# EECS332 Digital Image Analysis
# MP4 – Mikhail Todes

My coding is done is c++. I complied it in a bash terminal using g++. The code I used to compile and create the executable was “g++ colourseg.cpp -o colourseg.e `pkg-config --cflags --libs opencv`”. I then ran the executable from the terminal using ./colourseg .e.

There are two functions in my code.

Mat Training (void);
Mat ColourSegment (Mat input, Mat histo);

The Training function does not take in anything as an input. It returns a 2D histogram created in a cv::Mat form. It also outputs this as an image saved as Histogram.bmp (attached).

The function runs through all the images in the training_data directory. This consists of images that have been cropped to show only the skin of a person. The three given images were used as well as a few that were found online and cropped. It checks the Hue and Saturation of each pixel and adds it to the bin. Hue is a value out of 180 and Saturation is out of 255 when using OpenCV and C++. This bin is then used to create a normalised histogram.

The ColourSegment function takes in the image to be segmented as well as the histogram that was built in the training function. It outputs the skin colour segmented image only.

Because the Training function outputs the histogram in a cv::Mat format, this function only ever needs to be run once everytime a new skin sample is added. From there on the ColourSegment function will just read the new Histogram.bmp file.

There are very few if any false positives due to the set tested set threshold comparison in the ColourSegment function. There are some false negatives, but these decrease when more training data images are added.

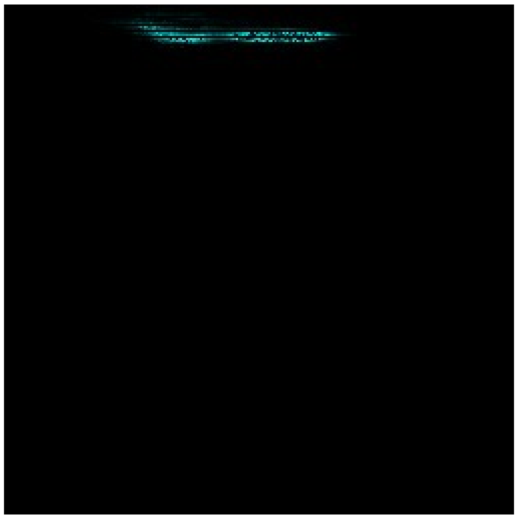
*Illustration 1: Histogram (Hue-Saturation) using the six training images*

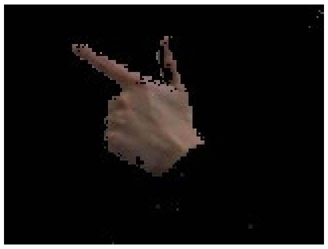
*Illustration 2: gun1.bmp after Colour Segmentation*

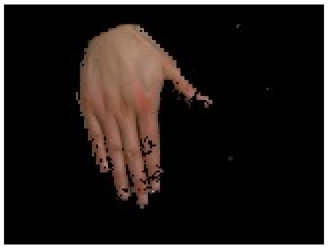
*Illustration 3: joy1.bmp after Colour Segmentation*

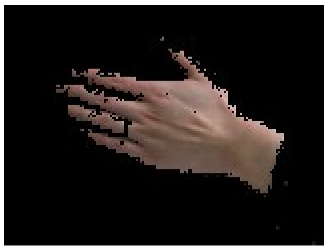
*Illustration 4: pointer1.bmp after Colour Segmentation*