ALANYA ALAADDİN KEYKUBAT UNIVERSITY
RAFET KAYIŞ ENGINEERING FACULTY

UNDERGRADUATE THESIS

CURSOR CONTROL WITH EYE AND HEAD GESTURES

İbrahim Yiğit EGEMEN

ALANYA, ANTALYA

2023

THESIS APPROVAL

The thesis named "**Cursor Control With Eye and Head Gestures**" and prepared by **İbrahim Yiğit Egemen** was approved on **02/07/2023** by the juries below with the **majority vote** as an **UNDERGRADUATE THESIS** in the Alanya Alaaddin Keykubat University Rafet Kayış Engineering Faculty Computer Engineering department.

Advisor:

Jury Members:

Head:

Member:

Member:

Member:

Member:

I am approving the results above.

Head of Institute

ETHICS

I hereby approve that the thesis I prepared was written according to the rules of Alanya Alaaddin Keykubat University Rafet Kayış Engineering Faculty, and all the information in it is complete and correct, and it was acted according to the scientific ethics during the development process, and I have referenced every source that was used.

02/07/2023

İbrahim Yiğit EGEMEN

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Yılmaz Kemal YÜCE for their assistance during the development process in every possible way, especially troubleshooting, guiding, and advising in each part.

And I would like to thank Dr. Özge Öztimur KARADAĞ for their input, suggestions, and tests in parts that involve image processing.

And I would like to thank Şerif İnanır for their help in modeling/planning and motion sensors.

ABSTRACT

Undergraduate Thesis

Cursor Control With Eye and Head Gestures

İbrahim Yiğit EGEMEN

Alanya Alaaddin Keykubat University
The Faculty of Engineering
Department of Computer Engineering

Supervisor: Yılmaz Kemal YÜCE

In this paper, we have used a camera to capture eye blinks, and a motion sensor to capture head movements to control the cursor in real-time. This allows people who can not use the current standard computer control interfaces (eg. people with physical disabilities) to control their computers. The result supports single left-click, double left-click, right-click, and the movement of the cursor to have all necessary controls over the cursor in the computer.

2023, 18 pages
Keywords: Eye, Cursor, Blinks, Face, Gestures, Image Processing

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

In the ergonomics and designs of a computer, there is a keyboard and a mouse interface to control it, which is the standard. However, not everyone has the physical ability to use those interfaces. People with;

- Tetra-amelia Syndrome [1],
- Amputations [2],
- Birth defects that cause physical disability [3],
- Permanent paralysis [4] do not have the ability to use a keyboard or a mouse.

Thus, this paper aims to provide an interface to control the cursor in the operating system, in real-time, that does not require any hand ability or movement below the head.

Additional planned features to have:

- Visual interface to show what is captured
- Support the most important mouse functions
    - Left-click
    - Right-click
    - Double left-click
- Precise cursor movement
- Consistent performance among different face/eye types
- Consistent performance in different environments
- Low processing time
- Easy to setup
- Low cost

Considering these goals, with a facing camera attached to the monitor, we can capture the eye blinks using methods that involve machine learning algorithms. This way, we can adapt to different types of environments and human ethnicities.

Additionally, we can attach a small wireless motion-detecting sensor to the user's head, which can help us capture the head movements precisely.

2. LITERATURE RESEARCH

In the paper "An Arduino based Gesture Control System for Human-Computer Interface" [5] a camera, an accelerometer device, several Arduino microcontrollers, and an Ultrasonic Distance Sensor are used to capture body activity. However, this involves the usage of hands and body movement.

In the paper "Head mouse control system for people with disabilities" [6] a customer model using Convolutional Neural Network (CNN) is trained, which finds the face and eyes at the same time. And both the head movements and eye blinks are captured from the camera to control the mouse. However, none of the required processing power or the processing times are mentioned.

In the paper "Eye-Mouse for Disabled" [7] a camera on the eyeglasses is used to capture the screen (which has IR markers on the corners) and one camera is used to capture the eye pupil to control the mouse.

3. DISCUSSIONS AND RESULTS

3.1 Cursor Movement

Our cursor movement module consists of two parts: A motion sensor to get the data, and the operating system APIs to move the cursor.

3.1.1 Motion Sensor

To capture the movement/position of the head, we have concluded to use MetaMotionRL from MbientLab. It has many sensors but the main sensor that we used is the Accelerometer.

An accelerometer can gauge the orientation of a stationary item in relation to Earth's surface, thus it is viable for our purpose.

The best location for the motion sensor to be in the face is the forehead, which is directly affected by any head movement that is useful to move the cursor.

Thus, with the accelerometer, we successfully captured the angle of the head. After processing the incoming data, we used the first values of the accelerometer as our starting point at the start of our application. To put it simply, we calibrate the sensor at launch.

Then, we have 2 variables to adjust the movement: sensor deadzone and sensitivity. The deadzone is a zone at the center which is ignored. This is required as it is near impossible for the user to center their head precisely each time they want to stop the movement of the cursor. The formula 3.1 decides the new value using the formula *f*, where x is the coordinates of the motion sensor on either X-axis or Y-axis, and t is the deadzone:

$$f(x) = \begin{cases} 0, & -t < x < t \\ x, & otherwise \end{cases} \tag{3.1}$$

Then, with the new decided value, we can decide on how many pixels to move in either direction with the formula *h*, where *y* is the processed value of any coordinate, and *s* is the pre-determined sensitivity variable:

$$h(y) = \frac{y}{1000 \,/\, s} \tag{3.2}$$

The accelerometer sensor on our MetaMotionRL has an accuracy of 99% according to the BMI160 data sheet by Bosch [8].

3.1.2 Operating System APIs

After we gather and process the incoming data from the motion sensor, we need to turn that into actual action by using the operating systems' APIs. The API is called *win32api* on Windows and *Xlib* on Linux. We have implemented both APIs in our project. This way our application can work on both Windows and Linux, simply cross-platform.

3.2 Capturing of Blinks

For the capturing of blinks, there are numerous machine learning algorithms to choose from. In this paper, we have tried Haar Cascade's frontal face and eye detector, and Dlib's HOG+ Linear SVM face detector.

3.2.1 First Approach: Haar Cascade

We used Haar Cascade as our initial approach. Haar is a machine learning approach that was introduced in the paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001 by Paul Viola and Michael Jones. Its goal is to detect objects in a given image with rapid processing times.

And Haar Cascade is a classifier that consists of numerous Haar-like features arranged and combined, which is used to find objects in an image. There are Haar Cascade classifiers to find the face and eye objects in a given image.

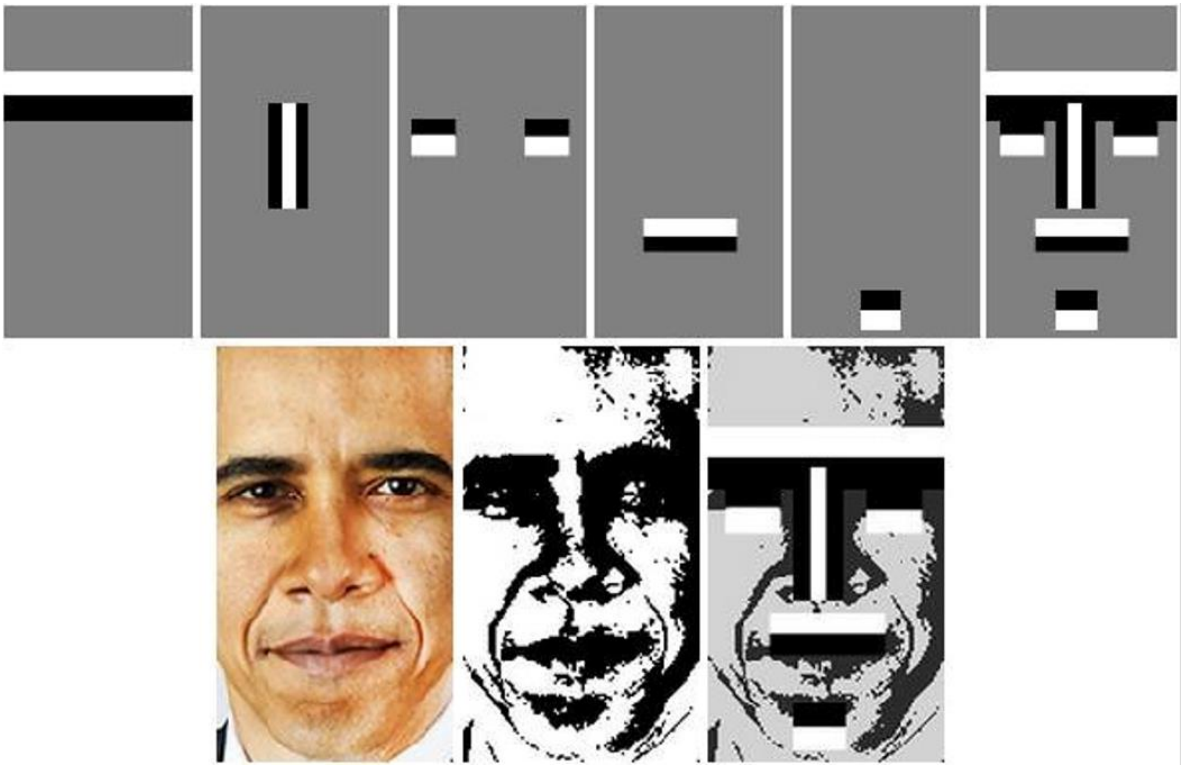

Figure 3.1 Haar Cascade Face Features Example

Thus, with this, we can first detect the face from the whole frame. Then, when we find the face, we can narrow down our search area to the face area and start looking for the eyes.

Before doing all of this, we make the picture black and white and apply Bilateral filter to remove impurities to improve the accuracy of the face and eye detection.

However, Haar Cascade can not detect an eye if it's closed. It only provides the eyes if they are open. This leads us to have the following algorithm to detect blinks:

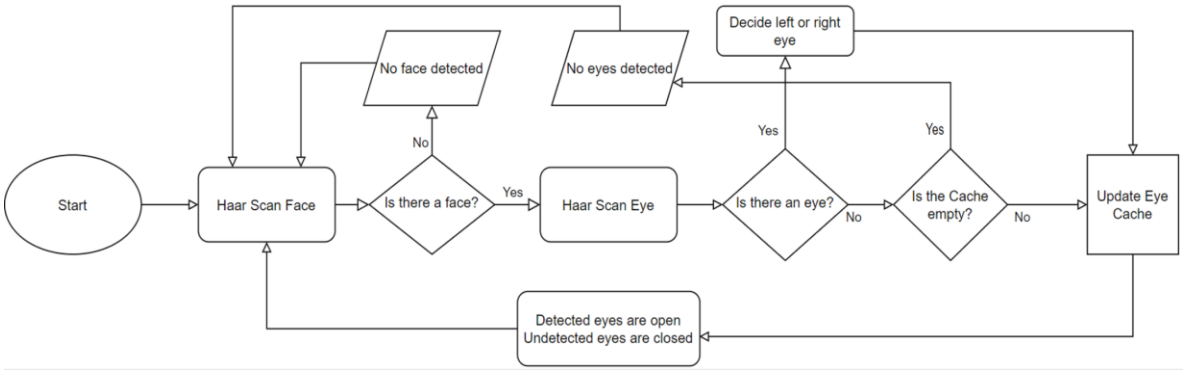

Figure 3.2 Haar Cascade Blink Detection Algorithm

This produces a somehow reliable detection of eye blinks, but most importantly this allows us to detect the closeness of each eye individually. With this, the gestures to left click and right click can be dependent on which eye you are blinking with. This algorithm produced the graph in Figure 3.3 and the table in Table 3.1.

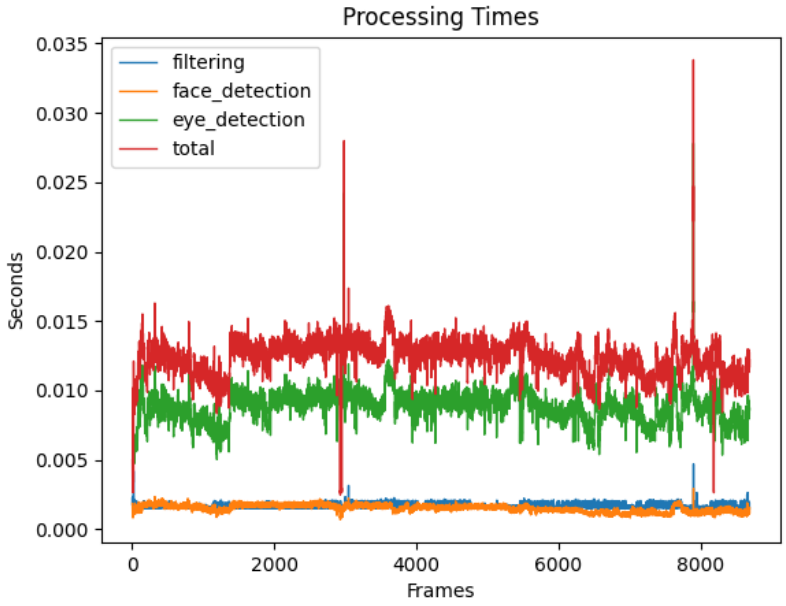

Figure 3.3 Haar Cascade Processing Times Graph

	Filtering	Face Detection	Eye Detection	Deciding Eye Type and Status	Total
Avg. Processing Time (milliseconds)	1.6	1.4	8.8	0.5	12.3

Table 3.1 Haar Cascade Processing Times Table

These results were captured by running the application for 5 minutes on Intel i7-9750H 4.00 GHz CPU with the camera's resolution set to 640x480. According to these results, this algorithm can support over 80 frames per second.

However, even though this is incredibly fast, its accuracy is low and not reliable enough to use in our project.

3.2.2 Second Approach: Dlib

Dlib is a C++ toolkit that contains machine learning algorithms and tools, and one of them is "Frontal Face Detector". The frontal face detector uses Histogram of Oriented Gradients (HOG) and Linear SVM classifier that is accurate and fast at the same time.

When the face is detected, Dlib's Frontal Face detector/predictor creates 68 facial landmark points all over the face, which allows us to easily know where any object in the face is located.

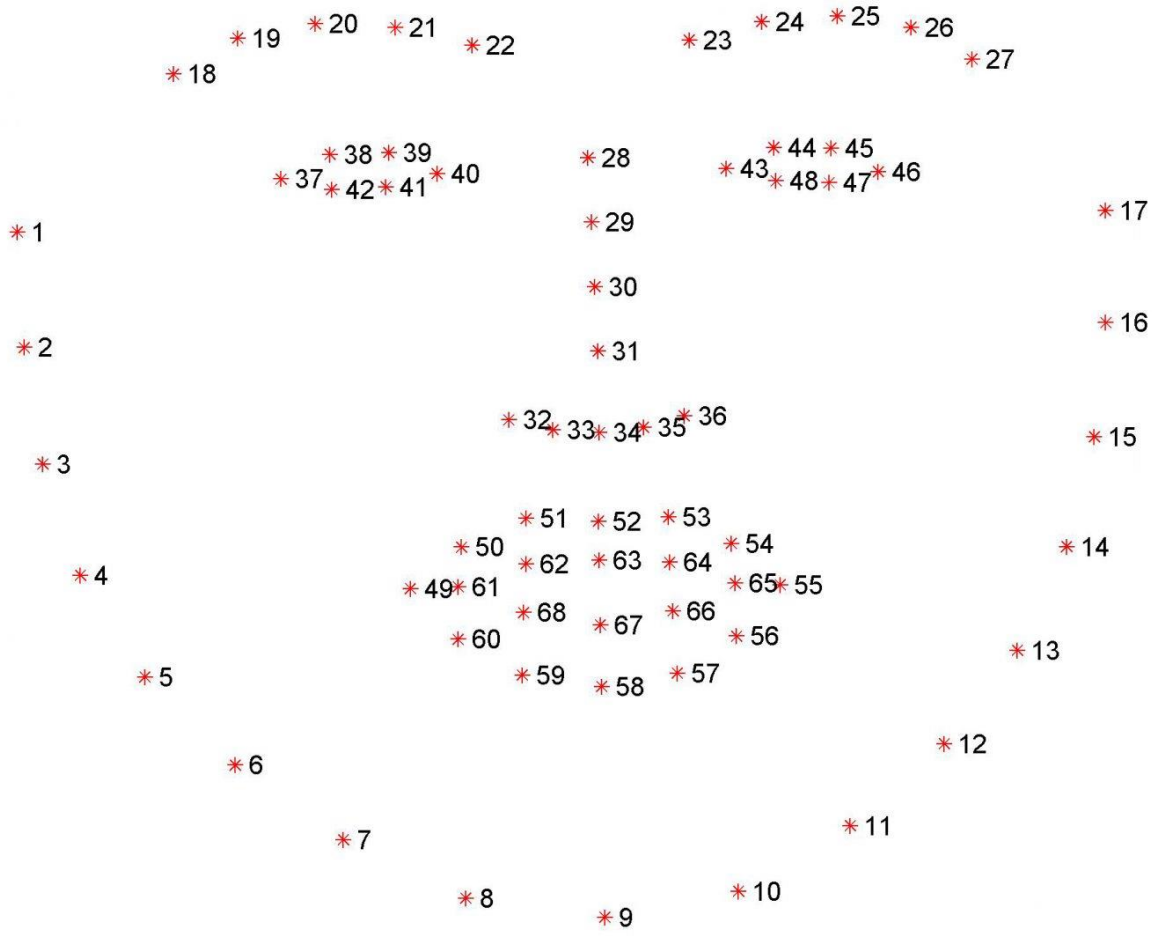

Figure 3.4 Dlib Frontal Face Detector Facial Landmarks

In Haar Cascade, when the eye is closed, we were not acquiring any eye at all. However, with Dlib's Frontal face predictor, we still get the eye and whether the eye is closed or not has to be computed. To achieve this, we can use the eye aspect ratio.

The ones we need among all those landmarks are 37, 38, 39, 40, 41, 42 for the right eye, and 43, 44, 45, 46, 47, 58 for the left eye. 37, 40, 43, and 46 are edge points, however, the rest of the landmarks are not centered. So, we need to calculate and consider their midpoints using

the Eucledian Distance formula. The result will give us the horizontal and vertical borders, which can be connected to acquire a rectangle.

Figure 3.5 Eye Landmarks Rectangle Creation

Then, if we calculate the Euclidean distance between opposing edges, and compare them, we can achieve a ratio that can represent the closeness of the eye, no matter how small/big or far the eye is.

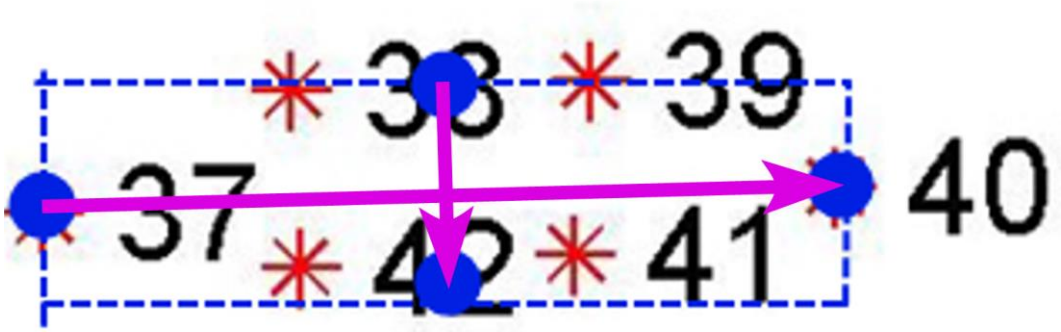

Figure 3.6 Eye Landmarks Aspect Ratio

If we divide the horizontal distance with the vertical distance, the more the eye is closed, the bigger number we have.

For our purpose, we would have liked to calculate the eye aspect ratio of each individual eye to support left-click and right-click gestures easily. However, Dlib's Frontal Face predictor does not work well when only one of the eyes is closed. Thus, this pushes us to use a formula to calculate the blinks of both eyes combined and create the gestures depending on this.

$$f(X,Y)$$

$$= \frac{\sqrt{\frac{(X_{0_x}-X_{3_x})^2+(X_{0_y}-X_{3_y})^2}{\frac{((X_{1_x}+X_{2_x})-(X_{4_x}+X_{5_x}))^2+((X_{1_y}+X_{2_y})-(X_{4_y}+X_{5_y}))^2}{2}}} + \sqrt{\frac{(Y_{0_x}-Y_{3_x})^2+(Y_{0_y}-Y_{3_y})^2}{\frac{((Y_{1_x}+Y_{2_x})-(Y_{4_x}+Y_{5_x}))^2+((Y_{1_y}+Y_{2_y})-(Y_{4_y}+Y_{5_y}))^2}{2}}}}{2} \quad (3.3)$$

In the formula 3.3, X is the left eye, Y is the right eye, superscripts from 0 to 5 indicate the landmark point, where 0 is the left edge, 1 and 2 are the top landmarks, 3 is the right edge, and 4 and 5 are the bottom landmarks. Additionally, x and y indicate the axis coordinate of each landmark. The result of this formula gives us the eye aspect ratio of both eyes combined.

Our flowchart with this algorithm is present in Figure 3.6.

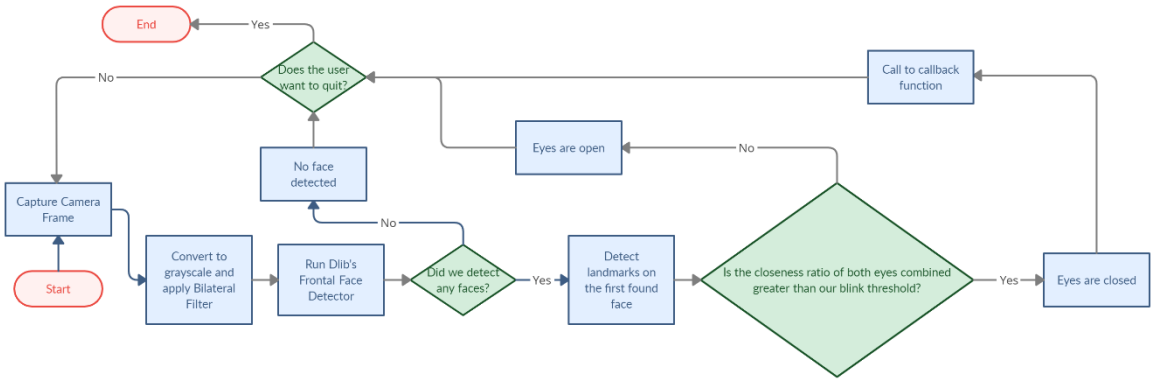

Figure 3.7 Frontal Face Detector Flowchart

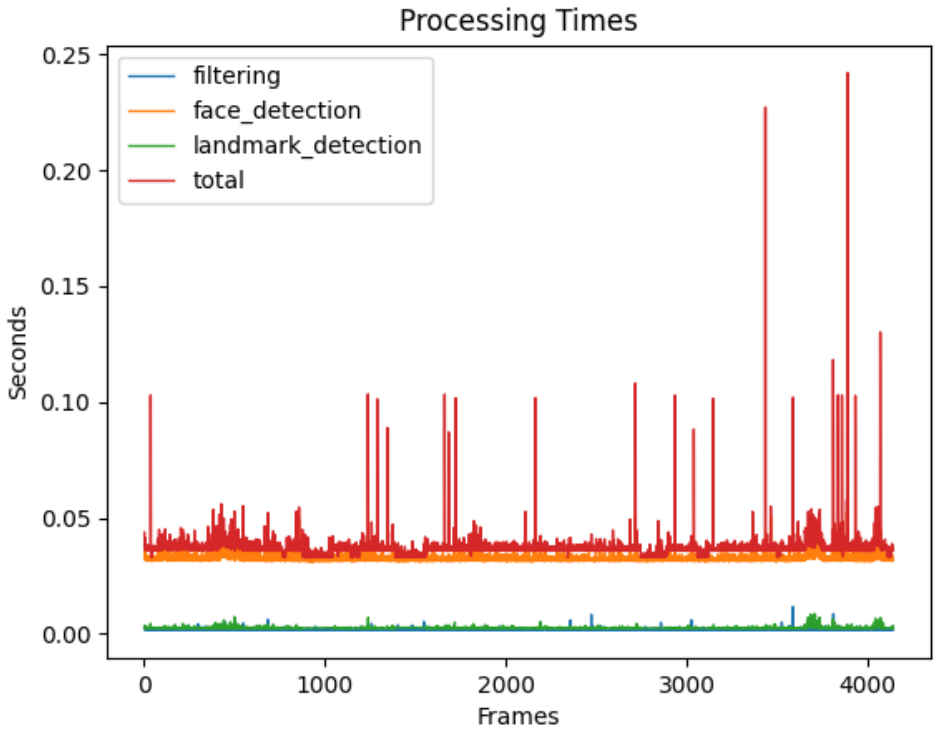

Figure 3.8 Frontal Face Detector Processing Times Graph

	Filtering	Face Detection	Landmark Placement	Deciding Eye Status and Other Processes	Total
Avg. Processing Time (milliseconds)	**1.7**	**32.0**	**2.0**	**1.3**	**37.0**

Table 3.2 Frontal Face Detector Average Processing Times

These results were captured with the same specifications in Figure 3.2. According to these results, this algorithm can support around 27 frames per second.

Even though this is slower compared to Haar Cascade, it is still fast enough, more accurate, and reliable. Thus, we will continue with Dlib's Frontal face predictor.

3.3 Decision Making

After we receive the blink, or rather, the callback that the eye is closed, then we can decide what to do. There are 3 algorithms that we can follow/use:

1. State-Driven Individual Blink Algorithm
2. State-Driven Double Blink Algorithm
3. Event-Driven Double Blink Algorithm

In the first algorithm, we use the status of each individual. However, since Dlib's Frontal Face Detector does not work well with individual eye status when only one of them is closed, we will not be using this.

In the second one, we use the state of both eyes combined and take action immediately. However, This will limit the number of actions to one, since we can only take one action with each event if we don't look for additional information.

In the third one, we get the state of the eye and process it. If the eye is closed, we measure how long it is closed.

According to the scientific article "High-speed camera characterization of voluntary eye blinking kinematics" [9] , a voluntary blink lasts about 134 milliseconds. And according to

the article “Asymmetry of Blinking” [10] , the change in blink periods averages only ± 2.23 ms. For our project, we have decided to use the blink threshold as 135 milliseconds.

Then, we constantly compare the latest state of the eye to the last state that we have received. If the state has changed (eg. if the eye was closed, but now it is open), then we measure how long it has lasted and depending on that if it exceeded our blink threshold, we add one blink to our cache and start a timer of 1 second.

During that one second, if we receive any more blinks, the action that will be made with the cursor changes, and the timer to take action get extended 1 more second. Here’s a list of gestures with this algorithm:

- One blink -> Left-click
- Two blinks -> Double left-click
- Three or more blinks -> Right-click

	False Positive	False Negative	True Positive	True Negative
Frame Count	37	41	363	559
Total Frame Count	400	600	400	600
Percentage	9,25%	6,84%	90,75%	93,16%

Table 3.3 Contingency Table of Blinking in Our Application

Looking at the contingency table, we have a precision value of 90,75% and an accuracy value of 92,20% while detecting the blinks, which is good and precise enough to be used in our project.

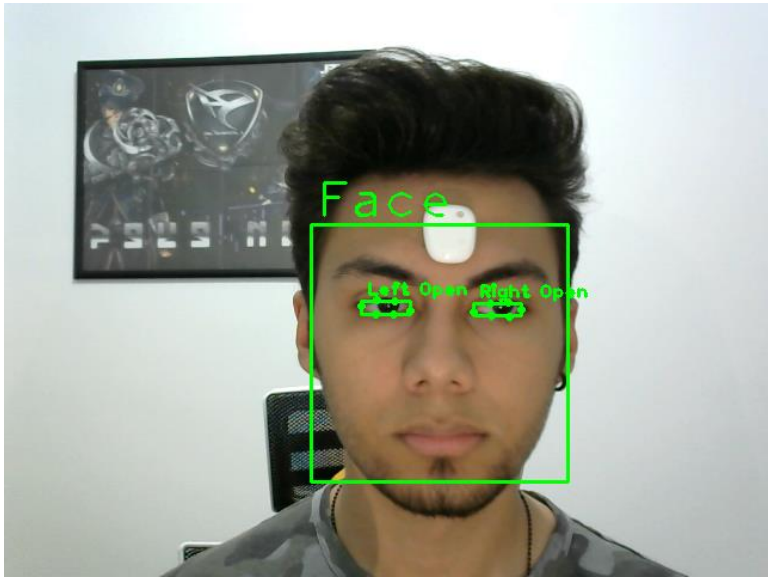

Figure 3.9 Graphical Interface which shows the current status

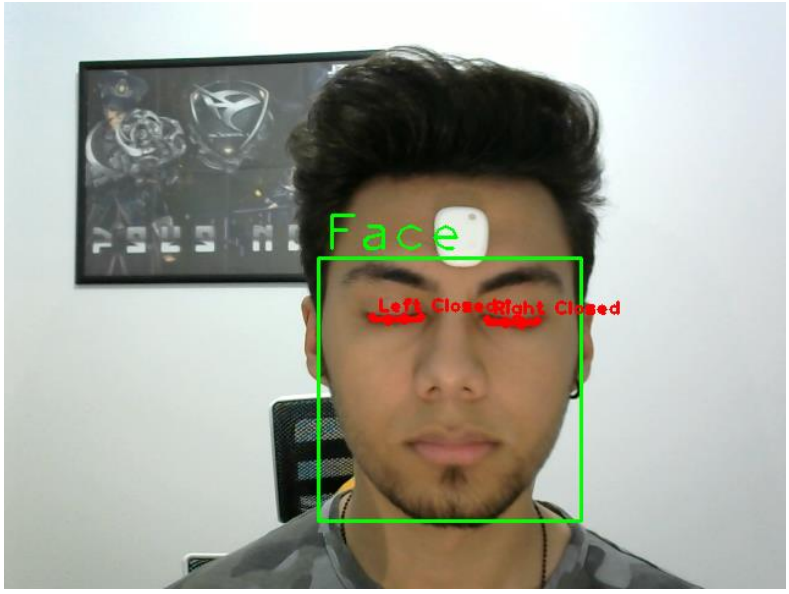

Figure 3.10 Graphical Interface while eyes are closed

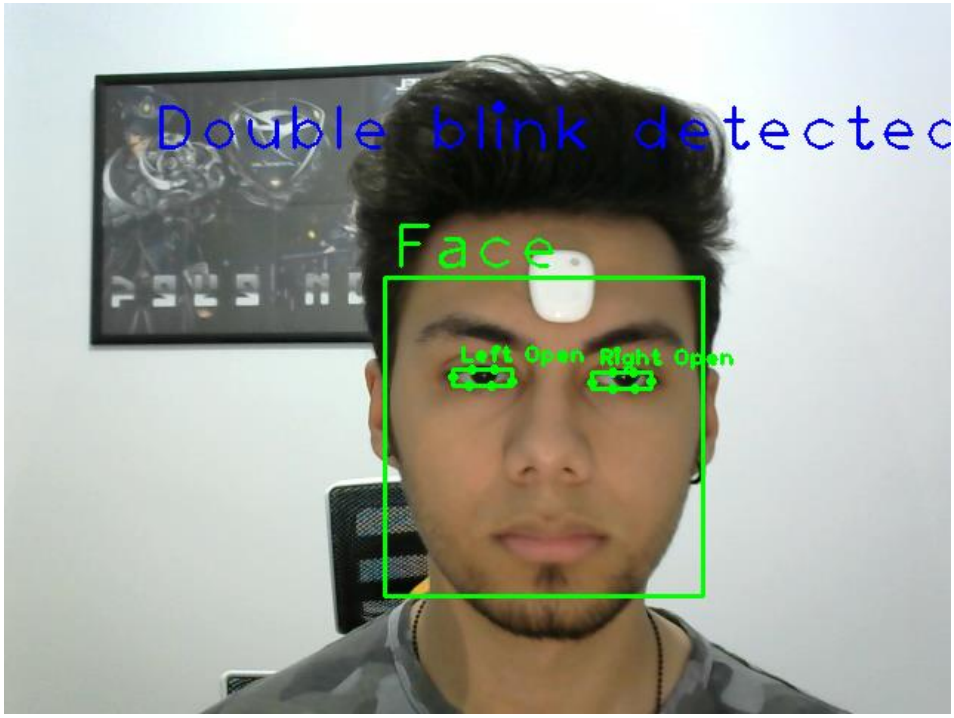

Figure 3.11 Graphical Interface when a double blink is detected (double left-click)

4. CONCLUSIONS

The control of the computers requires at least one interface, but not all of those interfaces are usable by everyone. With this project, by combining the detection of blinks and capturing the head movements, we have seen that it is possible for people with disabilities to control their computers by alternative means of interfaces, in real-time.

REFERENCES

[1] M. J. P. R. Dutta, «Tetra Amelia Syndrome - A Case Report,» 2012.

[2] J. Dormandy, L. Heeck ve S. Vig, «Major amputations: clinical patterns and predictors.,» 1999.

[3] Carmona ve H. Richard, «The global challenges of birth defects and disabilities.,» 2005.

[4] W. J. Meggs, «Permanent Paralysis at Sites of Dermal Exposure to Chlorpyrifos,» pp. 883-886, 2003.

[5] S. Belgamwar ve S. Agrawal, «An Arduino Based Gesture Control System for Human-Computer Interface,» pp. 1-3, 2018.

[6] M. Arslan ve R. Abiyev, «Head mouse control system for people with disabilities,» *Expert Systems*, cilt 37, no. 1, pp. 1-14, 2020.

[7] T. Kocejko, A. Bujnowski ve J. Wtorek, «Eye-Mouse for Disabled,» %1 içinde *Advances in Intelligent and Soft Computing*, 2009.

[8] B. Sensortec, «BMI160 DataSheet,» Bosch, 2020.

[9] Kwon ve Kyung-Ah, «High-speed camera characterization of voluntary eye blinking kinematics.,» *Journal of the Royal Society Interface*, 2013.

[10] Kassem, S. Iris ve C. Evinger, «Asymmetry of blinking.,» *Investigative ophthalmology & visual science*, cilt 47, no. 1, pp. 195-201, 2006.

RESUME

Name Surname : İbrahim Yiğit Egemen

Place of Birth : Antalya, Turkey

Date of Birth : 25/02/2000

Marital Status : Single

Languages : Native Turkish, C2 English

Education Status (Institute and Graduation Year)

Highschool : Aksu Science High School (2018)

Undergraduate : Alanya Alaaddin Keykubat University Rafet Kayış Engineering Faculty Computer Engineering Department (2022)