Report On

EYE-CURSOR
A new vision for disabled people

Submitted in partial fulfillment of the requirements of the Mini project in Semester VI of Third Year Computer Engineering

by
Rutuja Parab (Roll No. 19)
Shruti Sankhe (Roll No. 28)
Nehab Shaikh (Roll No. 32)

Supervisor
Prof. Vikrant A Agaskar Sir

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2019-20)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled **Eye Cursor (a vision to disabled people)** is a bonafide work of Rutuja Parab (Roll No. 19), Shruti Sankhe (Roll No. 28), Nehab Shaikh (Roll No. 32) submitted to the University of Mumbai in partial fulfillment of the requirement for the Mini project in semester VI of Third Year Computer Engineering.

Supervisor

Prof. Vikrant A Agaskar

Internal Examiner

External Examiner

Dr Megha Trivedi
Head of Department

Dr. H.V. Vankudre
Principal

ABSTRACT

The field of Human-Computer Interaction (HCI) has witnessed a tremendous growth in the past decade. The advent tablet PCs and cellphones allowing touch-based control has been hailed warmly. The researchers in this field have also explored the potential of eye-gaze as a possible mean of interaction. Some commercial solutions have already been launched, but they are as yet expensive and offer limited usability. This project strives to develop a low cost real time system for eye-gazed based human-computer interaction.

A high number of people, affected with neuro-locomotor disabilities or those paralyzed by injury cannot use computers for basic tasks such as sending or receiving messages, browsing the internet, watch their favorite TV show or movies. Through a previous research study, it was concluded that eyes are an excellent candidate for ubiquitous computing since they move anyway during interaction with computing machinery. Using this underlying information from eye movements could allow bringing the use of computers back to such patients. For this purpose, we propose an imouse gesture control system which is completely operated by human eyes only. The purpose of this work is to design an open-source generic eye-gesture control system that can effectively track eye-movements and enable the user to perform actions mapped to specific eye movements/gestures by using computer webcam. It detects the pupil from the user's face and then tracks its movements. It needs to be accurate in real-time so that the user is able to use it like other every-day devices with comfort.

The idea of eye controls of great use to not only the future of natural input but more importantly the handicapped and disabled. Moreover, implementing a controlling system in it enables them to operate computer without the help of another person.

INDEX

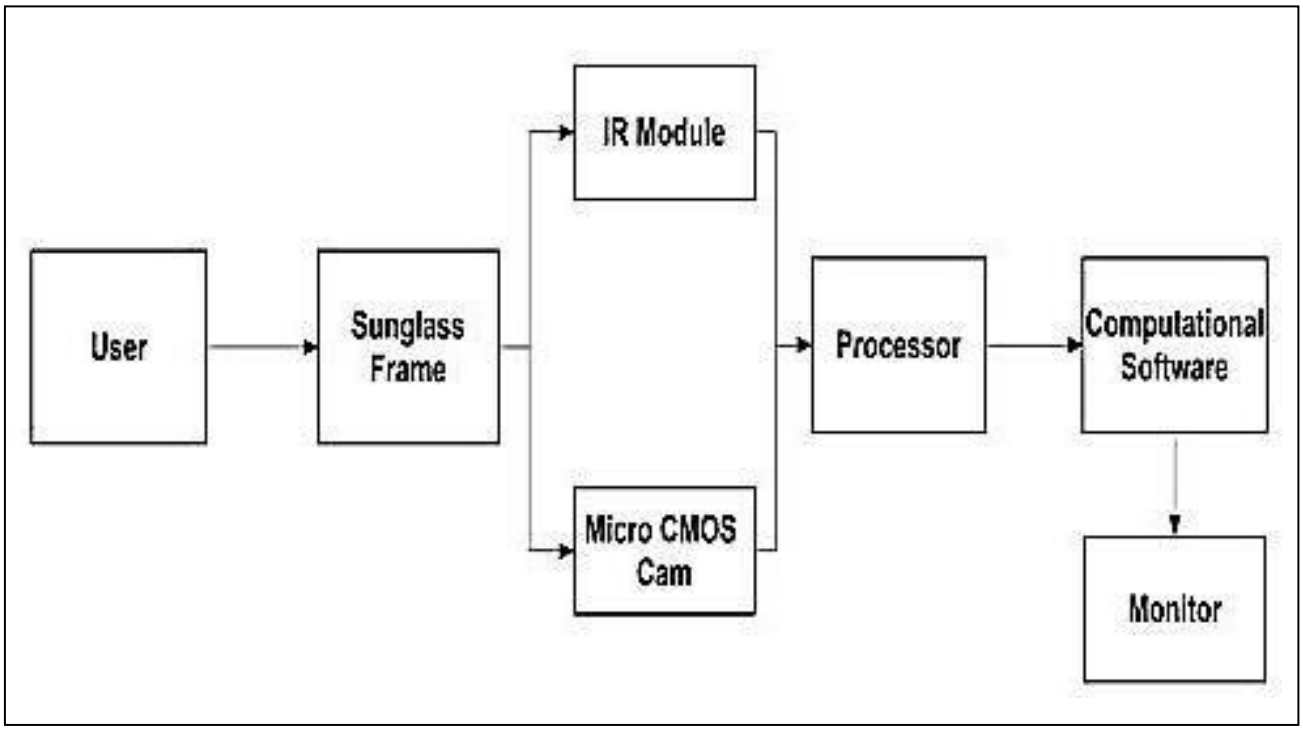
INTRODUCTION

- **PROBLEM STATEMENT:**

Some peoples cannot able to operate computers because of an illness. The idea of eye controls of great use to not only the future of natural input but more importantly the handicapped and disabled. Moreover, implementing a controlling system in it enables them to operate computer without the help of another person. It is more helpful to handicapped peoples. Those are need to operate computers without hand this one is most useful those can operate cursor by movement of eye.

An eye has a lot of communicative power. Eye contact and gaze direction are central and very important factors in human communication. The eye has also been said to be a mirror to the soul or window into the brain. Gaze behavior reflects cognitive processes that can give hints of our thinking and intentions. We often look at things before acting on them. To see an object we have to fixate our gaze at it long enough for the brains visual system to perceive it. Fixations are typically between 200 and 600 ms. During any fixation, we only see a fairly narrow area visual scene with high acuity. To perceive the scene accurately, we need to scan constantly it with rapid movement of an eye, so-called saccades. Saccades are quick, ballistic jumps of 2 Degrees or longer that take about 30 to 120 ms each. In addition to saccadic movements, the eyes can follow a moving target; this is known as (smooth) pursuit movement. The size of the fovea, subtends at an angle of about one degree from the eye. The diameter of this region corresponds to area of two degrees, which is about the size of a thumbnail when viewed with the arm extended. Everything inside the fovea is perceived with high acuity but the acuity decreases rapidly towards the periphery of the eye. The cause for this can be seen by examining the retina. The lens focuses the reflected light coming from the pupil on the center of the retina. The fovea contains cones, photoreceptive cells that are sensitive to color and provide acuity. In reverse case, the peripheral area contains mostly rods, i.e. cells that are sensitive to light, shade and motion.

- **BLOCK DIAGRAM:**



- **DESCRIPTION:**

The block diagram mainly consists of six parts:

1) User.

2) Sunglass frame.

3) IR Module.

4) Micro CMOS cam.

5) Processor.

6) Computational Software.

7) Monitor.

1) User: The user will be one who will provide the input to the system through his/her eyes.

2) Sunglass frame: Cheap and strong enough to sustain the weight of camera.

3) IR Module: Four IR LEDs mounted below the camera which will propagate the IR light on human eye which will help in tracking the eye movements more precisely.

4) Micro CMOS cam: IR sensitive and minimum resolution of 640 x 480 which will be mounted on the eyeglass frame.

5) Processor: Processor will process the images/pictures captured by camera.

6) Computational Software: The output from processor is then computed in Open CV.

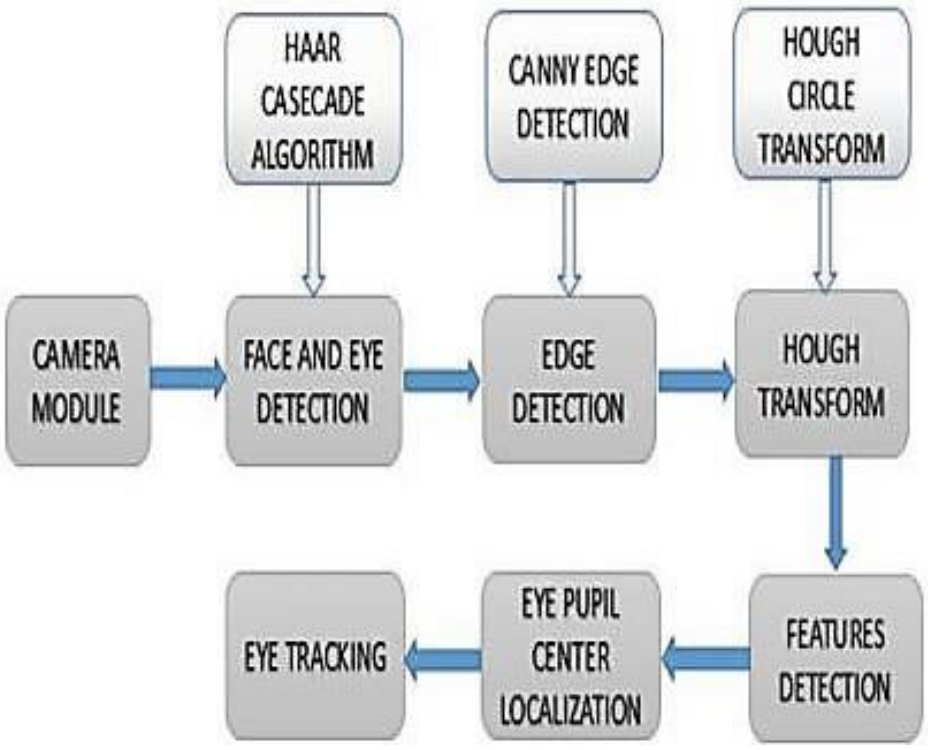
7) Monitor: It will be used to display the output generated from Computational software.

- **WORKING:**

The user has to sits in front of the display screen of private computer or pc, a specialized video camera established above the screen to study the consumer's eyes. The laptop constantly analysis the video photo of the attention and determines wherein the consumer is calling at the display screen. not anything is attached to the consumer's head or body. To "pick out" any key, the user seems at the key for an exact period of time and to "press" any key, the consumer just blink the eye. On this device, calibration procedure is not required. For this system enter is simplest eye. No outside hardware is connected or required.

Camera gets the input from the eye. After receiving these streaming movies from the cameras, it spoils into frames. After receiving frames, it will check for lights conditions because cameras require enough lighting fixtures from external sources in any other case blunders message will show at the screen. The captured frames which can be already in RGB mode are transformed into Black and White. Pics (frames) from the enter supply focusing the eye are analyzed for Iris detection (middle of eye).After this, amid point is calculated through taking the suggestion of left and right eye center point. Eventually the mouse will pass from one position to any other at the display and consumer will perform clicking with the aid of blinking their eyes for 5 seconds.

- **MODULE DESCRIPTION:**



The principle of this system is eye pupil detection and eye tracking based on computer vision technology. A new algorithm introduced for detecting the eye pupil location by Image processing. In this technique several stages used to find out the movement of eye, such as Face detection and Eye detection, color conversion, Edge detection, Hough Transformed, motion detection and object tracking. During initial stage the system acquired the captured Images by USB Web camera. The first direction is to detect the user Face accurately. If there is multiple faces are presented it will display the individuals and also showing the run time error. A system indicates and represents the face of user in a specific area of image. After that system will performed the several operations of image processing to track the Eye pupil. The figure 3 represents the complete methodology of proposed implementation. Here it will give the step by step information of the system working.

First camera module will start to capture the images. For the face detection Haar cascade algorithm is used. After detection of proper face, it will be trying to detect the eye inside the face region of interest. And again, Haar cascade algorithm is used like as face detection to detect eye. It will draw the rectangular box over the Eye. Now, the main target is to detects the eye pupil and define its center points. There is several image processing operations performed in system, such as blur Image, color conversion, thresholding, filtering, edge detection and Hough transform is used. For circle detection Hough transform method is used. By using the USB webcam allowed to capture the images on raspberry. There it will process and working without any processing delay. The system will crop the eye region of interest initially and it will detect all possible circle presented on that particular area. Than it will successfully detects the eye ball. After that corner detection method, we applied for eyes region of interest, and find out the corners. Where average of both two point defined its Center point. Now we measure distance between the Center point and eye circle Center point using coordinates system logic. According to the eye pupil movements, distance will be varied. A minimum distance indicates the eye pupil presented in left, and maximum values indicates the eye moved on right. And if there is no movements of the eye, than it concludes eye is in the middle position. Than the commands applied for all operation, when eye movement is left, the cursor moves left side will run. And when the eye moved is right the cursor on the screen is move right. If eye is blinked once, the cursor selects the content displayed on the screen. A system started with capturing images continuously by camera.

SOFTWARE AND HARDWARE USED

- **SOFTWARE USED:**

1. CAMERA.

Camera ideal for many imaging applications. Fist image will be capture by Camera. Focus on eye in image and detect the Centre position of pupil by open CV code. Take the center position value of pupil as reference, and then the next the different value of X, Y coordinates will be set for particular command.

2. OPERATING SYSTEM.

Windows is a series of operating systems developed by Microsoft. Each version of Windows includes a graphical user interface, with a desktop that allows users to view files and folders in windows. For the past two decades, Windows has been the most widely used operating system for personal computers PCs.

- **HARDWARE USED:**

1. OPENCV.

Open Source Computer Vision Library is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flashed etc.

2. PYTHON 3.6.

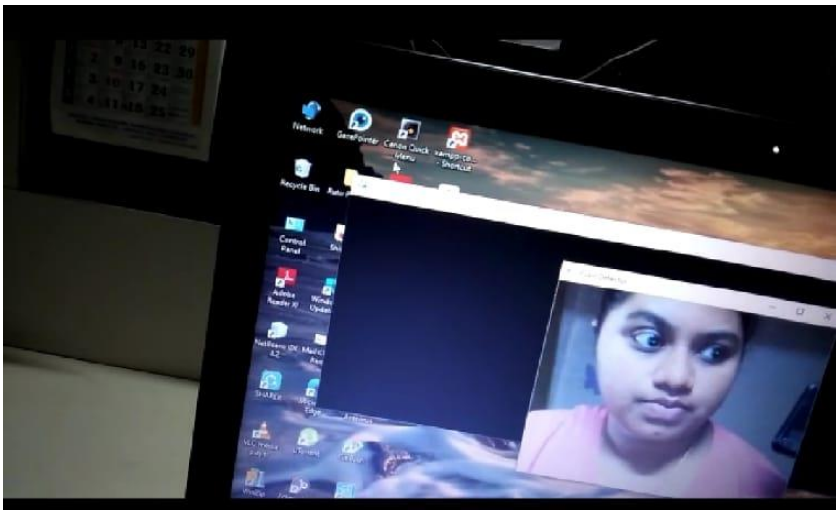
Python is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

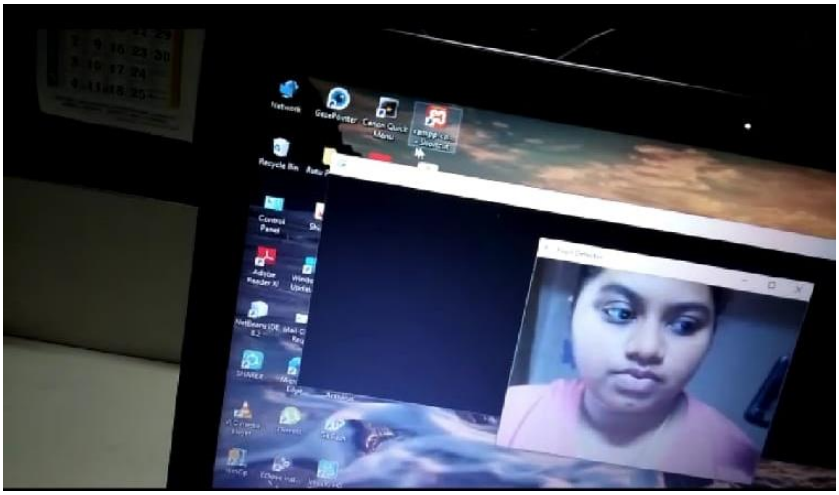
CODE

```
import pyautogui

import math

import cv2

scaling_factor = 0.7

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

cap = cv2.VideoCapture(0)

while True:

    ret, img = cap.read()

    img = cv2.resize(img, None, fx=scaling_factor, fy=scaling_factor,
interpolation=cv2.INTER_AREA)

    gray = cv2.cvtColor(~img, cv2.COLOR_BGR2GRAY)

    ret, thresh_gray = cv2.threshold(gray, 220, 255, cv2.THRESH_BINARY)

    contours, hierarchy = cv2.findContours(thresh_gray, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

    for contour in contours:

        area = cv2.contourArea(contour)

        rect = cv2.boundingRect(contour)

        x, y, width, height = rect

        radius = 0.25 * (width + height)

        print(radius)

        area_condition = (100 <= area <= 200)

        symmetry_condition = (abs(1 - float(width)/float(height)) <= 0.2)

        fill_condition = (abs(1 - (area / (math.pi * math.pow(radius, 2.0)))) <= 0.3)

        if area_condition and symmetry_condition and fill_condition:
```

```
            cv2.circle(img, (int(x + radius), int(y + radius)), int(1.3*radius), (0,180,0), -1)
            cv2.circle(img, (int(x + radius), int(y + radius)), 2, (0,180,0), -1)
            #print(pyautogui.size())
            pyautogui.moveTo(int(x + radius), int(y + radius), duration = 1)



    cv2.imshow('Pupil Detector', img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

RESULT





CONCLUSION

Traditional research has investigated the differences between debugging experts and novices in an effort to understand how to develop better pedagogical methodologies and tools. However, such research still requires solid and objective evidence to establish the authenticity of its findings.

In this project, the experimental results provide objective eye-tracking evidence that confirms the hypotheses made based on the findings of existing research: Most students recognize beacons and pay more attention to these areas when debugging. Low-performance students tend to debug programs aimlessly, whereas high-performance students debug programs in a more logical manner. The major differences between high- and low-performance students in terms of their cognitive processes lie in the former's ability to plan during the debugging process. High-performance students tend to organize the code into chunks, and their comprehension and debugging strategies are based on prior knowledge coupled with their ability to identify problems. Low-performance students, on the other hand, remain fixed on the syntactic details, ultimately failing to build the appropriate mental models for debugging. These conclusions were drawn after examining the differences between high- and low-performance students by applying statistical tests: the -test for spatial data, and sequential analysis for temporal data. Only significant statistical results have been reported in the conclusions, guaranteeing the conclusion validity. Previous research has revealed a relationship between working memory capacity and the cognitive activities related to debugging with regard to mental arithmetic, short-term memory, logical thinking, and problem solving. This study confirms this correlation. However, further study is needed to better understand the relationship, and such studies require firmer evidence. By exploring the differences between high- and low performance students with regard to their visual attention and gaze paths during debugging, a strong correlation was found between debugging skills and the associated cognitive activities, demonstrating the importance of the instructor's assistance in developing these cognitive abilities while teaching debugging skills. Adaptive instructional strategies and media can be designed based on these research findings. For example, the learning of scaffolding tools might be designed to help students plan debugging tasks, recall prior information, split the code into meaningful chunks, process computations, and trace the program in a logical manner.

REFERENCE

- **FOR BOOK/ARTICLE:**

[1] D. N. Perkins, C. Hancock, R. Hobbs, F. Martin, and R. Simmons, “Conditions of learning in novice programmers,” J. Educ. Comput. Res., vol. 2, pp. 37–55, 1986

[2] J. A. Villalobos and N. A. Calderón, “Developing programming skills by using interactive learning objects,” in Proc. ITiCSE, 2009, pp. 151–155

[3] E. Verdú et al., “A distributed system for learning programming on-line,” Computer. Educ., vol. 58, no. 1, pp. 1–10, 2012

[4] S. Fitzgerald et al., “Debugging: Finding, fixing and flailing, a multi institutional study of novice debuggers,” Comput. Sci. Educ., vol. 18, pp. 93–116, 200

[5] T. Putnam, D. Sleeman, J. A. Baxter, and L. Kuspa, “A summary of misconceptions of high school basic programmers,” J. Educ. Comput. Res., vol. 2, pp. 459– 472, 198

[6] S. Xu and V. Rajlich, “Cognitive process during program debugging,” in Proc. 3rd IEEE ICCI, 2004, pp. 176–182

- **FOR WEB REFERENCE:**

1. http://www.arringtonresearch.com/scene.html

2. https://opencv.org/about/

3. http://www.xavigimenez.net/blog/2010/02/face-detection-how-to-find-faces-with-opencv/

4. https://ieeexplore.ieee.org/document/8376907

5. https://www.researchgate.net/publication/325919286_Eye-controlled_mouse_cursor_for_physically_disabled_individual