU Speedup 8-cores (x)	GPU Runtime (s)	GPU Speedup (x)
151.948562	1x	81.43823	1.8658x	42.634876	3.5639x

Table 6: Execution times and speedups achieved for CPU, CPU-8 cores and GPU-based implementation for a single eye frame on NVIDIA Tesla T10 processor

Table 6 shows a comparison between the CPU-single core, CPU-8 core, and GPU runtimes for the proposed methodology. The multi core implementation gives us a 51.75% increase in speed. Whereas the GPU-based code provides an 85% increase in the speedup of the pupil detection program. In conclusion we can say that our GPU-based program's performance is better than CPU and multicore implementations.

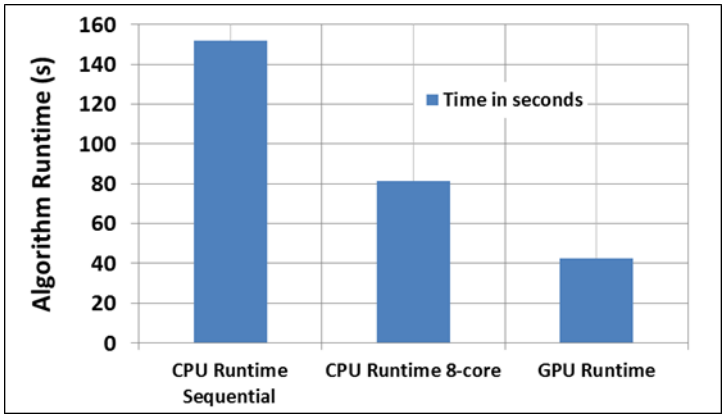


Figure 22: A comparison of CPU, CPU-8 core and GPU runtimes for proposed algorithm for a single eye frame on NVIDIA Tesla T10 processor

As Figure 12 shows, due to high performance on the GPU both competitive implementations (CPU and CPU-8 core) are less efficient. Unless the number of cores available for the multicore program is increased or a multi-node comparison is made, GPU processor provides the best efficiency results as of now.

CHAPTER 5

COGNITIVE LOAD ASSESSMENT

5.1 Background

In this work two pupil detection experiments were conducted. The developed software was used to convert and analyze the recorded pupil variation data. This chapter summarizes the results of the applied algorithm and analyzed data.

5.1.1 Participants

This research was carried out with the assistance of 2 participants. There were two different experiments. All subjects participated in all experiments. Participants were paid a rupee 100 honorarium. One of the participants was the project supervisor, one was author's friend. All of them had either normal or corrected-to-normal vision.

5.1.2 Apparatus

For all our experiments, we used a Logitech HD Pro Webcam C920, which has a sampling rate of 60 Hz. The resolution of this display was set to 1920x1080 pixels. The distance of participant's eye and the camera was approximately 0.5 m. A normal room temperature of 25^{o}C was maintained. The windows of the room were all covered to maintain a constant ambient luminance.

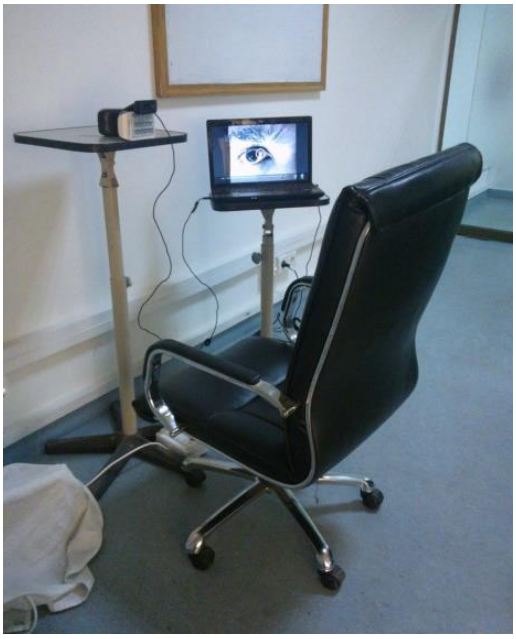


Figure 23: Experimental Setup

5.1.3 Procedure

The experimenter began each experiment by providing the participant with task instructions, positioning the subject's head to have a front and zoomed image of the subject's eye. Each trial began with the participants listening to recorded questions of a target pattern for about 10 seconds, during which the participants were to perform mental tasks and answer the questions.

5.2 Experiment Runs and Analysis

The proposed pupil detector system employs a desktop webcam which eradicates the need for distracting head-mounted equipment or chin rests. This enables non-intrusive measurement of cognitive load during normal computer use with a computer system that resembles standard desktop models. To

measure and study the task evoked pupillary response, we conducted two experiments.

5.2.1 Experiment I

The first experiment was setup as the experimenter could see the eye movements of the subject on the desktop screen. This experiment was a replica of an experiment performed by Klingner, Kumar, and Hanrahan[8]who measured the Task Evoked Pupillary Response (TEPR) for mental arithmetic. In each trial, subjects listened to two pronounced numbers between 1 and 20. Five seconds after the second number was spoken, subjects were prompted to speak out the numbers' product. The pupil video files were saved and later analyzed with the pupil variation algorithm.

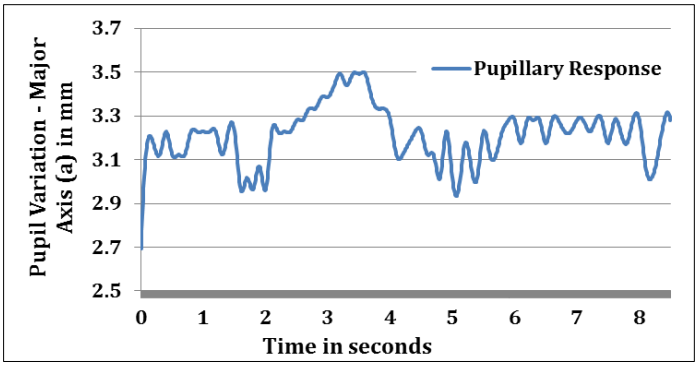


Figure 24: Pupillary response during the mental multiplication task on the major axis 'a' of the ellipse

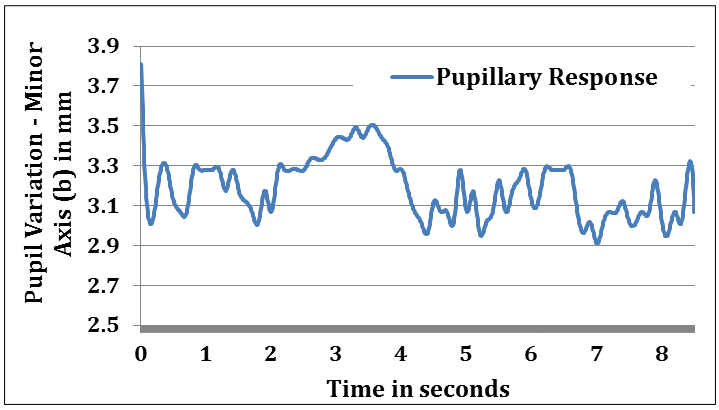


Figure 25: Pupillary response during the mental multiplication task on the minor axis 'b' of the ellipse

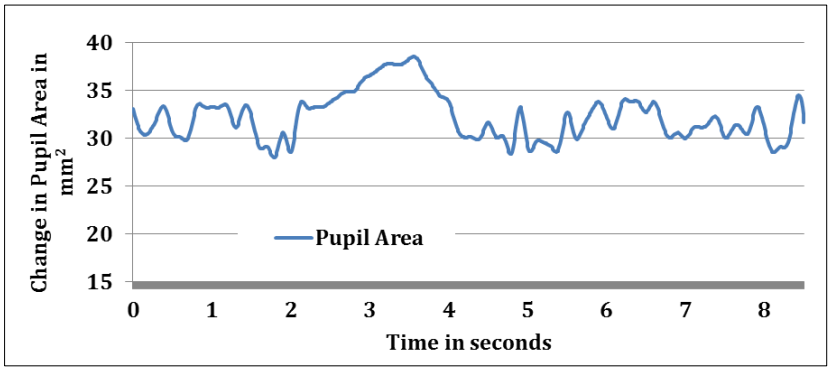


Figure 26: Pupillary Area Variation during the mental multiplication task

5.2.2 Analysis Experiment I

There is a small (0.2 mm) increase in pupil size as the subjects hear the multiplier and begin computing the product. Fig.0-4 shows the results from our replication of their experiment. Although we gave problems at all three difficulties, the easy level was the only one for which we collected sufficient correct responses for analysis. The pupillary response we observed for these easy problems resembles the prior result for medium and difficult problems.

5.2.3 Experiment II

In the second experiment, we measured the TEPR for an aural vigilance task. Subjects listened to an audio where the experimenter was counted from 1 to 12 and were told that the experimenter might make a mistake by dropping any number. Two such mistakes were inserted randomly, and subjects were required to speak out the dropped number as soon as they perceived the mistake.

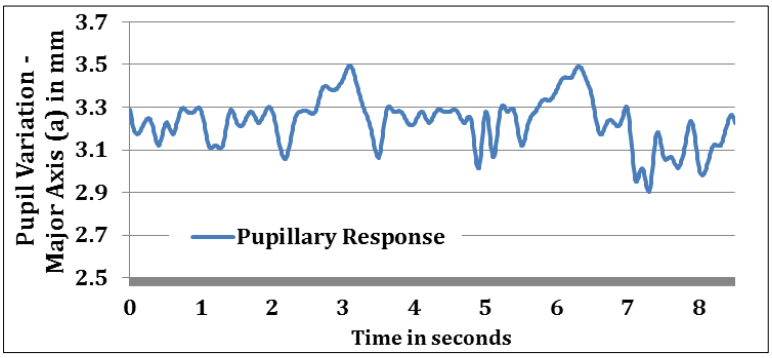


Figure 27: Pupillary response during the aural vigilance task on the major axis 'a' of the ellipse

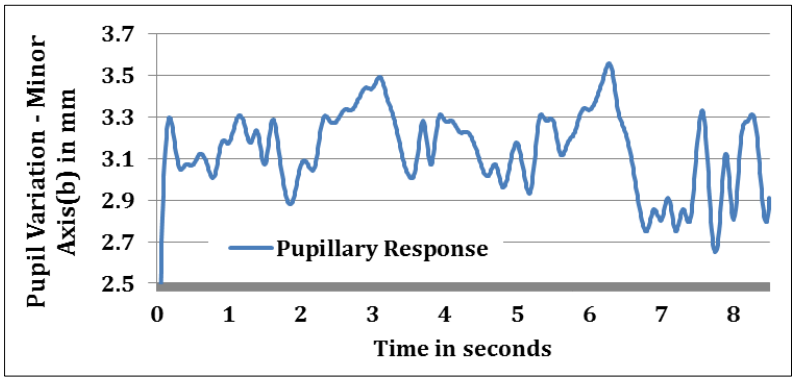


Figure 28: Pupillary response during the aural vigilance task on the minor axis 'b' of the ellipse

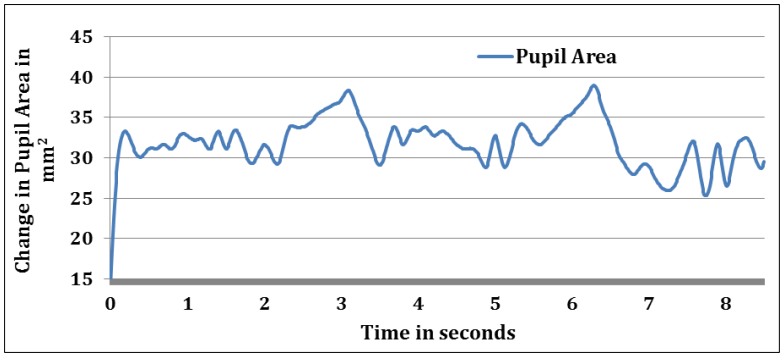


Figure 29 : Pupillary Area Variation during the aural vigilance task

5.2.4 Analysis Experiment II

Sharp spikes were observed in pupil diameter with consistent magnitude, duration, and shape following both the mistake points. Figure 29, 30 and 31 show pupillary response during the aural vigilance task. In both experiments, pupil diameter increases as the digits to be memorized are heard and encoded, peaks during the pause while they are retained, and declines as the subjects report them back.

CHAPTER 6

Conclusions & Future Work

6.1 Conclusions

The conclusions drawn from the work presented in this thesis can be divided into three major groups. The grouping is based on whether the conclusions are related to eye pattern recognition, high performance computation and human cognitive experiments.

6.1.1 Conclusions Related to Eye Pattern Segmentation

- Firstly, an automatic segmentation algorithm was presented, which would localize the pupil region from an eye image and isolate eyelid and eyelash areas. Automatic segmentation was achieved through the use of the elliptic Hough transform for localizing the iris and pupil regions, and the parabolic Hough transform for localizing occluding eyelids.
- It can be stated that segmentation is the critical stage of pupil detection, since diameter of a pupil during cognition changes by only 2.5 – 5 mm, there is a strong chance that the change in pupil diameter is falsely calculated which can corrupt the cognition results on the whole. We showed that this algorithm is effective even when applied to datasets obtained using low cost equipment and very noisy backgrounds.

- We used elliptic Hough transform for the measurement of pupil area instead of circular Hough transform to counteract the fact that a live pupil undergoes a constant state of small oscillation. Wildes[23] suggested that this situation can be avoided by checking for these small changes in pupil size between successive captures of the eye. Although, these factors are rarely considered in the open literature, they are critical to the accuracy and success of pupil detection as a biometric technology.

6.1.2 Conclusions Related to High Performance Computational Techniques

- Initially, speed was not one of the objectives for developing this system, but it was considered if the system had to be used for real-time applications. The most computation intensive stages indicated was the Houghvoting procedure. Since the system is implemented in MATLAB, speed benefits were achieved by implementing the computationally intensive parts through par-for loop, GPU and MPI programming methods.
- The results show an almost 50% speedup for parabolic Hough transform and an almost 65% speedup for elliptic Hough transform when all 8 cores of the machine are utilized instead of one which MATLAB uses by default to run sequential programs.
- A 60% speedup in parabolic Hough transform and a 70% speedup in elliptic Hough calculation is observed when the heavy computation parts of the code are shifted to GPU instead of CPU.

- The reason in both the abovementioned results being the problem size for parabolic Hough transform is smaller as compared to the elliptic Hough transform. The parabola to be detected lies in the range $0.0002<r<0.050$ which is the range set according to the physical dimension of the human eyelid. Hence, the number of combinations the Hough algorithm has to scan the input eye image is limited. Hough ellipse is designed to scan all possible combinations and find an accurate winner ellipse (pupil).

6.1.3 Conclusions Related to Human Cognition

- Most of the previous work utilizing eye trackers to estimate cognitive load has used head-mounted cameras, which provide high precision but can be cumbersome and annoying to users. Marshall[24] reported that some of her experimental subjects were bothered by wearing a head-mounted eye tracker, and that this may have distorted some of her results. Although remote systems that resemble standard desktop models typically have less precision and are subject to more measurement noise than head-mounted systems, we found that our remote eye tracker can be used for measuring cognitive load.

6.2 Future Work

In the course of designing and evaluating the pupil variation system and using resource management techniques for high-performance computing platforms, we have found several interesting issues that are still unresolved.

These section overviews some of these open issues that need further investigation:

- **Optimize Efficiency:** As the experimental results show, the proposed algorithm can detect the pupil even in tough occlusive cases. According to results, the algorithm is able to detect the pupil boundary when 50% of its contour is visible by the camera. An improved version of the algorithm can be implemented where the pupil is detected and the diameter is measured accurately with 20-30% of the contour visible in the edge map.
- **Optimize Speedup:** We have implemented the algorithm via high performance techniques using par-for loop and GPU. All these techniques include at least one for/par-for loop to calculate the Hough voting space. A completely vectored implementation for multicore, GPU and MPI can be done and to achieve an optimized speed up for both parabolic and elliptic Hough transform. Several other configurations such as GPU+par-for, Multi-GPU, GPU+MPI, par-for+MPI can be programmed and analyzed for speed and performance.

BIBLIOGRAPHY

1. Pomplun, M.a.S., Sindhura., *Pupil Dilation as an Indicator of Cognitive Workload in Human-Computer Interaction.* Proceedings of the International Conference on HCI, 2003.
2. Beatty, J., Lucero-Wagoner, Brennis., *Handbook of Psychophysiology*, in *The Pupillary System, Chapter 6*. 2000, Cambridge University Press.
3. Iani, C., Gopher, D., Grunwald, A. J., & Lavie, P., *Peripheral arterial tone as an on-line measure of load in a simulated flight task.* Ergonomic, 2007.
4. Andreassi, J.L., *Psychophysiology*, in *Human Behavior and Physiological Response*. 2000, Erlbaum Associates, Incorporated, Lawrence.
5. Davson, H. *human eye*. Encyclopedia Brittanica 2013; Available from: http://www.britannica.com/EBchecked/topic/1688997/human-eye.
6. Bayer, M., Sommer, Werner., Schacht, Annekathrin., *Emotional words impact the mind but not the body: Evidence from pupillary responses.* Society for Psychophysiological Research, 2011.
7. Partalaa, T., Surakka, Veikko., *Pupil size variation as an indication of affective processing.* International Journal of Human-Computer Studies, 2003.
8. Klingner, J.a.K., Rakshit and Hanrahan, Pat. *Measuring the Task-Evoked Pupillary Response with a Remote Eye Tracker*. in *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research & applications*. 2008: ACM.
9. Klingner, J., Tversky, Barbara. Hanrahan, Pat., *Effects of visual and verbal presentation on cognitive load in vigilance, memory, and arithmetic tasks.* Society for Psychophysiological Research, 2010.
10. Wang, J.T.-Y., Spezio, Michael., Camerer, Colin F., *Pinocchio's Pupil: Using Eyetracking and Pupil Dilation to Understand Truth Telling and Deception in Sender-Receiver Games.* American Economic Review, 2006.
11. Jianga, X., Zhengc, Bin., Tienb, Geoffrey., Atkins, M. Stella *Pupil Response to Precision in Surgical Task Execution* Studies in health technology and informatics, 2012.
12. Palinko, O.a.K., Andrew. . *Exploring the influence of light and cognitive load on pupil diameter in driving simulator studies* in *Proceedings of the Sixth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design* 2011.
13. Kun, A.L., Palinko, Oskar., Medenica, Zeljko., Heeman, Peter A. *On the Feasibility of Using Pupil Diameter to Estimate Cognitive Load Changes for In-Vehicle Spoken Dialogues* in *Proceedings of Interspeech*. 2013.
14. Gao, W., Kemao, Qian., Wang,Haixia., Lin, Feng., Seah, Hock Soon., *Parallel computing for fringe pattern processing: A multicore CPU approach in MATLAB® environment*, in *Optics and Lasers in Engineering*. 2009.
15. Sharma, G., and Martin, Jos, *MATLAB®: a language for parallel computing.* International Journal of Parallel Programming, 2009.

16. Wu, S.a.L., Xiangjiao, *Parallelization research of circle detection based on Hough transform.* International Journal of Computer Science Issues, IJCSI, 2012.
17. Braak, G.-J.v.d., Nugteren, Cedric., Mesman, Bart. and Corporaal, Henk, *Fast Hough Transform on GPUs: Exploration of Algorithm Trade-offs*, in *Advances Concepts for Intelligent Vision Systems*. 2011.
18. Masek, L., *Recognition of human iris patterns for biometric identification*. 2003, University of Western Australia.
19. Haralick, R.M. and L.G. Shapiro, *Computer and robot visión*. 1992: Addison-Wesley Pub. Co.
20. Gonzalez, R.C.a.W., Richard E., *Digital Image Processing*. 2007.
21. MathWorks, *MATLAB Parallel Computing Toolbox*.
22. Suh Jung, W.a.K., Youngmin., ed. *Accelerating MATLAB with GPU Computing: A Primer with Examples*

2014.
23. Wildes, R.P. *Iris recognition: an emerging biometric technology*. in *Proceedings of the IEEE*. 1997.
24. Marshall, S.P. *The Index of Cognitive Activity: Measuring cognitive workload*. in *Proceedings of the IEEE Conference on Human Factors and Power Plants*. 2002.

Appendix A

MATLAB Program Codes

---

1. Parabolic Hough Transform

```
function [w1,w2,Ep] = houghparabola(Ep)
[npmp] = size(Ep)
E = imcrop(Ep,[1 50 mp-1 np/2]);
rvals = 0.0002:0.0002:0.0048;
A = zeros(n,m,R);
Index = find(E);
forcnt = 1:length(Index(1,:))
for r=1:R
for x0 = 1:m
y0 = round(Index(1,cnt)-rvals(r)*(Index(2,cnt)-x0)^2);
if y0 < n && y0 >= 1
A(y0,x0,r) = A(y0,x0,r)+1;
end
end
end
end
cnt = 1:R;
H(cnt) = max(max(A(:,:,cnt)));
[maxval, maxind] = max(H);
```

2. Elliptic Hough Transform

```
function [stptx,stpty,ea,eb] = houghellipse(I,mina,minb,maxa,maxb)
[edgeredgec]=find(I);
windowa=maxa-mina+1;            windowb=maxb-minb+1;
maxab = windowa*windowb;
Accumulator = zeros(row,col,maxab);frame=1;
for a = mina:maxa;
a2 = a^2;
for b = minb:maxb;
b2 = b^2;
ratio=b2/a2;
foredg = 1:length(edgec);
xCenter = edgec(edg);      yCenter = edger(edg);
x = xmx:xpx;
y = real(round(yCenter - sqrt(b2 - (ratio*(xCenter - x)).^2)));
if(x>0 & x<col & y>0 & y<row)
    Accumulator(y,x,frame) = Accumulator(y,x,frame)+1;
end
if(frame<maxab)
frame=frame+1;
```

```
end
cnt = 1:maxab;
H(cnt) = max(max(Accumulator(:,:,cnt)));
startptx = mX-aa;          startpty = mY-bb;
ellipsea=2*aa;             ellipseb=2*bb;
rectangle('Position',[stptxstptyeaeb],'Curvature',[1,1],'EdgeColor','
r');
```

3. Parabolic Hough Transform with Parallel-for

```
function[ w1,w2,Ep ] = houghparabola_par(Ep,E)
[npmp] = size(Ep)
imshow(Ep);
E                =imcrop(Ep,[1 50 mp-1 (np/2)]);
[n m]            = size(E);
rvals            = [0.0002 0.0008 0.0016 0.0024 0.032 0.0040 0.050];
R                = length(rvals);
A                = zeros(n,m,R);
[yIndexxIndex] = find(E);
x0          = 1:m;
xIndexrep   = repmat(xIndex,1,m);

parfor r=1:R
n_temp=n;
m_temp=m;
xIndex_temp    = xIndex;
rvals_temp     = rvals;
rv             = rvals_temp(r);
    b         =rv.*(((bsxfun(@minus,xIndexrep_temp,x0rep_temp)).^2));
    y01       = round(bsxfun(@minus,yIndexrep_temp,b));
lencol    = m_temp;
lenrow    = length(xIndex_temp);
x0plot  = x0rep(i,j);
y0plot  = y0(i,j);
tmp_A   = zeros(n_temp,m_temp);
if (x0plot>=1 & y0plot>=1)
tmp_A(y0plot,x0plot) = tmp_A(y0plot,x0plot) + 1;
end
A(:,:,r) = A(:,:,r) + tmp_A;
fprintf ( 1, ' Thread %d executed on core # %d\n', r, t.ID );
end
H(cnt) = max(max(A(:,:,cnt)));
[maxval, maxind] = max(H);
figure, imshow(Ep); hold on;
plot(x((m/5):((4*m)/5)),y((m/5):((4*m)/5)),'Color','b');
end
```

4. Elliptic Hough Transform with Parallel-for

```
function [startptx,startpty,ellipsea,ellipseb] =
houghellipse_par(I,mina,minb,maxa,maxb)
[edgeredgec]= find(I);
[row col]   = size(I);
```

```
Accumulator = zeros(row,col,maxab);
parfor frame=1:length(combab)

combab_temp = combab;
    a2        = (combab_temp(frame,1))^2;
    b2        = (combab_temp(frame,2))^2;
ratio    = b2/a2;
edgenum  = 1:length(edge(:,1));
    x         = x0(edgenum,1):x0(edgenum,2);
    b         = ratio.*(((bsxfun(@minus,edgecrep,xrep)).^2));
y        = real(round(bsxfun(@minus,edgerrep,b)));
xplot= xrep(ie,je);
yplot    = y(ie,je);
temp_Accumulator = zeros(row,col);

if(xplot>0 &xplot<col &yplot>0 &yplot<row &ynplot>0 &ynplot<row)
temp_Accumulator(y,x) = temp_Accumulator(y,x) + 1;
end

Accumulator(:,:,frame) = Accumulator(:,:,frame) + temp_Accumulator;
fprintf ( 1, ' Thread %d executed on core # %d\n', frame, t.ID );
end
H(cnt)           = max(max(Accumulator(:,:,cnt)));
[maxval, maxind] = max(H);
ellipsea = 2*aa;            ellipseb = 2*bb;
rectangle('Position',[startptxstartpty-
offstellipseaellipseb],'Curvature',[1,1],'EdgeColor','r');
end
```

5. Parabolic Hough Transform with GPU

```
function [ w1,w2,Ep ] = houghparabola_gpu(Ep,E)
Epg = gpuArray(Ep);
[np, mp] = size(Epg)
rectangle('Position', [1 50 mp-1 np/2],'EdgeColor', 'r');
[n, m]           = size(E);
rvals           = [0.0002 0.0008 0.0016 0.0024 0.032 0.0040 0.050];
R               = length(rvals);
A               = zeros(n,m,R);
Ag              = gpuArray(A);
[yIndex, xIndex] = find(E);
xIndexrep   = repmat(xIndex,1,m);
yIndexrep   = repmat(yIndex,1,m);


for r=1:R
xIndex_temp    = xIndex;
rv             = rvals;
    b          = rv.*(((bsxfun(@minus,xIndexrep,x0rep)).^2));
    y01        = round(bsxfun(@minus,yIndexrep,b));
lencol    = m;
lenrow    = length(xIndex_temp);
    x0plot     = x0rep(i,j);
    y0plot     = y0(i,j);
    gx0plot    = gather(x0plot);
```

```
    gy0plot   = gather(y0plot);
if (gx0plot>=1 & gy0plot>=1)
Ag(y0plot,x0plot,r) = Ag(y0plot,x0plot,r) + 1;
end
end

cnt = 1:R;
H(cnt) = max(max(A(:,:,cnt)));
[maxval, maxind] = max(H);
gEp=gather(Epg);
figure, imshow(gEp); hold on;
plot(x((m/5):((4*m)/5)),y((m/5):((4*m)/5)),'Color','b');
imwrite(gEp,'figure1.jpg');
end
```

6. Elliptic Hough Transform with GPU

```
function [startptx,startpty,ellipsea,ellipseb] =
houghellipse_gpu(Im,mina,minb,maxa,maxb)
edge = find(I);
[row, col]   = size(I);
Accumulator = zeros(row,col,maxab);
Accumulatorg = gpuArray(Accumulator);
a      = mina:maxa;
b      = minb:maxb;

for frame=1:length(combab)

    a2       = (combab(frame,1)).^2;
    b2       = (combab(frame,2)).^2;
ratio    = b2./a2;
xpx      = edge(:,2) + combab(frame,1);
xmx      = edge(:,2) - combab(frame,1);
  x        = xmx:xpx;
gx=gpuArray(x);
edgeg=gpuArray(edge);
gx(gx<=0)  = 0;
xrep     = repmat(x,size(edgeg(:,2)),1);
edgerrep = repmat(edgeg(:,1),1,xlen);
b         = ratio.*(((bsxfun(@minus,edgecrep,xrep)).^2));
y         = real(round(bsxfun(@minus,edgerrep,b)));

if(gxplot>0 &gxplot<col &gyplot>0 &gyplot<row &gynplot>0
&gynplot<row)
Accumulatorg(y,x,frame) = Accumulatorg(y,x,frame) + 1;
end
end
gAccumulator=gather(Accumulatorg);
H(cnt)           = max(max(gAccumulator(:,:,cnt)));
gH=gather(H);
[maxval, maxind] = max(gH);
ellipsea = 2*aaellipseb = 2*bb
rectangle('Position',[startptxstartptyellipseaellipseb],'Curvature',[
1,1],'EdgeColor','r');
end
```