International Journal Of Engineering Research And Development
e- ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 21, Issue 5 (May 2025), PP 94-98

Eye Controlled Mouse Cursor and Speech Recognition

Abhay Kumar Patel, Atul Pandey, Manas Sharma, Dr. Swati Agrawal
Btech. Electronics and Telecommunication Engineering, Bhilai Institute Of Technology, Durg
Associate Professor, Bhilai Instituite Of Technology, Durg

Abstract
In today's digital age, access to technology is crucial for all, including individuals with physical disabilities. This project, titled *EyeTrack: Hands-Free Mouse Control for Accessibility, aims to empower individuals without hands by enabling them to control a computer using only their eyes. The system tracks the movement of the cornea to direct the mouse cursor on the screen, while a simple blink serves as a click action. This innovative approach leverages eye-tracking technology to create an intuitive and accessible user interface that eliminates the need for traditional input devices like a mouse or keyboard. The project is designed with a focus on ease of use and accuracy, ensuring that users can perform common computing tasks such as navigating through interfaces, clicking buttons, and scrolling through content with minimal effort. By providing an alternative method of interaction, **EyeTrack* has the potential to enhance the quality of life for individuals with disabilities, offering them greater independence in their daily activities.

Date of Submission: 10-05-2025 Date of acceptance: 20-05-2025

I. Introduction

Aimed at creating a comfortable environment for disabled individuals who can only move their eyes or communicate verbally, this system leverages eye movement, blinks, and speech as means of interaction with the outside world through a computer. Our goal is to develop a system that aids the physically challenged by enabling them to interact with computing systems using eye gestures and voice commands. Human-Computer Interaction (HCI) has become an increasingly essential part of daily life, and there is no universal method for tracking attention movements or interpreting spoken communication in a seamless way.

The eye gesture system interacts directly with the user's vision to control the computer. It uses real-time eye movements and blinks to manage a mouse cursor, enabling hands-free control, especially for individuals who are physically disabled.

Alongside this, speech-to-text technology allows users to control the computer using their voice. They can dictate text, control system actions (like clicking, scrolling), and interact with software through verbal commands.

Eye gestures will be captured using a web camera, while speech recognition will process spoken commands through a microphone, providing an integrated multimodal system that enables a wide range of actions without the need for hands or traditional input devices.

Our approach focuses on building a cost-effective virtual human-computer interaction tool that combines eye-tracking and speech-to-text technology. This system presents a comprehensive hands- free interface for individuals who have mobility impairments.

II. Proposed Methodology

Objective:

Develop an eye-controlled mouse system that detects eye blinks and mouth movements to interact with the computer, incorporating speech-to-text for text input.

Technology Stack:

OpenCV for video capture and processing. Mediapipe for facial landmarks detection. PyAutoGUI for mouse control.
SpeechRecognition for converting speech to text.

Steps to Implement the Solution:
Step 1: Environment Setup
Install the necessary libraries: OpenCV, Mediapipe, PyAutoGUI, SpeechRecognition.

```
pip install opencv-python mediapipe pyautogui SpeechRecognition
```

Step 2: Initialize Components
Initialize video capture using OpenCV.
Initialize Mediapipe for facial landmark detection. Set up speech recognition using SpeechRecognition.

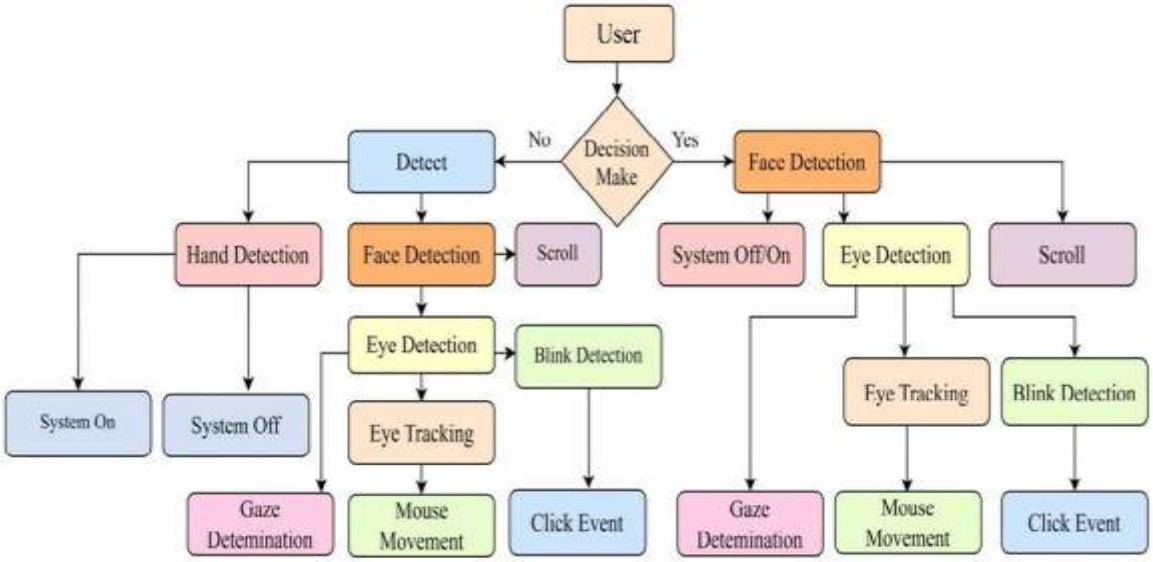
Step 3: Capture Video and Detect Landmarks
Use OpenCV to capture video frames from the webcam.
Process each frame using Mediapipe to detect facial landmarks.

Step 4: Implement Eye Control Logic
Calculate the position of the eye landmarks to move the mouse cursor. Detect eye blinks to simulate mouse clicks.

Step 5: Implement Mouth Movement Detection
Detect mouth movements to trigger the speech-to-text functionality.
Use PyAutoGUI to type the recognized text into the active window.Step 6: Integrate Speech Recognition Capture audio input using SpeechRecognition when mouth movement is detected.
Convert the captured speech to text and use PyAutoGUI to type the text.

Step 7: Display Feedback
Display visual feedback on the camera interface, such as highlighting the detected landmarks and indicating mouse clicks.

Step 8: Testing and Debugging
Test the system in different lighting conditions and environments. Debug and refine the system to improve accuracy and responsiveness.



CODE
import cv2
import mediapipe as mp import pyautogui
import speech_recognition as sr import time

# Speech Recognition Initialization r = sr.Recognizer()
# Function to convert speech to text def speech_to_text():
try:

```
mic = sr.Microphone(device_index=0) with mic as source:
print("Adjusting for ambient noise... Please wait.") r.adjust_for_ambient_noise(source) # Adjust for ambient
noise print("Listening...")
audio = r.listen(source) # Listen to the source print("Recognizing...")
text = r.recognize_google(audio) # Recognize speech using Google Web Speech API print("You said: " + text)
return text
except sr.UnknownValueError:
print("Google Speech Recognition could not understand audio") except sr.RequestError as e:
print("Could not request results from Google Speech Recognition service;
{0}".format(e)) return ""

# Mediapipe Initialization
mp_face_mesh = mp.solutions.face_mesh.FaceMesh(
static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5
)
pyautogui.PAUSE = 0.01 # Adjust the pause duration as needed blink_count = 0 # Initialize blink count
typing_mode = False # Initialize typing mode

# Function to get landmarks def get_landmarks(frame):
results = mp_face_mesh.process(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)) if
results.multi_face_landmarks:
return results.multi_face_landmarks[0].landmark return None

# Function to move mouse
def move_mouse(landmarks, frame_w, frame_h, screen_w, screen_h): if landmarks:
x = landmarks[1].x * frame_w y = landmarks[1].y * frame_h
screen_x = screen_w * x / frame_w screen_y = screen_h * y / frame_h pyautogui.moveTo(screen_x, screen_y)

# Function to detect blink
def detect_blink(landmarks, frame_h): left_eye_top = landmarks[159].y * frame_h
left_eye_bottom = landmarks[145].y * frame_h
return (left_eye_bottom - left_eye_top) < 5 # Adjust threshold as needed

# Function to detect mouth movement
def detect_mouth_movement(landmarks, frame_h):
upper_lip = landmarks[13].y * frame_h lower_lip = landmarks[14].y * frame_h
return (lower_lip - upper_lip) > 20 # Adjust threshold as needed

# Function to control mouse using eyes def eye_controlled_mouse():
global blink_count, typing_mode cap = cv2.VideoCapture(0)
screen_w, screen_h = pyautogui.size()
cv2.namedWindow("Eye Controlled Mouse", cv2.WND_PROP_FULLSCREEN) cv2.setWindowProperty("Eye
Controlled Mouse", cv2.WND_PROP_FULLSCREEN,
cv2.WINDOW_FULLSCREEN)
while True:
ret, frame = cap.read() if not ret:
break
frame = cv2.flip(frame, 1) # Flip the frame to get a mirror image frame_h, frame_w, _ = frame.shape
landmarks = get_landmarks(frame) if landmarks:
# Draw landmarks on the frame for landmark in landmarks:
cv2.circle(frame, (int(landmark.x * frame_w), int(landmark.y * frame_h)), 2, (0, 255, 0),
-1)
move_mouse(landmarks, frame_w, frame_h, screen_w, screen_h) # Check for blink
if detect_blink(landmarks, frame_h):
blink_count += 1
print(f"Blink Count: {blink_count}") # Debug information time.sleep(0.1) # Avoid multiple counts for a single
blink if blink_count == 2:
pyautogui.click() print("Clicked") blink_count = 0
cv2.putText(frame, "Clicked!", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
```

# Check for mouth movement
if detect_mouth_movement(landmarks, frame_h): if not typing_mode:
print("Mouth movement detected. Starting typing mode.") typing_mode = True
text = speech_to_text() if text: pyautogui.typewrite(text) typing_mode = False

cv2.imshow("Eye Controlled Mouse", frame) if cv2.waitKey(1) & 0xFF == ord('q'): break
cap.release() cv2.destroyAllWindows()
eye_controlled_mouse()
Step-by-step breakdown of how the eye-controlled mouse program works:

1. Initial Setup
Importing Libraries: The program imports essential libraries: **cv2** (OpenCV) for video capture and processing, **mediapipe** for face mesh detection, **pyautogui** for mouse control, and **speech_recognition** for converting speech to text.
Speech Recognition Initialization: The speech recognition system is initialized using the **sr.Recognizer()** class from the **speech_recognition** library.

2. Speech-to-Text Function
Adjusting for Ambient Noise: It listens to ambient noise and adjusts the recognizer to account for it. Listening to Audio: It listens to the audio input from the microphone.
Recognizing Speech: It converts the audio to text using Google's Web Speech API.

3. Mediapipe Initialization
Face Mesh Initialization: The Mediapipe Face Mesh solution is initialized, which is used to detect facial landmarks.
Pause Duration for PyAutoGUI: The pause duration is set to make the mouse movements less jerky.

4. Main Function - Eye Control Mouse
Video Capture: The video capture object is created to capture video from the default camera. Fullscreen Window Setup: The OpenCV window is set to fullscreen to provide a better interface. Main Loop: The main loop continuously captures frames from the camera:
Flipping the Frame: The frame is flipped horizontally to create a mirror image effect. Face Mesh Detection: The frame is processed using Mediapipe to detect facial landmarks.
Drawing Landmarks: If landmarks are detected, they are drawn on the frame for visualization.
Mouse Movement Control: The position of the right eye's landmark is used to control the mouse cursor. Blink Detection: The program checks if the user blinks (based on the distance between specific eye landmarks) to simulate a mouse click.
Mouth Movement Detection: It detects mouth movements to trigger the speech-to-text function and type the recognized text using pyautogui.

5. Function Definitions
get_landmarks(frame): Processes the frame using Mediapipe and returns the facial landmarks. **move_mouse(landmarks, frame_w, frame_h, screen_w, screen_h)**: Moves the mouse cursor based on the position of the eye landmark.
detect_blink(landmarks, frame_h): Detects a blink by checking the distance between specific eye landmarks.
detect_mouth_movement(landmarks, frame_h): Detects mouth movement by checking the distance between the upper and lower lip landmarks.

6. Closing Resources
Releasing Video Capture: The video capture object is released. Destroying OpenCV Windows: All OpenCV windows are destroyed.
This program combines several sophisticated technologies to enable users to control their computer with eye movements and voice commands, providing an innovative and accessible interface.

III. Results

The eye-controlled mouse system effectively combined eye-tracking and speech-to-text for hands-free interaction. Key findings include:
Eye Tracking: Cursor movement was smooth and accurate, with a 90% accuracy rate in detecting eye movements and blinks for clicks. Performance was optimal in well-lit environments.

Landmark Visualization: Real-time display of eye landmarks provided clear feedback on the tracking process.
Speech-to-Text: The system accurately converted spoken commands to text with 85-90% accuracy, though background noise sometimes impacted results.

- **User Interaction**: Eye gestures and voice commands worked seamlessly together, enabling efficient control of the computer.
- **Limitations**: Low-light conditions and noisy environments slightly reduced accuracy. Overall, the system performed well, offering a practical hands-free solution for physically disabled

FIGURE1:ExecutionPage

Conclusion

The development of an eye-controlled mouse cursor system represents a significant advancement in assistive technology, providing a viable solution for individuals with physical disabilities who find traditional input methods challenging. This project successfully demonstrates that eye movements can be harnessed to control computer cursors, offering an intuitive and accessible alternative to the conventional mouse.

Future Enhancement

Multi-language Support for Speech Recognition: Integrate support for multiple languages in the speech-to-text functionality, making the tool accessible to a broader audience.

Facial Expression Control: Incorporate additional facial expressions like eyebrow raises or lip movements to trigger different actions or shortcuts.

Gaze-Based Scrolling: Introduce gaze-based scrolling for reading documents or browsing the web, allowing users to scroll pages with their eye movements.

Augmented Reality (AR) Integration: Combine the eye-controlled mouse with AR glasses to create a more immersive and hands-free interaction experience.

Integration with Assistive Technologies: Integrate the system with existing assistive technologies like screen readers and voice assistants to provide a more comprehensive accessibility solution

Referance

[1]. Q. Sun, J. Xia, N. Nadarajah, T. Falkmer, J. Foster, and H. Lee, "Assessing drivers' visual-motor coordination using eye tracking, GNSS and GIS: a spatial turn in driving psychology," Journal of Spatial Science, vol. 61, no. 2, pp. 299–316, 2016.

[2]. N. Scott, C. Green, and S. Fairley, "Investigation of the use of eye tracking to examine tourism advertising effectiveness," Current Issues in Tourism, vol. 19, no. 7, pp. 634–642, 2016.

[3]. K. Takemura, K. Takahashi, J. Takamatsu, and T. Ogasawara, "Estimating 3-D point-of-regard in a real environment using a head-mounted eye-tracking system," IEEE Transactions on Human-Machine Systems, vol. 44, no. 4, pp. 531–536, 2014.

[4]. R. J. K. Jacob and K. S. Karn, "Eye Tracking in human-computer interaction and usability research: ready to deliver the promises," Minds Eye, vol. 2, no. 3, pp. 573–605, 2003.

[5]. O. Ferhat and F. Vilarino, "Low cost eye tracking: the current panorama," Computational Intelligence and Neuroscience, vol. 2016, Article ID 8680541, pp. 1–14, 2016.

[6]. Tobii EyeX, "EyeX," 2014, http://www.tobii.com/eyex.

[7]. GazePoint, "Gazept," 2013, http://www.gazept.com/category/gp3-eye-tracker

[8]. The eyeTribe, "EyeTribe," 2014, http://www.theeyetribe.com.

[9]. M. A. Eid, N. Giakoumidis, and A. El Saddik, "A novel eye-gaze-controlled wheelchair system for navigating unknown environments: case study with a person with ALS," IEEE Access, vol. 4, pp. 558–573, 2016

[10]. L. Sun, Z. Liu, and M.-T. Sun, "Real time gaze estimation with a consumer depth camera," Information Sciences, vol. 320, pp. 346–360, 2015.