This machine problem was to attempt to find the pixels in an image that are skin-colored. Please note that for whatever reason the results wound up displaying in black and white, where the white pixels are the skin pixels (mostly, there's also a little noise). Ten training images were created from the test images. The overall algorithm was divided into four functions, with a fifth for faster performance during experimentation with the proper boundaries to decide whether a pixel is skin-colored or not. There was a function RGBtoHSI to convert between color schemes (but the inverse was not needed and so not created). There was a function to create the 2D histogram for H and S. There was a function to train a model (which was just an amalgamation of the ten histograms generated by the training images, normalized). Then, the Limited_Scope_Skin_Color_Detector (named so because it had access to limited training data and so can't detect any skin tones other than those present in the test image uses the above functions to identify which pixels are pertinent. LSSCD2 is almost the same function, except by taking the model as an extra input, execution time decreased dramatically since all that was left to do was iterate over the image and figure out which pixels fit the model.

RGBtoHSI was relatively simple, coming almost straight from the equations on the slides. The important part of the 2D histogram function was that I realized that if both histograms were row vectors, then by transposing the H vector and doing matrix multiplication (H*S) I could get a matrix that held the probability of a given combination of H and S appearing in an image. Training as stated before simply creates 2D histograms by calling the function, and then adds it to its own histogram called "model" which is then normalized at the end. Another important feature of the Training function is that it can read through a bunch of images of the format "TrainingX.jpg" where X is a number between 1 and 10 inclusive. (This choice was arbitrary, I just decided to create ten training images is all, this could just as easily be 5 or 100 or 2.6 million.)

As for the identifier functions, all they do is go pixel-by-pixel, convert to HSI, and then check to see if said pixel falls within certain bounds determined experimentally according to the model. Those bounds involve both having a value above a threshold in the model and being within a certain distance from the max of the model (which was located at [1,1]). The latter was added to get rid of noise from white surfaces, such as the AC in 'gun1.bmp'.

As a closing note, I'd like to recommend that if you intend to do testing with multiple other images, you first run Training and store the result into a variable, and then run LSSCD2 with each image and model as the output. You could even set up iteration for this part, I'm just recommending this to save on speed because Limited_Scope_Skin_Color_Detector calls Training every single time, that's the only difference between that and LSSCD2 but it does translate into a good 30+ seconds each time Training is called. I'm also uploading the training images so that Training will actually run, though those images need to be put into the same directory as the code.