Tim Herrmann

EECS 332

MP4

Skin Color Detection

Programming Language:

- Python 2.7.6 with the following dependencies
  - matplotlib
  - numpy
  - cv2
  - pylab

Input:

skintone.py – a series of images cropped to only contain flesh tones.
skindetect.py – a color image followed by the outputfile from the skintone.py "training" function

Output:

skintone.py – a 100x100 grayscale image. Each pixel represents a bin from the 2d skin tone histogram. Bins that are above the specified threshold are white and bins below the threshold are black
skindetect.py – a masked image showing only the pixels that matched the "training" function's histogram

Method:

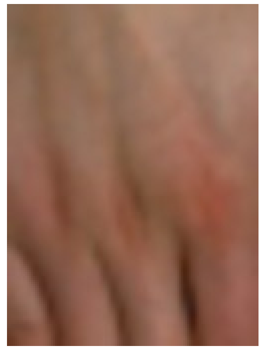
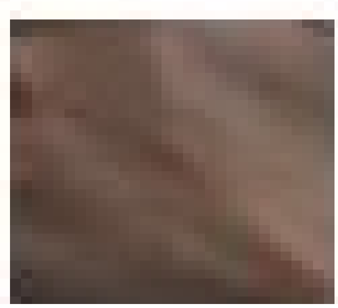
skintone.py
Step 1: read in an image
Step 2: append each pixel value from the image to a list
Step 3: loop through steps 1 and 2 for all images in the input list, creating a single master list of pixels
Step 4: convert each pixel in the master list to normalized BGR form
Step 5: divide each pixel value by the total number of pixels to create a normalized histogram where all the bins sum up to 1
Step 6: Plot a 2d histogram with 10000 bins set at .01 increments along each axis
Step 7: Threshold the histogram at a pre-determined value (after trying multiple, I chose .015. This represents bins containing at least 1.5% of the pixels.
Step 8: Output an image the same size as the histogram where all bins above the threshold are colored white and those below are black

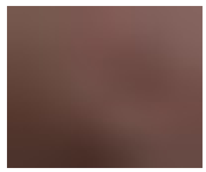
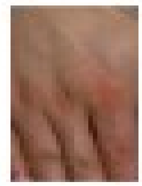
skindetect.py
Step 1: read in both the output from skintone.py and the image to be analyzed.
Step 2: loop through each pixel of the image to be analyzed, calculating its normalized BGR value.
Step 3: identify the corresponding histogram bin by multiplying the nBGR value by 100 and rounding to the nearest int. Check against the corresponding bin in the skintone.py image
Step 4: If the corresponding pixel is black, make the output pixel black as well, effectively masking the pixels that don't match the trained histogram
Step 5: Output the masked image.

<u>Examples and Analysis:</u>

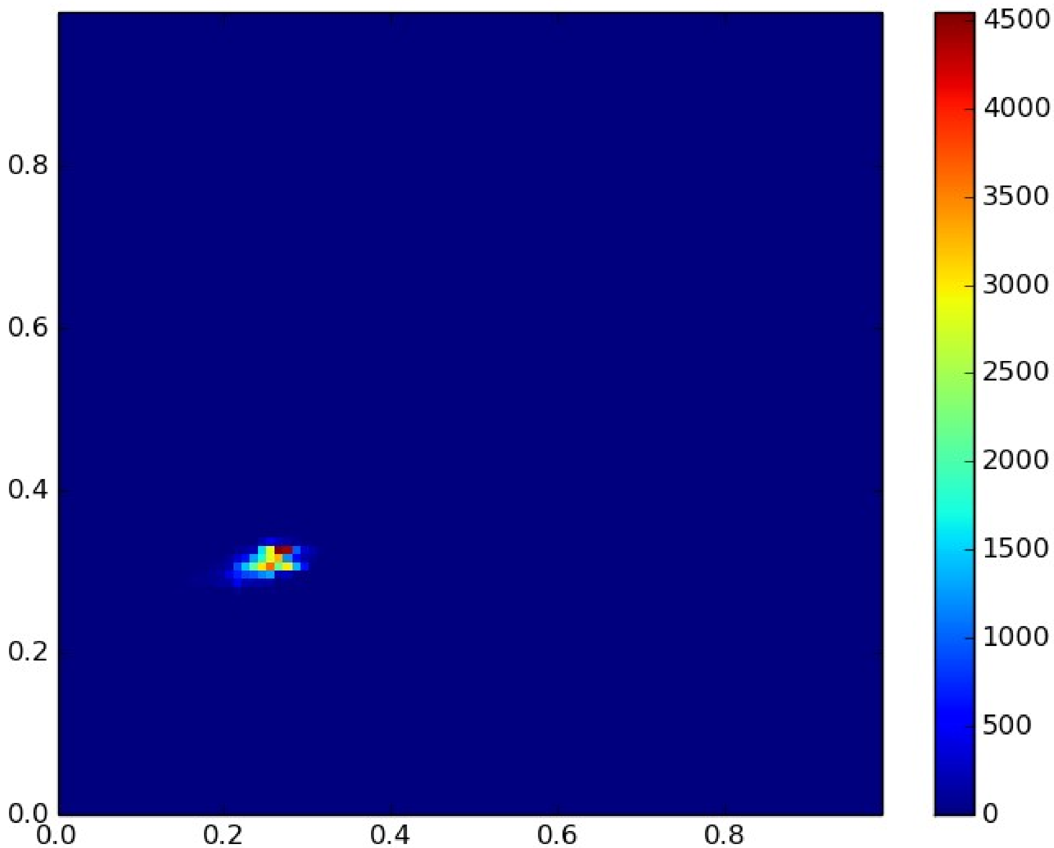
Sample training images



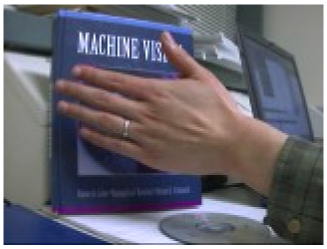



Output Histogram – The data gives a very localized cluster of thresholded values which indicate the region that skin tones exist.



Examples

Original Image	Filtered Image
	
	
	
	
	

Analysis

As expected, the algorithm performs best on the images that were part of the original “training” data, but it also performs reasonably well on other clear images. Where it started struggle a bit was with the final image above with the girl running outdoors. The multitude of neutral nature colors in the background sometimes get picked up as false positives on the skin detect. Additional sampling and “training” would certainly help increase the accuracy. In all tested images, it was clear where the bulk of the skin in the picture resided.