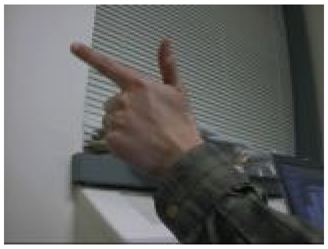
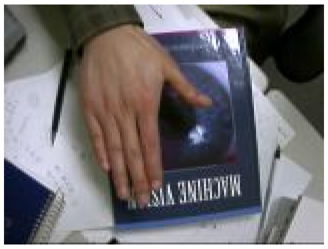
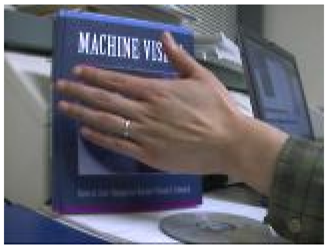
Kyle Qian

Histogram-Based Skin Color Detection

For this project we worked on implementing color image segmentation applying this principle to skin color to detect flesh from an image. The images we used as testing and are trying to segment are shown below:

From left to right, we will refer to the images as gun1, joy1, and pointer1. Outside of these testing images, I also found a training image online as well. Because I assumed that these hands were from professor Wu, I thought it only appropriate to choose a profile image of professor Wu as the training image, shown below:

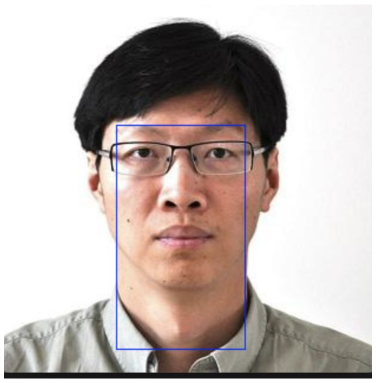
This will give us good training colors as all of the skin tones should come from the same person.

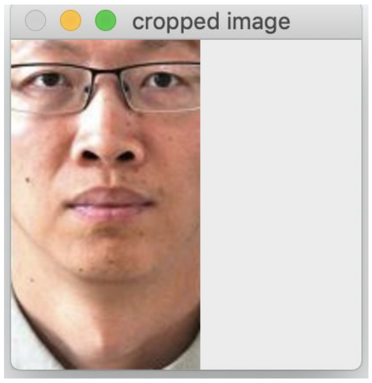
The Algorithm

The idea of this algorithm is quite similar to Histogram Equalization. First we collect training data by cropping some skin tones from an image, or by photographing just skin. Then we take this training data and map it to a color space of our choosing— either RGB, N-RGB, or HSI. After that, we train a 2D color histogram based on our training data. Finally, we test images by seeing if the color of each pixel is above a certain threshold within the histogram we developed from our training data.

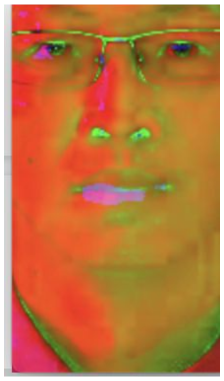
Implementation

In order to collect data, I wrote a script that would allow the user to crop a bounding box around the area of an input image to be used as training data. Clicking and dragging on opposite corners of a box will draw an outline to indicate to the user how their image will be cropped, as shown below:

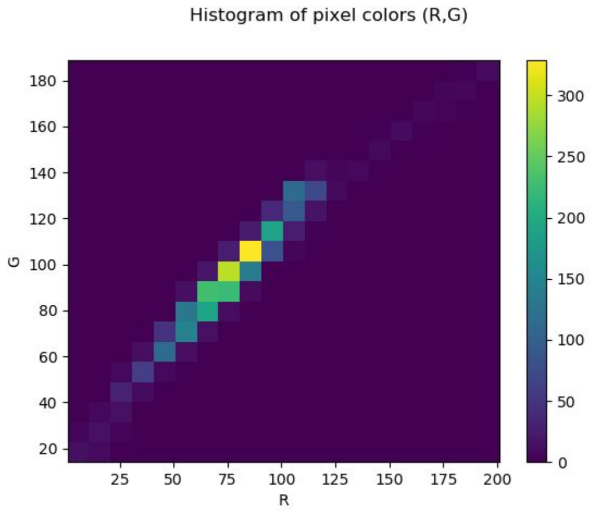


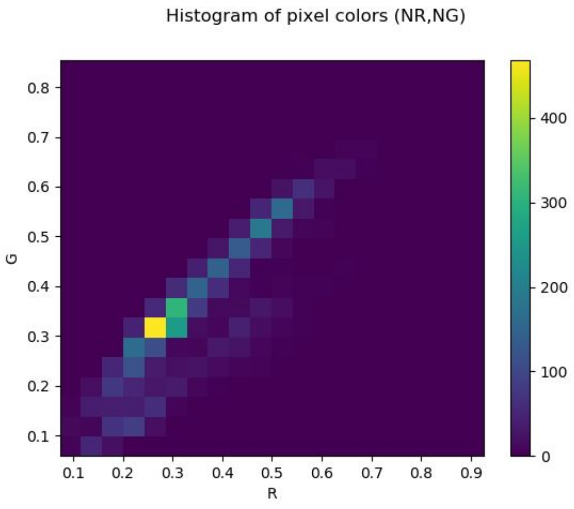


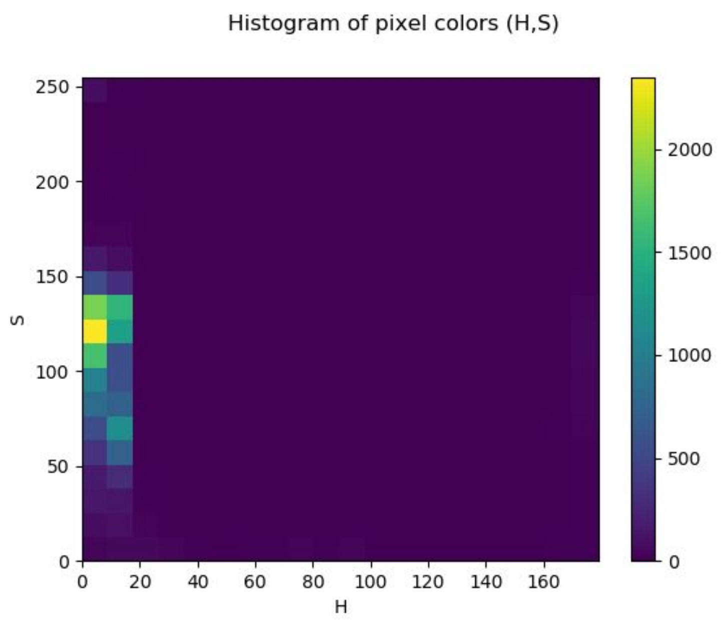
We can see that the image has been cropped by the blue bounding box, and this new region has become our training data. Next, we move our image into whatever color space we choose. Here is the HSV color space as an example:



Once that is done we can produce the 2D histogram based on our color space. This histogram will allow us to determine common skin colors. Here is the histograms for HSI, RGB, and N-RGB respectively:

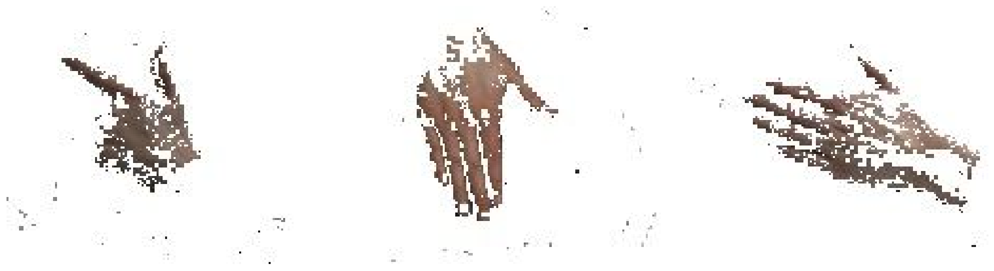






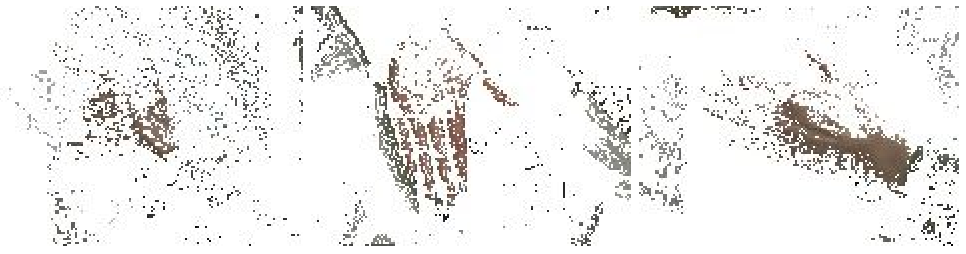
These histograms give us the frequency that each color combination occurs, depending on the color scheme. We then normalize this data from 0 to 1 so that we can create a threshold independent of the training data frequencies. Once this is completed, we can begin testing on our model. For each testing image, we will go through pixel by pixel. If the color coordinate from the pixel in the testing data is above a certain threshold in the histogram we created then we include it in the final image. Here are some of the results for multiple color spaces, with the ordering of images being gun1, joy1, and pointer1 respectively:

HSI color space



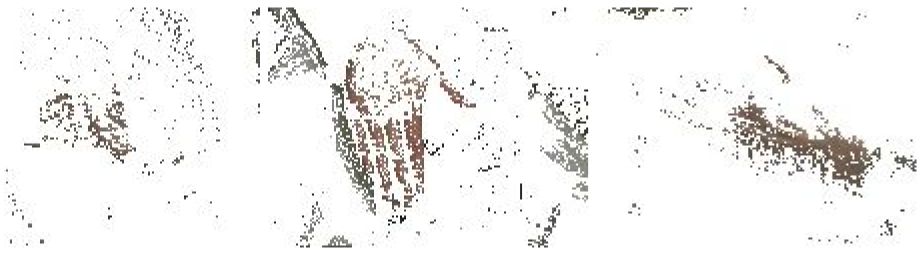
RGB color space

(trained with joy1 for gun1 and pointe1, and gun1 for joy1)



N-RGB color space

(trained with joy1 for gun1 and pointe1, and gun1 for joy1)



From these results, we can see that HSI color space led to the cleanest image segmentation. HSI was also the only color space that gave good results from the training image, as RGB and N-RGB basically did not work at all if trained with the training image that I used for HSI. To substitute this, I had to train the RGB and N-RGB images with the other training images—ensuring that the training and testing image weren't the same. There is significant noise on the RGB and N-RGB color spaces, with more noise coming from the RGB color space, while the HSI color space produced both more complete hands and less noise.