Jacob Bloom, Nathaniel Carter

Andrew Neer

CS 420 Computer Vision

05/09/25

Drawing using Eye Tracking

For our project the goal was to make it so that a person can draw on a computer with their eyes. To achieve this, we used a computer camera to find a person’s eyes, track where they were looking, then draw on a canvas moving in the direction the user looked.

The design of the project follows multiple different steps. First, we detect the person’s face, then locate the eyes, and finally identify the center of each eye and the pupil. Using this information, we calculate the direction in which the person is looking. This direction is used to move a “cursor” on the screen and draw a rectangle at that point, turning the canvas into an Etch A Sketch controlled by eye movement. Included with this we also detect blinks to allow the user to change the drawing color.

The first major step is detecting the user’s eyes. To optimize performance and accuracy, we begin by detecting the entire face, narrowing the region where the eyes will be searched. This improves runtime and reduces the risk of false detections. We further refined the detection process by limiting what can be classified as an eye, which improved accuracy. Once the eyes are located, we identify the center of the eye and the pupil. These two points allow us to calculate the direction the user is looking. At this stage, we also detect whether the eyes are open or closed to enable blink detection. As a note, each step with eye detection is done using OpenCV's built-in classifiers (source: OpenCV documentation), which allowed us to focus on applying the technology rather than training custom models. For specifically blink detection, we had to use dlib landmark detection for faces, as well as install the associated database which is linked in our readme. Its unfortunate that we had to use two different models to detect desired features and if we were to go back I think we would have done this a little differently.

To determine the direction in which the user is looking, we calculate a vector from the center of the eye to the pupil. However, we encountered a challenge: the coordinates of the eye center were relative to the face, while the pupil coordinates were relative to the eye. This discrepancy meant the data existed on different coordinate planes and needed to be normalized. Once normalized we then can then calculate where the pupil sits compared to the center of the eye, being expressed as a 2D vector (a, b), where “a” indicates horizontal direction (negative for left, positive for right) and “b” indicates vertical direction (positive for up, negative for down). To detect blinking we use the landmarks found with dlib and calculate the eye aspect ratio between the top and bottom of the eye. if this average is below a certain percentage threshold then it is detected as a blink. Since we do this process individually for each eye, to detect winks we can check if one eye is below the threshold while the other eye is above the threshold.

With this data, the final step is to draw on the canvas. We maintain a point representing the user's current position. Based on the gaze direction, we increment or decrement this point accordingly. A rectangle is then drawn at the new position, simulating the effect of drawing with eye movement. By repeating this process in real time, the user can navigate the canvas and create drawings using only their eyes. The user can also use the implemented blink and wink detection to use additional features. Blinking allows the user to change the drawn color, left win restarts the canvas, and right wink fills the canvas with the current color. With all of these features combined we have created an interactive visualization of eye and pupil detection.

In conclusion, this project successfully demonstrates how eye-tracking technology can be used for interacting with a program. By accurately detecting the direction someone is looking and translating it into on-screen movement, we created a way to use this technology in drawing on a canvas, completing our goal.

SOURCES

<https://www.geeksforgeeks.org/how-to-install-dlib-library-for-python-in-windows-10/>

<https://www.geeksforgeeks.org/opencv-python-program-face-detection/>

<https://www.geeksforgeeks.org/eye-blink-detection-with-opencv-python-and-dlib/>

TIME CHART (min 18hr)

|  |  |  |
| --- | --- | --- |
| Person | Objective | Time |
| Nate | Eye Detection Consistency | 2 hrs. |
| Nate | Eye Direction Calculation | 4 hrs. |
| Nate | Canvas Implementation | 4 hrs. |
| Jacob | Face Detection | 2 hrs. |
| Jacob | Eye Detection | 2 hrs. |
| Jacob | Pupil Detection | 3 hrs. |
| Jacob | Blink / Wink Detection | 2 hrs. |
| Jacob | Blink / Wink functionality | 2 hrs. |