CSCI 1430 Final Project Report: All Eyes On Me

Team name: Isabelle Sharon, Elizabeth Wu, Orion Bloomfield, Hannah Julius.
TA name: Qian Zhang. Brown University

Abstract

For this project, we wanted to implement webcam eye-tracking to control the computer cursor you see on your screen. We were able to identify major facial landmarks to use for webcam position tracking, but we weren't able to get eye (pupil) tracking to work properly, as we couldn't get granular enough feature points that allowed for accuracy and consistency. As a result, we used the popular 68-point model from Dlib with major facial landmarks to control the cursor's position using the tip of your nose and clicking by opening your mouth and winking. This allows for more accuracy and robustness in the mouse movements, because the facial movements to control the cursor are much bigger than pupil movements. While this does detract from the overall accessibility of our project, we believe that with more time, we could create a custom feature detection model built off of the one we ended up using that would allow us to implement pupil detection (and thus gaze tracking) to control the cursor.

1. Introduction

The problem we are trying to solve in our project is the issue of those with mobility limitations' ability to use their computer mouse with all of its functionality. Our project aims to use face tracking to control the cursor without use of one's hands. Our original goal was to control the mouse via eye gaze tracking; however, it's very difficult to get pupil tracking to work well/consistently, even with a better webcam. When looking for facial landmark models, we also weren't able to find many that could identify where the pupil is. This project is also difficult because we had to implement other mouse functionality (besides having it around on the screen) that would have to be connected to specific facial movements. When adding mouse clicking, we struggled to decide what the best strategy would be. Tracking something like blinking for a certain amount of time requires that the camera is of high enough quality to detect this well. On the other hand, tracking something like head nodding may not be the most accessible, because it requires quick, exaggerated head movements that could hurt your neck. The approach we decided to take was moving the head to move the mouse (since we got nose tracking to work), opening the mouth to click and drag on the screen, and winking to right click. We think these movements led to more robust and accurate cursor control, since they're so 'big' that it's hard for the mouse to misidentify what the user is doing. If we had more time/resources, though, we'd love to be able to properly implement gaze tracking, as that would make our project truly accessible to users with limited mobility.

2. Related Work

We researched the existing solutions to this problem and similar problems, one of the sources we looked at was from access computing, How can people with mobility impairments operate computers?.

We also looked into state-of-the-art ML pipelines for inspiration on iris tracking before switching direction. (In particular, Google's MediaPipe Iris[1])

When we started coding, we first used a Towards Data Science tutorial[2] to get a stencil standing up that we could change and build off of going forward. We also referenced this[3] Github repo to learn how to detect when webcam users open their mouth, as well as this[4] YouTube video to get a similar stencil for face detection that we could change and build off going forward.

We didn't rely on any research papers or their implementations, but we did reference the Webgazer.js[5] for general project inspiration, as well as to get an idea for what eye-tracking research looks like.

3. Method

Our approach to solving this issue was to first find a library that would accurately plot facial landmark points. We researched and tested many different implementations, and modified the parameters of each to match how we wanted the tracker to work. This included restricting the points to just the face and nose, as we would only be considering these points to track movement of the mouse. Once we had these points, we locally store them to check to see if there is movement between points of interest. We check to see movement in the previous and current nose points to detect a change which results in movement of the mouse. While we were really impressed by Webgazer's capabilities[5], we also noticed that there were still some discrepancies (e.g. lag when following eye position). Given the scope of this project, we felt that it made the most sense for us to implement nose/face tracking, rather than gaze tracking, because we'd be able to create a decently accurate/robust demo in our time frame.

We also check to see whether users open their mouth (and make a face similar to " :O ") to control the mouse's left click. If users keep their mouths open, the left click will still be held, and if they begin moving around with their mouths open, the cursor will follow them whilst holding down left click (so it's equivalent to click and drag, which we use in situations like highlighting text). To right click, we check for users to wink with their right eye. For both opening the mouth and winking the right eye, we get the aspect ratio of the mouth and right area area, then just compare them to manually-defined thresholds that we found gave us the most accurate readings.

4. Results

The final result we ended up with included nose tracking for movement of the cursor, as well as opening the mouth in order to click on the screen, and finally a wink with the right eye to right click on something. Before this, we tried to implement eye tracking for movement of the cursor, where the pupil would determine where the cursor moves. But, we were unable to find a way to determine the pupil point that was realistic within the scope of our project. So, we shifted towards using eye points to move the cursor, where we found that the points were slightly inaccurate when using our laptop's webcams. We experimented with this implementation with a clearer camera, and found that it did slightly improve the points, but still was not very accurate on those with glasses. To improve accessibility and the accuracy for our project, we decided to shift to nose tracking. We chose to do this because the nose point was generally more accurate than the eye points. With more time we would try to find ways to improve upon the accuracy and implement gaze tracking, but within the time frame we ended up settling on nose tracking. For click functionality, we had two implementations as well. We considered nodding as well as opening one's mouth. We decided against nodding because it was a much more unnatural motion, and required moving the head back and forth in a way that is also not very accessible.

Some of the results we have are shown in videos that we submitted to gradescope along with our code. We submitted the demo video as a zip file within our group's repo called **iTrack.mp4.zip**. We demonstrate clicking with a left click, clicking with a right click, as well as selecting text and drawing through left click and drag.

4.1. Technical Discussion

Because we chose to use nose-tracking to control the movement of the mouse, it requires pretty exaggerated head movements that bring the accessibility of the program to question. Indeed, we did have to sacrifice the accessibility of the project somewhat by not implementing pure gaze-tracking. However, given the scope and time frame of this project, as well as the difficulty of implementing accurate and stable pupil detection (i.e. pupil detection that isn't jittery because our eyes are constantly making very small movements — even when we don't think so), we think our decision to use nose tracking makes sense. We still relieve users from the necessity of controlling a mouse with their hands, and with bigger/more exaggerated head movements, the cursor is able to accurately follow where the user directs it. Plus, the lack of a DL model makes it so that the cursor can pretty much have a one-to-one line up with the user's nose position, since the code can just run faster.

4.2. Socially-responsible Computing Discussion via Proposal Swap

1. The first criticism is about data that could be inadvertently collected about users, such as eye movements, gaze patterns, and other biometric data. This is a very valid concern, and one that we, if we were planning to implement this in a practical setting, would focus on to ensure that we aren't saving and collecting unnecessary data. We currently don't store any user data — the webcam sees what's happening as the program is running, but nothing is ever saved afterwards. However, were we to put this project out for the public, we would add security measures to ensure that the information we use to track differences in movements such as opening one's mouth to click is still not stored anywhere. We would only use this information to detect a click or right click while the program is running, and then perform the corresponding action.

   If we were to expand on the project, a potential issue could be if we add gaze-tracking functionality via a deep learning model that learns thresholds based on the user. For example, if a user tends to regularly open their mouth a specific amount in order to click, the program could learn this amount and adjust the thresholds in order to accept a more or less obvious opening of the mouth. With this extension, there could be more issues with storing data, and we would brainstorm further on ways to avoid storing this information. A potential solution here could be that the information is encrypted and used strictly within the program, rather than being stored in a plaintext format corresponding to the user. However, without the extension, we reiterate that we currently do not store any information about our users, especially not any sensitive information.

2. The next concern is ensuring that our project accommodates a wide range of disabilities and varying degrees of visual and motor impairments. This is a really important concern, but also something we felt that we would only be able to account for given more time and resources. By switching from eye-tracking to nose-tracking, we were able to increase support for users with glasses, since the accuracy for detecting eye-feature points while wearing glasses was pretty shoddy. However, our project does currently operate on the assumption that users can see the screen and where they're moving the cursor, but that may be more difficult for users that are already in a position of having very low vision or mobility. If we had more time, we think it would've been cool to explore features like voice control to allow access to even more users. The group critiquing our project mentioned that it would be useful to discuss our project with an accessibility expert, which we think could also be really helpful and productive. Again, though, this is only something we could have done with more time. Within the project deadlines, we were focused on functionality for our base case, but we would love to spend more time making the project more robust and accessible in the future.

3. The third concern touches on the idea that our project may enforce an over-reliance on technology. The group cites that our project may promote a more sedentary lifestyle. We disagree with this concern. We think that implementing this project with a contextual focus on accessibility outweighs the potential side-effects of over-reliance on technology. Creating equal access and ability to use technology for those with accessibility issues does not directly encourage over-reliance on technology. The issues are unrelated in our opinion, and there can be other ways to reduce reliance on technology. We considered the effects of this project for those without disabilities as well, because the group seems to suggest that these alternate methods for using technology are more convenient and may increase overall usage. However, the fix here is not any more convenient than using a track pad — it's just another way to control your cursor. However, concerns about pupil tracking do remind us of the use of VR, and how pupil tracking is used to create a more human experience when interacting with other players in online games. This is a very valid concern, as the human-like nature of games is sure to make VR games that much more addicting. However, given the scope of our project — where we are mainly implementing these features in order to increase accessibility of basic computer functionality (and not even implementing pupil tracking) — we do not find this to be a major concern of ours. A more sedentary lifestyle is also inherent to society's increasing reliance on computers, so increasing accessibility will not promote unhealthy lifestyles more than computers already do.

5. Conclusion

Our project resulted in a functional eye tracker that used nose facial landmark points to shift the cursor on the screen. We also added functionality for left as well as right clicks, by winking and by opening one's mouth. Finally, one is able to drag the mouse by opening their mouth and moving their head concurrently. Our project implements an important accessibility feature, aiding those with limited mobility in their hands. We also find it important for the general public, as it has the potential to reduce adverse health effects of non-ergonomic keyboards. The impact going forward could be even more increased accessibility if we are able to implement gaze tracking.

References

[1] Artsiom Ablavatski, Andrey Vakunov, Ivan Grishchenko, Karthik Raveendran, and Matsvei Zhdanovich. Real-time pupil tracking from monocular video for digital puppetry, 2020. Google's iris-tracking implementation for camera effects. 1

[2] Irfan Alghani Khalid. Face landmark detection using python, 2021. Article from Towards Data Science. 1

[3] Ross Mauck (mauckc). mouth-open, 2019. Github repo that detects when a human's mouth is open. 1

[4] Nicolai Nielsen. Face detection and 3d position estimation in opencv, 2020. YouTube video tutorial on how to implement face detection using OpenCV. 1

[5] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, and James Hays. Webgazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3839–3845. AAAI, 2016. 1, 2

Appendix

Team contributions

Please describe in one paragraph per team member what each of you contributed to the project.

Isabelle Isabelle worked on finding a computer vision library in python in order to get access to points that correspond with different parts of the face. Isabelle also researched opencv applications that we could use for the project, and worked on implementing the facial landmarks section of our project.

Orion Orion provided a stencil for opencv in python to connect with a laptop camera feed. Orion also found a resource and reference for mouth recognition which would work to use the mouth and mouth opening for click functionality.

Hannah Hannah researched opencv related resources that could be helpful in implementing the project. Hannah also contributed to testing the code as well as working on the poster and project report.

Lizzy Lizzy also aided in researching potential libraries we could use for the project, as well as finding a resource to get a higher quality webcam than the computer webcam. Lizzy also worked on the poster and project report, as well as helping to debug.