International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 4, July - Aug 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

# Cursor Control Using Eye Movement Using Open CV

Dupati Praneetha*, Uppuganti Anjali Nanda**, P. Nancy Celine***
*(Computer Science Of Engineering, Vasavi College Of Engineering, Hyderabad, Telangana, India
Email: dupatipraneetha@gmail.com)
**(Computer Science Of Engineering, Vasavi College Of Engineering, Hyderabad
Email: anjalinanda2004@gmail.com)
***(Computer Science Of Engineering, Vasavi College Of Engineering, Hyderabad
Email: Celine@staff.vce.ac.in)

----------**********----------

## Abstract:
This work aims to show the process of controlling a computer cursor by eye movement. A number of modules were created to interface the digital and physical worlds already available. We present a novel approach to human-computer interaction using a real-time camera and color pointers as opposed to the conventional cursor control. Using camera and computer vision technology such as image segmentation, background subtraction, and color tracking, we demonstrate the capability of a computer mouse to perform these tasks like current mouse devices. This can be done because a color pointer takes care of object identification and tracking so that a module can be constructed without actual physical hookup. A color pointer was used to mark and follow objects to facilitate the construction of the module without direct system integration. The development of the program was carried out using Python Atmosphere and Windows 10. The overall objective of this method is to economically produce a hands-free interface between a computer and a us-er using a webcam to construct a virtual tool for human-computer interaction.

Keywords: **OpenCV, HCI, Python, colour pointer, eye movement detection**.

----------**********----------

## I. INTRODUCTION

Human-computer interaction has tried to produce better accessible and natural interfaces but financial limitations preclude large-scale deployment of touchscreens in desktops since such screens are utilized solely with smartphones. The material discusses Eye Cursor as an application for eye tracking to generate virtual mouse interface integration. Eye Cursor provides low-cost hands-free human-computer interaction by converting sight into cursor zones to enable novel accessible interfaces and customize advertisement modes.

Eye Cursor accuracy increases in each session by geometric detection that involves learning user-specific data as well as OpenCV's Haar cascades.

The study explores Eye Cursor design elements with its evaluation implementation to show how it operates as a substitute for the mouse and overcomes hardware limitations and cost minimization.

## II. LITERATURE SURVEY

### *A. SURF (Speeded-Up Robust Features)*

SURF is a fast and stable local feature detector and descriptor in computer vision. It finds and describes interest points in images, allowing for applications such as object recognition and 3D reconstruction. SURF is more efficient and faster than its successor, SIFT, by applying integral images and approximating the Laplacian of Gaussian to detect interest points. It becomes scale and rotation invariant, hence appropriate for diverse image transformations.

### B. *Image Processing Techniques*

Image processing is the transformation of digital images to retrieve or highlight information.

Decomposition of images into RGB components is one of the methods, which enables detection and analysis with great accuracy. The method serves as a foundation for most applications, including object tracking and recognition.

### C. *Image Analysis Color Detection*

Image analysis is the process of extracting valuable information from images. In realtime processing, the RGB values of pixels are examined to identify specific colors, which is a basic application in object tracking and motion detection. Image processing converting grayscale images and eliminating noise using median filtering can eliminate noise, making subsequent analysis more precise.

## III. DESIGN

### A. *Algorithms Used:*

#### *1.1. Haar Cascades*

An object recognition algorithm called Haar Cascades is used to find things in pictures or video streams. To identify features like faces or eyes, it employs machine learning based classifiers that have been trained on both positive and negative images.

a. Recognition of Faces

First, you use a pre-trained Haar cascade, such as haarcascade_frontalface_default.xml, to identify the face region. This makes the area to be searched for eyeballs more manageable.

```
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_de
fault.xml') faces =
face_cascade.detectMultiScale(gray_frame,
scaleFactor=1.3, minNeighbors=5)
```

b. Detection of Eyes

After the face has been identified, the eyes are located by applying a second Haar cascade (such as haarcascade_eye.xml) inside the face ROI.

```
eye_cascade =
cv2.CascadeClassifier('haarcasca
de_eye.xml') for (x, y, w, h) in
faces:
   roi_gray = gray_frame[y:y+h, x:x+w]
   eyes = eye_cascade.detectMultiScale(roi_gray)
```

c. Localization of Iris or Pupils (Optional Enhancement)

To detect the pupil, a better indicator of gaze direction, you may process the eye region (e.g., using thresholding or Hough Circles) if greater precision is required.

After determining the location of the eye's center or pupil(s):

a. Based on the pupil's relative location within the eye socket, you can estimate the direction of look.

b. Map this location or direction to the coordinates on the screen.

c. To adjust the mouse pointer, use a library such as autopy or pyautogui.

#### *1.2. Dlib's HOG-based face detector and shape predictor*

detector = dlib.get_frontal_face_detector() → detects the face using HOG + SVM.

```
predictor =
dlib.shape_predictor("model/shape_predictor_68_fa
ce_landmarks.dat")
```

→ gives 68 facial landmarks, including eyes, mouth, and nose.

These landmarks are then used to:

*a.* Detect eye blinks/winks (via EAR),

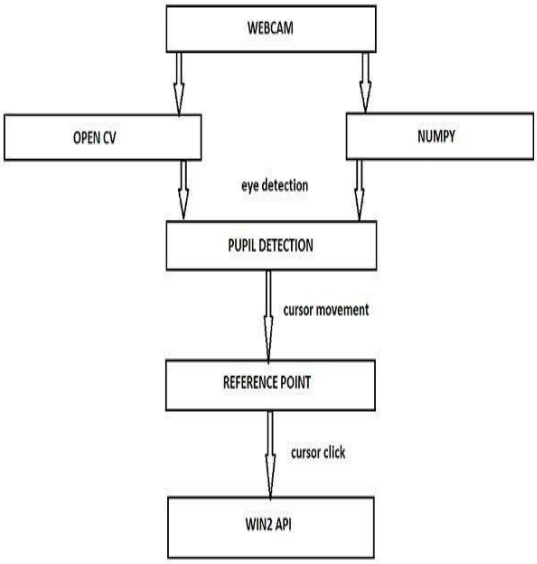
*b.* Detect mouth open (via MAR),

*c.* Track nose movement to move the mouse,

*d.* Trigger clicks or scrolling based on expressions.

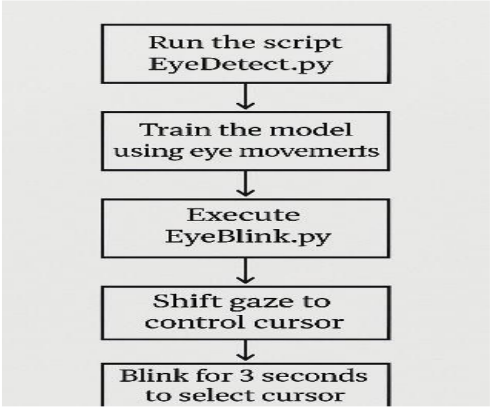
  

## B. *Architecture*




The main component of this system's architecture is a computer with a webcam that can record videos. We can easily detect the eyes and face thanks to the OpenCV haarcascade library. The user's side of the script then displays the user's live video on their screen and attempts to obtain the reference point by first locating the pupil center and then determining the joint pupil probability. By using a reference point to read the user's movement, we attempt to determine the gaze position, which is then used to move the cursor to that location. Additionally, we click the cursor on the blink using the PythonAutogui library; however, the blink should last for five seconds in order to distinguish it from a natural blink.

## C. *Implemetation For Flow Diagram*




1. ***Launch the python script (EyeDetect.py):***

•Action: The user starts the eye-tracking program by executing the "EyeDetect.py" Python program.

•Purpose: This program probably initializes the webcam(camera), configures the image processing pipeline for detecting eyes, and maybe addresses the initial calibration or setup process. It initiates the main functionality of the eye- tracking system.

2. ***Train the dataset using the assistance of eye movement*:**

•Action: The system captures and monitors the eye movements of the user.

•Purpose: In this step, there is a training or calibration process. Watching how the user moves their eyes, the software learns the user's unique features of eye movements and associates them with intended actions (such as cursor movement). This individualization is necessary to achieve accurate and effective eye tracking. The "dataset" in training probably corresponds to a model or set of parameters that the software employs in interpreting the user's gaze.

3. **Run the script EyeBlink.py:**

•Action: The user starts a second Python script, "EyeBlink.py".

•Purpose: The script is responsible for the blink detection and click simulation feature. It is separated from the main eye-tracking script,to account various features of the interaction.

4. **Gaze movement to move the cursor:** •Action: The user moves their gaze to move the cursor on screen.

• Purpose: This is the primary mode of interaction. The system converts eye movement (as a function of training data) and moves the cursor accordingly, so that the cursor can be operated without hands.

***5. Hold blink for 3 seconds to click cursor:***

- Action: The user does a blink held for 3 seconds.
- Purpose: This action simulates a mouse click. A blinks duration that is longer than normal is employed to differentiate from normal blinks so that unintentional clicks are avoided. This permits the user to activate functions or pick items on the screen without having to utilize a physical mouse button.

**overall Flow:**

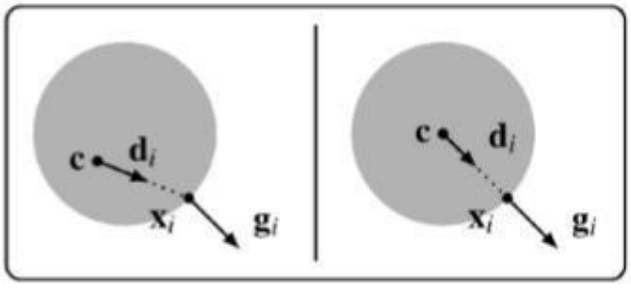
The image illustrates a simple procedure:

1. Start: Start the eye-tracking engine up.
2. Train: Adjust the system to suit the user's vision.
3. Activate: Blink-detection module initialization.
4. Interact: Use your eyes to move the pointer.

## IV. MATHEMATICAL CALCULATIONS

### A. *Locating Pupil Center*

Now we were required to locate the pupil in the eyes. Pupil is the focal and centering portion of eye, situated at focal point of iris. Light goes into eye through pupil, and finding the situation of understudy is foremost purpose of enthusiasm for proposed method. Eye stare projection depends on the relative uprooting of student from focus of eye. Student should be recognized to extend client's eye stare in the 'Test Area'.




If we take a unit vector di which points from the given image location to a gradient location, then the squared dot product with the normalized gradient vector i.e. gi, expressed as ([diT ]; gi)2 This will be as large as possible as if we are at the center of the pupil. We try to do

### B. *Equation of Pupil:*

$$\mathbf{c}^* = \arg\max_{\mathbf{c}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \left( \mathbf{d}_i^T \mathbf{g}_i \right)^2 \right\},$$

$$\mathbf{d}_i = \frac{\mathbf{x}_i - \mathbf{c}}{\|\mathbf{x}_i - \mathbf{c}\|_2}, \quad \forall i : \|\mathbf{g}_i\|_2 = 1$$

***Reference Point:***

Information from both eyes was integrated to increase the accuracy of pupil location probability calculations. Accurately integrating these probabilities was a major difficulty because the bounding boxes surrounding the eyes were insufficiently precise for direct picture alignment. A new technique was developed by taking

advantage of the fact that, while head position is fixed, the vector between the left and right pupils is rather constant (particularly when gazing at a planar surface like a computer screen). The probability picture for the right eye was superimposed on the probability image for the left eye instead of utilizing bounding boxes.

## V. RESULTS

We must now move the cursor using the reference point above and use blink to detect clicks. The win2api in Python, which is used as pythonautogui to control mouse movements like a click, was used for this. In this manner, we succeeded in achieving our goal.

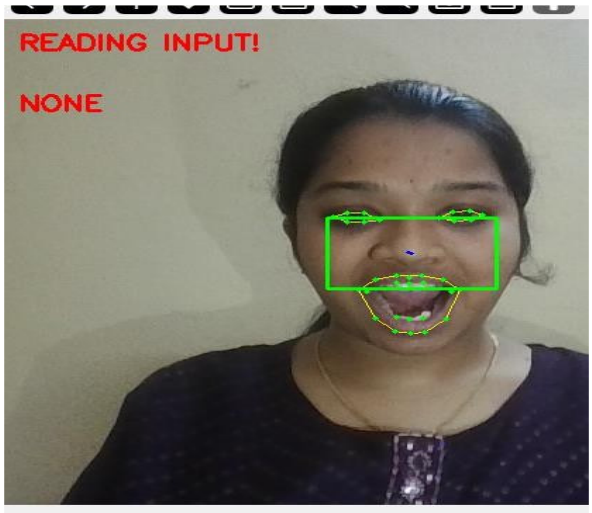



Fig.1. Reading Input

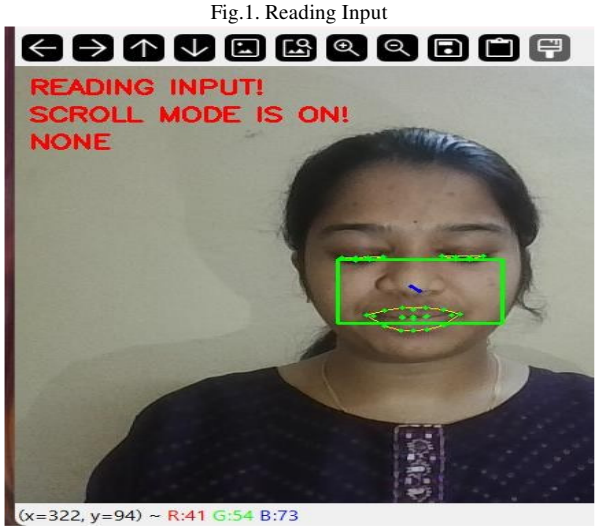



Fig.2. Scroll Mode ON

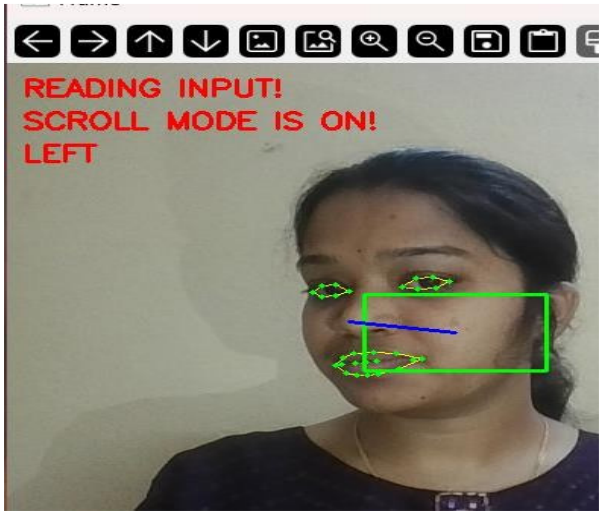



Fig.3. Scroll Mode Left

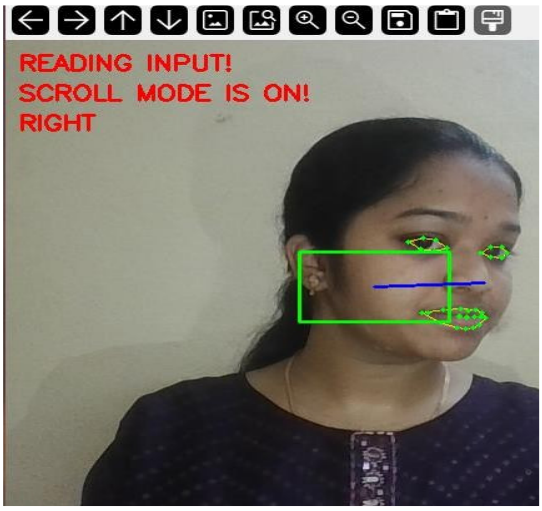



Fig.4. Scroll Mode Right

## VI. CONCLUSION AND FUTURE WORK

EyeCursor emerges as an innovative and cost-effective eye-tracking software that offers real-time gaze monitoring using computer vision. It supports hands-free computer control, making it particularly beneficial for individuals with disabilities, while also serving the needs of human-computer interaction, assistive technologies, market research, and security. The system captures valuable user data, such as attention spans and preferences, offering insights for analytics and research.

Currently, EyeCursor's performance depends on image resolution and lighting conditions. Future improvements include integrating lighting compensation algorithms and enhancing webcam resolution to boost robustness. Head pose estimation and gaze tracking using particle filters will increase accuracy and enable personalization. Challenges like tracking with head movements can be addressed by estimating head orientation through techniques like Haar cascades and facial geometry.

Eye tracking's relevance extends to VR/AR environments, gaming, and web usability testing, where it enhances user experience and supports cognitive research. To optimize system reliability and accuracy, adopting best practices—such as

proper lighting, hardware readiness, and trained staff—is essential. Overall, EyeCursor lays a strong foundation for affordable and scalable eye-tracking solutions that will continue to evolve and contribute to research and application development across diverse domains.

## ACKNOWLEDGEMENT

We are sincerely grateful for informative research in hand gesture recognition due to its beneficial impact on human-computer interaction with available findings at hand.

Previous research works provide a sound foundation for our study of eye-tracking approaches to EyeCursor design. Previous interactions findings alongside processing issues directed us towards creating our research framework and system development process.

## REFERENCES

[1] A. E. Barth, F. T. “Accurate eye centre localization by means of gradients.

[2] A. Gevers, V. (2008). “"accurate eye center location and tracking using isophote curva-ture".” IEEE Conference on Computer Vision and Pattern Recognition

[3] Bay, A. E. T. T. L. V. G. H. (2008). “Surf: Speeded up robust features.” Computer Vision and Image Understanding, vol. 110.

[4] Jones, V. A. “Robust real-time object detection.” IEEE Conference on Object Detection.

[5] Ji, J. C. Q. “Probabilistic gaze estimation without active personal calibration.” Scopus journal on Object recognizition