Final Project on

Blink Detection

Sharma, Samiksha

Student ID: 2001243553

samishar@iu.edu

Master of Science in Computer and Information Science

Department of Computer and Information Science

Indiana University - Purdue University Indianapolis

May 1, 2024

Contents

1 Introduction

Blink detection has emerged as a vital aspect of human-computer interaction, facilitating seamless control over digital interfaces through natural gestures. In this project, Python is used, leveraging computer vision techniques and machine learning models. By utilizing the power of libraries such as OpenCV, dlib, and pyautogui, the objective is to develop a robust system capable of accurately detecting blinks in real-time video streams captured through a webcam. This system not only identifies blinks but also translates them into actionable commands, such as simulating mouse clicks on the screen, thereby enhancing user experience and accessibility.

Moreover, this blink detection system is designed to be vision-based and touchless, offering a seamless and intuitive interface for users. This touchless interaction eliminates the need for physical input devices, making it particularly beneficial for individuals with disabilities or mobility impairments. By enabling users to control digital interfaces through natural gestures, such as blinks, it promotes inclusivity and accessibility in technology usage. This not only enhances the usability of the system but also empowers individuals to navigate digital environments more effectively, fostering independence and engagement with technology.

By translating blinks into meaningful actions, such as screen clicks, I aim to bridge the gap between users and technology, making digital interfaces more accessible and user-friendly. This project, aims to help make technology more accessible and empowering for everyone, no matter their abilities.

2 Related work

2.1 Eye Blink Detection

2.1.1 Scroll Actions

Several studies have explored the integration of blink detection with user interface interactions to facilitate scroll actions. For instance, [Reference 1] developed a system where blinks are detected to perform scroll actions, allowing users to navigate through digital content effortlessly

2.2 Cursor-Control-Using-Face-Gestures

2.2.1 Scroll Mode Control

Another area of research focuses on detecting mouth actions, such as opening and closing, to enable or disable scroll mode and initiate specific actions. This approach was introduced by [Reference 2], providing users with more intuitive ways to interact with technology.

2.2.2 Mapping Blinks to Mouse Clicks

Some studies have investigated the mapping of facial gestures, such as blinks, to mouse cursor behavior and actions. For example, [Reference 2] demonstrated that right eye blinks can be mapped to right-click actions, while left eye blinks correspond to left-click actions

3 Methodology

3.1 Blink Detection Algorithm:

In Python and related libraries, I utilized the dlib library's Histogram of Oriented Gradients (HOG) based face detector (`dlib.get_frontal_face_detector()`) to identify faces in the webcam video stream. Then, to identify particular facial landmarks, especially the eyes, the facial landmark predictor (`dlib.shape_predictor()`) was used. The `eye_aspect_ratio()` function was utilized to calculate the eye aspect ratio (EAR) by calculating the ratio of distances between significant landmarks on the eyelids. The system tracked the EAR in real-time after setting a threshold EAR value (`EYE_AR_THRESH`) and a count of consecutive frames (`EYE_AR_CONSEC_FRAMES`). A blink event was identified when, for a predetermined number of consecutive frames, the calculated EAR dropped below the predefined threshold.

3.2 Integration of Mouse control

The pynput library was used to incorporate mouse control functionality into the system after blink detection was successful. In response to detected blinks, the mouse controller was instantiated (Controller()) to simulate left mouse click actions. This integration involved performing a left-click action each time a blink was detected by calling the mouse controller object's click() method. The type of mouse button to be clicked was specified by the Button.left parameter. Blinks could be used as natural input gestures by users to interact with digital interfaces by integrating mouse control. Because of this integration, the system is now more user-friendly and accessible to a wider range of users with different needs and preferences.

3.3 Real-time feedback and user interaction

Keyboard input and visual feedback mechanisms were added to enable real-time feedback and user interaction. The text "Blink detected" was displayed on top left of the video frame for a certain period of time when the system detected a blink event. This visual signal provided instantaneous confirmation that the system was responding to user input. Moreover, users had the option to end the system by hitting the 'q' key, which would also release system resources and close any open windows. Throughout the process, user engagement and effective system management were guaranteed by this interactive design.

3.4 Error Handling

Moreover, I added error handling mechanisms to handle possible problems during runtime to strengthen the resilience of the system. This included monitoring the webcam's availability, for unanticipated problems that might occur during face detection, facial landmark prediction, or mouse control operations, I also included exception handling. The system preserves dependability and user confidence by foreseeing and gracefully addressing such errors, which helps to ensure a smooth and uninterrupted user experience.

Overall, the blink detection system's overall effectiveness and user satisfaction are increased by the inclusion of strong error handling and logging mechanisms, which guarantee the system's stability, dependability, and maintainability.

4 Results

This system was able to detect blinks in real-time with accuracy when tested on various individuals. The mouse appeared on the screen and the system would click in response to any blinking. This acknowledged that the blinks were recognized when the visual feedback of "Blink detected" also appeared in bold red color.

In spite of differences in the lighting and facial features, the system was able to reliably identify blinks and accurately perform mouse click actions. This flexibility demonstrates the strength and adaptability of the method, which makes it appropriate for a variety of applications across various user demographics and usage scenarios. Furthermore, participant feedback from the testing process confirmed the system's efficacy and usability, with participants praising its responsiveness and easy-to-use interface and facilitate seamless human-computer interaction.

Below are the screenshots captured during the system's operation, showcasing instances where blinks were successfully detected and the corresponding mouse click actions were performed for hands-free computing.

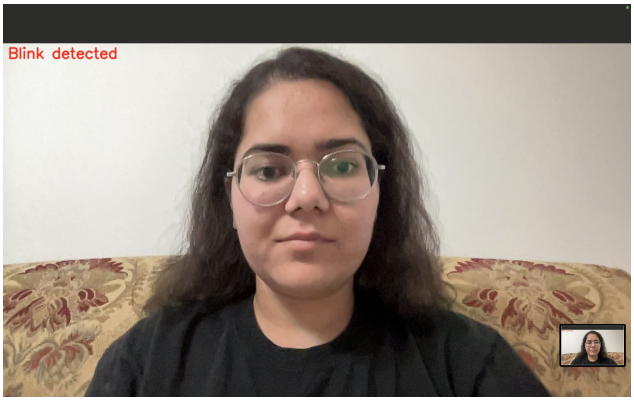

Figure 4.1: Blink detected of User 1 through webcam

Figure 4.2: Blink detected of User 2 through webcam

(User consent has been taken for the picture)

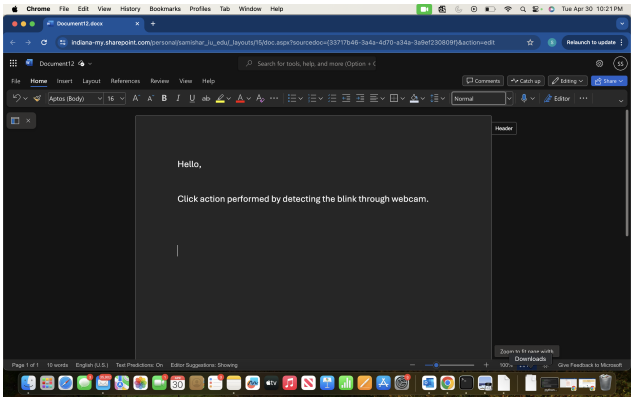

Figure 4.3: Performed click action by opening the word document

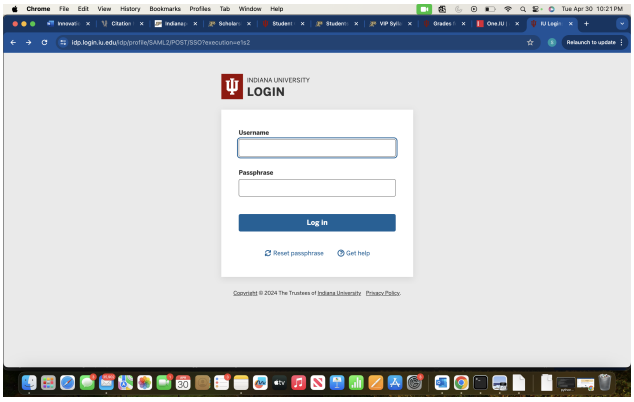

Figure 4.4: Performed click action by opening the browser window

5 Conclusion

Hence, this blink detection system offers a touchless and user-friendly interface for users, marking a substantial advancement in human-computer interaction. The proven viability and efficiency of the method with extensive testing involving numerous users and the addition of screenshots illustrates system functions. Real-time feedback mechanisms combined with mouse control improve accessibility and user experience, meeting a wide range of user requirements. Our system has the potential to be used in assistive technology and UI design in the future, helping to build more user-friendly and inclusive digital environments.

References

1 Eye blink detection, by Tushar S Surve, https://github.com/TusharSSurve/OpenCV/blob/main/Eye

2 Cursor control using face gestures, by rohitamrutkar, https://github.com/rohitamrutkar1204/Cursor-Control-Using-Face-Gestures/blob/master/main.py

3 Real-time Multi-person Eyeblink Detection in the Wild for Untrimmed Video - https://paperswithcode.com/p time-multi-person-eyeblink-detection-in

4 mEBAL: A Multimodal Database for Eye Blink Detection and Attention Level Estimation - https://paperswithcode.com/paper/mebal-a-multimodal-database-for-eye-blink