Fingernail Painter

Hao Yee, Chan
Department of Computer Science
Stanford University
Stanford, CA ,USA

Nicholas Yap
Department of Electrical Engineering
Stanford University
Stanford, CA, USA

Abstract—**This paper describes a real-time fingertip detection algorithm. We also attempted to detect fingernails and paint it with a specified color. We describes the various challenges faced while trying to do fingernail detection.**

Keywords-fingernail detection, fingertip detection

I. INTRODUCTION

A. Motivation

When purchasing nail polish, customers often have to assess how the nail polish color matches their skin tone under various lighting conditions, before deciding if a nail polish is suitable for purchase. However, currently, customers typically have to either visualize the nail polish on their nails or actually try painting a portion of their nail to see how it looks like, which can be troublesome.

Hence, we aim to develop a mobile application on the android platform that users can utilize to simulate how a nail polish would look on themselves. The mobile application would be video-based and real-time, only utilizing the resources available on the phone to detect the fingernail regions quickly and apply the desired color. In the application, we first detect the user's hand and isolate the fingertip regions. We then detect the fingernail before applying the color on the fingernail.

The remainder of the report is organized into the following sections. In Section II, we examine prior work on skin, hand, fingertip and fingernail detection. In Section III, we describe the implementation of our application with our proposed system. In Section IV, we present and discuss our results before concluding the report in Section V.

II. PRIOR AND RELATED WORK

A. Skin Detection

Skin detection has been done using a variety of methods in different color spaces such as RGB, normalized RGB, YUV, YCrCb, HSV, TSL and CIE-*lab*. Often, methods aim to separate chrominance from luminance to increase invariance against illuminations effects with each color space having its own advantages and disadvantages. Methods include simple bounds within a color space, building up skin models, such as training color histograms, and classifying them based on statistical methods, such as the Gaussian classifier[1-3]. The main problems faced in color segmentation as a way of detecting skin is distinguishing against objects with a color distribution similar to human skin [1]. While most of these techniques are pixel based, there is ongoing work on regions of pixels.

B. Hand Detection

Work on hand detection normally begins with segmentation by skin color, after which much work seems to be focused on identifying the shape of the hand through template matching, often using the contours of the hand or other hand models (such as active shape models). [1,4-8].

C. Fingertip Detection

Methods to detect fingertips typically used the k-curvature method first suggested by Segen [9] or based on template matching, which is typically computationally expensive [1,10-11].

D. Fingernail Detection

Most work in this area have a controlled experimental setup, with fixed object to camera distances, high resolution pictures and controlled lighting. Most methods revolve around using an edge detector, typically the Canny edge detector, or by feature matching[11-13].

III. DESIGN OF APPLICATION

A. User interface

On loading the application, the user will see the current scene in the viewfinder. It starts up in the "Choose Color" Mode. By tapping on an area, the user will select the color of the tapped surface as the desired fingernail color.

Next, the user will choose "Color Nail" from the popup menu. By viewing his extended hands, the application will detect the fingertips and overlay the chosen fingernail color over the fingernail. If the overlay is not accurate, the user is able to recalibrate the application by tapping on the image of his hand in the viewfinder. The application should then adjust the settings to do detection more robustly. Currently, the application is limited to an extended hand with palm facing down and will not work on a fist.

The user is able to select a new fingernail color at any time by choosing the "Choose Color" option in the menu (See Fig. 1) An additional functionality, which has yet to be added, is including a color palette that is based on information supplied by nail polish suppliers.

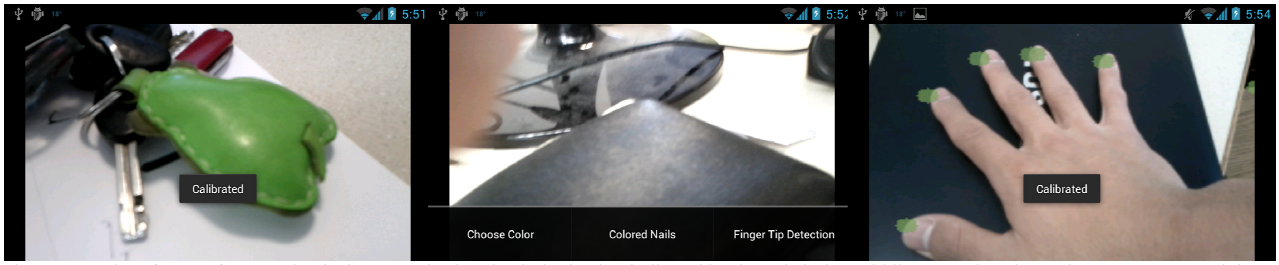

Fig. 1. User interface: Left screenshot is the user selecting the desired color, indicated by the red circle.. Middle screenshot shows the pop-up menu. Right screenshot shoes the nails being colored. Note that regions are off, which will be discussed in later sections.

B. *System Overview*

Each frame of the video is first converted to RGB color as there is no function to convert from YUV space, which is the color space the camera is in, directly to HSV. We then scaled the RGB frame down by 4 on both axes to reduce pixels. Each frame is then first passed into our fingernail detection algorithm and the found fingernail regions would then be passed to a coloring function. Ideally, the coloring function will perform a weighted combination of the original frame with the color in the fingernail region. The fingernail detection algorithm was designed with computational efficiency and speed in mind, as we wanted the application to run on the phone itself, without a need to rely on external servers. The application was written in Java, using the OpenCV library. See Fig. 2 for a pictorial representation of the flow. All systems were first tested on MATLAB before implementation in OpenCV.

C. *Fingernail detection algorithm*

There are 3 main parts to the fingernail detection algorithm. First, we detect the hand by looking for skin colored regions. Second, we look for the fingertips. Third, we window about the fingertips and look for the fingernails. Each step is described in more detail below. (Fig. 2)

1) *Hand Detection*

We aim to detect the hand via color segmentation. Techniques that used template matching or fitting, such as active shape models, were deliberately avoided to reduce computational complexity. The next issue was to decide on which color space to work in. We decided to use the perceptual based color space of HSV for its robustness against a variety of illumination [1]. We looked at chromaticity information captured in the *Hue* and *Saturation* values, which defines the dominant color of an area and the colorfulness of an area respectively [14]. The *Value* values which represent color luminance were ignored. We then did a pixel-based segmentation approach, deciding if a pixel is a skin or non-skin pixels if $0^o \leq H < 50^o$ and $0.23 \leq S < 0.68$ [15]. An additional calibration functionality was added to allow for cases where the user has a skin color that is not easily detected. Here, the user simply taps the region where his/her hand is, and we grab the value $(H_o S_o V_o)$ where the user tapped. We then modify the decision criteria to $H_o - 25^o \leq H\ H_o + 25^o$; $S_o - 0.196 \leq S < S_o + 0.196$.

After determining the skin and non-skin pixels, we erode by a 5 by 5 square window to remove noise. We assume that the hand will be the largest skin-colored region in the frame as the user would be using our application to look at his/her hands. Therefore, we select the largest region of skin-colored pixels and take the outline of the region. At this point, we have both the binary mask and outline of the hand

2) *Fingertip detection*

Fingertip detection was meant as a way to create a window to find the fingernail region in our next step. Again, we avoided using template matching as it was computationally more intensive. Thus, to find the fingertips, we modify the k-curvature method introduced by Segen [9]. We first find the convex hull of the outline of the hand and find points where the hull and the outline intersect. This is done to reduce the set of points where we have to find the curvature. For each point in this reduced set of points, with coordinates (x_m, y_m) we look at a fixed number of steps (k) forward along the outline to obtain another point (x_{m+k}, y_{m+k}) and do the same thing in the opposite direction along the outline, getting (x_{m-k}, y_{m-k}) Based on the spatial location of the points, we are then able to get the k-curvature which is defined as the angle between vectors $[x_{m+k} - x_m, y_{m-} y_{m+k}]$ and $[x_{m+k} - x_m, y_{m-} y_{m+k}]$. If the angle is smaller than a threshold, we determine that the point is a fingertip (See Fig. 3).

3) *Fingernail detection*

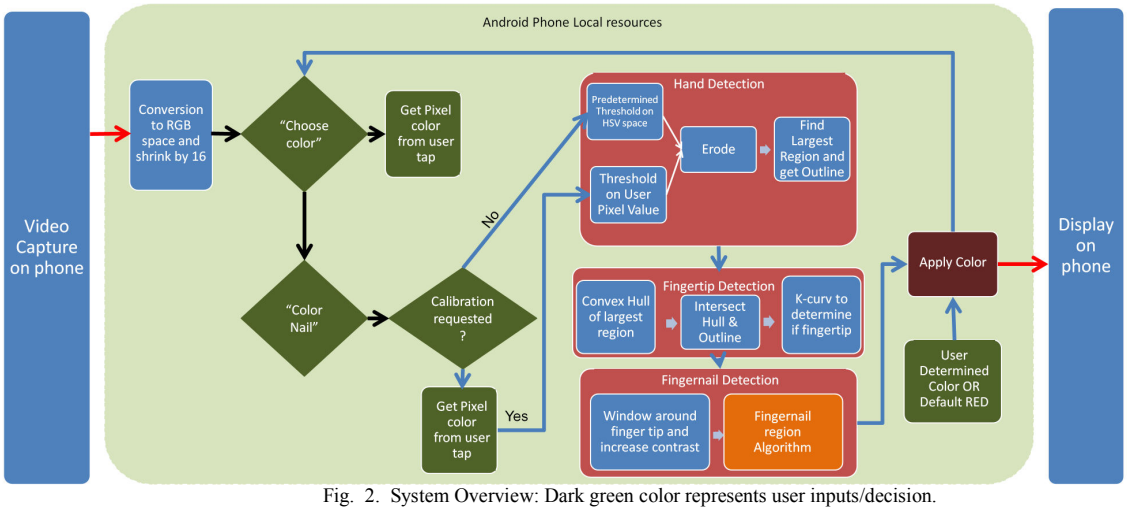

Fig. 2. System Overview: Dark green color represents user inputs/decision.

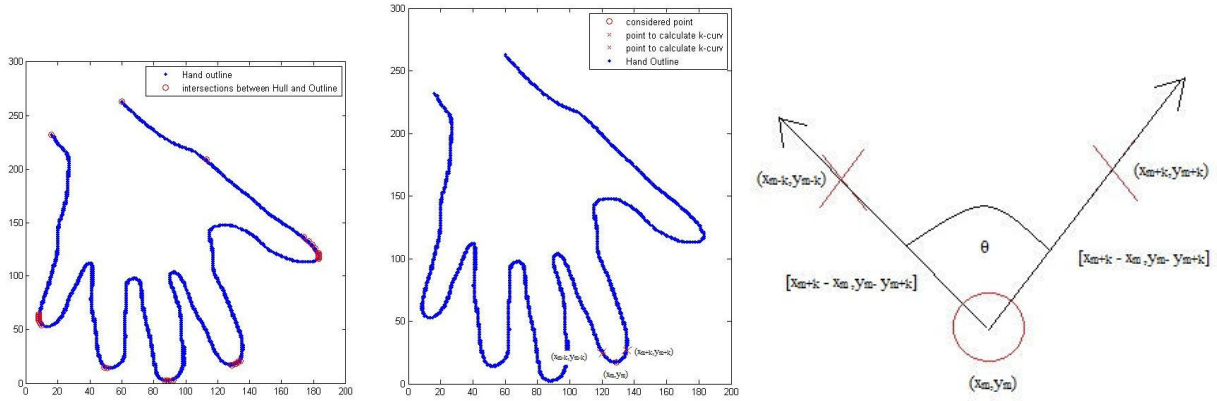

Fig. 3. Illustration of fingertip detection. Figure on the left shows the intersection between the hull and the outline. The center figure shows a considered point and the points used to calculate the k-curvature. The figure on the right shows the considered angle.

To detect the fingernail, we utilize information about the fingertip and consider a window around the fingertip. Within this window, we approach the problem of detecting the fingernail in two ways.

The first approach uses edge detection. Visually, the edges of the fingernail is rather obvious, hence we thought that edge detection would have a high success rate. Furthermore, previous work by Yu have successfully identified the fingernail region through canny edge detection followed by cubic b-spline [12]. Hence, we first increased contrast through local adaptive thresholding and then try a variety of edge detection methods including canny, sobel, prewitt, roberts and laplacian. We tried a number of ways of applying edge detection, which include apply the edge detection on the 3 channels in HSV and RGB separately before performing an "OR" or "AND", performing edge detection on images with and without local adaptive thresholding. Details are described in Table I. We observe varying degrees of success which will be discussed in the next section.

The second approach uses Maximally Stable Extremal Regions (MSERs), which extracts from an image a number of co-variant regions called MSERs, which are stable connected components of some level sets of the image. [16]. We hope that subtle differences in color can be detected via MSER, as the fingernail color is often slightly different from skin color. We also tested MSER out on nails that are already painted, which should give a better MSER result.

D. *Color Application*

To apply color, we either use a default red color or the color which the user indicated earlier. We perform a weighted combination of the original image and the color, with the majority of the weight being applied on the fingernail color. In other words, the fingernail region in the output image would be (α x original fingernail region) + {(1-α) x applied color}. This is done so that the applied color would look have a more natural blend with the finger. We hoped that illumination effects, such as shine would also be seen through this weighting process.

IV. Results and Discussion

Fig. 4 shows the screen shots from various stages of the application till the fingertip detection. Fingernail detection is skipped as explained below in C, and we arbitrarily fill a colored region.

A. *Hand Detection*

When tested with MATLAB, skin detection using simple boundaries on HSV works rather well under a variety of lighting conditions. In cases, where the pre-determined settings do not seem to work well, the calibration step often solves the problem. We note that if there are skin toned objects in the background or the lighting that makes the background objects apply skin-like, the skin detections does not work very well, even with calibration. See Fig. 5 for some of the results. The same effects were observed when hand detection was tested on the phone itself.

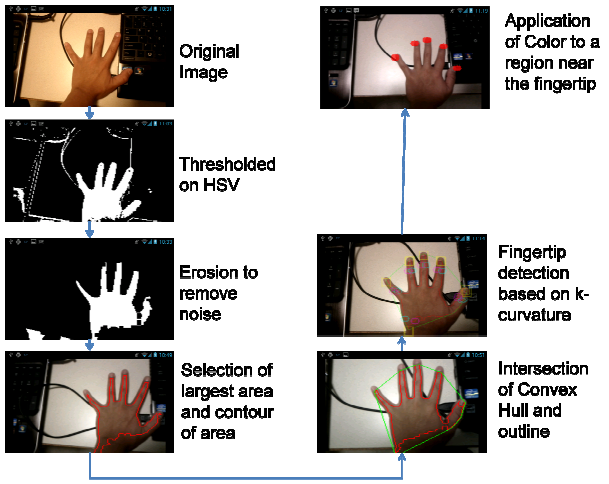

Fig. 4. Screen shot of results of the application at various stages. Reminder that fingernail region has been arbitrarily set to test out color filling function. The red outline represents the contour. The green line represents the convex hull. The pink and cyan circles are the points used in the calculation of the k-curvature and the yellow bounding box shows the location of the fingertips.

Fig. 5. MATLAB results from hand detection. Each set of three images are of the same hand taken under various backgrounds and lighting conditions. For each set, the picture on the left is the original image, the center is the masked image based on pre-determined HSV threshold values and the picture on the right is the masked image using user-input calibration. The top left and right sets of images are taken with simple backgrounds and reasonably good white lights. The bottom left set of images are taken with simple background but orange light. The bottom right set of images are taken with complex background and relatively good lighting conditions.

B. *Fingertip Detection*

We observe a slight difference in the convex hull/ contour function in MATLAB and OpenCV. MATLAB seems to be capturing many more points with the intersection of the hull and contour/outline. The outlines are also more jagged resulting in more false positives like those shown in Fig. 6. These however, do not happen in our application implementation, and there is no requirement of smoothening of the contour, which helps to speed things up.

C. *Fingernail Detection*

Results from edge detection were varied from sample image to sample image, with different set of images requiring edge parameters tunes specifically to them. We also note that different set of images have different performances when we use the different methods described in Table. I(See Fig. 7) .

We have only shown some of our results in this paper, and interested parties can contact us to see more results. For one set of images, that created a lot of edges in areas other than the fingernail region, we are able to get a relatively good estimation of the fingernail region, by first running an edge detector on each of the HSV channels before performing an "AND" operation on all three channels. We then add up the number of edges in a window centered about a pixel. If the total number of edges is more than a threshold, we treat that pixel as the background (or skin regions). Fig. 8 shows the results, on one set of windows from one hand image, using the above described algorithm. The parameters are optimized for each type of edge detector, and we observe that the canny edge filter seems to do a better job. This is observed in other sets of images where an optimized canny edge detector detects the nail better than other edge detection techniques.

Results with MSER in a grayscale image are generally not good with unpainted fingernails. Much detail is lost when the image is converted to gray scale. However, they show much promise for painted fingernails(See Fig. 9). This is a logical as painted nails tend to be more consistent within the fingernail region and are visually different from the

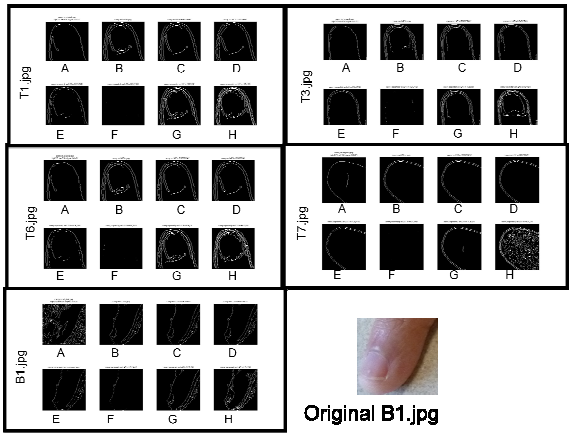

Fig. 7.Peformances of edge detectors using Canny edge with methods described in Table. I. We note that while method H works well for T1, T3 and T6, method H works better for B1, while no method really works for T7.

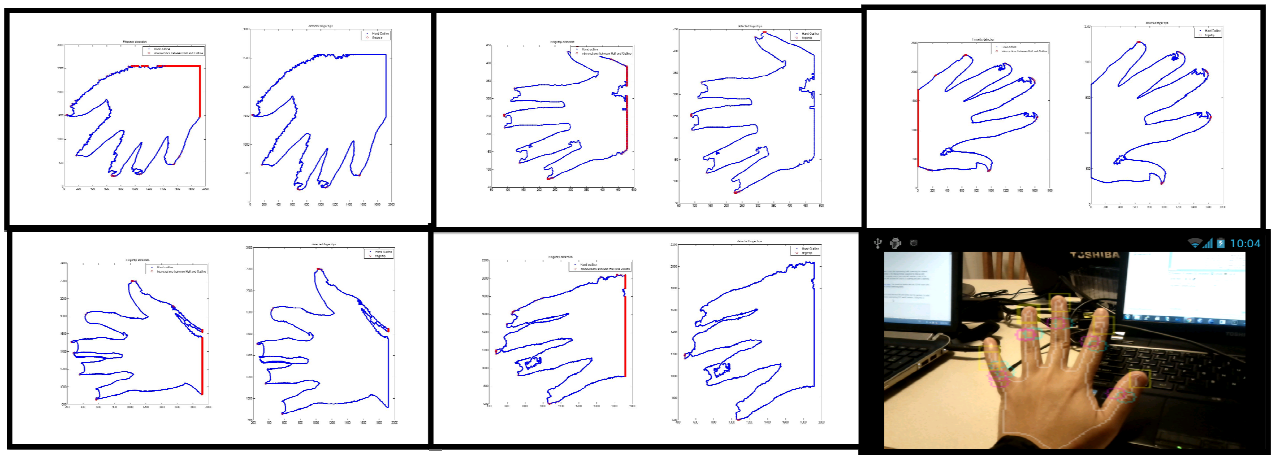

Fig. 6. Fingertip detection results. Each set of images bounded by a black box come from the same hand image. The left figure is the intersection of the hull and the outline, while the right figure is the fingertips determined by k-curvature. The bottom right picture is a screen shot of fingertip detection, with the points used to calculate k-curvature shown in pink and cyan, and the fingertip point bounded by a yellow box. We note that in MATLAB, cases where multiple figure tips are detected in the same region occur rather frequently, but this happens with less frequency in our application.

TABLE I. WAYS OF APPLYING EDGE DETECTION

Methods	Steps			
	1	*2*	*3*	*4*
A	Convert RGB to gray scale	Edge detection		
B	Convert RGB to grayscale	Local adaptive threshold	Edge detection	
C	Local Adaptive threshold on each channel in RGB	Convert RGB to gray scale	Edge detection	
D	Convert RGB to HSV.	Local adaptive threshold on each channel in HSV	Convert from HSV to gray scale	Edge detection
E	Local Adaptive threshold on each channel in RGB	Edge detection on each channel in RGB	"AND" results from all 3 channels	
F	Convert RGB to HSV.	Local Adaptive threshold on each channel in HSV	Edge detection on each channel in HSV	"AND" results from all 3 channels
G	Local Adaptive threshold on each channel in RGB	Edge detection on each channel in RGB	"OR" results from all 3 channels	
H	Convert RGB to HSV.	Local Adaptive threshold on each channel in HSV	Edge detection on each channel in HSV	"OR" results from all 3 channels

surrounding areas. However, it is noted that reflections may affect MSER. Also, we observe the same problem of having to tune the MSER parameters for a different set of images. We then tried examining MSER in each channel of the HSV color space, however, the results vary from finger to finger.

D. Color Application

Due to our inability to accurately determine fingernail detection and in order to test our proof of concept, we draw an ellipse based on the detected fingertip location. As we had not accounted for multiple points, sometimes we observe multiple ellipses as indicated by Fig. 10. The color application is passable, but we feel that it can be done more realistically.

V. CONCLUSION

We have managed to achieve a real-time detection of fingertips. However, the wide ranging conditions available, make detecting the fingernail region very difficult. Hence, more research has to be done to fine tune and test out algorithms for fingernail detection.

A. Future Work

With regards to the skin detection, a better and still rather quick method to use could be to build up statistical skin models through training. With regards, to color space choice, while different papers have differing views on the appropriate choice of color spaces, we will continue to use HSV color space, which has worked reasonably well, till there is a more compelling reason to switch to another color space. Also, should there be more computational power available on the phone, work can be done to develop a way of quickly fitting active shape models to the detected skin regions to improve hand detection and hence differentiate the background from the region of interest. However, it is important to note that since the focus here is in detecting the fingernail regions on an extended hand, getting a very accurate mask is not that important as we would later be using the convex hull to obtain possible fingertip locations

Work can be done to improve the algorithm should there be a cluster of probable fingertip locations in the same space, which could happen if the fingernail is not detected as a skin, while the sides of the finger nail are, creating a dip, which typically shows up as two points as shown previously.

Tremendous amounts of work can still be done in terms of fingernail detection. As observed above, it seems that parameters for edge detection have to be tuned rather specifically for a nail. Work has to be done to come up with an adaptive algorithm that dynamically sets parameter according to the windowed finger. More research has to be done on which color space is generally better for edge detection (current results seem to vary). MSER works largely with colored nails, hence we could incorporate this into our application, applying MSER if we somehow determine that the nails are already painted, else use edge detection or other methods to detect the fingernail.

With regards to color application, more research has to be done in to applying the color more realistically, perhaps through meshing or other methods.

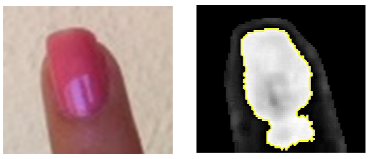

Fig. 9.MSER on painted fingernail in HSV space

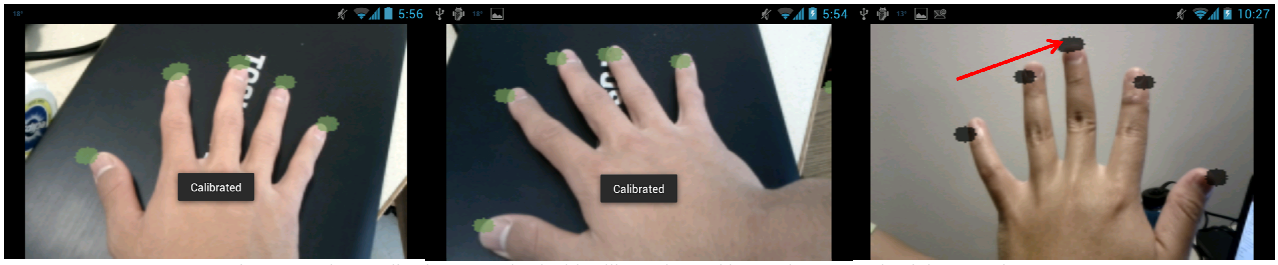

Fig. 10. Color Application. Note the double ellipses denoted by a red arrow on the right most picture.

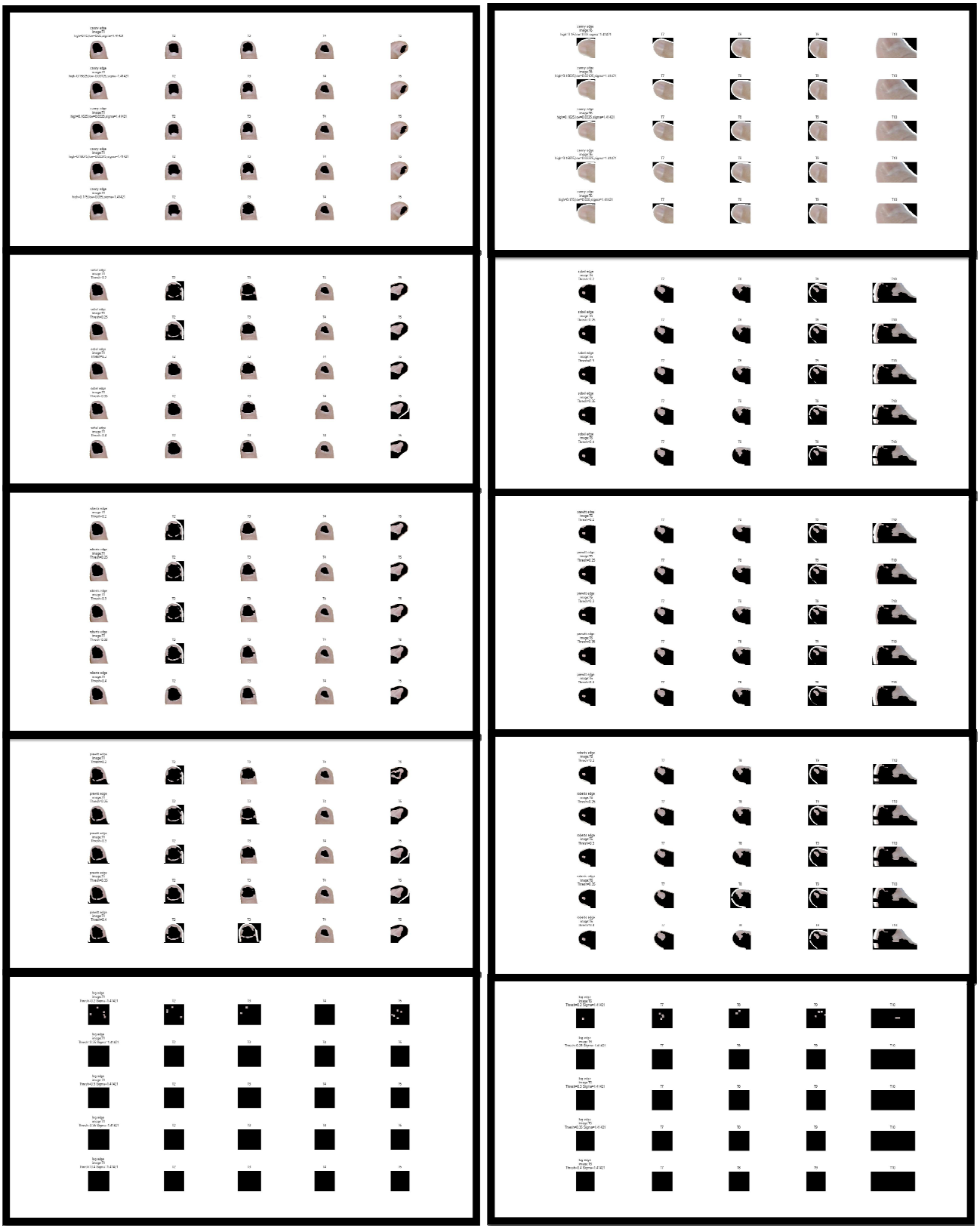

Fig. 8. Fingernail detection using described algorithm. The black regions indicate where our fingernail detection algorithm would fill. Each black box represents an edge detection, with order from top to bottom: Canny, Sobel, Roberts, Prewitt and Laplacian. Each column in the box represents a windowed finger from the hand image. Each row varies the respective parameters of the edge detector. The left set of black boxes are for one hand image, and the right set of black boxes are for another hand image, with each row having the same perimeters. We note, that in general thumbs do not work well, and that canny edge seems to work best for the other 4 thumbs. The key takeaway is that this algorithm does not work well on all images.

Acknowledgment

We would like to thank our mentor, Andre Araujo, and Huizhong Chen for their helpful advice and support. Special thanks to David Chen for his lightning-speed responses to questions and for being so encouraging!

REFERENCES

[1] X.Zabulis, H.Baltzakist, A.Argyros;"Vision-based Hand Gesture Recognition for Human-Computer Interaction.

[2] J. Terrillon, M. Shirazi, H. Fukamachi, S. Akamatsu. "Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human facesin color images" In Proc. International Conference on Automatic Face and Gesture Recognition (FG), 2000.

[3] M. J. Jones and J. M. Rehg. "Statistical Color Models with Application to Skin Detection." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999

[4] M.Kolsch and M.Turk."Robust Hand Detection"

[5] C.-C . Chang, J.-J. Chen, W.-K. Tai, and C.-C. Han;"New approach for static gesture recognition" Journal ofInformation Science and Engineering, 22(5):1047-1057,September 2006.

[6] H.Koike et.al "Integrating Paper and Digital Information on EnhancedDesk: A Method for Realtime Finger Tracking on an Augmented Desk System". ACM Transactions on Computer-Human interaction Dec 2001

[7] J Zhang et.al "A Fast Algorithm for Hand Gesture Recognition Using Relief". Sixth International Conference on Fuzzy Systems and Knowledge discovery, 2009.

[8] Y.Fang et.al "A real-time hand gesture recognition method". IEEE 2007

[9] Segen, J. & Kumar, S. Human-computer interaction using gesture recognition and 3D hand tracking. *Proceedings 1998 International Conference on Image Processing ICIP98 Cat No98CB36269* **3**, 188-192 (1998).

[10] Oka, K., Sato, Y., & Koike, H. (2002). Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems. In *Proceedings of IEEE Conference on Automatic Face and Gesture Recognition (FG)*. pp. 429-434.

[11] Yu Sun; Hollerbach, J.M.; Mascaro, S.A.; , "Predicting Fingertip Forces by Imaging Coloration Changes in the Fingernail and Surrounding Skin," *Biomedical Engineering, IEEE Transactions on* , vol.55, no.10, pp.2363-2371, Oct. 2008 doi: 10.1109/TBME.2008.925691

[12] Sun, Yu; Hollerbach, John M.; Mascaro, Stephen A.; , "Finger Force Direction Recognition by Principal Component Analysis of Fingernail Coloration Pattern," *EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint* , vol., no., pp.90-95, 22-24 March 2007 doi: 10.1109/WHC.2007.53

[13] Sugita, N.; Iwai, D.; Sato, K.; , "Touch sensing by image analysis of fingernail," *SICE Annual Conference, 2008* , vol., no., pp.1520-1525, 20-22 Aug. 2008 doi: 10.1109/SICE.2008.4654901

[14] Vezhnevets V., Andreeva A., "A Comparative Assessment of Pixel-based Skin Detection Methods". Technical report, Graphics and Media Lab., Moscow State University, Moscow, Russia, 2006

[15] Oliveira, Conci. "Skin Detection Using HSV Color Space". Computation Institute, Universidade Federal Fluminense, Niterói, Brazil.

[16] Tutorials on MSER on VLFeat.org. http://www.vlfeat.org/overview/mser.html

APPENDIX-WORK DISTRIBUTION

Hao Yee did the programming in C while Nicholas did the programming in MATLAB. However, discussion, evaluation and formulation of all processes were done together. We worked on the poster and report together.