Real Time Human Computer Interface Application Based on Eye Gaze Tracking and Head Detection

M.Sc. Thesis

in

Mechanical Engineering

University of Gaziantep

Supervisor

Prof. Dr. Sadettin KAPUCU

by

Sinan KESKİN

March 2018

REPUBLIC OF TURKEY

UNIVERSITY OF GAZİANTEP

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

MECHANICAL ENGINEERING

Name of the thesis: Real Time Human Computer Interface Application Based on Eye Gaze Tracking and Head Detection

Name of the student: Sinan KESKİN

Exam date: 15.03.2018

Approval of the Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. A.Necmettin YAZICI

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Mehmet Sait SÖYLEMEZ

Head of Department

This is to certify that we have read this thesis and that in our consensus opinion it is fully adequate in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Prof.Dr Sadettin KAPUCU

Supervisor

Examining Committee Members	İmzası
Prof.Dr. Sedat BAYSEÇ	[illegible]
Prof.Dr. Sadettin KAPUCU	[illegible]
Prof.Dr. L. Canan DÜLGER	[illegible]
Asst.Prof.Dr. Çağlar CONKER	[illegible]
Asst.Prof.Dr. Ali KILIÇ	[illegible]

ABSTRACT

REAL TIME HUMAN COMPUTER INTERFACE APPLICATION BASED ON EYE GAZE TRACKING AND HEAD DETECTION

KESKİN, Sinan
M.Sc. in Mechanical Engineering
Supervisor: Prof. Dr. Sadettin KAPUCU
March 2018
72 Pages

The localization of eye centers and the knowledge of eye movements is becoming an important search area on human-computer interactions with increasing developments in the computer vision branch. As software, obtaining the position of the small region which the user look on the screen is a simple and efficient alternative way for use of mice and touchy devices to control the cursor.

In this study, firstly, Viola Jones Algorithm is used to detect eye areas through acquired webcam image on Matlab platform. Then, undistorting that is for making the eye center points more accurate, a camera calibration algorithm is included. Grey level and morphology level operations are achieved step by step. Eye centers are described using Circular Hough Transform Method.

Finally, an application of kids interface is developed. Letters, objects, colors, numbers become to be learned simply and funny by this interface. The applicability of the proposed algorithm is tested and gathered results are presented.

Key Words: Eye Gaze interface, eye detecting and tracking, computer vision, Viola Jones Algorithm, Hough Transform

ÖZET

GÖZ TAKİBİ VE BAŞ HAREKETİ SAPTAMIYLA GERÇEK ZAMANLI İNSAN BİLGİSAYAR ARAYÜZÜ UYGULAMASI

KESKİN, Sinan
Yüksek Lisans Tezi, Makine Müh. Bölümü
Tez Yöneticisi: Prof. Dr. Sadettin KAPUCU
Mart 2018
72 Sayfa

Göz merkezlerinin tayini ve göz hareketlerinin bilgisi bilgisayar da görü dalında artan gelişmelerle birlikte insan bilgisayar etkileşiminde önemli bir araştırma konusu haline gelmiştir.Yazılımsal olarak, kullanıcının ekranda bakmış olduğu küçük bölgelerin pozisyonunun belirlenmesi imleci kontrol etmek için fare yada dokumatik cihazların kullanımına basit ve etkili bir alternatif olmaktadır.

Bu çalışmada, ilk olarak Viola Jones Algoritması Matlab platformunda web kameradan elde edilmiş imaj üzerinden göz bölgelerinin saptanması için kullanılmıştır.Sonra, imaj düzeltme (Göz merkez noktalarını daha hassas yapmak kamera kalibrasyonu algoritması dahil edilmiştir.), gri seviye ve morfolojik seviye işlemler sırasıyla gerçekleştirilmiştir. Göz merkezleri dairesel Hough transformasyonu metoduyla belirlenmiştir.

Son olarak ise bir çocuk arayüz uygulaması geliştirilmiştir. Harfler, nesneler, renkler ve rakamlar bu arayüz ile daha kolay ve eğlenceli bir şekilde öğrenilir olmuştur. Önerilen algoritmanın uygulanabilirliği test edilmiş ve elde edilen sonuçlar sunulmuştur.

Anahtar Kelimeler: Göz arayüzü, göz saptama ve takibi, bilgisayarda görü, Viola-Jones algoritması, Hough transformasyonu

To gaws…..

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Prof. Dr. Sadettin KAPUCU and Asst. Prof. Dr. Ali KILIÇ without their guidance, motivation, and encouragement this thesis would not have become a reality.They motivated me to pursue my studies in all stages of the thesis.

I would also like to express the warmest gratitude to my friends for their moral support.

This study would have never been completed without the moral support, continuous help and encouragement of my dearest family. Therefore, my very special thanks are due to them.

.

TABLE OF CONTENT

Page

LIST OF FIGURES

LIST OF ABBREVIATIONS

EEG	Electroencephalography
HCI	Human Computer Interface
RSI	Repetitive Strain Injury
HMI	Human Machine Interface
GUI	Graphical User Interface
EOG	Electrooculography
GLHS	Generalized Lightness Hue, Saturation Model
HDR	High Dynamic Ratio
IDE	Integrated Development Environment
GPU	Graphical Processing Unit
LCN	Local Contrast Normalization
CHT	Circular Hough Transform
CV	Computer Vision
HOG	Histogram Of Oriented Gradients
LBP	Local Binary Pattern
RTOS	Real Time Operating System
RGB	Red, Green Blue Color Space

CHAPTER 1

INTRODUCTION

1.1 Introduction

Recently, increasing the variety of computer-based products and the production of high-speed processors has triggered a growing interest and focus of work in developing natural interaction between human and computer. Getting the knowledge of the position or region that user focus on the screen solves a lot of problems. So far, some techniques as hand-arm, head-body movements, voice recognition, EEG signals, brain control interfaces and vision-based software's are compete in a better interaction as simple, cheap, efficient and comfortable for a use of Human Computer interface (HCI).

Human-computer interaction mainly grouped into two categories as hardware based and software based. But these two groups cannot be considered quite different from each other. Software (vision) type systems are generally more convenient to develop.

In last decades, several devices such as phones, tablets, TVs get progress in object oriented and vision based programming issues. This provides people to become more sociable and have an opportunity to contact people from all over the world on the same media.

Nowadays, almost all students have smartphones for communication and entertainment. This can be usable for teacher and parent interaction. It is happened on the web platform to give kids the best education. At this point, instructive and enjoyable solving techniques to home works, tutorials becoming more and more important. A user, especially young people aged 5-20, looks at the screen and does homework's like playing puzzles. Also, the sounds are effective to make learning easy. Developers keep that stuffs in mind as helpers of educationalists.

That developer-educationalist cooperation can be provided with web technology in a better way and kids who get difficulty in reading, or learning letters and languages get progress on it. These days, there are a lot of vision-based applications that use image processing and Computer vision techniques succeed in communication industries. While touchscreen technology uses hands, alternatively, this control of processor event is about to be achieved by eye localization technique. The eye first detected, tracked and decided where the user looks as information of control. Eye tracking gives also information how long, where and when a user looks at which region on the screen. This information gives statistical data for advertisements, human psychologies and interests.

1.2 Human Computer Interaction

Human-computer interaction turned out by different methods for years. Mouse, joysticks, sound processing techniques, vision-based techniques and brainwaves based controls are all used as very modern technic in a time of history. Brainwaves are not ready for today’s technology. So, eye tracking systems are accepted very modern alternative for any other way of communication. Touch screens are expensive, sound processing methods are hard to apply depending on environmental conditions. Eye tracking devices as hardware are not comfortable because of being wearable products. Eye gaze plays an important role in human communications. Actually, there is a slogan like “What we look is actually what we get”. Eye position informs us where to look. This simply sums up the software. First, detecting the eye and taking the positions from one point to another means where the human look. This principle results in many eye detecting and tracking approaches to compete.

Eye tracking approaches have a huge diversity on this area and they are separated into two main groups as hardware based and software based. (Discussed in chapter 2)

1.3 Eye Gaze as a Tool

Eye-gaze interface looks hopeful as a new assistive technique. In general, people with disabilities who are unable to move any organ except their own eyes, especially students who have difficulty reading numbers of students, students and persons, language-scientists who memorize vocabulary easily and enjoyable. The system is simple and very fast. For this reason, the algorithm should be of a very good design in terms of the user itself and especially the help of the disabled peoples and children. Eye gaze HCI presents several benefits to users:

- *Ease to use*

  Eye gaze tracking combines the hand movements for mouse and keyboard use over the scene and the eye movements on the screen earlier and performs the action with eye movements only so that the user can get rid of the arm and elbow pain. There is no extra load on the eye. There is a calibration process of several seconds.
- *Reliable interaction*

  Especially, when using the touchscreen, such as smartphones, tablets. It can get rid of any false keystrokes. Random searches, unwanted page entries are prevented. Because every part of the tracking application is software, so is the situation caused by the slow operation of the processor on the computer and the processor and computer become more efficient.
- *Clean interface*

  It would be beneficial for people with disabilities to use the eye-gaze interface of people injured or injured in places that require high hygiene such as hospitals. In places where common computer use is common in various places, such as internet cafes, the use of the same devices will take place with the entry of the eye-gaze interface.
- *Maintenance free*

  The vision-based eye tracking system does not require any maintenance because it does not use any device. Cleaning and repair of devices such as a mouse, keyboards are not possible. This is especially good way on behalf of protecting devices from kid's vandalism. Kids can only use their eyes on the screen and use their eyes with no harm.
- *Interaction speed-up*

  Eyes are bodies that perform the fastest physical movements in the human body. If the methods of gaze-writing are slow, the eye-gaze tracking method will provide significant benefits for improving it.

- *Remote control*

  Increasing zoom cameras and high-resolution camcorders will allow computers to be controlled from a distance of 100-150 cm, not 30-40 cm. So, the user's eye health will be more secure. The user will be able to use the computer more effectively and for longer periods of time.

- *User identification as mentality*

  Eyes tell a lot of things about the user emotions and interests. For example, as an internet seller, some programs can detect what colors, designs; shapes are more suitable to sell. Eye gazes provide a huge database for sales on the internet. There is a huge development on the artificial intelligence in today's century. In the next few decades, computers that recognize the user and work according to the user's tastes, interests and expectations will be produced.

  In addition, the user will react and perform both mentally (sad, happy, excited, nervous) and physically (sleepless, sleeping or sleeping, looking at the screen or not). On the other hand, there are a few negative aspects of using Eye-Gaze Tracking method. These are:

- *Saccading issue*

  A saccade is a quick, simultaneous movement of both eyes between two or more phases of fixation in the same direction [1]. Eyes as defined build unexpected movements itself. To overcome this saccading issue, it is necessary computers, cameras are much more expensive, more sensitive and at higher speeds. In this study, the idea of saccading is accepted at a very sensitive level and is neglected since the application is completed with fewer resolution things.

- *Midas touch problem*

  It is accepted as one of the biggest challenges for eye-gaze interaction as known "Midas touch problem". The problem is simply defined eyes are inspecting the object on the screen or starting an action. Misinterpretation of the looks, i.e. gestures can cause unexpected actions to use the interface. The developer of an eye gaze interaction system should define whether the looking contain an intention or natural movements as a reflex, distraction mode, blinking issues.

- *Repetitive strain injury* (*RSI)*

  RSI is defined a physical injury to muscles or muscle groups that may be caused by repetitive tasks, forceful exertions. The problem is also eye muscles can be caused. This has to be taken seriously for any constant and repetitive movement study of eyes.

In a paper, in 2000, Sibert and Jacob [2], declared that eye gaze interaction is useful input for an HMI and can be designed advance programs in the future. In this study, eyes natural movements are studied and affirmed that eyes should be used for the interface but not become an unnatural movements machine.

In this dissertation, we investigate how an eye gaze tracking technique becomes a simple, accurate and funny way of communication and socializing.

1.4 Objectives and Methods

In this thesis, an application for kids training on the Pc without using special device is designed and application is studied in an uncontrolled environment. Obviously, the most difficulty in developing an application for kids training is to make it a feasible, funny and hardware independent.

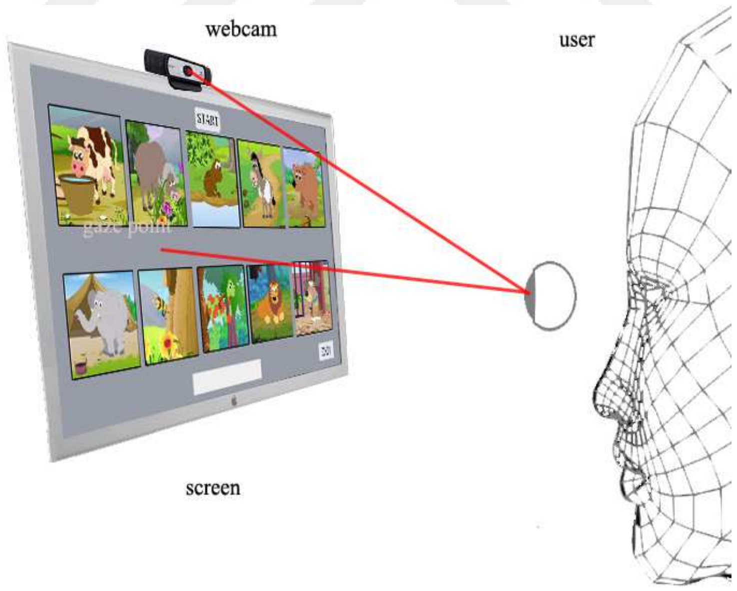


Figure 1.1 Setup for eye gaze tracking

In this study, the main objective is to design an application for kids to teach objects like animals using an interface HCI with respect to the gaze points of the mouse. The vision-based software approaches for precise and accurate eye detection are generally regarded as 5 categories [3].

- Eye template matching method, the kernel acts as a sliding window and is considered the center of maximum matching.   .
- Eye shape, edge, corner characteristics.
- Machine learning methods are used to decide whether or not all the regions in the image are an optimal match on a plane.
- Parallax regions reflected in pupil with infrared lights.
- Studies on the color and brightness of the eye.

Appearance-based methods are done with traditional computer vision techniques or some training algorithms [5,7].These technics require building a dataset for a very great perceptivity of eyes with different variations. Eyes strategic location color, shape and environment conditions make this problem very challenging. The appearance-based method is very popular for detection any object but not recommendable to attain precise coordinates.

Feature-based methods use the geometry, shape, color, texture of the eye. These approaches are effectively applicable depending on the resolution of the cameras. Snake algorithms [8], isophote curvatures [9], skin color subtraction [10] all need very close camera captures or high-resolution webcams. In this study, an algorithm to detect eye circles and control gaze points, efficient, fast and funny for a kid training game is presented. Both right and left iris areas are localized with using a low-resolution camera. The overview of the proposed Eye Tracking Game is illustrated in Figure 1.1. The system objects consist of a webcam and a lap top. A user, staying away from the monitor 50-80 cm, looks at the top-left corner before bottom-right corner for the calibration step by step, and then the game is ready for a play. Proposed algorithms are analyzed different scenarios like vertical, horizontal and diagonal gaze situations. So this system cannot be used with eyeglass. Head movements are excluded and lighting conditions are not feasible for any conditions.

1.5 Scope

The most common limitation for an eye gaze tracking application is to get the precise data of coordinate of eyes. Many studies using image process gives the result of eye gaze as a direction with up, down, right, left. That data is not enough for a kid training system. So that many possibilities that impair the algorithm is studied in this dissertation but not for a higher educational level. As a nature of the user, some aspects such as ethnicity, gender are not included. This study is not suitable for outdoor environments. Different lighting effects are excluded. Biometric features like eye sizes, eye colors that are changing person to person are not be considered as in the scope of this dissertation. Any wearable tool or accessories like eyeglasses that comes to the eye region or make-up on these areas like darken or enlighten of one's eyelid things are out of the subject. Finally, for an image acquisition error, any frame cannot be included coordinate of the eyes depend on the frame rate or being the eye-closed position. This condition is disregarded.

1.6 Structure

The remainder of this dissertation is organized as follows:

The introduction (this chapter) gives a huge overview of the study, then, main purpose, the methods, approaches, discusses on eye gaze as an input and thesis roadmap is presented. The contributions to the field of research are finally listed.

Chapter two as background starts with a definition of eye structure, motivation followed by eye tracking history. It describes existing systems and technologies with their application. It also presents the eye tracker used in this thesis. Furthermore, it discusses the current challenges and the scientific work is done up to now on a general level. Special related work sections are part of the corresponding chapters.

In chapter three, camera calibration process will be presented. The camera calibration results as internal and external parameters of the camera are classified and explained in detail. Finally, calibration results and acquired data are presented. All things to be acquired for the use of a camera application are computed. For image processing, camera parameters are used to correct the distorted image.

In chapter four, design and implementation steps are all explained step by step. All background is presented from image acquisition to eye localizing for the eye tracking application. Applications as GUI are presented as eye gaze alphabet, animals and color.

Chapter five is a presentation of the experimental results of eye gaze motion of acquired diagonal, horizontal and vertical from the image plane to screen plane.

Finally, conclusion and discussion is covered with the relations between solutions and results. Recommendations are provided for the further researchers.

1.7. Contributions

In this thesis, the screen is divided vertically and diagonally into the smallest unit squares. The letters, numbers and various object detection tests for various users have tested. The usability of the eye tracking system is analyzed. Briefly, in this thesis:

Eye gaze interaction techniques: it is tried to find out if the proposed algorithm can detect the coordinates of the objects correctly in different colors and different sizes on the screen. The main aim of this study is to solve the problem in a very simple and amusing way by comparing our approach with the approaches of the other. The child-specific interfaces using pictures, colors and various sounds are prepared. Finally, the eye gaze can really be used as a PC appliance input.

*Gaze tool technologies**:*** the difficulties and problems that were encountered, the possible methods (discussed in Section1,3) are discussed. Sensitivity was also improved by implementing camera calibration. Many thoughts and predictions for the future were shared.

CHAPTER 2

BACKGROUND

2.1 Introduction

Eye tracking can be defined to get the knowledge of the coordinates of the gazes on the screen. Eye as a software tool is very handy for a much technological area. Automotive, psychology, computer science are very known areas for eye detection and tracking systems. There is no completely vision-based eye tracking techniques. That is, all methods use hardware. Since low-resolution camera based studies use the only webcam, they are very famous and valuable as a topic to research.

2.2 Human Eye Structure

Eye is an interactive agency for a human body that senses light. Eye muscles with white striped have mechanically the fastest gestures. Eyes reflect not only the light but also the emotions, personal interests, and some psychological situation of a human to a research subject. It is given in Figure 2.1.

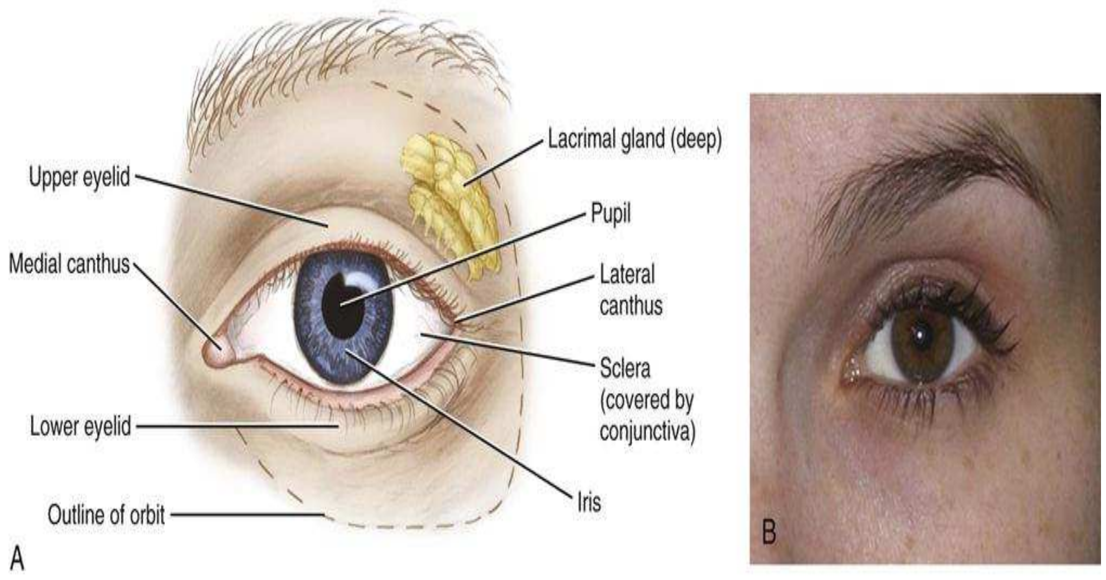


Figure 2.1 Eye regions by parts (A) with original eye (B) [11]

The eye is also described as the most easily and repeatedly segment able part of a face part. Moreover, eye has the most diversity of features for a computer vision area. Circles, ellipses, colors, sharp borders, reflectivity and movement factors make it unique for a face recognition and eye tracking field. Sclera is white area of the eye. Iris is colored that change person to person, used in biometric.

2.3 Motivation

Eye tracking applications mostly happen with some devices known "Eye Trackers". These devices are not comfortable with a wearing technology even a person does not need support for its activities.

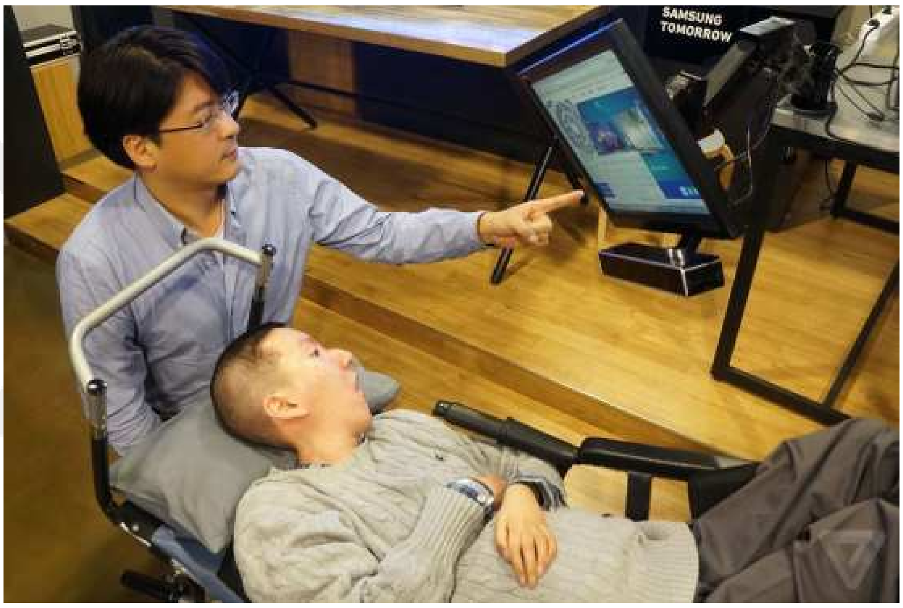


Figure 2.2 Eye tracking for disabled people [12]

Vision-based interfaces are fast, cheap and easiest way to interact with getting a frame of a visual device. Especially, these software based solutions play a key role on HCI as an assistive technology using a GUI environment for severely disabled peoples, paralyzed patients, elders as seen in Figure 2.2. Letting these peoples work, socialize and entertain is the most notable goal of this study.

On the other hand, Eye gaze tracking related to HCI is very modern as a vision-based system for the education of some child getting difficulty in reading and kids to learn. On the side, Eye tracking game is played with the kids to learn. Mother says the name of the object and kid points it with his eyes. Thus, mother communicates with her child as seen on Fig 2.3, understand what he or she desires and teaches the name of the objects. This study also gives to know the kids interests as seen on Figure 2.3. Moreover, eye tracking solutions are preferable for learning language letters.

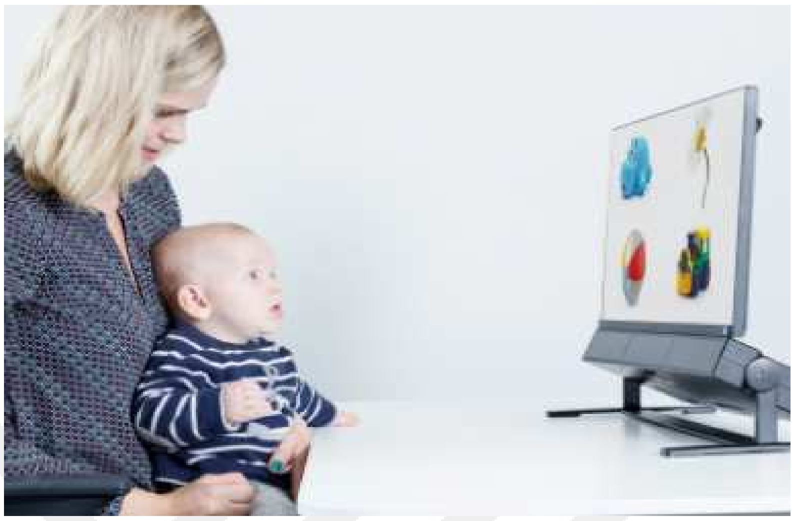


Figure 2.3 Eye gaze tracker game for kids [14]

As seen in Figure 2.4. A child who is getting difficulty in reading is pictured. Eye gaze indirectly helps the students to read in a short time. Pc is used for another distinct purpose as an entertaining device between parent and child. Only eye gaze movements are used for this desktop study. As result, e-learn makes all studies effective and enjoyable.

Figure 2.4 Getting difficulty on reading [14]

2.4 Eye Tracking Integration in e-Learning

Eye tracking technology in the e-learn platform has been grown to decide user’s intention, behaviors, even in the mood. The first and most effective study is adaptive e-learning system developed as assistance for translation in language courses.

The main objectives about e-learn are observing the behavior of learners in learning processes in real time, by monitoring characteristics such as areas of interest, time spent watching them, a frequency of visits and sequences or patterns, evaluating the learner’s awareness according to which content is studied.

In the last years, various technologies (like collaborative Software, cloud computing, screen casting, portfolios, virtual classroom) and different devices (e.g. mobile devices, webcams, audio/video systems or smart boards) were used to facilitate e-learning development and to increase the effectiveness and accessibility of e-learning platforms [15]. E-learning gets importance decade to decade, especially for kids and disabled peoples that watching multimedia elements, such as images, video and animations.

2.5 Gaze Tracking Techniques

Eye tracking techniques are categorized into two main groups as hardware based methods and software-based methods.

Gaze tracking is a subject of video oculography as software-based systems to infer the knowledge of image data from cameras. Firstly, it is the detection of the eye area, then, gaze estimation is determined. Pupil as eye part lets the light in. Iris controls the sclera and pupil areas. This eye parts and their functions play an important role in eye tracking issues. These factors are very challenging due to the vision and eye as a region that mentioned about in chapter 1.

Video-based eye tracking uses dark or bright pupil technique. And this technique is separated into two groups: feature and appearance based gaze tracking.

2.5.1 Feature-based Gaze Estimation

Feature-based methods approach to the problem of gaze track for some distinct features of eyes on the face such as ellipse as sclera contour, circle as pupil contours, corneas reflections.

The goal of the feature-based method is to attain a robust feature or feature combination to describe the gaze direction independent from the environment and device effects. If light independency does not improve, the accuracy of gaze detection fails. There are two types of feature-based approaches exists Model-based (geometric) and interpolation-based (regression-based) Hansen and Ji [16].

2.5.1.1 Model Based Approaches

Model-based approaches utilize geometry of eye for estimating of gaze direction. This method uses camera calibration. Pre-generated 3D eye models are used to compute eye direction. Models base parts are the position, tilt and rotation. These models are constructed with the rotation of eye shape degree by degree. The system does not use any learning algorithm. To compare this model to others, and it is more convenient to high-resolution devices.

2.5.1.2 Interpolation-Based Approaches

These type methods focus on the mapping from image features to 2D gaze coordinates. This can be polynomial (parametric form) or neural networks (non-parametric form). Some linear mapping studies occurred as video-based trackers but did not get popularity like polynomial mappings. The interpolation-based approaches do not use the geometry of the eye; instead, they use more general image vectors as gaze points. Neural network based methods; some points are used as features predefined in the face. Then, a model is chosen to train. Features data are trained and eye gaze coordinates are determined.

2.5.2 Appearance - Based Gaze Estimation

Appearance-based methods without the need of any camera calibration, occurred mapping directly based on the photometric view of the objects. This method made with gaze data by looking different angles to the screen.

Some studies about appearance-based methods: gaze tracking is proposed based on Run Length Coding (RLC). The appearance-based methods detect and track eyes directly based on the photometric appearance. Appearance-based techniques use image data to get the gaze direction by mapping image data to screen coordinates [17]. These methods require preprocessing and post-processing steps including image analysis but not camera calibration.

2.5.3 Eye Tracking Applications

Eye tracking technology is used many technical studies such as Pc and gaming, psychology, market research, academic and educational research.

But even today it is not directly applicable. It does exceed laboratory experiments. Many devices become more reliable and hopeful about this area. But many of the interests are related to the HCI based eye gaze. Because, artificial intelligence give the best researches areas to involve a software-based application development. It is more convenient due to its natural. The software-based methods directly play an important role in daily life in a way of e-learn, e-game, e-socialize. These software-based techniques mainly divided into two categories: intrusive eye gaze trackers and camera-based eye trackers.

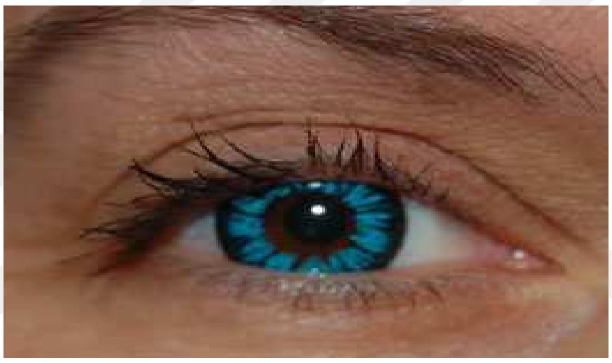


Figure 2.5 Contact lenses for eye tracking [18]

2.5.3.1 Intrusive Eye Gaze Trackers

Intrusive eye tracking techniques can be contact lenses, EOG signals. These are very close to the eye as seen in Figure 2.5. Intrusive gaze trackers are more accurate because of it is closeness.

The contact lenses type eye tracking system as seen in Figure 2.5 consists of an enclosure material, a three point electrooculography EOG sensor and a controller. Two type enclosure materials concave and convex are mounted on the eye region. There is a sensor that indicates the voltage levels of this material. The capacitive sensor system is disposed within the enclosure material. The capacitive sensor system has at least one capacitance value that varies with changes in a gazing direction of controller both controlling of sensor power and also evaluating the eye position.

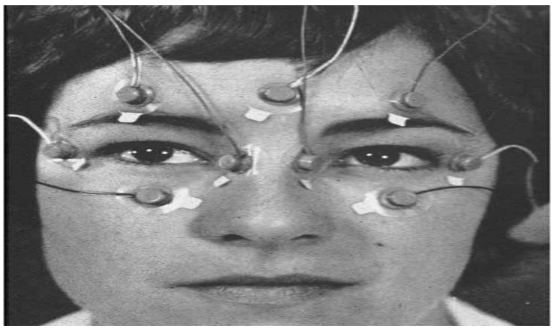


Figure 2.6 EOG for eye tracking [19]

The electro-oculogram signals (EOG) are both cheap and easy to use. It measures skin potentials as seen in Figure 2.6. It is more common in medical applications not for human-computer interaction. Some electrodes are placed around the eye with some distances. These electrodes measure the change of these distances changing the skin.

2.5.3.2 Camera Based Eye Gaze Trackers

Camera-based eye gaze tracking uses a feature of an eye that is detected and tracked by a camera or optic devices. This method is non-intrusively.

As a feature of eye tracking sclera, limbus and pupil detection is used. Limbus does not capture when eyelids cover. So it is not reliable for full-time detection. The pupil is hard to segment but unique due to circularity. To detect eye easily IR light is used as assistive. It is turned on and off in a range of time. So pupil is detected easily.

2.6 State of The Art

There are many approaches to a use of eye tracking efficiently. For instance, Sclera-based ellipse detection proposed by Hansen et al [20] and [21]. This approach is directly related to the resolution. Very near taken frames are required. Technically two circle measures can give more accurate result. But it takes more calculation time. Wrinkled regions and eyelash edges are ellipsoidal. That makes difficult to find the most obvious geometry on the face is iris circle.

M.Ramezanpour et al [22] studied color feature to localize eye point as a new method. They apply a new color space, GLHS that is converted from RGB data. This method works well. The user faces a huge accuracy problem and the energy of algorithm by means of calculating power is too big to compute. That is, algorithm is too long and exaggerates the calculating time

A light reflection based method proposed by Shubham Singh et al [23]. Light comes to the eye region in a condition spot lightening. This method based on controlled-environment. Bright and dark images taken by light source and camera subtracted. This method is uncomfortable and makes a general application hard due to different shapes of the eye region.

Bülent Turan et al [24] proposed a method which does not require any additional hardware but a webcam. Neural networks are used for gaze point control. Because of declaring saccading and fixing problems inevitable, they declared 100% efficiency on their own dataset study. But, this method troubled with saccadic movements on mice points because of uncontrolled environmental conditions.

Over the last decade, many techniques are developed on the task of vision-based eye detection [25]. These are separated into two main categories, respectively appearance-based and feature-based methods.

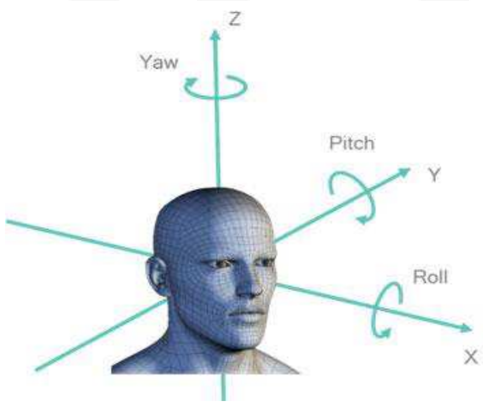


Figure 2.7 Head movements effects [26]

2.7 Challenges

This study also discusses some difficulties to handle use of an eye gaze as Pc input. There are plenty of challenging things to overcome to use an eye as a pointer to a Pc or any devices with a processor. The most difficulty is illumination of the environment as seen in Figure 2.8 and head movements as a posed problem as described in Figure 2.7.

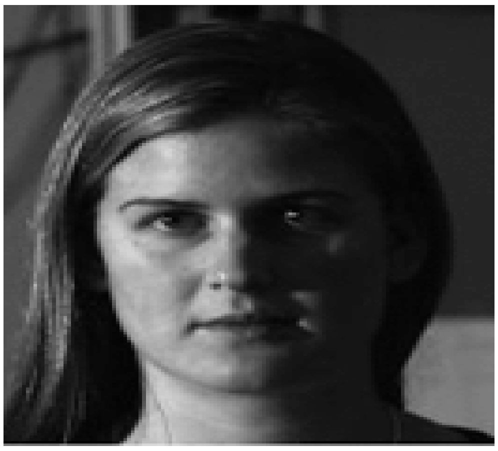


Figure 2.8 Lightening effects [27]

Uncontrolled environment lighting and object diversity, the biggest problem that needs to be overcome in applications using image background. The variety of objects causes heterogeneous light distributions as seen in Figure 2.8, different reflections and indirectly local light changes on the face. This is also evident as a bad quality image.

Pose, the user can hold his hand on his head, rub his eyes and be in distractibility. If the user is using a portable device, the user can keep it at different angles and thus, it occur a posing problem.

Eyelash (Half, fully open state), there is a half or full or unmeasured open and closed state of each one of the left and right eyes. This faces a huge problem to tackle.

Distance, the user may need to use the application at changing distances in different HMI. For example, when a patient is lying in a bed, on a passenger seat, or when a student is studying. The point detected at different distances is expressed by a different number of pixels. As result, for a geometrical image analysis, dimension problem occurs.

Camera Properties, with the webcam focus feature, the image has the HDR feature as the lighting feature. HDR features of cameras are very important in uncontrolled environments. Since the eye movements are very fast so, Fps specifications of the cameras are also blurring and boosting at this point. The resolution of the camera must be a certain standard in order to overcome sensitivity problems due to regional differences to be used for this application.

Calculation time, almost all HMIs have video cards. Frames processed in image processing techniques, even if they are not in high resolution, must be at a certain speed in order to be able to respond to eye movements and the user's instant requests. This point is also making it more important to do high-speed calculations in parallel with the main processor using graphics cards.

CHAPTER 3

CAMERA CALIBRATION

3.1 Introduction

Camera is a much-known occurrence for daily and industry life for a long-long time. Camera measurements are pixels. It is necessary to convert pixel values to metric values for a physical estimation as a process before coding. Camera's internal and external properties are calculated and taken a session of data have to use on the acquisition of a calibrated image. There is an image in Figure 3.1 which is represented the way of the calibration.

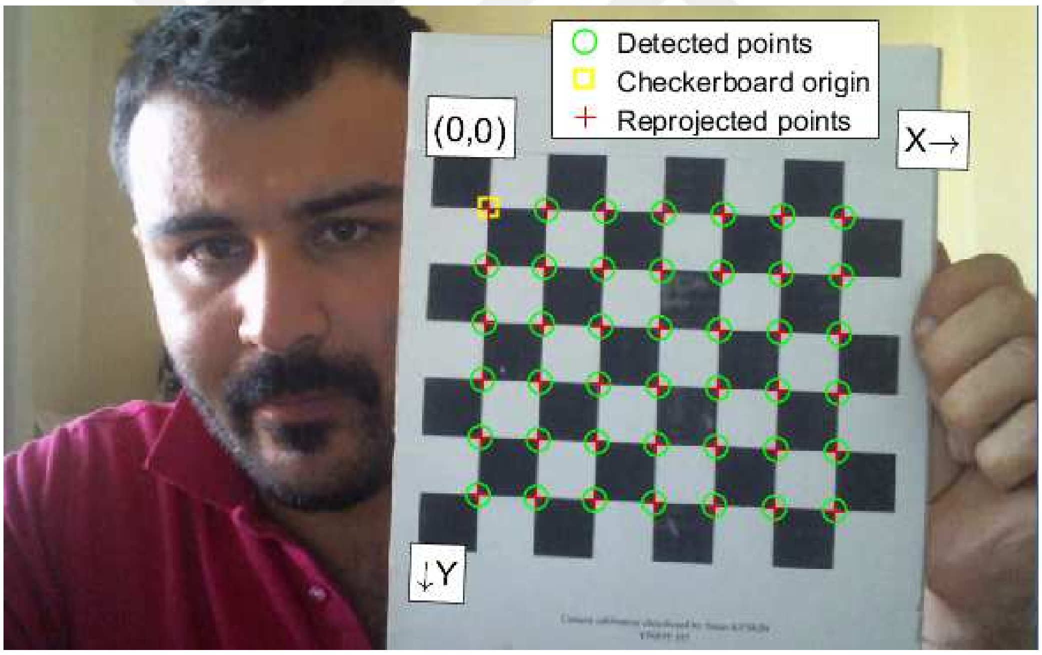


Figure 3.1 Camera calibration setup

As for internal quantities that affect imaging process, image center is not on the half of height and width size. Focal length must be recalculated. Scaling factors for the pixels are not exact. That is, row and column sizes are not proportioned to a stable value.

Skewness of the pixels is changeable to the camera to camera that has to be computed. Very much type of lenses has too many diversity of distortion. Lens distortion meets an overcoming problem to use camera, effectively. As for the external quantities of the camera is related to the rotational and translational positioning of the camera due to the world frame that is a fixed coordinate system for representing an object in the world. The external parameters are very significant if the camera or taken object is moving.

3.2 Estimating Parameters

Camera calibration can be defined to the determination of internal camera optical and geometric behaviors and 3-D orientation and localization of the camera for world coordinate system. Computer vision and machine vision applications highly require the accurate camera calibration process.

Several methods for camera calibration are in the literature. The classic methods [28] solve the problem by decreasing the non-linear error. The time algorithm efficiency is taken into account, some other method like closed-form solutions are suggested (e.g. [28], [29] and [30]). But, these methods take the problem so easy and therefore, they do not provide better results as nonlinear minimization. There are also calibration procedures where both nonlinear minimization and a closed form solution are used. [31] In this study, it is used Matlab calibration application depending on the paper [32], [33], [34]. The calibration parameters are simply taken several steps. Calibration process in the background of the codes occurs below:

Object coordinates (3D) >> world coordinates (3D, extrinsic) >> camera coordinates (3D, extrinsic) >> image plane coordinates (2D, intrinsic)>> pixel coordinates (2D, intrinsic).

3.2.1 Intrinsic Parameters

Camera intrinsic matrix is acquired after calibration. This matrix transformation occurs post-projection and can be decomposed shear, scaling and translation transformations. This matrix gives the parameters of the skewness, focal length and principal point offset.

3.2.1.1 Skewness

Skew factor is added to the intrinsic matrix to calculate also pixel orientation. If this orientation is not corrected as a rectangle, distortion is not accurate. Images row/column value is named ‘aspect ratio’. This shapes the pixel how much amount of deviation from a square geometry is.

3.3.1.2 Lens Distortion

Distortion parameter changes camera to camera. It is about lens manufacturing technology. Moreover, cameras or lenses have usability for many purposes in many fields. In some areas, lens distortion becomes more important, for example, vision applications. There are two types of well-known lens distortion.

Barrel distortion, when straight lines are curved inwards in a shape of a barrel, this type of aberration is called “barrel distortion”. Commonly seen on wide angle lenses, barrel distortion happens because the field of view of the lens is much wider than the size of the image sensor and hence it needs to be “squeezed” to fit. As a result, straight lines are visibly curved inwards, especially towards the extreme edges of the frame. [36]

Pincushion distortion is the opposite of barrel distortion as seen in Figure 3.2.Pincushion distortion is the exact opposite of barrel distortion – straight lines are curved outwards from the center. This type of distortion is commonly seen on telephoto lenses, and it occurs due to image magnification increasing towards the edges of the frame from the optical axis. This time, the field of view is smaller than the size of the image sensor and it thus needs to be “stretched” to fit. As a result, straight lines appear to be pulled upwards in the corners, as seen below:

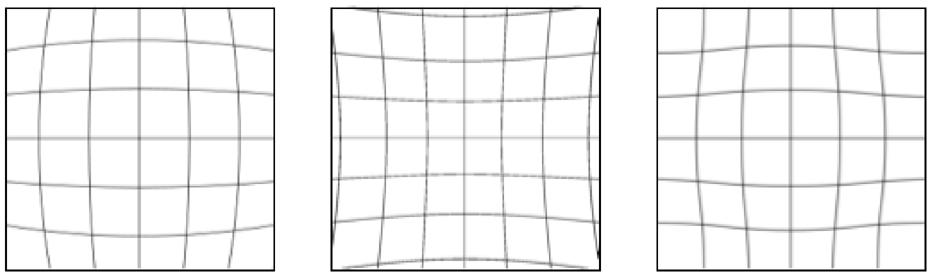


Figure 3.2 distortion factors for camera calibration [36]

3.3.1.3 Focal Length

Focal length is defined as a distance between the pinhole and the film. It is shown in Figure 3.3. Focal length measurement is pixel which it plays an important role in image characteristics. For example;

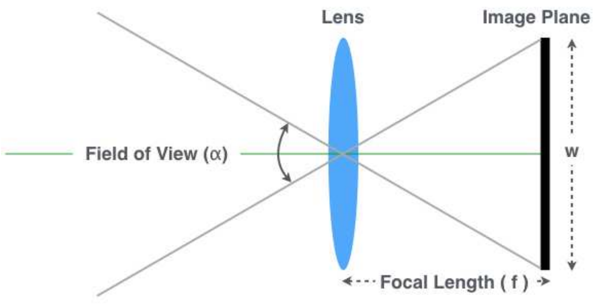


Figure 3.3 Focal length for calibration [37]

- The image has been non-uniformly scaled in post-processing.
- The camera's lens introduces unintentional distortion.
- The camera uses an anamorphic format where the lens compresses a widescreen scene into a standard-sized sensor.
- Errors in camera calibration, in all of these cases, the resulting image has non-square pixels. There are plenty of pinhole cameras.

3.3.2 Extrinsic Parameters

There are two reference frames to get the extrinsic parameters of the camera. First, camera reference frame and second reference world frame. Transformation matrix between these two frames determines the translation and rotation parameters of the camera. Recall the fundamental equations of perspective projection – assumed the orientation of the camera and world frame known–this is actually a difficult problem known as extrinsic pose problem – using only image information recover the relative position and orientation of the camera and world frames [38]. Scale factor w, image points x, y and X, Y, Z world points are given in equation (3.1)

$$W[x\ y\ 1] = [X\ Y\ Z\ 1]\ P \tag{3.1}$$

Camera matrix P, rotation R, translation t, K intrinsic matrix given in equation (3.2);

$$P = \begin{bmatrix} R \\ t \end{bmatrix} \mathrm{K} \tag{3.2}$$

3.3 Calibration Process

Camera calibration is the process of estimating the parameters of the lens and the image sensor. Camera calibration is done for getting a better accuracy to compute gaze points. Matlab Camera Calibrator App is used to estimate that camera intrinsic, extrinsic, and lens distortion parameters [39]. For the calibration data, sampled 20 frames with a checkerboard (7x9) are computed to get camera parameters. The acquired .mat file is included by main eye detection algorithm. The world points are transformed to camera coordinates using the extrinsic parameters. The camera coordinates are mapped into the image plane using the intrinsic parameters. Image with respect to camera is plotted 3D as shown on Figure 3.4.

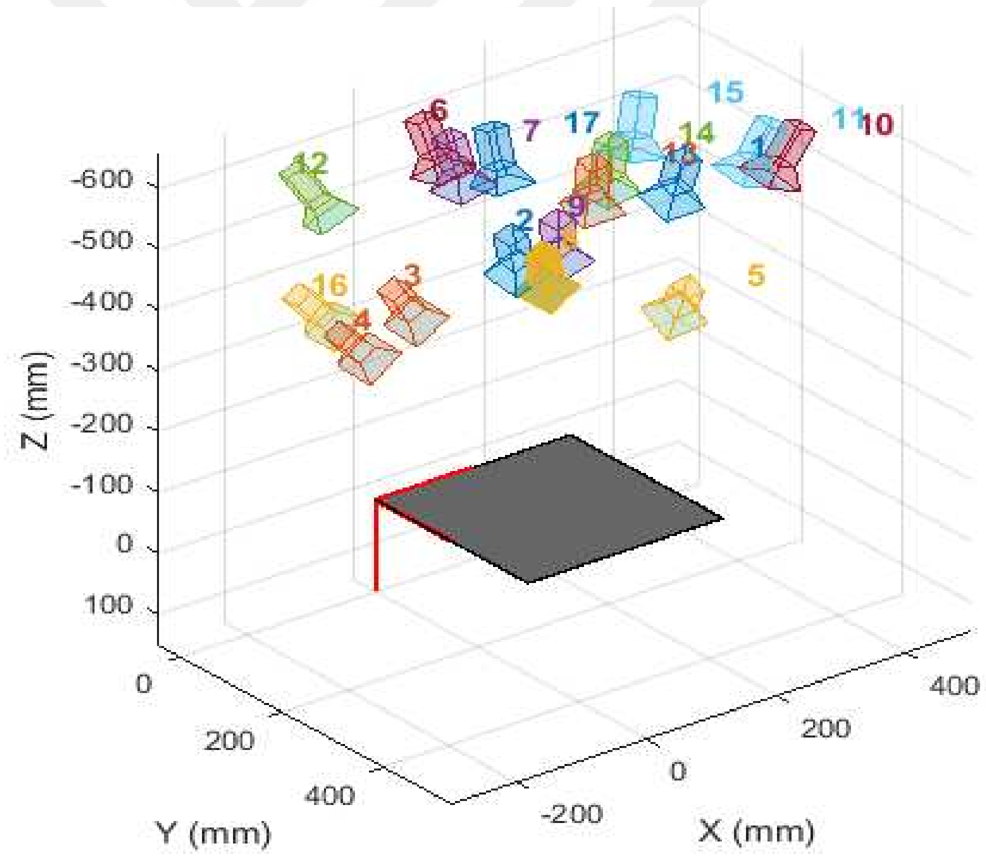


Figure 3.4 Extrinsic parameter visualization

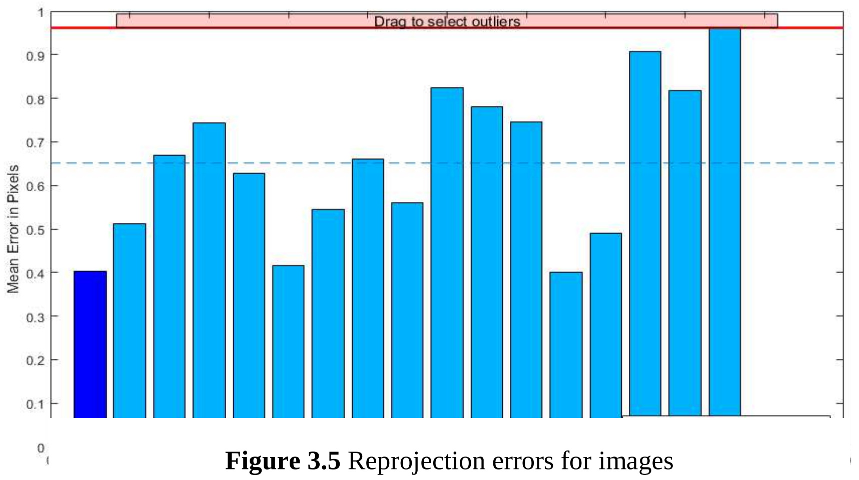


Figure 3.5 Reprojection errors for images

As camera internal properties center point, focal length, skewness factor) and external properties (rotation, translation) are included in the algorithm as camera parameters data in Figure 3.5. The bar graph indicates the accuracy of the calibration. Each bar shows the mean reprojection error for the corresponding calibration image. The reprojection errors are the distances between the corner points detected in the image, and the corresponding ideal world points projected into the image. Camera parameters are to use lens distortion and undistorted image as a part of the algorithm and provided that better results.

All the calibration samples are shown in Figure 3.6.

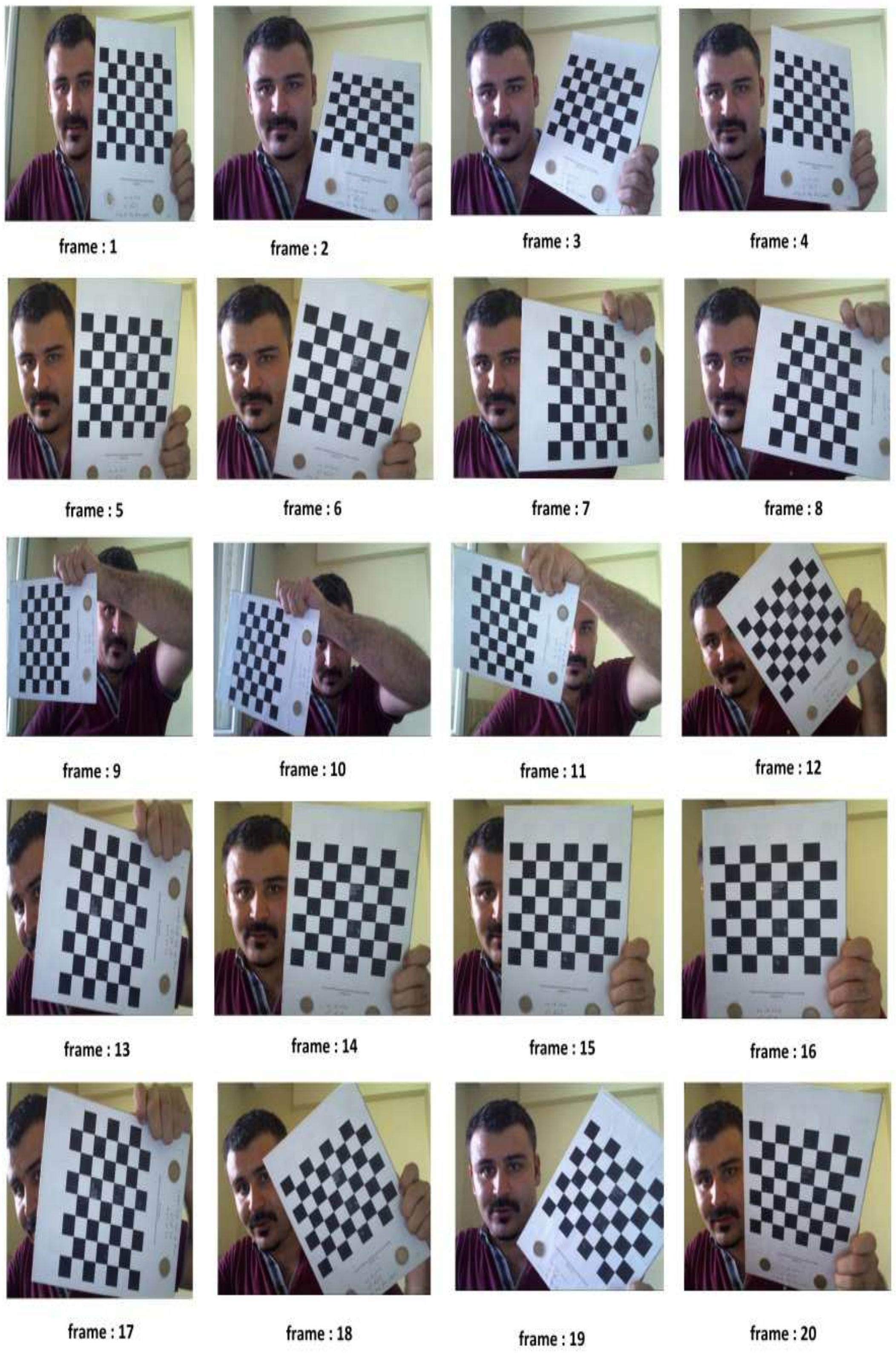


Figure 3.6 Calibration images

CHAPTER 4

DESIGN & IMPLEMENTATION

4.1 Introduction

This chapter consists of two main parts. First, the algorithm, second, interface design and implementation of algorithm are presented, respectively. Eye tracking and eye processing techniques has several steps to apply. At this work, very famous Viola & Jones algorithm is used to detect face and eye pair region. At this point, as a strategy of preprocessing, all edges, corners noises in the eye region are smoothed and removed. And there is only three kind regions. Finally, a suitable threshold is applied. After some morphological operations, Circular Hough Transform is applied to this region and eye-ball is easily detected. To sum up, this study, mainly, consist of capturing, distortion, processing and analyzing of image step by step.

4.1.1 Algorithm

The algorithm is based on the Matlab programming language and as a compiler. Image acquisition, Image processing and Computer Vision Toolboxes and some GUI applications are used. Script is simply a list of commands to run different from function that is return a value or needs some arguments. This command means a function or an object that is already built in Matlab. It is constructed with a lot of process in the algorithm so that functions and objects are returned in an order just writing their name. If the script that ends in the (.m file) is placed in our Matlab directory, it runs successfully. The algorithm consists of several basic image processes such as image acquisition, image calibration, and image processing and eye localization for tracking. The image is processed in order to find the point on the screen to find the gaze point. In this sequence, gray conversion, extracting of eye regions, image intensity processes and some filtering operations (Median, Gaussian) are performed. There are two main stages at this point.

i) The presence of the center of the eye
ii) Determination of the coordinates of the eye points on the screen.

4.1.2 Matlab as a Development Environment

Matlab, which stands for Matrix Laboratory, is a complete programming environment that encompasses its own programming language, IDE (integrated development environment), libraries (called toolboxes in Matlab), amongst many other things [40]. Matlab is a high-level language that means the developer does not have to deal with some detailed stuff. As a language, it is very easy to build an algorithm, applications and any kind of calculation because of its well documented resources and Mathworks site (file exchange, webinars etc.). Matlab provides very effective tools to handle with image processing environments. Developing an image processing algorithm is made easy due to all toolboxes are in the same ecosystem. There is a lot of GUI s to help for simple calculations and predictions and so on. For example for image processing, many process such as camera calibration, morphological operations, segmentation, color analysis, image acquisition are handled with GUI's.

4.2 Image Acquisition

Image acquisition is both first stage of the image processing and the most significant part of an application developed with image processing techniques. There are plenty of camera module provides different data formats, color spaces, color patterns and so on. But they simply produce image that is the 2D array in our case. Chroma resampling that is the way to sample image data sent by an image sensor. For instance, 4:4:4, 4:4:2 (r, g, b); color correction that is used to adjust white balance, brightness; gamma correction that is used to encode linear luminance or RGB values to match the non- linear characteristics of display devices [Xilinx]; deinterlacing is used to incoming interlaced frame joining a progressive image; timing controller is very important for image processing which types the image is shuttered and exposed. One of the ways of image data streaming is known Real-Time image acquisition. It can be defined retrieving image data automatically from a source such as, a camera, telescope, and microscope etc. There is a huge problem when acquisition from a webcam is timing due to many frames storage and processing time. The camera fps changes with time. If triggering happens in a range, then timing and latency problems may occur.

But in this study, there is no more focus on these issues. Device properties are fixed by Matlab Image Acquisition Toolbox and Image Acquisition GUI. The GUI detects and configures hardware options. The toolbox enables acquisition modes, such as processing in-the-loop, hardware triggering, background acquisition, and synchronizing acquisition across multiple devices. Acquired image for this application is shown in Figure 4.1.The image is taken from the webcam as 800x600 resolutions, 30fps in RGB color space. Acquisition parameters such as fps, exposure, and frame size and color mode are firstly fixed. 640x480 resolution of the camera is easy to calculate but gives not accurate results.1280x1024 resolution is very good but hard to process for our algorithm. Approximately 4 frames per second is used for our program adding calculation time of the algorithm. Real-time image process did not happen because of processing time and camera changing frame rates. Reasons are discussed in the future work section.

4.3 Image Calibration

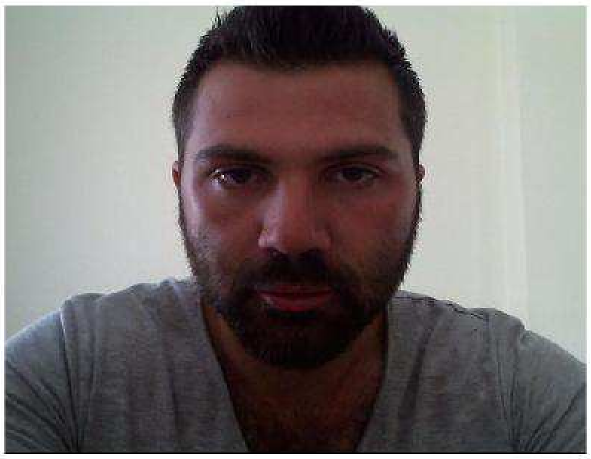


Figure 4.1 Input image

Cameras are used for a long time in the HCI environments. However, the cheap *pinhole* cameras became a common occurrence in our everyday life. This cheapness brought its significant distortions which are constants and with a calibration and some remapping that it could be fixed.

Furthermore, with calibration it is possible to determine the relation between the camera’s natural units (pixels) and the real world units (for example millimeters) [41]. Camera calibration for a measurement is so important. Distortion parameters for a use of image sensor and lens as internal parameters changes from the camera to camera. To obtain these parameters and use of them in an image acquisition algorithm gives more accurate results. Shortly, a raw image enters and an undistorted image is acquired.

4.3.1 Image Correction

There are two physical defects on the cameras while providing us an image. More accurate parabolic lenses are hard to manufacture and this causes to positioning a lens in a camera after through the image sensor. Finally, these two types of distortion are inevitable in the end. Radial distortion is about the how spherical the lens is and tangential distortion is relevant to the lens if it is parallel to imaging plane or non-parallel. Without undistorting the image, we cannot measure a distance between two objects like eye pairs. In Matlab, user does not know what happens in the

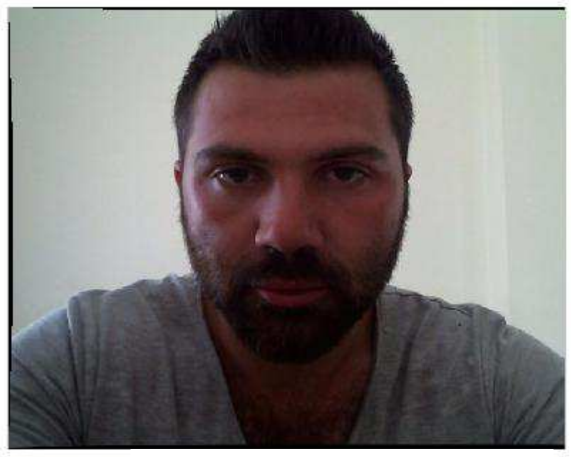


Figure 4.2 Undistorted image

background while calling system objects, classes and structs. Undistorted image objects give the camera parameters to apply for a frame. As seen in Figure 4.2. Undistorted image is attained implementing camera parameters to the raw image. Camera parameters are included to algorithm after image acquired. External properties (rotation and translation) of webcam and internal properties such as skew coefficient, focal length, and optic center are used to correct the image. Real world object points are mapped on a new plane called image points. A new origin is defined. Finally, the frame is linearized as it can be that means the distances between objects are proportional to the image as in the 3D world. So, the fully calibrated image is obtained and image pixels are then more accurate for the measurement**.**

4.4 Grey Level Operations

Cameras can be grouped as monochrome and color. The monochrome camera as a device gives the 2D gray level image. Gray level means 2D image to be processed with its values changing 255 from 0.It can be any image or image component such as R, G, B; Y, Cb, Cr etc. taken from the camera. Color image is 3D but can be easily converted to gray level with hardware or software wise design. After converting, the image is going to be ready for spatial and frequency domain filtering, intensity-based operations, enhancement, smoothing, sharpening and so on. The monochrome cameras have higher sensitivity and give more details. Webcams produce the RGB image that loads the conversion to the processor. A color image sensor captures 1/3 of light because of its sensors. There are three filter layers red, green and blue. Thus, light coming through lens is divided. And then, finally one image is produced. Output depends on the demosaicing method. That proves every software application including data conversion can replace using a binary image sensor. At least, it is more appropriate for the prototyping process. That is, webcams with color image sensors can be accepted wrong choice for computer vision applications. It decreases of calculation time and pixel data precision. Especially, to detect an object or some features of it must be extracted. The feature can be defined anything that differs the target object or region on an image. Eye circle, eye color, eye edges can be used as features. Feature extraction algorithms are consisting of grey level operations, segmentation and binary level operations. Grey level operations in this study can be summed into first, neighborhood kernel operations as Gaussian, averaging filters; second, as a shading correction operation contrast stretching; third, as a Thresholding method, Otsu. Grey level operations ends with Thresholding. A value locally or globally turns to image binary values

4.4.1 Eye Pair Detection

After calibration process, Image is first converted to gray level. Well-known computer vision technique, Face detection by Paul & Viola Jones algorithm [37] is applied and eye pair is clipped as right eye end left eye image from eye pair image as seen in Figure 4.3.

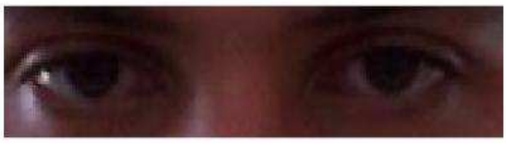


Figure 4.3 Extracted eye pair region image

Pre-built system objects for Computer vision toolbox are used for detection. After detection operation, Eye pairs are respectively preprocessed by Gaussian and Averaging disk filters to get rid of noises and entropy of image. Then, Contrast normalization is done. After all, a global threshold level is computed by Otsu Method and it is used to convert this intensity image to a binary image.

4.4.1.1 Viola-Jones Algorithm

In paper [43], Yi-Qing Wang has studied Viola-Jones algorithm in detail. Viola-Jones algorithm is a revolutionary work has been done in the field of computer vision using Haar as seen in Figure 4.4.

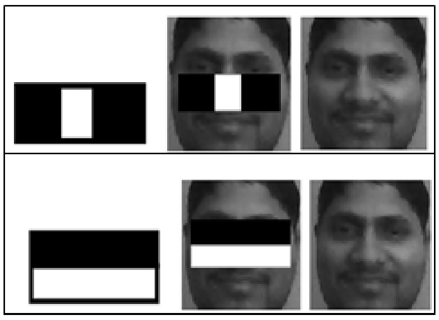


Figure 4.4 Haar cascade features [42]

Robust and fast detection of face and face parts make it so applicable. They discovered the best features that are on a face but not in any other object. Lightening conditions are minimized, almost compensated and the most suitable features as Haar cascades are described using machine learning algorithms to tackle that problem. To sum up this framework; first, a database is built from both positive (There is a face on the image) and negative images known as train images and also some more to test in another folder known as test images. Second, training a model and it produces a classifier that detects face in an image taken from all around. Classifier is an .xml file that is produced. It stores the all image characteristics and used as comparing the input image to this stored values. The main algorithm consists of four steps as Haar features, integral image, AdaBoost, cascading.

Haar features, there are many types of features of black and whites. All features are like vertical, horizontal, diagonal stripes. These features are applied to the image. If there is any match of these features on the image, this region is pointed. For example, eye regions are black. To find eye areas first feature is applied to the image as seen in Figure 4.5. Noise region is most bright are on the face. This information is extracted using second features. Viola-Jones algorithm uses 24x24 images. And a sliding window of this features scan all of the pixels one by one.

Integral image, Image features that match the Haar features is calculated by summing both black and white pixels. This is an intense computation of the processor. Viola-Jones uses a trick of computing of only corner pixels. Logic is any rectangular area on an image is adjacent to another.

AdaBoost algorithm is to eliminate the redundant features. After features detected, there are 160 000 features to construct a face. To complete a face, relevant and irrelevant features must be separated.

Cascading, after AdaBoost 2 500 features are to handle. These features are grouped and decided if it is face or not.

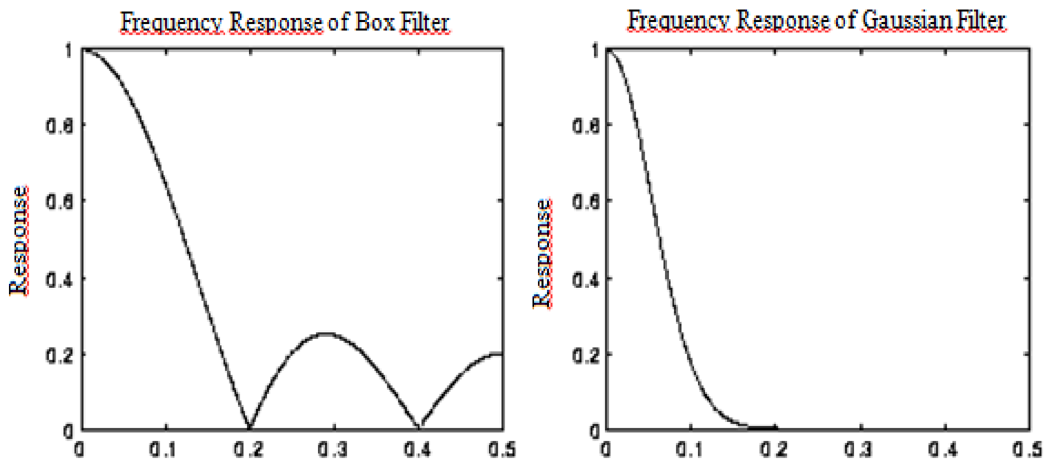
4.4.2 Smoothing Operations

In this assertation, two smoothing operations are made. First, as a non-linear filtering operation, Gaussian, second; as a linear filtering average. If you compare these two filters, both filters attenuate high frequencies more than low frequencies,

Figure 4.5 Filtered image

But the mean filter exhibits oscillations in its frequency response. The Gaussian on the other hand shows no oscillations. In fact, the shape of the frequency response curve is itself (half a) Gaussian as seen in Figure 4.5. So by choosing an appropriately sized Gaussian filter it may be fairly confident about what range of spatial frequencies are still present in the image after filtering, which is not the case of the mean filter 43].

4.4.2.1 Gaussian Filtering



d
n
n
e
n

Figure 4.6 Box filter as linear vs. Gaussian filter as a non-linear [43]

Sigma is used for the degree of smoothing. Gaussian kernels are decreased far from the center of kernel. Main purpose to use of Gaussian filters is frequency domain responses and it is low pass filters. As seen in Figure 4.6, the image is smoothed,while sclera and eyeball edges and corners are weakening, also, eyeball and sclera is regionally distributed uniform. Edges after applying the Gaussian filters become geometrically more smooth. As a pre-process before thresholding gaussian filtering is so significant to utilize. The Gaussian filter includes a little more mathematical calculations so; it is not preferable for the GPU Calculations compared to CPU.

4.4.2.2 Average Disk Filtering

Common names of this filter are known box, mean, average to be used to reduce of intensity difference between one pixel and the next.

The average filter is a smoothing operation implemented for getting noise reduced and blurred the image. It is a linear filter and easy to apply. Center pixel becomes the average of all neighbor pixels related to selected window which must be an odd number. Generally, it is used as 3x3 kernels but default 5x5. Smoothing degree comes with calculation problems. So the smaller window is better. In this application, circular average filter is used to get circular object more segmental.

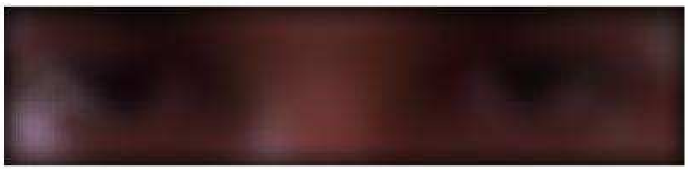


Figure 4.7 average filtered images

This function increases the accuracy of detected eye and gives better results. As compared Figure 4.5 to Figure 4.7, it is obvious to realize that Figure 4.7 is not more complex, edges are removed. There are only black, white and skin color regions. Now there is only one more step remaining is to apply contrast normalization through the whole image. Average filters act as low pass filters because of positive values. The sum of all the elements should be one. If gain is greater than one, the filter attenuate, otherwise the filters amplify.

4.4.3 Local Contrast Normalization (LCN)

Contrast stretching is used generally in medical imaging applications. It can be grouped as local and global, partial, dark and bright contrast stretching [44]. It is a good study to compare different contrast enhancement technique. In this study, local contrast enhancement is implemented to the filtered image. Contrast is a useable factor for a human eye in an image. Contrast normalization is used non-linearly change of an image to normalize. The most important part of the grey level operation is to eliminate the intensity distribution through the image. Local contrast normalization makes the dark areas darker, bright areas brighter. As an alternative to histogram equalization, at this point, gives locally better results to clear eye localization while normal or adaptive histogram equalization methods are global. LCN is pixel basis. Every pixel is particularly processed by removing the mean of the neighborhood and divide by its variation. It can be said that this Contrast normalization technique as seen in the Figure 4.8 gives better results for the well-blurred image.

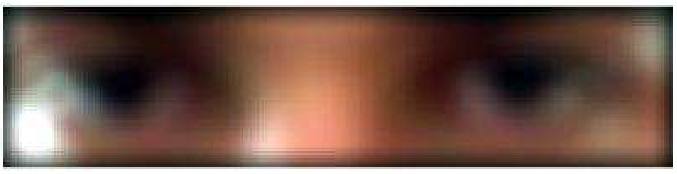


Figure 4.8 LCN applied image

It admits whole intensity values as a low and high and then mapped to the whole image. It can be noticed that eye balls are blacker and around are not. Matlab contrast adjusting tool is a good way of seeing the contrast values. This tool presents a scaled histogram of pixel data. Maximum and minimum values are known and a range can be defined.

4.5 Thresholding

Segmentation is a separation of an image into regions related to target objects. So, Thresholding is the simplest segmentation method to find eye region which is used to obtain binary images from grey images. It simply happens to define a constant and to compare it with every single pixel. If the pixel value is greater, it is allowed this pixel to white, otherwise, admitted as black. Thresholding techniques can be categorized into six groups as presented below.

Histogram shape-based method is based on the shape of the histogram. For example, convex hulls take the deepest concavity point; the peaks, valleys, and curvatures of the smoothed histogram are analyzed and taken one or two points as the threshold.

Clustering - Based methods where the gray-level samples are clustered in two parts as background and foreground object, or alternatively are modeled as a mixture of two Gaussians.

Entropy-based methods result in algorithms that use the entropy of the foreground and background regions, the cross-entropy between the original and binaries image, etc.

O*bject attribute-based methods* search a measure of similarity between the gray-level and the binaries images, such as fuzzy shape similarity, edge coincidence, etc.

Spatial methods use higher-order probability distribution and/or correlation between pixels. These methods use grey value distribution and pixel neighborhood. For example, context probabilities, correlation functions, 2-D entropy etc.

Local methods adapt the threshold value on each pixel to the local image characteristics. In these methods, a different threshold is selected for each pixel in the image. The threshold is computed with every pixel depend on some local statistics like range, the variance of neighborhood pixels. Versions of local contrast and variance are developed.

4.5.1 Otsu Thresholding

Image processing continues after grey level operations converting into a binary image. There are several methods to decide a threshold value. Some thresholds are local or global, some thresholds are multi-level. The aim is to acquire best non-overlapping regions or images whose foreground is exactly separated from the background. In this study, Thresholding is used to binaries the preprocessed image.

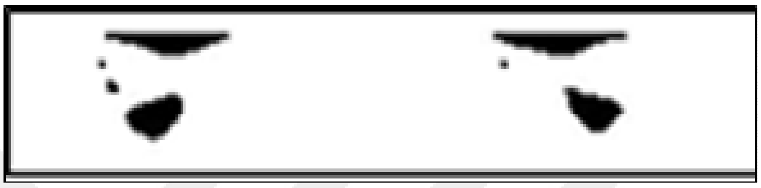


Figure 4.9 Threshold image

To explain Otsu method, first all possible threshold values are calculated as variance and weight. Foreground and background pixel values are separated. To both region, minimum variance value is purposed. Finally, a mean-variance class is calculated with the weights to foreground and background images. If the mean -variance class is min, that value is picked as a threshold value. As seen in Figure 4.9, after Thresholding process, gray image is separated to foreground and background to show the target object.

4.6 Morphological Operations

Morphological operations can be defined an image built with black and white regions instead of pixel values. These operations that only need an image and a structuring element are simple to implement. What the important is pixel order, connectivity and neighborhood. The structuring element is binary image or kernel which assign to the structure of neighborhood. Structuring elements play the same role as convolution kernels in linear filtering operations. The structuring element has dimensions, shape and an origin to apply. Matrix size changes depending on the size of an image or asked shape of the feature. The structure shape consists of 1 and 0 values and applied to every pixel with sliding window method. There are actually two main operations as Dilation and Erosion but different from of these operations, there are other operations such as opening, closing, top-hat, gradient etc. Dilation is simply adding pixels to the boundary and erosion just opposites of it depend on the kernel shape and size. Opening is used to remove noises, small areas between regions and closing is reverse of opening. Gradient can be defined as the difference between erosion and dilation. Top - hat is the difference of the image and the opening of this image.

4.6.1 Logical - Not Function (~)

As binarization process, Logical - not function is most important to apply to the image and so, noises, small pieces are removed both white and black regions. To inverse of any pixel value (1 to 0 or 0 to 1) and then apply the structuring element to the image provides to overcome big challenges. Finally, It is gotten a more clear, descriptive and usable image.

4.6.2 Small Pieces Remove Operation

Preprocessing operations are aimed to get a clean binary image. However, it does not

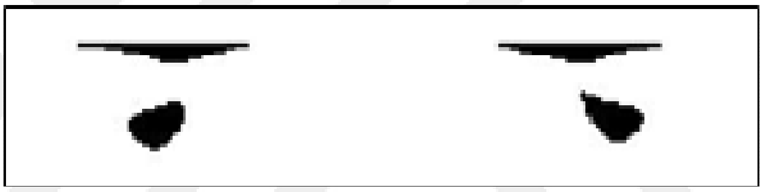


Figure 4.10 Small Pieces removed image

happen complete as expected. Because, there is not perfect segmentation and contours of the objects. So, small pieces and noises removing operations must be applied as the first operation of the binary process. As seen in Figure 4.11, some small pieces, noises are eradicated applying this process if it is compared to Figure 4.10.Small pieces removing algorithms work that way; first, connected components are defined and computed the area of each component, then removed small pieces.

Figure 4.11 Edges chopped off image

The connected components are composed of the steps of searching unlabeled pixels, using flood-fill to label all the pixels in the connected component containing this pixel, repeat this for all pixels. For area calculations, segmentation of grey image is necessary to discover the binary objects and analyze the original gray scale pixel values corresponding to each object in the binary image. There are three general neighborhood parameters 4-6-8. To label of an object after calculating the area is first encoded with run-length, scanned and assigned the preliminary labels to a table separated the classes to be relabeled again. Finally objects built with 4 connected components are labeled [39].

At the end of this step, one more operation as named chopping is made using morphological binary tools. The detected region before is eroded with a structuring element. This is done to get more accurate circle just chopping off the edges. An object that composed of connected components shrinks after this operation.

4.6.3 Circular Hough Transform (CHT)

Hough transform is a feature extraction technique that is used to find eye circles. Hough transform is first utilized to identify the line positions, but then developed to

Figure 4.12 Eye localization image

localize circles, ellipses and some other arbitrary shapes. The transformation begins first to find the aligned points that create lines. This is achieved by applying edge detection operation and lines are extracted. The main logic is what defines a line or a circle. Circular Hough transform is used to find eye circles as shown in Figure 4.12. Eyes are simple and easy to find because of it is unique in the face area as the circle. Circles can be detected dark and bright areas which to the choice. To implement Hough circle extraction, three parameters are necessary such as x, y coordinate and radius.

4.7 Application Design

Pointing (showing something with your body action) to user interfaces in human-computer interaction is a very high-level communication. Pointing to an object is easy to say or write its name. Most user interfaces use the mouse, trackball, track point, joystick, touchpad or a touchscreen while scrolling a folder, for example,

setting the cursor position when writing, selecting an option from the menu. The use of mice from traditional pointing devices can cause physical problems. Many designers and computer players suffer from the carpal tunnel syndrome. In most cases, it is very difficult to locate the pharynx while the dual screen is running.
The big part of our study is to develop an eye gaze interface which is going to be suitable, friendly and funny for the kids.

4.7.1 Algorithm Process

As a developing process, firstly, all documents published, books, journals to control the cursor and to track an eye gaze are scanned. These materials proved that there exists a lot of way of making eye controlled. Main blocks are shown in Figure 4.13.

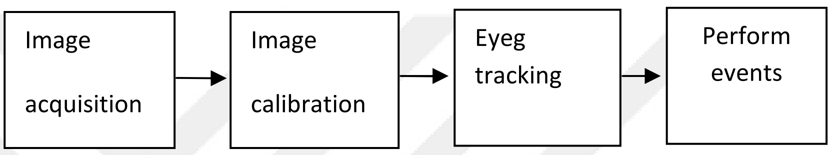


Figure 4.13 Block diagram of the application

Image acquisition application is used to take videos with putting some points on the screen as horizontal, vertical and diagonal with equal distances. Distances are pointed to get better results with eye gaze skipping as far as the gaze pointing sensibility is enough. Videos divided into individual frames based on the gaze point on focus. Videos are taken in different environmental conditions and different rooms with different backgrounds and lightening. Then, video and frames for calibration are acquired with same methods. Since the taken images are distorted, they are to be corrected firstly and then processed and analyzed consequently. After algorithm is developed, it is run with a GUI. To start an application, calibration step is first launched after clicking on the start button with mouse and also pointing this icon with eyes to localize itself with respect to PC screen. During calibration, piano music rings two times. User must look first top left corner on the screen with first ring and then, to bottom right corner. Now, eye gaze interface is ready for e-learn.

4.7.2 Calibration Process

As a priority, every user needs to undergo a personal calibration process after the user looks at the screen. This is because there are a lot of people have a different shape, color, size vs. characteristics. In this application the user stares at 1 second for predefined points as shown in Figure 4.14 and totally calibration ends up within 5 seconds.

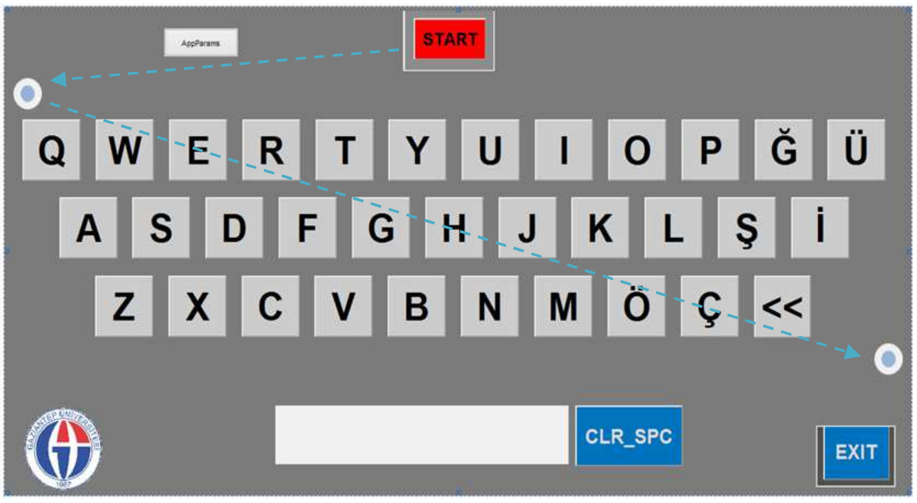


Figure 4.14 Calibration process route

If this is compared to other studies, it is noticeable that the process is incredibly short. As a calibration process, two points are chosen that the user look at both top left corner and bottom right corner. It is important to keep the calibration process short and easy because it might be done several times. As shown in the figure, the calibration process is completed in three steps. First, gaze is focused on the start button, second, top left corner and then bottom right. At his point, while focusing on these dots, time- based information could be sent making the dots exploded and turn into pieces spread. A click mechanism is taken 0,5 s per a station as seen in the Figure 4.14.

4.7.3 Graphical User Interface (GUI)

GUI is a kind of communication between human and computer. It makes the things easier. Many interfaces existed and these interfaces are completed for user experiences. GUI is can be called the second generation for HCI after command line interactions with typing commands. Next interaction is going to be eye gaze systems to make things easiest and efficient. Matlab provides an easy environment that eliminates the use of language and type commands to run an application to build a GUI. In this eye-gaze application, pushbuttons to the image process, axes to show the image and text spaces to show the coordinates of eye gaze are used. One more ability to our application is to use voice to communicate with a computer. The use of voice makes the application more dynamic. When a kid looks at the bear, and hearing the roar of the bear helps the kid to learn more effectively. It is believed that learning is a process which is related to how many sense organs are active in it. The more the sense organs work, the easier kid learns. So that also can be admitted another reason to add some voices to the application.

4.8 Eye Gaze Interface

Involving technology needs to make the use of hands faster or another method because of new generation processors and high-speed computing units. In this study, eye gaze as the fastest organ of the human body is used to give commands to the computer. Four types of eye gaze interface are designed for kid's education to learn letters, to teach animals, to learn colors and to teach the kids the numbers. There are a lot of challenges to be handled like clicking, calibration, environment issues and physical issues. The clicking is the most challenging task. This is achieved by timing issues with the computer operating system facilities. Further studies can be done using an RTOS (real time operating system) due to its task managing and timing accuracy. The application depends on the calculation of the position of user gaze on screen and test results are presented as a written text feedback.

4.8.1 Eye Gaze Animals

Many types of communication and gaming devices are nowadays in the hands of the kids. They are mostly in use of game and social media interaction programs. Many enjoyable things on the internet harm the education because of the intention of joy. Also, urbanization makes the next generation out of the nature of plants and animals. In this interface is first designed for the kids to learn objects like animals as a beginning. It could also be developed by indoor and outdoor objects on different pages on this application to meet the children with the world and, but it was the out of the target.

Simply, kids are toughed to look the first calibration then asked to look an animal said by its name. Figure 4.15 shows the eye gaze interface for animal names.

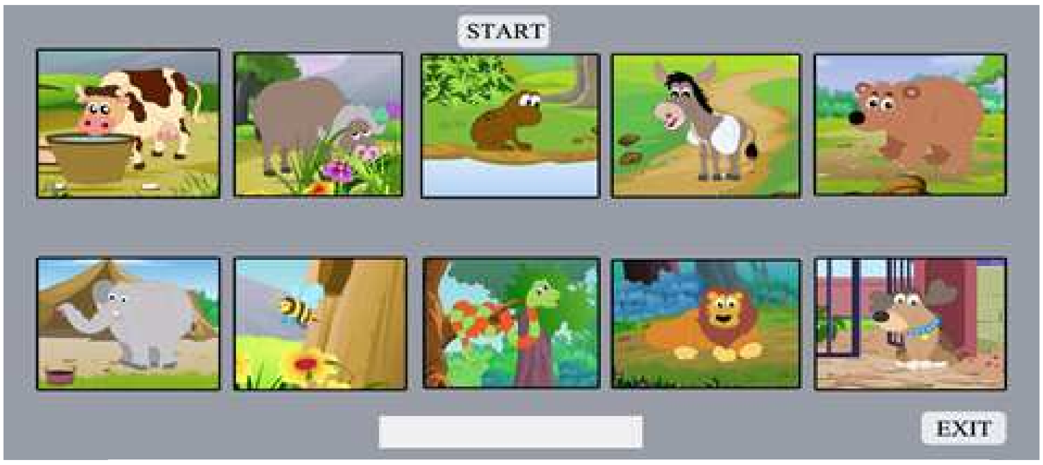


Figure 4.15 Eye gaze interface for animals

If the kid is wrong, hears the different voice of animals and tries again. Both voices hearing and seeing the letters of the animal makes the learning fast. It also increases the reading capability of kids. The interface is not only for the kids, and also could be used for learning languages to adults.

4.8.2 Eye Gaze Alphabet

Alphabet interface is one of our best studies for preschoolers to learn letters in a playful and funny manner. Learning A, B and C happen quickly, easily and correctly pronounced vowels and consonants of the ABC for toddlers. Some sounding of voices in the background lets the toddler pronounce truly.

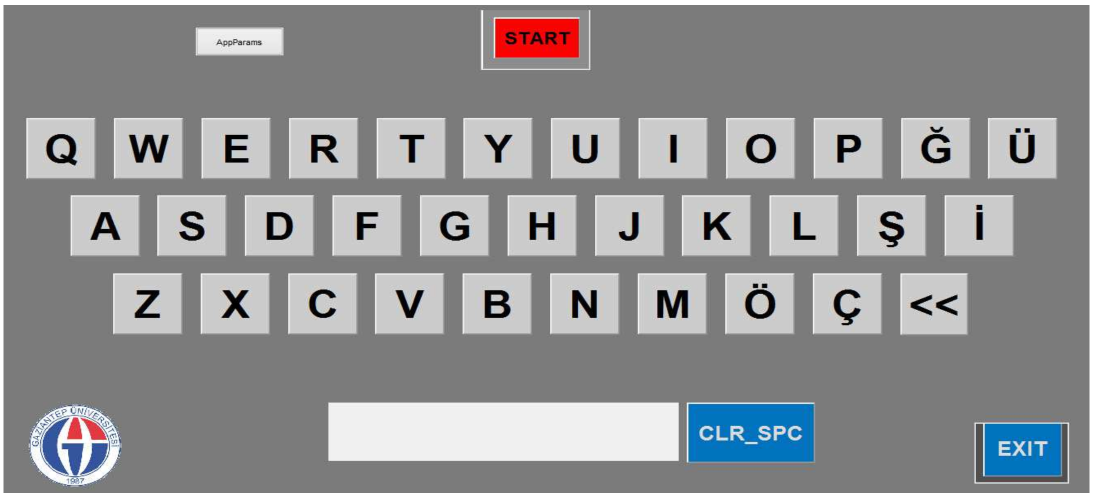


Figure 4.16 Eye gaze interface for alphabet

This interface helps the kids to stimulate fine motor skills and useful for future study at school. The game is played after calibration process looking at three points on the screen. Then, kid looks a letter and hears what it is. This game also assists for the adults to get forced to learn letters of a language such as Arabic, Russian and Chinese etc. as seen in Figure 4.16. The study showed that the distance between the letters should be enough to detect and for better result higher resolution cameras are needed for getting better results.

4.8.3 Eye Gaze Color

Eye gaze color interface for the babies, kids and toddlers provide to name the colors and ability to tell by heart. It is simple and joyful to learn. This application could be joined with animal objects and teach kids both color and animals. Color names are universal but hard to explain in different densities. Color is a valuable pointing feature in daily life. So that it is important to kids to socialize. Kids learn ten basic colors with this eye gaze color app. In learn colors section, the kids can be trained with memorizing of ten basic colors. Kids easily navigate among color icons as seen in Figure 4.17. Toddlers look at a color button and see the name of it and hear.

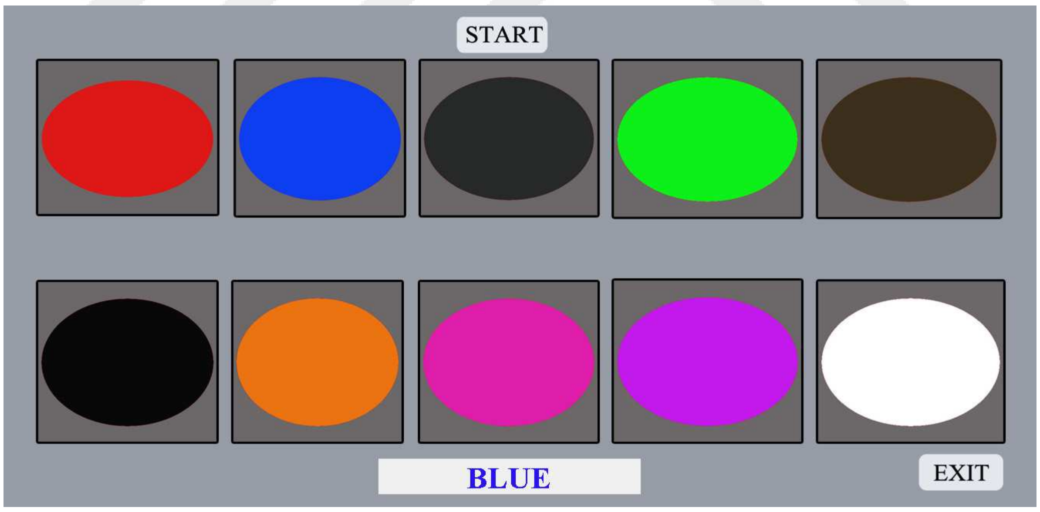


Figure 4.17 Eye gaze interface for colors

4.8.4 Eye Gaze Number

Kids are very pure mind to learn easily and must be tended to learn with love. This can be done with fun and easy manner with supported a kind of music and pictures like a game. In this interface, there is a start button to begin to play the game. First, calibration is done and then e-learn GUI is ready. Alternatively, this application as number usage can be also modified for telephones. Telephones allow the user to place and receive calls. If the interface is figured as a mobile application, frequently used numbers can be stored in a telephone "book”.

On-verbal users or disabled peoples may access the numbers as shown in Figure 4.18 with their eyes non-using a speech synthesizer to talk [45].

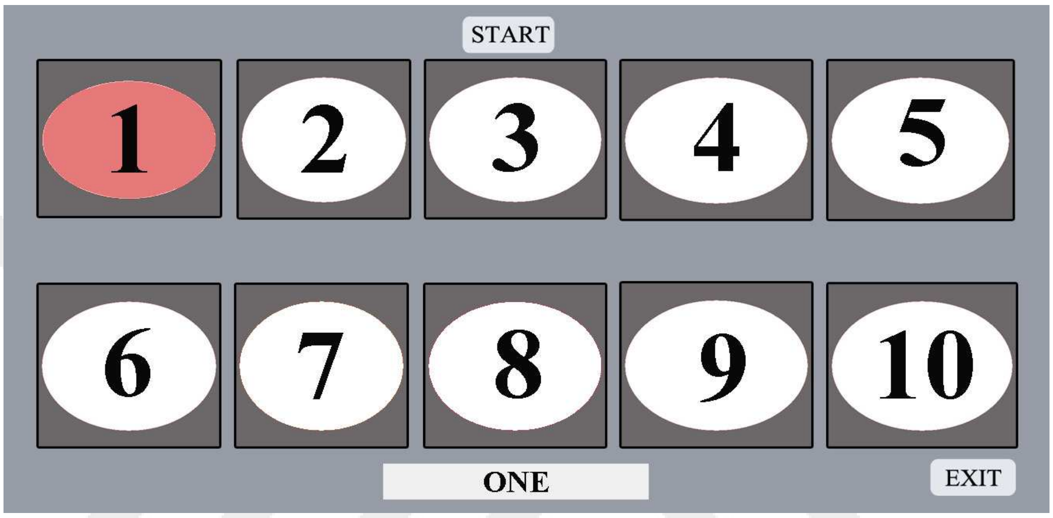


Figure 4.18 Eye gaze interface for numbers

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Introduction

The application is primarily developed on a Windows 7 PC with Intel Core i7-3630QM, 2,4 GHz. Monitor size is 13.6x7.6 in. Image is acquired by Logitech webcam at 30 fps. The image is processed as gray scale of 800x600 pixels using some structs of Matlab Toolboxes. The images are acquired and processed by Matlab Image Processing and Computer Vision Toolbox after the calibration.

5.2 Eye Tracking Experiment

Screen coordinates are mapped to image points before application tests. The algorithm is verified one point by one point through the screen in three directions.

Matlab image acquisition application is used, an A3 paper is painted and glued to the screen and thus, eye gazes are navigated from left screen edge to right, up screen edge to down and up left screen edge to downright edge and diagonal eye localizations are computed with respect to screen points as a millimeter.

5.3 Vertical Eye Points

PC screen is equally divided into 12 points vertical and eyes are focused on these points one after one. Images are acquired that eyes are looking at different points of the screen vertically.

It is noticed that eye pair sizes are different from each due to Viola-Jones face detection. Some points are clearly very exact. It can be inferred from the images that eye centers are not much exact in y-direction because of eyelashes and eyelid. The eye circles are not detected very accurate.

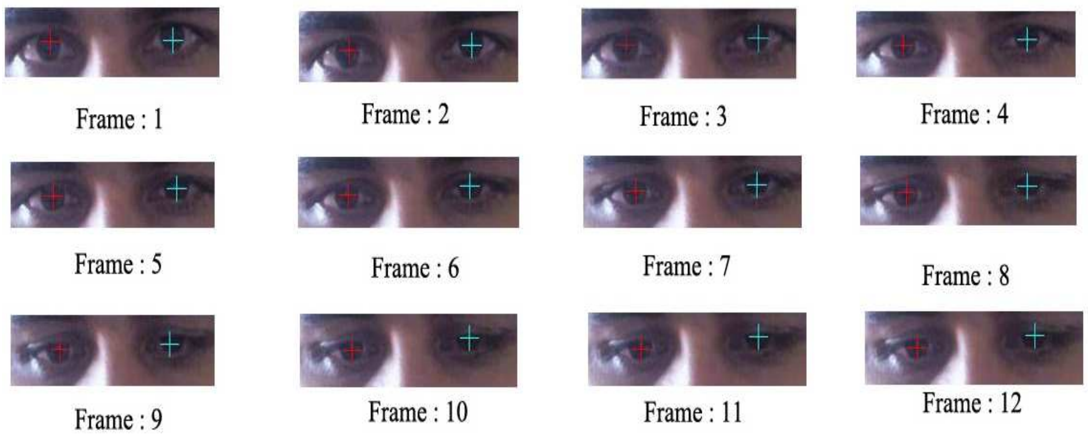


Figure 5.1 Vertical eye gaze localization points

Semi black circle in a white area is not clear. Figure 5.1 illustrates the images taken and Figure 5.2 demonstrates the position of the eyes with respect to the image points. Eye circles accuracy is weak at down point of the screen. When the points are matched, the error is less than 2mm and this can be negligible for the application.

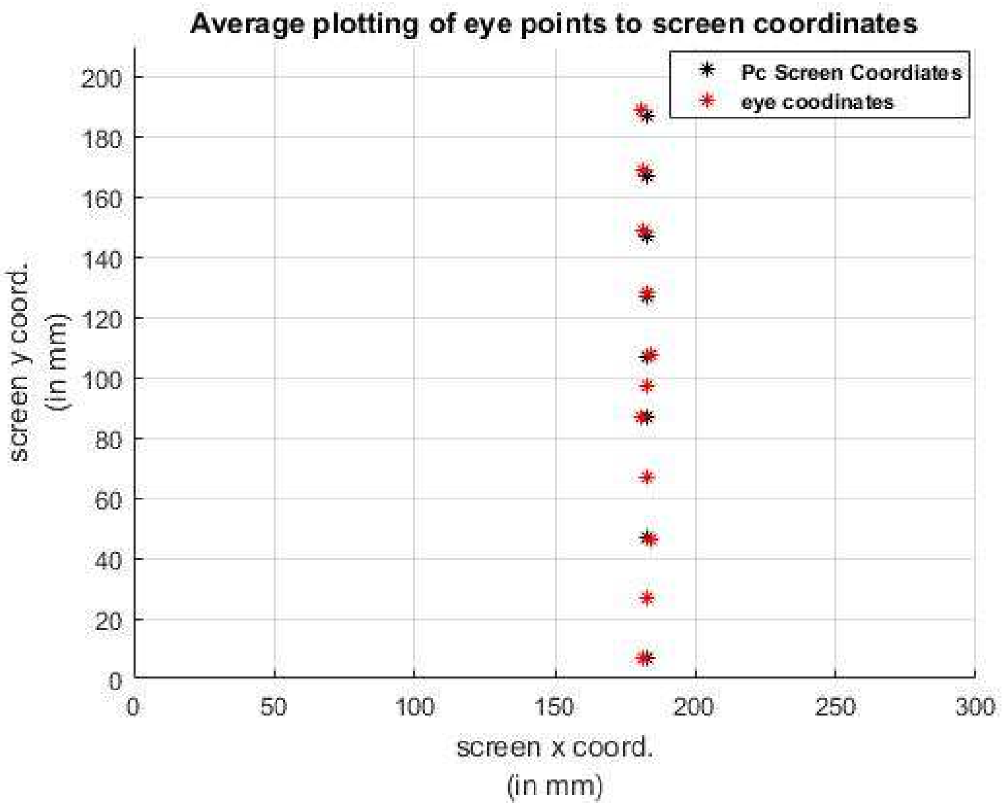


Figure 5.2 Vertically gaze points mapping to screen points

5.4 Horizontal Eye Points

As a pre-study, eye localization is mapped through the screen using the algorithm. First, PC screen is divided into 12 points and eyes are focused on these points. The images are acquired that eyes are looking at different positions of the screen horizontally. Figure 5.3 illustrates the images taken and Figure 5.4 demonstrates the position of the eyes with respect to the image points.

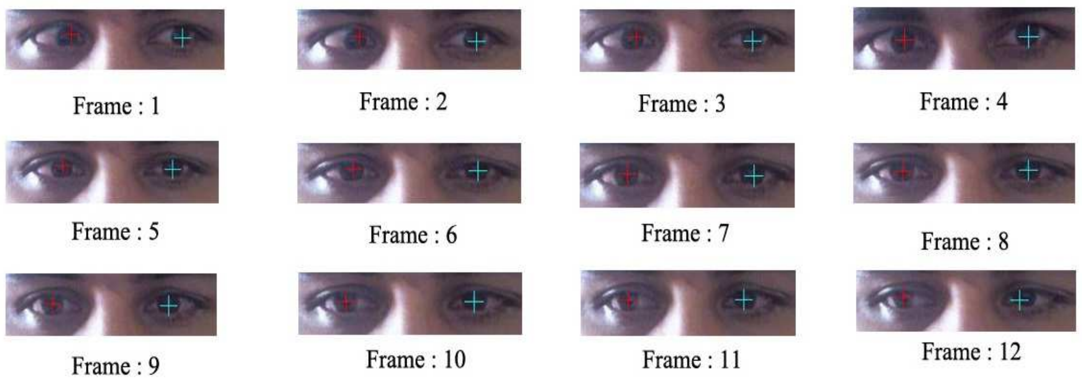


Figure 5.3 Horizontally gaze points mapping to screen points

It is noticed that eye pair sizes are different from each due to Viola-Jones face detection. It is obtained that there is a good detection/match at the left side of the screen points when compared with right side. Error is added to the algorithm and divided by two. This error is not big enough to fit for the eye positioning.

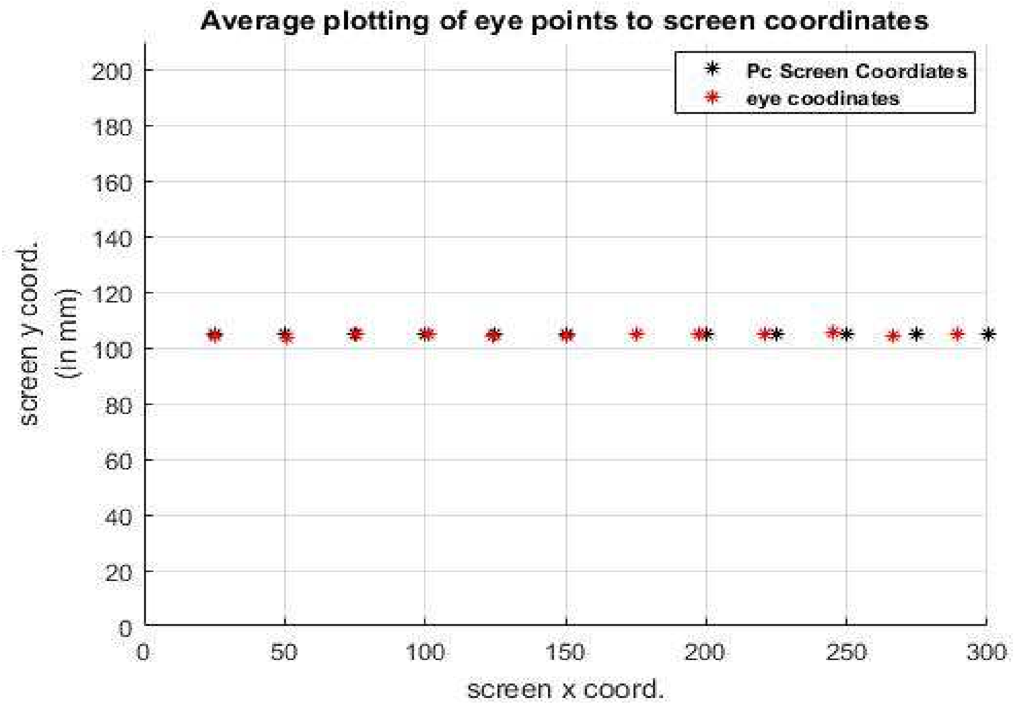


Figure 5.4 Horizontal eye gaze points

5.5 Diagonal Eye Points

PC screen is divided into 11 points and eyes are focused on these points. Images are acquired that eyes are looking at different positions of the screen diagonally. Figure 5.5 illustrates the images taken and Figure 5.6 demonstrates the position of the eyes with respect to the image points. At first glance, results seem not accurate but for a middle-level application, results are good. At the of the screen points, matches are not good enough so that algorithm is not going to work well at this points. Button sizes are as big as it is fitted with the algorithm.

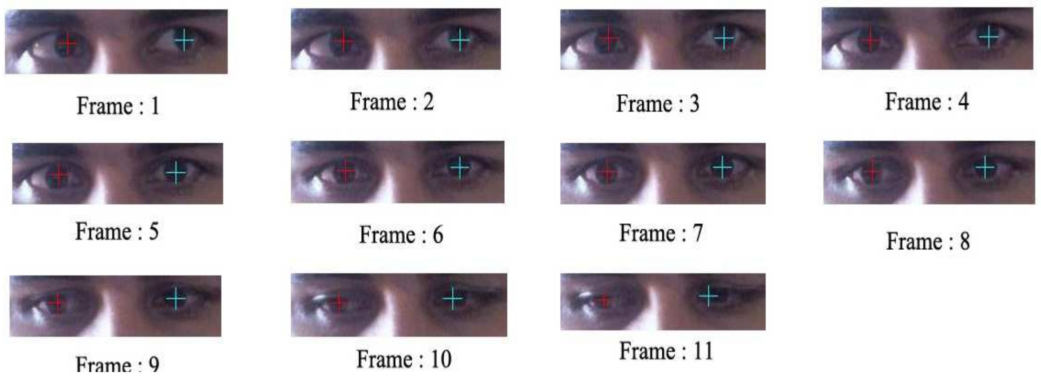


Figure 5.5 Diagonal eye gaze points

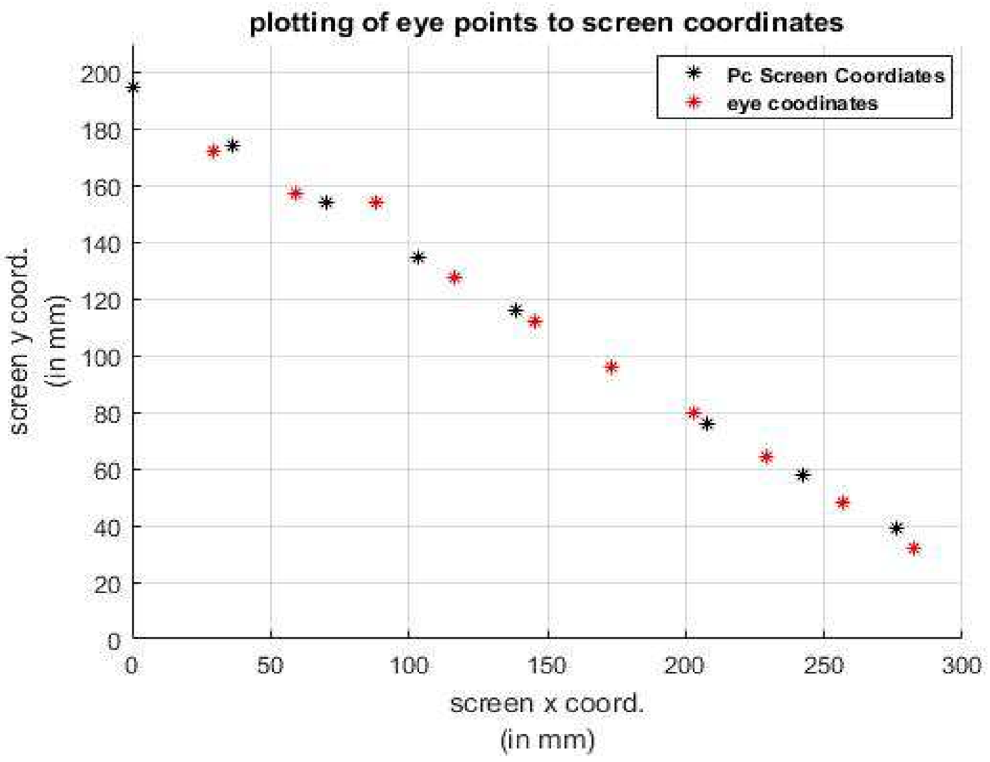


Figure 5.6 Diagonally gaze points mapping to screen points

5.6 2D Gaze Controlling Experiment

Users are trained about how to calibrate, start and stop the application. Whenever kids want to use this GUI some like shown in Figure 5.7, the first calibration process is rapidly run. After calibration, the time user or kid looks at the picture on the screen, animal voices sound and animals name are seen on the screen as a response. The user can listen to all the voices of the animals which (s) he points out. As a response, the name of the animal is typed in the area below. Exit button is clicked and application is quitted. It must be specified that application cannot be used without a mouse. This makes senses when all use of a device is eye gazed. For example, if a disabled people have no ability to talk, (s) he can show the letters on the screen in such a way.

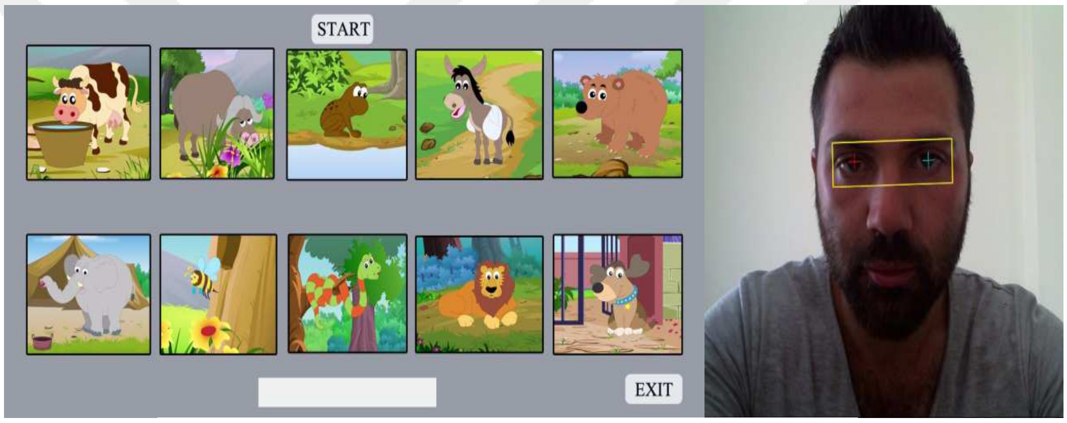


Figure 5.7 Eye gaze tracking game

CHAPTER 6

CONCLUSION AND DISCUSSION

6.1 Summary of the Results

The purpose of the study was to whether eye gaze assisting technology would create a new interaction method for some peoples. The work is clearly understood that using the only conventional computer vision (CV) techniques (Viola-Jones Algorithm) is not sufficient to solve this problem, completely. But the results presented here are promising interaction technique for the future. The eye is a moving object so that changing the accuracy of the eye is limited. When people stare at something on the monitor, they are generally stationary. If could be possible to exclude head movements with some sensors, this might assist the proficiency of the gaze system.

Behind the chapter four scenes, working with a single frame at any fps can give irrelevant results from time to time, so it will increase usability by getting more than one frame at a time and reaching a result with a mean of these frames. This is achieved by more sample frames and averaging. Maybe a more microprocessor helps at least on acquisition part. Eye detection cannot be precise to curser between two letters. But large items are suitable for the beginning. The problem is achieved enlarging the GUI elements. The huge picture buttons are studied in this study. As a step forward, gif files could be used to pictures, some animations like lighting fireworks. On the other hand, reading, awareness, physical and emotional states, desires, attention would be the valuable further topics in this field. Eye complexity and gaze tracking issues requires special instrumentation and/or devices more than one computer and a webcam as devices.

6.2 Conclusion for Eye Gaze Systems

In this dissertation, a simple eye-tracking technology is described. The most unique of this system is completely software wise. That is, no wearable, any touchy things required. We proposed a low-cost camera for the users to learn objects having fun with the sounds. Pupils at a circular geometry are scanned through eye pair region. The system can be implemented depending on well-lightened environment and tilt of the camera. The experimental results approved that usability of our proposed kids' training method although, conscious and unconscious moves of eye occur. But still, mouse and keyboard are more efficient and this eye gaze technology needs more expensive tools to compete with others. Furthermore, head movements can be included in the algorithm and more accurate detection can be achieved. In the end, more funny and flexible algorithm for the game is gained.

6.3 Further Work

To the results obtained the point of gaze on the computer screen or the accuracy of an area processed will contribute to ease and time efficiency. Because, the user looks at the screen as fast as possible and communicates with it only applying a calibration process as an entry. For this purpose, it is foreseen that hardware-based work is unnecessary. It can be further work to use eye blink as right and left clicks to assign the mice moves. An advanced Viola-Jones Algorithm in the OpenCV library can be tried for next step study in this direction. Also, a constant threshold value can be calculated by performing tests on different lighting media, and adaptive Thresholding can be done to increase the sensitivity of the eyeball. Designing a specific eye detector using a traditional CV techniques (CASCADE, HOG, LBP), a system object can be programmed.

It can also be run on specialized face detection libraries as a pre-process to go even further for getting clear and clear results, but, face landmarks could be more accurate but no efficiency. Obviously, deep learning as an artificial intelligence is the most effective way to solve many problems of image processing. It is only necessary to create a database and create shape browser software. It is proven by this article that deep learning can make more real and definitive decisions compared to the human mind today. One more thing, a parallel computing library might be included to go further.

REFERENCES

[1]. Saccading as eye movement: https://www.wikizero.com/en/Saccade 20.11.2017

[2]. Sibert and Jacob.2000. Evaluation of Eye Gaze Interaction : www.cs.tufts.edu 20.11.2017

[3]. Kyung-Nam Kim.1999.Vision-Based Eye-Gaze Tracking for Human Computer, Department of Information and Communications, Kwangju Institute of Science and Technology, Kwangju, 500-712

[5]. Bülent Turan.2015. Eye Tracing and Cursor Control Method with Domestic Webcam doctorate thesis, https://tez.yok.gov.tr/UlusalTezMerkezi /tezSorguSonuc 17.10.2017

[6]. D.W.Hansen and Q.Ji.2010. In the eye of the beholder: a survey of models for eyes and gaze. Pattern analysis and Machine intelligence, IEEE transactions on, pp. 478-500

[7]. I.Fasel, B.Fortenberry and J.Movellan.2005.A Generative Framework for real time object Detection and Classification *",Computer Vision and Image Understanding*,**98** (1),182-210

[8]. Kass, M. Witkin, A. Terzopoulos, D. Snakes.1998.Active contour models, *International Journal of Com5puter Vision***,** **1**(4), pp.321-331.

[9]. R.Valenti and T.Gevers.2008.Accurate eye center localization and tracking using isophote curvature" in international Conference on Computer Vision and Pattern Recognition (CVPR'08),Amsterdam, the Netherlands, pp.1-8

[10].R.L.Hsu, M. Abdel-Mottaleb, and A.K.Jain.2002.Face detection in color images, Pattern Analysis and Machine Intelligence, IEEE Transactions on , pp-.696-706

[11]. Eye regions figüre**:** https://pocketdentistry.com/2-surface-anatomy/ 20.11.2017

[12]. Eye Tracking for Disabled People, https://www.psfk.com/2014/12/new-eye-tracking-mouse-opens-computing-to-users-with-disabilities.html 20.10.2017

[13]. Eye tracking in infant and child research, https://www.tobiipro.com/fields-of-use/infant-child-research/ 20.09.2017

[14]. Eye Gaze Tracker Game for Kids, http://www.eyegaze.com/eye-tracking-assistive-technology-device/ 20.11.2017

[15]. S.V. Sheela et al. 2011. *An Appearance based Method for Eye Gaze Tracking Journal of Computer Science* **7(**8): 1194-1203

[16]. D.W.Hansen and Q.Ji.2010. In the eye of the beholder: a survey of models for eyes and gaze. Pattern analysis and Machine intelligence, IEEE transactions on, **32,** pp.478-500

[17]. Rikert and Jones, Morphable Model. 1998.Kar-Han et al.2002.Appearance manifold Gaussian interpolation Sugano et al.2012. and Flavio et al. H.R. Chennamma et. al. cross-ratio / *Indian Journal of Computer Science and Engineering (IJCSE)* ISSN: 0976-5166 **4**(5) Oct-Nov 2013 391

[18]. Contact lenses, http://www.colormecontacts.com/colored-lens-haul-splash-of-color-freshlook-and-acuvue/ 20.11.2017

[19]. EOG electrodes,http://aibolita.com/sundries/23641-overview-of-major-eyemovement - recording-technologies.html 20.11.2017

[20]. Yoo, D.H et al. 2006.Non-intrusive eye gaze estimation using a projective invariant under head movement, In Proceedings of IEEE International Conference on Robotics and Automation (ICRA'06), pp. 3443- 3448

[21]**.** M.Ramezanpour et al. 2010. A New Method for Eye Detection in Color Images. *Journal of Advance in Computer Research*, http://jacr.iausari.ac.ir/article_2393 e3b9 f237 e0b7e3dd32803.pdf 20.11.2017

[22]. Shubham Singh et al.2014.Cursor Control Using Pupil Tracking. *International Journal of Engineering and Technical Research (IJETR)* ISSN: 2321-0869, **2,** Issue-11,

[24]. Bülent Turan “Eye Tracing and Cursor Control Method with Domestic Webcam” Doctorate Thesis

[25]. Visal Kith et al.2008.A feature and appearance based method for eye detection on gray intensity face images, Computer Engineering & Systems, ICCES. International Conference on, http://ieeexplore.ieee.org/document /4772963 . 20.11.2017

[26]. Head movements Figure, https://www.qualcomm.com/ news/on/2016/ 08/11 /device-motion-tracking 20.11.2017

[27]. Different lighting effects on a face Figure, https://www.computer.org/csdl /trans/tp/ 2009/11/ttp2009111968-abs.html 20.11.2017

[28]. Slama, C. C. (ed.) (1980) Manual of Photogrammetry, 4th edition, American Society of Photogrammetry, Falls Church, Virginia.

[29]. Abdel-Aziz, Y. I. & Karara, H. M.1971. Direct linear transformation into object space coordinates in close-range photogrammetry. Proc. Symposium on Close-Range Photogrammetry, Urbana, Illinois, p. 1-18.

[30]. Melen, T. 1994. Geometrical modelling and calibration of video cameras for underwater navigation. Dr. ing thesis, Norges tekniske høgskole, Institutt for teknisk kybernetikk.

[31]. Zhang, Z.2000. A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence. **22**(11) pp. 1330–1334

[32]. Heikkila et al.1997. A Four-step Camera Calibration Procedure with Implicit Image Correction. IEEE International Conference on Computer Vision and Pattern Recognition.

[33]. Bradski, G et al. 2008. Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol, CA: O'Reilly.

[34]. Bouguet, J. Y. Camera Calibration Toolbox for Matlab. Computational Vision at the California Institute of Technology. Camera Calibration Toolbox for MATLAB

[35]. Richard Hartley, Multiple View Geometry in Computer Vision, Second Edition

[36]. Distortion types Figure, https://photographylife.com/what-is-distortion 20.10.2017

[37]. Focal length Figure, https://www.learnopencv.com/tag/focal-length/20.10.2017

[38]. Course notes, http://ftp.cs.toronto.edu/pub/psala/VM/camera-parameters.pdf 20.10.2017

[39]. Matlab single camera calibrate webpace, https://www.mathworks.com/single-camera-calibrator
20.10.2017

[40]. Matab as a programming language,http://www.matlabtips.com/what-is-matlab-where-how-and-when-to-use-it, 20.10.2017

[41]. OpenCV camera calibration, https://docs.opencv.org/2.4 /doc/tutorials/calib3d/ camera_calibration/ 20.10.2017

[42]. Yi-Qing Wang.2014. An Analysis of the Viola - Jones Face Detection Algorithm, Published in Image Processing On Line (IPOL) on 06–26.

[43]. Gaussian Filtering, https://homepages. inf. ed.ac. uk/rbf/HIPR2/gsmooth. htm. 20.10.2017

[44]. N.R**.** Mokhtar.2009. Image Enhancement Techniques Using Local, Global, Bright, Dark and Partial Contrast Stretching For Acute Leukemia Images, Proceedings of the World Congress on Engineering 2009 I, WCE 2009, July 1 - 3, London, U.K

45].Eye gaze system use, http://www.gschlosser.de/eyegaze_english.htm.20.10.2017

[46]. Rafael C. Gonzalez, Richard E.Woods, Digital Image Processings, third edition

APPENDIX A

Matlab Code (.m file)

A.1 Main GUI

```
% -----------------------------------------------------------------------------------------------
%  * Matlab Script (TEZ_GUI. m) which is main script for the application.
%  * Purpose: Creating a user interface with axes, buttons to eye interact and also,
%                   use of callbacks for other scripts.
%   ********** Author: Sinan KESKIN
%   ********** Date: August 12, 2017
%   ********** Version: 1,0
%   ********** (c) 2017,University of Gaziantep, Turkey
%   ********** E-mail: sinankeskin15@gmail.com
% -----------------------------------------------------------------------------------------------
function varargout = TEZ_GUI(varargin)
% TEZ_GUI MATLAB code for TEZ_GUI. fig
%      TEZ_GUI, by itself, creates a new TEZ_GUI or raises the existing
%      singleton*.
%      H = TEZ_GUI returns the handle to a new TEZ_GUI or the handle to
%      the existing singleton*.
%      TEZ_GUI('CALLBACK',hObject, eventData, handles, ...) calls the local
%      function named CALLBACK in TEZ_GUI. M with the given input arguments.
%      TEZ_GUI('Property','Value', ...) creates a new TEZ_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before TEZ_GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to TEZ_GUI_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help TEZ_GUI
% Last Modified by GUIDE v2.5 02-Sep-2017 13:46:09
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State=struct('gui_Name'mfilename,
                'gui_Singleton',  gui_Singleton, ...
                'gui_OpeningFcn', @TEZ_GUI_OpeningFcn, ...
                'gui_OutputFcn',  @TEZ_GUI_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})

    gui_State. gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}]=gui_mainfcn(gui_State,
varargin{:});
Else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
 % --- Executes just before TEZ_GUI is made visible.
function TEZ_GUI_OpeningFcn(hObject, ~, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to TEZ_GUI (see VARARGIN)
% Choose default command line output for TEZ_GUI
% Images are loaded to the user interface in the beginning.
handles. output = hObject;
bg_image = imread('cow. png');
bg_image=imresize(bg_image,[170,170]);
Set(handles. q, 'CData', bg_image);
bg_image1 = imread('elephant. png');
bg_image1=imresize(bg_image1,[170,170]);
Set(handles. C, 'CData', bg_image1);
bg_image1 = imread('dog. png');
bg_image1=imresize(bg_image1,[170,170]);
Set(handles. E, 'CData', bg_image1);
bg_image1 = imread('frog. png');
bg_image1=imresize(bg_image1,[170,170]);
set(handles. T, 'CData', bg_image1);
bg_image1 = imread('zebra. png');
bg_image1=imresize(bg_image1,[170,170]);
set(handles. U, 'CData', bg_image1);
bg_image1 = imread('kangaroo. png');
bg_image1=imresize(bg_image1,[170,170]);
set(handles. O, 'CData', bg_image1);
bg_image1 = imread('bear. png');
bg_image1=imresize(bg_image1,[170,170]);
set(handles. M, 'CData', bg_image1);
bg_image1 = imread('lion. png');
bg_image1=imresize(bg_image1,[170,170]);
set(handles. B, 'CData', bg_image1);
bg_image1 = imread('dankey. png');
bg_image1=imresize(bg_image1,[170,170]);
set(handles. Z, 'CData', bg_image1);
bg_image1 = imread('snake. png');
bg_image1=imresize(bg_image1,[170,170]);
set(handles. L, 'CData', bg_image1);
% Animal sounds are loaded to the memory in the beginning.
handles. c=audioread('cow.mp3');
handles. bu=audioread('buffalo.mp3');
handles. f=audioread('frog.mp3');
handles. z=audioread('zebra.mp3');
handles. k=audioread('kangaroo.mp3');
handles. d=audioread('dog.mp3');
handles. e=audioread('elephant.mp3');
handles. l=audioread('lion.mp3');
handles. be=audioread('bear.mp3');
handles. s=audioread('snake.mp3');
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes main wait for user response (see UIRESUME)
% uiwait(handles.figure1);
setappdata(0,'hMainGUI',gcf);
setappdata(gcf,'mainHandles',handles);
setappdata(0,'mainHandles',handles);
% --- Outputs from this function are returned to the command line.
function varargout = TEZ_GUI_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles. output;
% Enlarge GUI to full screen and logo is loaded.
set (gcf, 'Position', get(0,'Screensize'));
set (gcf,'name','CURSOR MOVEMENT','numbertitle','off')
Axes (handles. logo);
imshow('C:\Users\WORLD\Documents\MATLAB\MATLAB-TEZ-GUI\calısma_02\logo.
png')
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
a='C';
set(handles.edit1,'String',a);
% Update handles structure
% --- Executes on button press in q.
function q_Callback(hObject, eventdata, handles)
% hObject handle to q (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
handles. output = hObject;
oldString=get(handles. typing,'String');
newString='COW';
textstring=strcat(oldString, newString);
sound(handles. c);
pause(2,4);
clear sound
set(handles. typing,'String',textstring);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
```

```
İf is pc && is equal (get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
 % --- Executes on button press in exit.
function exit_Callback(hObject, eventdata, handles)
% hObject handle to exit (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
delete(handles.figure1);
% function img_acq_Callback(hObject, eventdata, handles)
% % hObject handle to img_acq (see GCBO)
% % eventdata reserved - to be defined in a future version of MATLAB
% % handles structure with handles and user data (see GUIDATA)
% img_acq;
% --- Executes on button press in E.
function E_Callback(hObject, eventdata, handles)
% hObject handle to E (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='BUFFALO';
sound(handles. bu);
pause(2,4);
clear sound
textstring=strcat(oldString, newString);
Set(handles. typing,'String',textstring);
% --- Executes on button press in T.
function T_Callback(hObject, eventdata, handles)
% hObject handle to T (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='FROG';
sound(handles. f);
pause(2,4);
clear sound
textstring=strcat(oldString, newString);
Set(handles. typing,'String',textstring);
% --- Executes on button press in U.
function U_Callback(hObject, eventdata, handles)
% hObject handle to U (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='ZEBRA';
sound(handles. z);
pause(2,4);
clear sound
textstring=strcat(oldString, newString);
set(handles. typing,'String',textstring);
% --- Executes on button press in O.
function O_Callback(hObject, eventdata, handles)
% hObject handle to O (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='KANGAROO';
sound(handles. k);
pause(2.4);
clear sound
textstring=strcat(oldString, newString);
set(handles. typing,'String',textstring);
% --- Executes on button press in L.
function L_Callback(hObject, eventdata, handles)
% hObject handle to L (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='SNAKE';
sound(handles. s);
pause(2,4);
clear sound
textstring=strcat(oldString, newString);
set(handles. typing,'String',textstring);
% --- Executes on button press in Z.
function Z_Callback(hObject, eventdata, handles)
% hObject handle to Z (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='DANKEY';
sound(handles. d);
pause(2,4);
clear sound
textstring=strcat(oldString, newString);
set(handles. typing,'String',textstring);
% --- Executes on button press in C.
function C_Callback(hObject, eventdata, handles)
% hObject    handle to C (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='ELEPHANT';
sound(handles. e);
pause(2,4);
clear sound
textstring=strcat(oldString, newString);
set(handles. typing,'String',textstring);

 % --- Executes on button press in B.
function B_Callback (hObject, eventdata, handles)
% hObject handle to B (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='LION';
sound(handles. l);
```

```
pause(2.4);
clear sound
textstring=strcat(oldString, newString);
set(handles. typing,'String',textstring);

 % --- Executes on button press in M.
function M_Callback(hObject, eventdata, handles)
% hObject handle to M (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
oldString=get(handles. typing,'String');
newString='BEAR';
sound(handles. be);
pause(2,4);
clear sound
textstring=strcat(oldString, newString);
set(handles. typing,'String',textstring);
% --- Executes on button press in pushbutton100.
function pushbutton100_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton100 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
 set(handles. typing,'String','');
% --- Executes on button press in Del.
function Del_Callback(hObject, eventdata, handles)
% hObject handle to Del (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
yazilan=get(handles. typing,'String');
ekle='';
yeni=strcat(yazilan(1:end-1),ekle);
set(handles. typing,'String',yeni);
% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject handle to start (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Calibration process before app runs.
[a,b]=One_Frame();     % Top left corner coordinates of the UI
a=abs(a);
b=abs(b);
pause(1);
[m,n]= One_Frame ();   % Bottom right corner coordinates of the UI
m=abs(m);
n=abs(n);
% abs(a-m)
% abs(n-b)
pause(1);
%% Application process run.
While(1);
[x,y] = One_Frame ();
x=abs(x);
y=abs(y);
pause(0.5);
```

```
end
%% For verification of the coordinates acquired.
%  a,m,x
%  b,n,y
% Close All;
% Outputs for top row images
if (x<a+abs(a-m)*3/28 & x>a+abs(a-m)-20  & y>b-abs(b-n)*0.5 & y<b+20)
 oldString=get(handles. typing,'String');
 newString='COW';
 textstring=strcat(oldString, newString);
 set(handles. typing,'String',textstring);
end
 if (x>=a+abs(a-m)*3/28 & x<a+abs(a-m)*105/280  & y>b-abs(b-n)*0.5 & y<b+20)
 oldString=get(handles. typing,'String');
 newString='BUFFALO';
 textstring=strcat(oldString, newString);
 set(handles. typing,'String',textstring);
end
 if (x>=a+abs(a-m)*105/280 & x<a+abs(a-m)*175/280  & y>b-abs(b-n)*0.5 & y<b+20)
 oldString=get(handles. typing,'String');
 newString='FROG';
 textstring=strcat(oldString, newString);
 set(handles. typing,'String',textstring);
end
if (x>=a+abs(a-m)*175/280 & x<a+abs(a-m)*245/280  &y>b-abs(b-n)*0.5 & y<b+20)
 oldString=get(handles. typing,'String');
 newString='ZEBRA';
 textstring=strcat(oldString, newString);
 set(handles. typing,'String',textstring);
end
 if (x>=a+abs(a-m)*245/280 & x<a+abs(a-m)+100  & y>b-abs(b-n)*0.5 & y<b+20)
 oldString=get(handles. typing,'String');
 newString='KANGAROO';
 textstring=strcat(oldString, newString);
 set(handles. typing,'String',textstring);
end
% Outputs for bottom row images
 if (x<a+abs(a-m)*3/28 & x>a+abs(a-m)-20  & y<=b-abs(b-n)*0.5 & y>n-20)
 oldString=get(handles. typing,'String');
 newString='DANKEY';
 textstring = strcat(oldString, newString);
 set(handles. typing,'String',textstring);
 end
 if (x>=a+abs(a-m)*3/28 & x<a+abs(a-m)*105/280  & y<=b-abs(b-n)*0.5 & y>n-20)
 oldString=get(handles. typing,'String');
 newString='ELEPHANT';
 textstring = strcat(oldString, newString);
 set(handles. typing,'String',textstring);
 end
 if (x>=a+abs(a-m)*105/280 & x<a+abs(a-m)*175/280 & y<=b-abs(b-n)*0.5 & y>n-20)
 oldString=get(handles. typing,'String');
 newString='BEE';
 textstring = strcat(oldString, newString);
 set(handles. typing,'String',textstring);
 end
 if (x>=a+abs(a-m)*175/280 & x<a+abs(a-m)*245/280  & y<=b-abs(b-n)*0.5 &   y>n-20)
 oldString=get(handles. typing,'String');
 newString='BEAR';
 textstring = strcat(oldString, newString);
 set(handles. typing,'String',textstring);
 end
  if (x>=a+abs(a-m)*245/280 & x<a+abs(a-m)+100  & y<=b-abs(b-n)*0.5 & y>n-20)
 oldString=get(handles. typing,'String');
 newString='SNAKE';
 textstring = strcat(oldString, newString);
 set(handles. typing,'String',textstring);
 end
```

A.2 One Frame Detect Function (One_Frame. m)

% --
% * Matlab Script (One_Frame. m) which is eye localization script for the
%   application.
% * **Purpose**: This function takes a one frame of video and find the coordinates of
%   eye circle using Circular Hough transform.
% ********** **Author:** Sinan KESKIN
% ********** **Date:** August 12, 2017
% ********** **Version**: 1,0
% ********** (c) 2017, University of Gaziantep, Turkey
% ********** **E-mail:** sinankeskin15@gmail.com
% --

```
% Image Acquisition Camera Settings
% Date: 14:04:2017
% A point between eyes is described using right and left eye center points.
function[x_ori, y_ori]=ilkFrame()
%% Voice is loaded and a frame is acquired.
a=audioread('instr_piano. wav');
vidobj=videoinput('winvideo',2,'MJPG_640x480');
start (vidobj);
sound(a);
pause(1);
data = getsnapshot(vidobj);
% Imwrite(data,'1.jpg');
figure, imshow(data);
%   Camera Calibration Settings
% Calibrated camera parameters are loaded and new origin is assigned                  %
After undistorting image.
load cameraParamsyd2.mat;
[imx, newOrigin] = undistortImage(data,cameraParamsyd2, 'OutputView', 'full');
% Face Detection Part
% Viola Jones algorithm is used to find eye regions.
faceDetector = vision. CascadeObjectDetector('EyePairBig');
faceDetector. MergeThreshold=3;
faceDetector. ScaleFactor=1.1;
boxFace = step(faceDetector, imx);

% Eye pairs are extracted.
```

```
for i = 1:size(boxFace,1)
detectedFace = imx(boxFace(i,2): boxFace(i,2)+ boxFace(i,4),...
            boxFace(i,1): boxFace(i,1)+boxFace(i,3),:);
 end
figure, imshow(detectedFace);
% Localizing of Iris positions
% Right Eye localization
rightFace = detectedFace(5:round(size(detectedFace,1) * 4.5/ 5),...
                  5:round(size(detectedFace,2) / 2));
% figure, imshow (rightFace); title('RightFace');
[r1,r2]=right(rightFace, newOrigin,cameraParamsyd2,boxFace, vidobj);
% Left Eye localization
leftFace = detectedFace(1:round(size(detectedFace,1)*4/5),...
             round(size(detectedFace,2) / 2):size(detectedFace,2));
%  figure, imshow(leftFace);title('LeftEye')
[l1,l2]=left(leftFace, newOrigin,cameraParamsyd2,boxFace, vidobj);
x_ori=(l1+r1)/2;
y_ori=(l2+r2)/2;
stop (vidobj);
end
```

A.3 Left eye Detect Function

```
% -----------------------------------------------------------------------------------------------
%  * Matlab Script (left. m) which is auxiliary script for the application.
%    * Purpose: Finding the left eye coordinates of the acquired image.
%   ********** Author: Sinan KESKIN
%   ********** Date: August 12, 2017
%   ********** Version: 1,0
%   ********** (c) 2017,University of Gaziantep, Turkey
%   ********** E-mail: sinankeskin15@gmail.com
% -----------------------------------------------------------------------------------------------
% localization of left eye center
function[x_ori, y_ori]=left(leftFace, newOrigin,cameraParamsyd2,boxFace, vidobj)
% Preprocessing Steps
% Apply Gaussian Filter
gaussSigma = 0.01*length(leftFace);
filter = fspecial('gaussian',[5 5], gaussSigma);
ima = imfilter(leftFace, filter);
% Apply Average Disc Filter
filt2 = fspecial('Disk',5);
im1 = imfilter(ima,filt2);
% Contrast Adjusting Step
im3 = imadjust(im1,stretchlim(im1),[]);
thresh = 14;
med = median(double(im3(:)));
imBW = im2bw(im3,med / 255 * thresh / 100);
figure, imshow(imBW);
% Morphological Process
% Remove of small objects
closingParameter = 15;
imBW = ~imBW;
imBW = bwareaopen(imBW, closingParameter);
imBW = bwareaopen(imBW, closingParameter);
imBW = ~imBW;
% Cut off the Edges
imBW = ~imBW;
imBW = bwmorph(imBW,'majority');
imBW = ~imBW;
% Circular Hough Transform
Rmin = 4;
Rmax = 20;
% [centersBright, radiiBright] = imfindcircles(imBW,[Rmin Rmax], 'ObjectPolarity',
'bright');
[centersDark, radiiDark] = imfindcircles(imBW,[Rmin Rmax],'ObjectPolarity','Dark');
if isempty(centersDark)
error('no circle found');
else
% viscircles(centersDark, radiiDark,'LineStyle','--');
end
%Calibration Distance Measurement Steps
load impyd2;  % Image Points
load wopyd2;  % World Points
% Compute extrinsics
% Detect the checkerboard, Compute rotation and translation of the camera.
[R, t] = extrinsics(imagePoints, worldPoints,cameraParamsyd2);
% Get the top-left and the top-right corners.
box1 = [centersDark(: ,1),centersDark(: ,2),-radiiDark: radiiDark,-adiiDark: radiiDark];
box1(: , 1: 2) = bsxfun(@plus, box1(: , 1 :2), newOrigin);
imagePoints1 = [box1(1: 2); ...
                        box1(1) + box1(3), box1(2)];
% Get the world coordinates of the corners
worldPoints1 = pointsToWorld(cameraParamsyd2, R, t, imagePoints1);
% Compute the diameter of the coin in millimeters.
d = worldPoints1(2, :) - worldPoints1(1, :);
diameterInMillimeters = hypot(d(1), d(2));
fprintf('Measured diameter of one eye = %0.2f mm\n',diameterInMillimeters);
% Distance Measure
 box2 = [newOrigin(:,1),newOrigin(:,2),centersDark(:,1),centersDark(:,2)];
 box2(:,1:2) = bsxfun(@plus, box2(:, 1:2), newOrigin);
 imagePoints2 = [box2(1:2);...
                        box2(1) + box2(3)+boxFace(1,1), box2(2)];
 worldPoints2 = pointsToWorld(cameraParamsyd2, R, t, imagePoints2);
 x_ori = worldPoints2(2, :) - worldPoints2(1, :);
 x_ori=hypot( x_ori(1),  x_ori(2));
%  fprintf('Measured x_distance of eye center = %0.2f mm\n', x_ori);
 imagePoints3 = [box2(1: 2); ...
                        box2(1),(box2(2)+ box2(4)+boxFace(1,2))];
 worldPoints3 = pointsToWorld(cameraParamsyd2, R, t, imagePoints3);
 y_ori = worldPoints3(2, :) - worldPoints3(1, :);
 y_ori=hypot( y_ori(1), y_ori(2));
%  fprintf('Measured y_distance of Eye center = %0.2f mm\n', y_ori);
% centerX=x_ori;
% centerY=y_dist;
End
```

A.4 Right Eye Detect Function

```
% -----------------------------------------------------------------------------------------------
%  * Matlab Script (TEZ_GUI. m) which is main script for the application.
% * Purpose: Finding the right eye coordinates of the acquired image.
%   ********** Author: Sinan KESKIN
%   ********** Date: August 12, 2017
%   ********** Version: 1. 0
%   ********** (c) 2017,University of Gaziantep, Turkey
%   ********** E-mail: sinankeskin15@gmail.com
% -----------------------------------------------------------------------------------------------
% Localization of left eye center
function[x_ori,y_ori]=right(rightFace, newOrigin,cameraParamsyd2,boxFace,vidobj)
%  Preprocessing Steps
% Apply Gaussian Filter
gaussSigma = 0.01*length(rightFace);
filter = fspecial('gaussian',[5 5], gaussSigma);
ima = imfilter(rightFace, filter);
% Apply Average Disc Filter
filt2 = fspecial('Disk',5);
im1 = imfilter(ima,filt2);
% Contrast Adjusting
im3 = imadjust(im1,stretchlim(im1),[]);
thresh = 14;
med = median(double(im3(:)));
imBW = im2bw(im3,med / 255 * thresh / 100);
% figure, imshow(imBW);
%  Morphological Process
closingParameter=15;
imBW = ~imBW;
imBW = bwareaopen(imBW, closingParameter);
imBW = bwareaopen(imBW, closingParameter);
imBW = ~imBW;
% Skeletonization
imBW = ~imBW;
imBW = bwmorph(imBW,'majority');
imBW = ~imBW;
% Circular Hough Transform
Rmin = 4;
Rmax = 20;
[centersBright, radiiBright] = imfindcircles(imBW,[Rmin Rmax],'ObjectPolarity','bright');
[centersDark, radiiDark] = imfindcircles(imBW,[Rmin Rmax],'ObjectPolarity','Dark');
if isempty(centersDark)
error('no circle found');
else
viscircles(centersDark, radiiDark,'LineStyle','--');
end

%Calibration Distance Measurement Steps
load impyd2;  % Image Points
load wopyd2;  % World Points
% Compute extrinsics, Detect the checkerboard
% Compute rotation and translation of the camera.
[R, t] = extrinsics(imagePoints, worldPoints,cameraParamsyd2);
```

```
% Get the top-left and the top-right corners.
box1 = [centersDark(: ,1),centersDark(: ,2),-radiiDark: radiiDark,-radiiDark: radiiDark];
box1(: , 1: 2) = bsxfun(@plus, box1(: , 1:2), newOrigin);
imagePoints1 = [box1(1: 2); ...
                     box1(1) + box1(3), box1(2)];
 % Get the world coordinates of the corners
 worldPoints1 = pointsToWorld(cameraParamsyd2, R, t, imagePoints1);
 % Compute the diameter of the coin in millimeters.
 d = worldPoints1(2, :) - worldPoints1(1, :);
 diameterInMillimeters = hypot(d(1), d(2));
 fprintf('Measured diameter of one eye = % 0.2f mm\n', diameterInMillimeters);
 % Distance Measure
 box2 = [newOrigin(:,1),newOrigin(:,2),centersDark(:,1),centersDark(:,2)];
 box2(: ,1: 2) = bsxfun(@plus, box2(: , 1. 2), newOrigin);
 imagePoints2 = [box2(1: 2); ...
                     box2(1) + box2(3)+boxFace(1,1), box2(2)];
 worldPoints2 = pointsToWorld(cameraParamsyd2, R, t, imagePoints2);
 x_ori = worldPoints2(2, :) - worldPoints2(1, :);
 x_ori=hypot( x_ori(1),  x_ori(2));
%  fprintf('Measured x_distance of eye center = %0.2f mm\n', x_ori);
 imagePoints3 = [box2(1: 2); ...
                     box2(1),(box2(2)+ box2(4)+boxFace(1,2))];
 worldPoints3 = pointsToWorld(cameraParamsyd2, R, t, imagePoints3);
 y_ori = worldPoints3(2, :) - worldPoints3(1, :);
 y_ori=hypot( y_ori(1), y_ori(2));
%  fprintf('Measured y_distance of Eye center = %0.2f mm\n', y_ori);
% centerX=x_ori;
% centerY=y_dist;
end
```

A.5 Algorithm Development Environment with GUI

```
%   ---------------------------------------------------------------------------------------------
%   * Matlab Script (algorithm_dev. m) which is main script for the application.
%   ***********Purpose: Creating a GUI for fast and efficient algorithm develop.
%   ********** Author: Sinan KESKIN
%   ********** Date: August 12, 2017
%   ********** Version: 1. 0
%   ********** (c) 2017,University of Gaziantep, Turkey
%   ********** E-mail: sinankeskin15@gmail.com
%   -------------------------------------------------------------------------------------------

function algorithm_dev(varargin)
% Create a figure;
clear, close, clc;
h= figure();
setappdata(h,'slidervalue',50);
% Enlarge figure to full screen.
set(gcf, 'Position', get(0,'Screensize'));
set(gcf,'name','CURSOR MOVEMENT','numbertitle','off')
b = axes(h,'Position',[0.7,0.5,0.29,0.49]);
% 4 Tane Push Button Oluştur
start = uicontrol('Parent',h,'Style', 'pushbutton', ...
          'String','START', ...
```

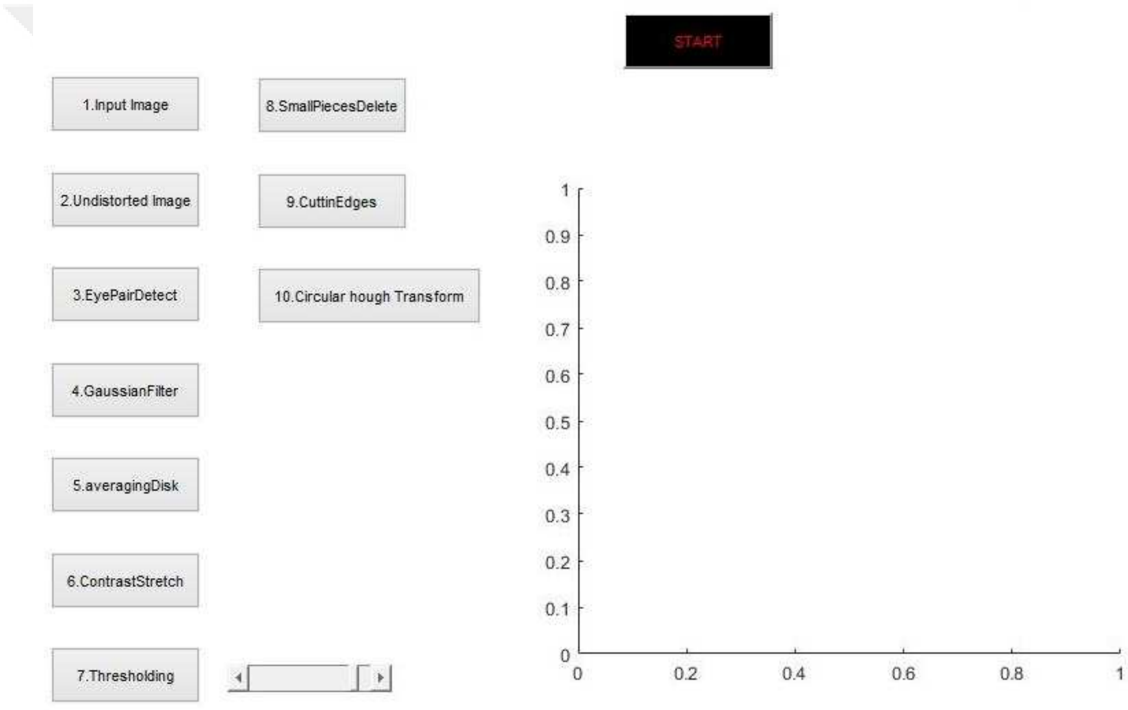
```
            'Units','normalized', ...
            'Position', [0.45 0.93 0.08 0.06]);
             start. ForegroundColor=[1 0 0];
             start. BackgroundColor=[0 0 0];
btn1 = uicontrol('Parent',h,'Style', 'pushbutton', ...
               'String','1.Input Image', ...
               'Units','normalized', ...
               'Position', [0.01 0.86 0.08 0.06]);
Set (btn1,'Callback',@inputim);
btn2 = uicontrol('Style', 'pushbutton', ...
               'String','2.Undistorted Image', ...
               'Units','normalized', ...
               'Position', [0.01 0.76 0.08 0.06], ...
               'Callback', {@undistorted,btn1});
btn3 = uicontrol('Style', 'pushbutton', ...
               'String','3.EyePairDetect', ...
               'Units','normalized', ...
               'Position', [0.01 0.66 0.08 0.06], ...
               'Callback', {@detect,btn2});
btn4 = uicontrol('Style', 'pushbutton', ...
               'String','4.GaussianFilter', ...
               'Units','normalized', ...
               'Position', [0.01 0.56 0.08 0.06], ...
               'Callback', {@blur,btn3});
btn5 = uicontrol('Style', 'pushbutton', ...
               'String','5.averagingDisk', ...
               'Units','normalized', ...
               'Position', [0.01 0.46 0.08 0.06], ...
               'Callback', {@averaging,btn4});
btn6 = uicontrol('Style', 'pushbutton', ...
               'String','6.ContrastStretch', ...
               'Units','normalized', ...
               'Position', [0.01 0.36 0.08 0.06], ...
               'Callback', {@contrast,btn5});

btn7 = uicontrol('Parent', h,'Style', 'pushbutton', ...
                'String','7.Thresholding', ...
                'Units','normalized', ...
                'Position', [0.01 0.26 0.08 0.06], ...
                'Callback', {@threshold,btn6});
% Create slider
sld = uicontrol('Parent', h,'Style', 'slider', ...
        'Min',1,'Max',50,'Value',45, ...
        'Position', [170 195 120 20], ...
        'Callback', {@slidore,btn7});
%     yazi = uicontrol('Style','Edit', ...
%              'String','50','Position',[220, 220, 30, 20], ...
%              'CallBack', {@yazi, sld}) ;
btn8 = uicontrol('Style', 'pushbutton', ...
               'String','8.SmallPiecesDelete', ...
               'Units','normalized', ...
               'Position', [0.15 0.86 0.08 0.06], ...
               'Callback', {@tbwareaopen,btn7});
btn9 = uicontrol('Style', 'pushbutton', ...
            'String','9.CuttinEdges', ...
            'Units','normalized', ...
            'Position', [0.15 0.76 0.08 0.06], ...
            'Callback', {@tbwmorph,btn8});
btn10 = uicontrol('Style', 'pushbutton', ...
            'String','10.Circular hough Transform', ...
            'Units','normalized', ...
            'Position', [0.15 0.66 0.12 0.06], ...
            'Callback', {@though,btn9});
% btn11= uicontrol('Style', 'pushbutton', ...
%           'String','8.SmallPiecesDelete', ...
%           'Units','normalized', ...
%           'Position', [0.15 0.56 0.08 0.06], ...
%           'Callback', {@tbwareaopen,btn10});
function inputim(hObject, eventdata, handles)
handles = guidata(hObject)
handles. vidobj=videoinput('winvideo',2);
start (handles. vidobj) ;
handles. image=getsnapshot(handles. vidobj);
imshow(handles. image);
stop (handles. vidobj);
guidata(hObject, handles);
function undistorted(hObject, eventdata, handles)
load cameraParamsyd2.mat
handles=guidata(handles);
handles. b=handles. image;
[handles. imx, handles. newOrigin] = undistortImage(handles. b, cameraParamsyd2,
'OutputView', 'full');
imshow(handles. imx);
guidata(hObject, handles);

function detect(hObject, eventdata, handles)
handles=guidata(handles);
handles. detect=handles. imx;
faceDetector = vision. CascadeObjectDetector('EyePairSmall');
faceDetector. MergeThreshold=3;
faceDetector. ScaleFactor=1.1;
handles. boxFace = step(faceDetector, handles. detect);
for i = 1:size(handles. boxFace,1)handles. detectedFace = handles. imx(handles.
boxFace(i,2): handles. boxFace(i,2)+ handles. boxFace(i,4), handles. boxFace(i,1) :handles.
boxFace(i,1) +handles. boxFace(i,3),:);
end
imshow(handles. detectedFace);
guidata(hObject, handles);
function blur(hObject, eventdata, handles)
handles=guidata(handles);
a=handles. detectedFace;
gaussSigma=0.01*length(a);
filter = fspecial('gaussian',[5 5], gaussSigma);
handles. ima = imfilter(a,filter);
imshow(handles. ima);
guidata(hObject, handles);
function averaging(hObject, eventdata, handles)
handles=guidata(handles);
```

```
a=handles. ima;
filt2 = fspecial('Disk',7);
handles.im1 = imfilter(a,filt2)
imshow(handles.im1);
guidata(hObject, handles);
function contrast(hObject, eventdata, handles)
handles=guidata(handles);
handles.im3=imadjust(handles.im1,stretchlim(handles.im1),[]);
imshow(handles.im3);
guidata(hObject, handles);
function threshold(hObject, eventdata, handles)
handles=guidata(handles);
im3=handles.im3;
handles. gray=rgb2gray(im3);
handles. thresh=getappdata(hObject. Parent,'slidervalue');
med = median(double(handles. gray(:)));
handles. imBW = im2bw(handles. gray, med / 255 * handles. thresh / 100);
imshow(handles. imBW);
guidata(hObject, handles);
function tbwareaopen(hObject, eventdata, handles)
handles=guidata(handles);
imBW=handles. imBW;
closingParameter=15;
imBW = ~imBW;
imBW = bwareaopen(imBW, closingParameter);
handles.imBW2 = bwareaopen(imBW,closingParameter);
handles.imBW2 = ~handles.imBW2;
imshow(handles.imBW2);
guidata(hObject, handles);
function tbwmorph(hObject, eventdata, handles)
handles=guidata(handles);
imBW3=handles.imBW2;
imBW3 = ~imBW3;
handles.imBW4 = bwmorph(imBW3,'majority');
handles.imBW4 = ~handles.imBW4;
imshow(handles.imBW4);
guidata(hObject, handles);
function though(hObject,eventdata,handles)
handles=guidata(handles);
circlein=handles.imBW4;
Rmin = 4;
Rmax = 20;
% [centersBright, radiiBright] = imfindcircles(imBW,[Rmin Rmax],'ObjectPolarity','bright');
[handles. centersDark, handles. radiiDark] = imfindcircles(circlein,[Rmin
Rmax],'ObjectPolarity','Dark');
if isempty(handles. centersDark)
error('no circle found');
else
% viscircles(centersBright, radiiBright,'EdgeColor','b');
viscircles(handles. centersDark, handles. radiiDark,'LineStyle','--');
end
guidata(hObject, handles);
function slidore(hObject, eventdata, handles)
handles = guidata(hObject)
```

```
handles. thresh = 100 - hObject. Value;
% im3=handles.im3;
% handles. gray=rgb2gray(im3);
% handles. thresh=getappdata(hObject. Parent,'slidervalue');
med = median(double(handles. gray(:)));
handles. imBW = im2bw(handles. gray, med / 255 *handles. thresh / 100);
imshow(handles. imBW);
disp(handles. thresh);
%  set(yazi, 'String', num2str(handles. thresh));
%  function yazi(varargin)
%  num = str2num(get(yazi,'String'));
%  if length(num) == 1 & num <=100 & num >=0
%  set(Slider,'Value',num);
%  else
% msgbox('The value should be a number in the range [0,100]','Error','error','modal');
% end
```



APPENDIX B
Databases

B.1 Camera Parameters

RadialDistortion	: [0,0322 -0,1293]
TangentialDistortion	: [0 0]
WorldPoints	: 42x2 double
WorldUnits	: ‘mm’
EstimateSkew	: 0
Num Radial Distortion Coefficients	: 2
EstimateTangentialDistortion	: 0
TranslationVectors	: 18x3 double
ReprojectionErrors	: 42x2x18 double
RotationVectors	: 18x3 double
NumPatterns	: 18
IntrinsicMatrix	: [886,8111 0 0;0889,38740;539, 0331 280,1611 1]
FocalLength	: [889,3874 886,8111]
PrincipalPoint	: [539,0331 280,1611]
Skew	: 0
MeanReprojectionError	: 0,3548
ReprojectedPoints	: 42x2x18 double
RotationMatrices	: 3x3x18 double

B.2 Image Files, Sound Files are all stored in the CD.