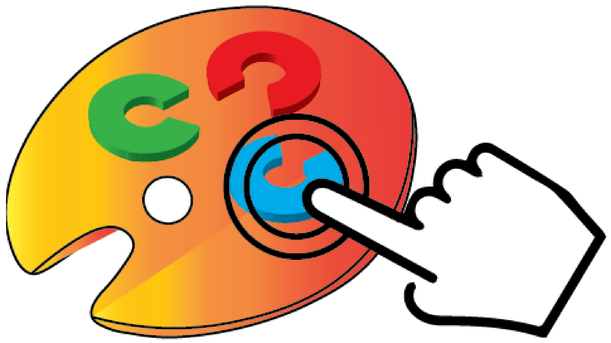
OpenCV Paint

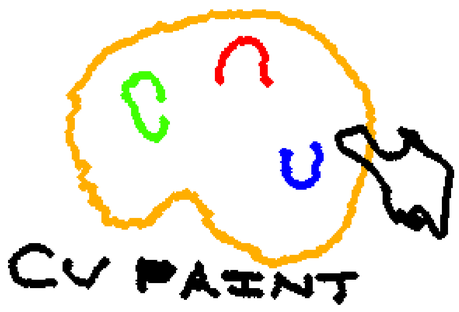
Rachel Won, Noah D' Souza, Chase Joyner
Olin College of Engineering | Spring 2018| Software Design



C.V. Paint

Project Details

The program uses a range of colors that it recognizes and finds in the camera view. In our case, the code will recognize the color green and focus on that point. Once locked on to the point, as you move the (green) object, your cursor will move too. As you move the cursor, a paint stroke will appear and you will have the option to pause the stroke, and be able to change the color of the brush. The code also allows you to save your masterpiece and crops out the color bar.



An accurate mock up of our logo using the CVPaint program.

ABSTRACT

Inspired by the infamous Microsoft Paint, we wanted to create a fun interactive computer vision program where you can draw a picture using your hand as a cursor. The program has the capabilities of changing the color of your cursor and have and eraser to delete and misguided/unintended strokes. You will have the function to save your creation afterwards!

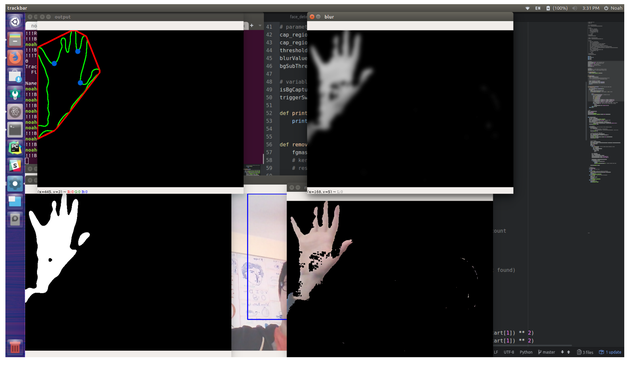
Program with the final color bar.

IMPLEMENTATION

We integrated pygame to implement a nice GUI. We tracked our points using a single list of nested tuples containing all the information, including x and y location, and mode (drawing or not). We decided to not split the model, view, and controller into different classes because all the variables were incredibly codependent on each other.

Issues/Challenges

One challenge we had was getting hand tracking to work. It detects and tracks the hand well, however it had issues where the mask began to fade out as the program was refreshing. It is likely refreshing to fast and is causing the mask to unload. We managed to fix this issue eventually, but not adequately enough to properly implement it into our code.



Why did we pivot?

Originally, we were going to use OpenCV to track human movement with an XBox Kinect. However, we had issues getting the pointcloud library to work on our computers. Without the pointcloud, would have a hard time utilizing the Kinect's skeletal tracking. Unfortunately, due to time constraints, we felt we should pivot to something more feasible. Chase is still working on PCL and got it to function. However, we eventually realized that in fact, concepts/parts of CVPuppets could be salvaged. With more time, we would have liked to fully integrate the Kinect motion tracking into our CVPaint code so the user could use their fingers to control the paint cursor and used the depth map to manipulate the thickness of the lines being drawn.

An depth image taken using OpenCV

A pointcould of the depth image.

Why OpenCV?

The purpose of the final project is to be a culmination of our learning, and to focus in on one topic the whole group is interersted in. Our group chose Computer Vision, which is the process of computationally analyzing and working with visual inputs from various image capturing tools.