Digital Notes With Any Pen on Any Surface

Group Members: Asher Mai (hanlinm2)
Pauline Lu (pauline4)

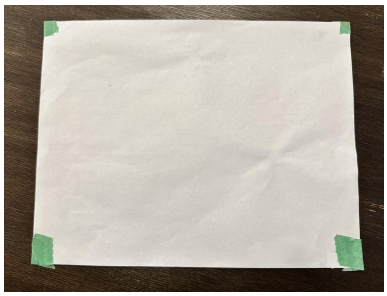
Video Demo

Motivation and Impact:

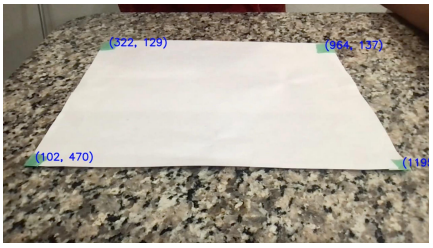
Our final project is to allow users to use any stylus to draw or write on any surface. The method would involve using a vertical camera, like a webcam, to detect the location of a person's moving pen tip to discern what that person is drawing or writing on any surface in front of the camera. We chose this topic because we always wanted to utilize the benefits of digital note-taking in middle and high school but we did not have access to the resources such as an expensive stylus and tablets. A software like this will allow users to enjoy the benefits of digital note-taking without buying expensive styluses and special tablets. When we first started with this project, we embarked with the hope of learning more about single-view geometry and Harris corner detection, and intended for both of these topics to form the integral technical components of this project. However, we realized that we needed to use different approaches for better and faster results.

Approach:

We prepare an A4 paper with four corners taped with green masking tape, and place it on the table in front of the webcam.

We use color thresholding to get the four clusters of paper corner locations. We make the assumption that each paper corner point cluster falls in each of the four quadrants of the camera frame, and we separately average the threshold point clusters in each of the four quadrants to get four precise corner points shown in blue text below.

We use these four paper corner points and the four points of an on-screen canvas as correspondents to solve for a homography transform matrix H. This matrix H will be used to transform the on-camera pen location x to the on-screen canvas location x' (i.e. $x' = Hx$)

The user can press Q to confirm the calibration, move the paper away, and start drawing with a pen that also has green masking tape on the pen tip for color thresholding shown below. The drawing will be shown on the screen, and the user can press Q to quit or press C to clear canvas.


Results:

Video Demo

Our system works at around 15 frames per second. The text and drawing shown on canvas closely match the user's intended pen trajectory, although higher frame rate is needed to improve smoothness. We find that the drawings are more precise when the pen is closer to the camera (top of canvas) due to there being more canvas pixels per inch of the table closer to the camera. The color thresholding for the paper and the pen are very robust and enable the user to very easily calibrate the working surface as well as very easily move the pen in natural writing and drawing motions without much loss in accuracy.

The significance of our project is that, without spending money on expensive tablets and styluses, anyone can set up this drawing environment anywhere with any pen as long as they have the taped paper and pen in their backpack.

Implementation Details:

Our project uses Python, as well as the OpenCV, and Numpy libraries. We used OpenCV for video capture, color thresholding, adding text, and drawing on the canvas. Numpy is used to store and process the camera frames and canvas, and for solving the homography transform matrix. We referenced lecture 18 on how to set up the linear equations and solve for homography:

Computing homography

Direct Linear Transform

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1u_1' & v_1u_1' & u_1' \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v_1' & v_1v_1' & v_1' \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv_n' & v_nv_n' & v_n' \end{bmatrix} \mathbf{h} = \mathbf{0} \Rightarrow \mathbf{Ah} = \mathbf{0}$$

- Apply SVD: $\boldsymbol{UDV}^T = \boldsymbol{A}$
- $\boldsymbol{h} = \boldsymbol{V}_{\text{smallest}}$ (column of $\boldsymbol{V}$ corr. to smallest singular value)

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_9 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Matlab

```
[U, S, V] = svd(A);
h = V(:, end);
```

We referenced the link below for how to use color thresholding and drawing lines on canvas after we obtain the homography transform matrix.
https://learnopencv.com/creating-a-virtual-pen-and-eraser-with-opencv/

Challenge/Innovation:

Our original idea involved using PyAutoGUI to open up a blank GoodNotes notebook and have the mouse select the pen tool and move it around in accordance with the pen movements picked up by the camera. However, we discovered that the latency with the mouse movements on PyAutoGUI was too high. One possible solution to the PyAutoGUI latency problem is to implement it with threading, but we ended up simplifying our implementation a bit by pivoting to just using the OpenCV canvas and line drawing functions for the on-screen interpretation of the pen movements.

We additionally ran into some problems with our original idea of using an object tracker to track the pen location and Harris corner detection for the pen tip, and edge detection to detect the edges of the paper on the table. Upon implementation of these corner detections, we quickly realized that the Harris corner detection was picking up on a lot of extraneous corner points that

were not relevant to the pen tip and any movements that were not slow and deliberate with the pen would result in a blur, causing the object tracker to lose track of the pen. We faced similar problems with the edge detection that we tried to use to detect the physical paper on the table. In order to overcome these issues, we decided to make the switch to using color thresholding instead, where bright green tape is used to mark the four corners of the canvas, and a object tracking box was manually applied to the pen tip during the calibration step and locked on to the tip of the pen as it moves during the writing process.

With color thresholding, we were able to calibrate the canvas surface, but the object tracker would still often lose track of our pen tip. Thus, we decided to apply color thresholding to the pen tip as well, with the same bright green tape used to mark the pen tip. With color thresholding, the tracking became a lot more robust, it enabled us to move the pen around with more ease and speed, and the frame rate was improved.

After color thresholding the calibration paper corners, we thought about using the k-means algorithm (with $k = 4$) to turn the four corner clusters of points into 4 precise corner points. However, the iterative nature of the k-means algorithm meant that it would slow down the frame rate for the calibration step. We came up with the assumption that each of the four point clusters falls into each of the four quadrants of the camera frame, which sped up our calculation a lot.

We believe that since we were able to explore different possible solutions and implementations and analyze the trade-offs and results of each method, and eventually implement color thresholding on top of homography transform for the whole system to work successfully in real time, which we believe constitutes a moderately complex technique, that we deserve at least 15 out of the 20 possible challenge and innovation points for this project.