**MOD002643**

**A System for the Recognition of Hand Systems**

**IMAGE PROCESSING**

**Name: James Wong**

**SID: 1507313**

**Date: 18/12/2018**

Contents

[Section A 3](#_Toc533151775)

[Task A1 3](#_Toc533151776)

[Task A2 4](#_Toc533151777)

[Task A3 7](#_Toc533151778)

[Task A4 10](#_Toc533151779)

[Task A5 13](#_Toc533151780)

[Task A6 17](#_Toc533151781)

[Section B 21](#_Toc533151782)

[Task B1 21](#_Toc533151783)

[Task B2 29](#_Toc533151784)

# Section A

## Task A1

clear all ;

fileName = uigetfile('palmdown.png');

palmdownImageFile = imread(fileName);

%Converting RGB Image to GrayScale.

palmdownGreyScale = rgb2gray(palmdownImageFile);

%Displaying images side by side.

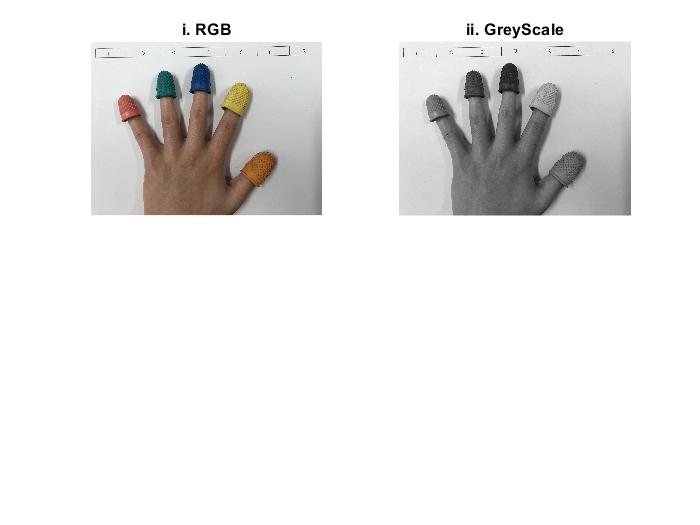
subplot(2,2,1);

imshow(palmdownImageFile);colormap gray;axis image;title('i. RGB');

subplot(2,2,2);

imshow(palmdownGreyScale);colormap gray;axis image;title('ii. GreyScale');

%State the equation and the name

**This figure shows the output from Task A1. The original image loaded into MATLAB and a greyscale version.**

It converts the truecolor images to grayscale by eliminating the hue and saturation information while retaining the luminance. It is called luminance. It converts the RGB values to grayscale values by forming a weighted sum of the R, G and B components. 0.2989 \* R + 0.5870 \* G + 0.1140 \* B .Uploading an image to MATLAB and producing a greyscale version of it worked perfectly.

## Task A2

close all;

clc;

fileName = uigetfile('palmdown.png');

palmdownImageFile = imread(fileName);

%Converting from rgb to greyscale.

palmdownGreyScale = rgb2gray(palmdownImageFile);

palmdownGreyScaleHistogram = imhist(palmdownGreyScale);

%Splitting up histogram with RGB components so there are three coloured

%lines for the intensity.

R = palmdownImageFile(:,:,1);

G = palmdownImageFile(:,:,2);

B = palmdownImageFile(:,:,3);

RedChannelHist = imhist(R);

GreenChannelHist = imhist(G);

BlueChannelHist = imhist(B);

%ArithmeticMean of each individual RGB.

greyScaleArithmeticMean = mean2(palmdownGreyScale);

rArithmeticMean = mean2(R);

gArithmeticMean = mean2(G);

bArithmeticMean = mean2(B);

figure;

subplot(1,2,1);

plot(0:255, RedChannelHist, 'r','LineWidth', 1);hold on

plot(0:255, GreenChannelHist, 'g','LineWidth', 1);hold on

plot(0:255, BlueChannelHist, 'b','LineWidth', 1);

%Labelling of X axis

xlabel('Intensity', 'FontSize', 14); ylabel('Normalised Count','FontSize',14);

xlim([0 255]);

rgbHistYLength = get(gca,'YLim');

%Show vertical dotted bars down as Artihmetic Means.

ylim([0 max(RedChannelHist)]);

line([rArithmeticMean,rArithmeticMean],rgbHistYLength,'Color','r','LineWidth', 1,'LineStyle', '--');hold on

ylim([0 max(GreenChannelHist)]);

line([gArithmeticMean,gArithmeticMean],rgbHistYLength,'Color','g','LineWidth', 1,'LineStyle', '--');hold on

ylim([0 max(BlueChannelHist)]);

line([bArithmeticMean,bArithmeticMean],rgbHistYLength,'Color','b','LineWidth', 1,'LineStyle', '--');

%Displaying the Histogram showing Lumninance with only one arithmetic mean

%= GreyScale

subplot(1,2,2);

plot(0:255, palmdownGreyScaleHistogram, 'black','LineWidth', 1);

%Labelling of Y axis.

xlabel('Luminance', 'FontSize', 14);ylabel('Normalised Count','FontSize',14);

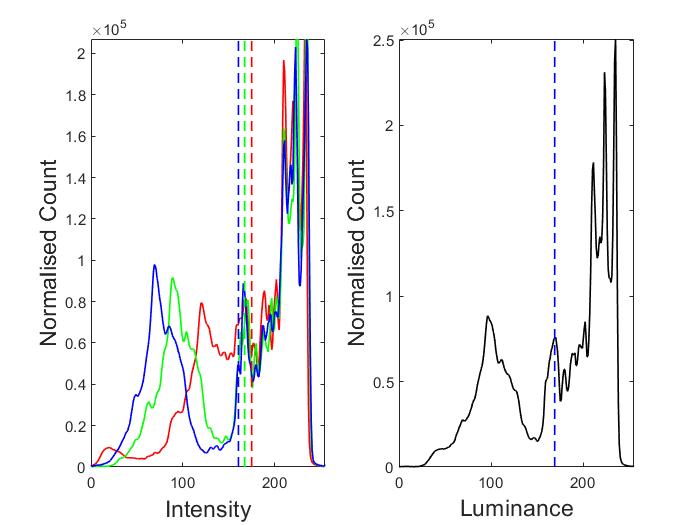
xlim([0 255]);

%Show vertical dotted bar down as Artihmetic Mean.

ylim([0 max(palmdownGreyScaleHistogram)]);

greyScaleHistYLength = get(gca,'YLim');

line([greyScaleArithmeticMean,greyScaleArithmeticMean],greyScaleHistYLength,'Color','black','LineWidth', 1,'LineStyle','--');



**This figure shows the RGB and Greyscale Normalised Histograms side by side.**

With MATLAB having in-built functions such as ‘imhist’ made it relatively simple. The difficult part was labelling all the axis perfectly since I was not adept at creating subplots. With various things to include such as ‘Linestyle,Color,Width.’.

## Task A3

clear all;

close all;

clc;

fileName = uigetfile('palmdown.png');

palmdownImageFile = imread(fileName);

r = palmdownImageFile(:, :, 1);

g = palmdownImageFile(:, :, 2);

b = palmdownImageFile(:, :, 3);

%Thresholds of all isolated RGB pixels.

orangeBinary = r > 142 & r < 210 & g > 65 & g < 107 & b < 52;

orangeIsolated = palmdownImageFile;

orangeIsolated(repmat(~orangeBinary,[1 1 3])) = 0;

yellowBinary = r > 141 & r < 202 & g > 123 & g < 194 & b > 61 & b < 81;

yellowIsolated = palmdownImageFile;

yellowIsolated(repmat(~yellowBinary,[1 1 3])) = 0;

blueBinary = r > 5 & r < 30 & g > 40 & g < 60 & b > 70 & b < 170;

blueIsolated = palmdownImageFile;

blueIsolated(repmat(~blueBinary,[1 1 3])) = 0;

greenBinary = r > 0 & r < 50 & g > 70 & g < 100 & b > 70 & b < 105;

greenIsolated = palmdownImageFile;

greenIsolated(repmat(~greenBinary,[1 1 3])) = 0;

redBinary = r > 160 & r < 200 & g > 90 & g < 120 & b > 84 & b < 119;

redIsolated = palmdownImageFile;

redIsolated(repmat(~redBinary,[1 1 3])) = 0;

%Displaying images side by side.

figure;

subplot(6,2,1);

imshow(orangeBinary);colormap gray;axis image;title('i. Binary Masks');

subplot(6,2,2);

imshow(orangeIsolated);colormap gray;axis image;title('ii. Isolated Pixels');

subplot(6,2,3);

imshow(yellowBinary);

subplot(6,2,4);

imshow(yellowIsolated);

subplot(6,2,5);

imshow(blueBinary);

subplot(6,2,6);

imshow(blueIsolated);

subplot(6,2,7);

imshow(greenBinary);

subplot(6,2,8);

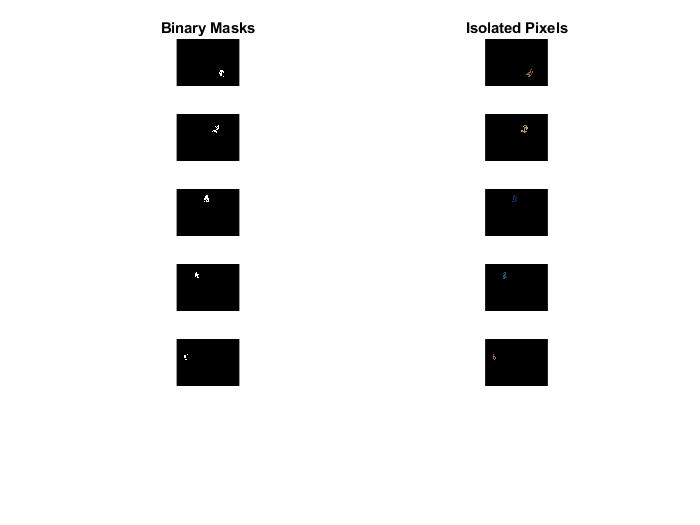
imshow(greenIsolated);

subplot(6,2,9);

imshow(redBinary);

subplot(6,2,10);

imshow(redIsolated);



**This figure shows the binarized images isolated from the RGB Image and the isolated RGB Pixels.**

Having to individually use high, low, band pass and band reject thresholding was very troublesome at the start. From the binary masks above, you can see that only some of them show very clearly. Such as the red pinky finger and the green ring finger. Messing around with the red green and blue threshold values was very time consuming to only achieve this with trial and error.

## Task A4

close all;

clc;

flathand = imread('palmdown.png');

greyScaleFlatHand = rgb2gray(flathand);

catGreyScale = cat(3, greyScaleFlatHand , greyScaleFlatHand, greyScaleFlatHand );

greyWithColour = imadd(catGreyScale, orangeIsolated);

greyWithColour = imadd(greyWithColour, redIsolated);

greyWithColour = imadd(greyWithColour, blueIsolated);

greyWithColour = imadd(greyWithColour, greenIsolated);

greyWithColour = imadd(greyWithColour, yellowIsolated);

imshow(greyWithColour);

%Cardinality of each set.

OrangeCardinality = sum(orangeBinary(:));

YellowCardinality = sum(yellowBinary(:));

BlueCardinality = sum(blueBinary(:));

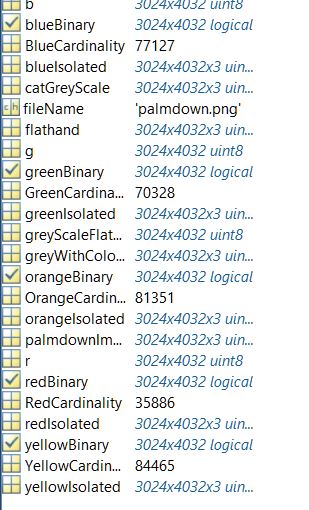
RedCardinality = sum(redBinary(:));

GreenCardinality = sum(greenBinary(:));



**This figure shows the coloured fingertips placed on top of the greyscale version from Task A3.**

Referring back to TaskA3 where I had to find the RGB thresholds for each colour, this shows an imperfect attempt at this task. Orange, Green and Blue show up decent but red and yellow do not. They look rather pale. If I had more time, I would have messed around with the thresholds more to get the desired colour.



**This figure is located within the workspace which outputs to the user the cardinality of each set.**

## Task A5

close all;

clc;

flathand = imread('palmdown.png');

greyScaleFlatHand = rgb2gray(flathand);

imshow(greyWithColour);

%Co-ordinates shown marked with a red circle from x and y values.

[y,x] = find(orangeBinary);

Orangexmean = mean(x);

Orangeymean = mean(y);

hold on;

plot(Orangexmean, Orangeymean, 'ro', 'MarkerSize', 30);

[y,x] = find(greenBinary);

Greenxmean = mean(x);

Greenymean = mean(y);

hold on;

plot(Greenxmean, Greenymean, 'ro', 'MarkerSize', 30);

[y,x] = find(blueBinary);

Bluexmean = mean(x);

Blueymean = mean(y);

hold on;

plot(Bluexmean, Blueymean, 'ro', 'MarkerSize', 30);

[y,x] = find(redBinary);

Redxmean = mean(x);

Redymean = mean(y);

hold on;

plot(Redxmean, Redymean, 'ro', 'MarkerSize', 30);

[y,x] = find(yellowBinary);

Yellowxmean = mean(x);

Yellowymean = mean(y);

hold on;

plot(Yellowxmean, Yellowymean, 'ro', 'MarkerSize', 30);

%Green line connecting adjacent fingers.

plot([Orangexmean, Yellowxmean] ,[Orangeymean ,Yellowymean], 'g')

plot([Yellowxmean, Bluexmean], [Yellowymean, Blueymean], 'g')

plot([Bluexmean, Greenxmean], [Blueymean, Greenymean], 'g')

plot([Greenxmean, Redxmean], [Greenymean ,Redymean], 'g')

%Euclidean distance from each adjacent pair of fingertips.

orange2Yellow = pdist([Orangexmean, Orangeymean; Yellowxmean,Yellowymean], 'euclidean');

yellow2Blue = pdist([Yellowxmean, Yellowymean ; Bluexmean,Blueymean], 'euclidean');

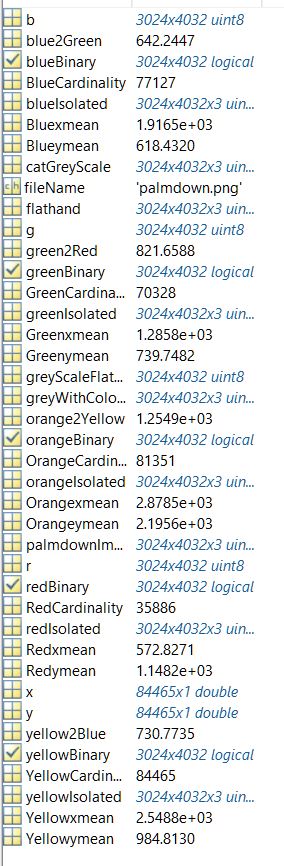
blue2Green = pdist([Bluexmean, Blueymean ; Greenxmean,Greenymean], 'euclidean');

green2Red = pdist([Greenxmean, Greenymean ; Redxmean,Redymean], 'euclidean');



**This figure is the output image from Task A5. It identifies the centre of all pixel clusters and is joined with Euclidean-distance lines.**

Overall, this task worked successfully from the instructions given. Luckily, with the functions given in MATLAB such as ‘plot’ and having the two arguments which were the x and y values of each fingertip. There is a lot of stuff you can do with ‘plot’, in this case, it was the red circles and the green line connecting to the adjacent fingertip. Unfortunately, I was not able to output the co-ordinates when hovering over the figure.



**This figure is outputted from the workspace that shows the distances from each adjacent fingertip with Euclidean distance**.

## Task A6

RGB = imread('palmdown.png');

%Converting from rgb to hsv.

I = rgb2hsv(RGB);

channel1Min = 0.046;

channel1Max = 0.080;

channel2Min = 0.235;

channel2Max = 0.549;

channel3Min = 0.284;

channel3Max = 0.923;

sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...

(I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...

(I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);

BW = sliderBW;

maskedRGBImage = RGB;

maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

%imshow(BW);

maskedSkin = rgb2gray(maskedRGBImage);

%Edge Detection

edges\_prewitt = edge(maskedSkin,'Prewitt');

edges\_roberts = edge(maskedSkin, 'Roberts');

edges\_sobel = edge(maskedSkin, 'Sobel');

edges\_canny = edge(maskedSkin, 'Canny');

%Red line

redLine = strel('line',11,90);

prewittContour = imdilate((reshape(uint8([255 0 0]),[1,1,3]) .\* uint8(edges\_prewitt)),redLine);

robertContour = imdilate((reshape(uint8([255 0 0]),[1,1,3]) .\* uint8(edges\_roberts)),redLine);

sobelContour = imdilate((reshape(uint8([255 0 0]),[1,1,3]) .\* uint8(edges\_sobel)),redLine);

cannyContour = imdilate((reshape(uint8([255 0 0]),[1,1,3]) .\* uint8(edges\_canny)),redLine);

%Displaying each edge detection method side by side.

subplot(2,2,1);

imshow(RGB + prewittContour);colormap gray;axis image;title('Prewitt');

subplot(2,2,2);

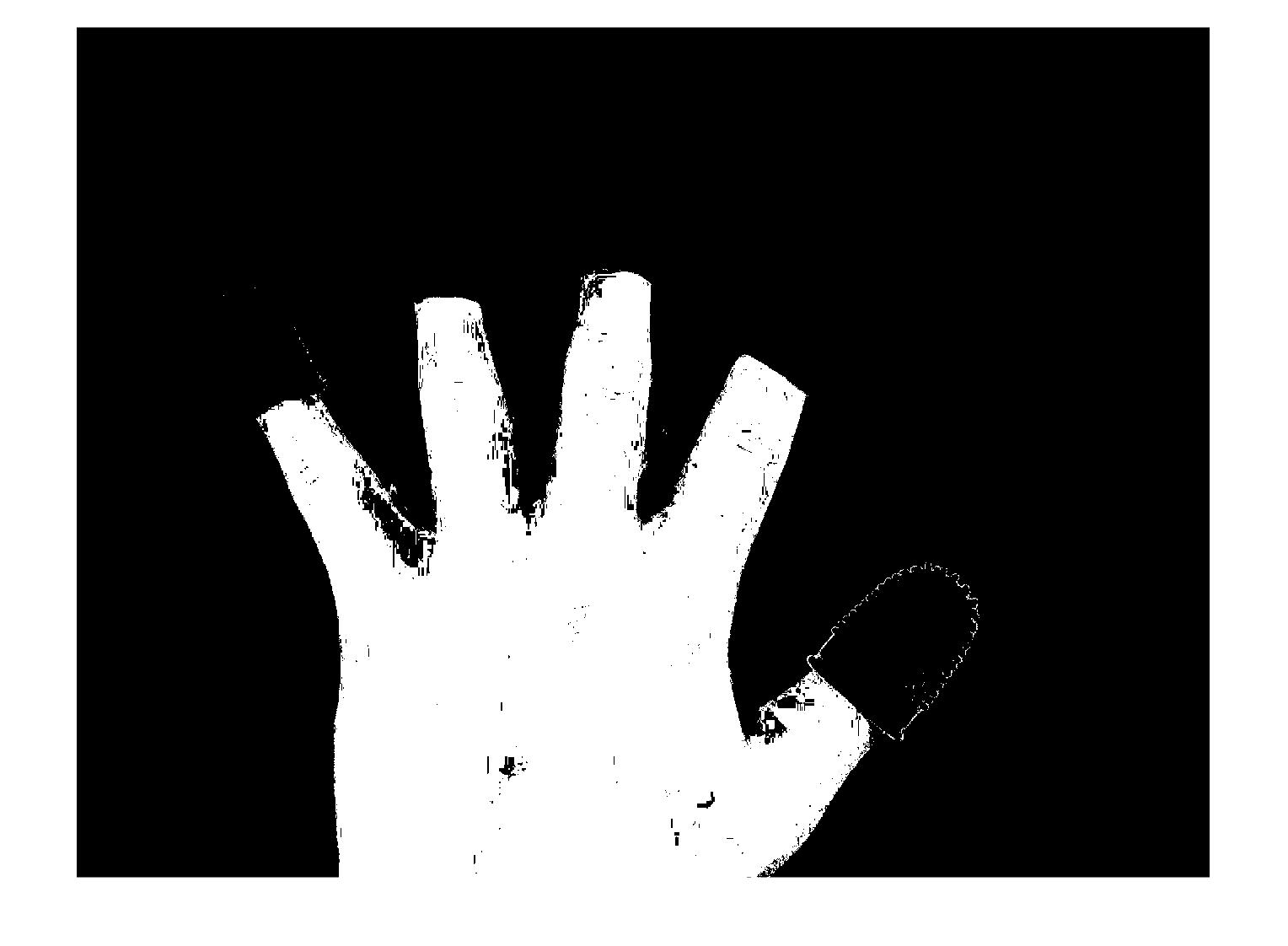
imshow(RGB + robertContour);colormap gray;axis image;title('Roberts');

subplot(2,2,3);

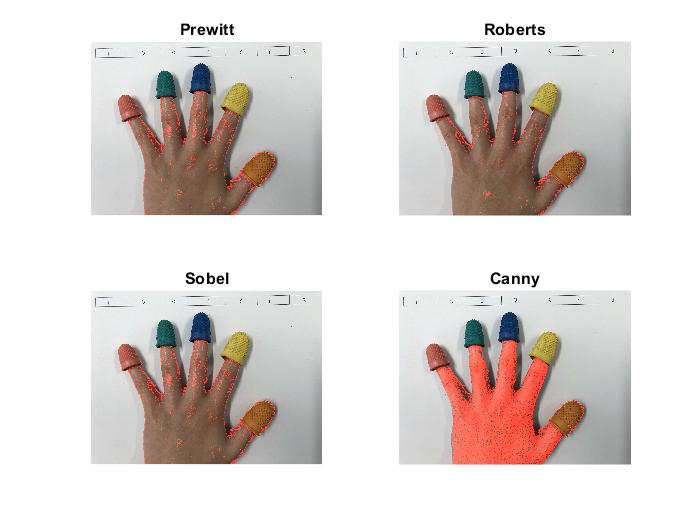
imshow(RGB + sobelContour);colormap gray;axis image;title('Sobel');

subplot(2,2,4);

imshow(RGB + cannyContour);colormap gray;axis image;title('Canny');



**This figure shows the binarized version of the RGB Image where the skin is 1 and the background and fingertips are 0.**



**This figure shows the trace boundaries of skin using Robert’s, Prewitt, Sobel and Canny. This is shown with the red edge contour on each picture.**

As from the figure above, this is an imperfect attempt at this task given. I used a tool wwithin MATLAB called ‘Color Thresholder’ to isolate the skin from the rest of the picture. From messing around with the thresholds. I was not able to successfully get rid of the outline for the orange Thumb. In the figure above, this is shown in all of the edge detection thresholds as they all trace back around the orange thumb. In addition to this, I was not able to get canny to work properly like the other thresholds.

# Section B

## Task B1

fileName = uigetfile();

HandImage = imread(fileName);

r = HandImage(:, :, 1);

g = HandImage(:, :, 2);

b = HandImage(:, :, 3);

greyScaleHand = rgb2gray(HandImage);

catGreyScale = cat(3, greyScaleHand , greyScaleHand, greyScaleHand );

orangeBinary = r > 142 & r < 208 & g > 68 & g < 109 & b < 54;

orangeIsolated = HandImage;

orangeIsolated(repmat(~orangeBinary,[1 1 3])) = 0;

yellowBinary = r > 174 & r < 235 & g > 142 & g < 206 & b > 33 & b < 108;

yellowIsolated = HandImage;

yellowIsolated(repmat(~yellowBinary,[1 1 3])) = 0;

blueBinary = r > 5 & r < 30 & g > 40 & g < 60 & b > 70 & b < 170;

blueIsolated = HandImage;

blueIsolated(repmat(~blueBinary,[1 1 3])) = 0;

greenBinary = r > 0 & r < 50 & g > 70 & g < 100 & b > 70 & b < 105;

greenIsolated = HandImage;

greenIsolated(repmat(~greenBinary,[1 1 3])) = 0;

redBinary = r > 140 & r < 180 & g > 55 & g < 80 & b > 50 & b < 80;

redIsolated = HandImage;

redIsolated(repmat(~redBinary,[1 1 3])) = 0;

greyWithColour = imadd(catGreyScale, orangeIsolated);

greyWithColour = imadd(greyWithColour, redIsolated);

greyWithColour = imadd(greyWithColour, blueIsolated);

greyWithColour = imadd(greyWithColour, greenIsolated);

greyWithColour = imadd(greyWithColour, yellowIsolated);

imshow(greyWithColour);

OrangeCardinality = sum(orangeBinary(:));

YellowCardinality = sum(yellowBinary(:));

BlueCardinality = sum(blueBinary(:));

RedCardinality = sum(redBinary(:));

GreenCardinality = sum(greenBinary(:));

[y,x] = find(orangeBinary);

Orangexmean = mean(x);

Orangeymean = mean(y);

hold on;

plot(Orangexmean, Orangeymean, 'ro', 'MarkerSize', 30);

[y,x] = find(greenBinary);

Greenxmean = mean(x);

Greenymean = mean(y);

hold on;

plot(Greenxmean, Greenymean, 'ro', 'MarkerSize', 30);

[y,x] = find(blueBinary);

Bluexmean = mean(x);

Blueymean = mean(y);

hold on;

plot(Bluexmean, Blueymean, 'ro', 'MarkerSize', 30);

[y,x] = find(redBinary);

Redxmean = mean(x);

Redymean = mean(y);

hold on;

plot(Redxmean, Redymean, 'ro', 'MarkerSize', 30);

[y,x] = find(yellowBinary);

Yellowxmean = mean(x);

Yellowymean = mean(y);

hold on;

plot(Yellowxmean, Yellowymean, 'ro', 'MarkerSize', 30);

plot([Orangexmean, Yellowxmean] ,[Orangeymean ,Yellowymean], 'g')

plot([Yellowxmean, Bluexmean], [Yellowymean, Blueymean], 'g')

plot([Bluexmean, Greenxmean], [Blueymean, Greenymean], 'g')

plot([Greenxmean, Redxmean], [Greenymean ,Redymean], 'g')

[y,x] = find(greenBinary);

Greenxmean = mean(x);

Greenymean = mean(y);

[y,x] = find(blueBinary);

Bluexmean = mean(x);

Blueymean = mean(y);

[y,x] = find(redBinary);

Redxmean = mean(x);

Redymean = mean(y);

[y,x] = find(yellowBinary);

Yellowxmean = mean(x);

Yellowymean = mean(y);

orange2Yellow = pdist([Orangexmean, Orangeymean; Yellowxmean,Yellowymean], 'euclidean');

orange2Blue = pdist([Orangexmean, Orangeymean ; Bluexmean,Blueymean], 'euclidean');

orange2Red = pdist([Orangexmean, Orangeymean ; Redxmean,Redymean], 'euclidean');

orange2Green = pdist([Orangexmean, Orangeymean ; Greenxmean,Greenymean], 'euclidean');

%Displaying back to user output commands of each gesture.

if orange2Yellow<orange2Blue && orange2Yellow<orange2Red && orange2Yellow<orange2Green

disp('This is thumb to Index Finger.')

end

if orange2Blue<orange2Yellow && orange2Blue<orange2Red && orange2Blue<orange2Green

disp('This is thumb to Middle Finger.')

end

if orange2Green<orange2Yellow && orange2Green<orange2Red && orange2Green<orange2Blue

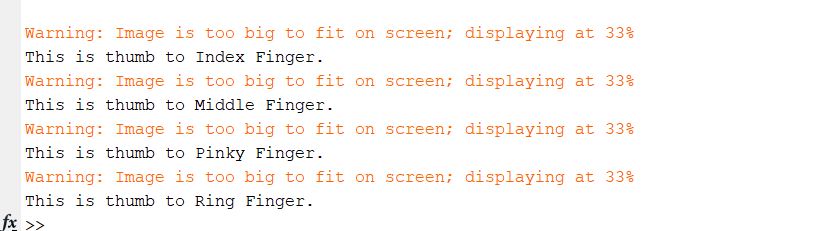
disp('This is thumb to Ring Finger.')

end

if orange2Red<orange2Yellow && orange2Red<orange2Blue && orange2Red<orange2Green

disp('This is thumb to Pinky Finger.')

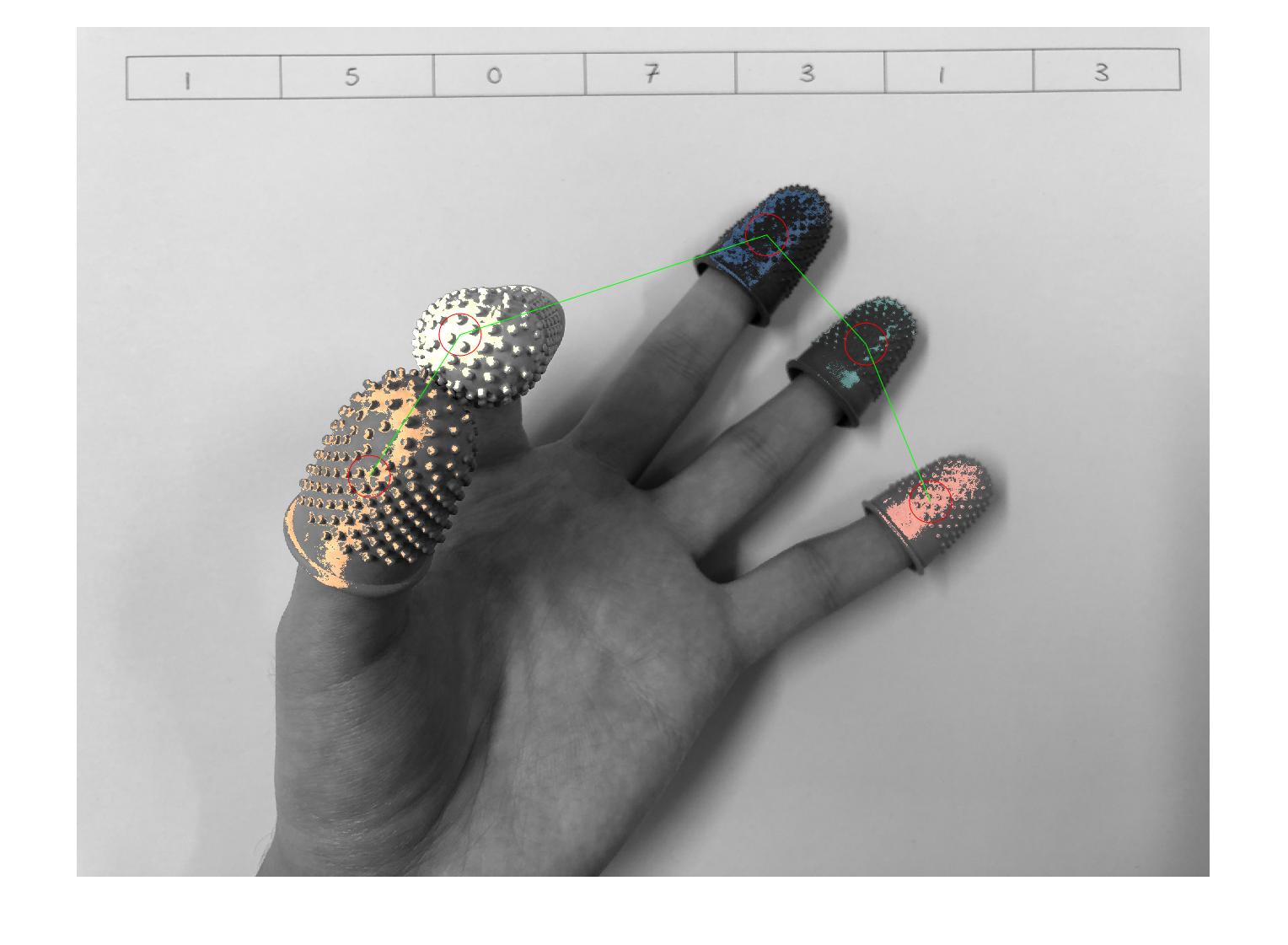
end



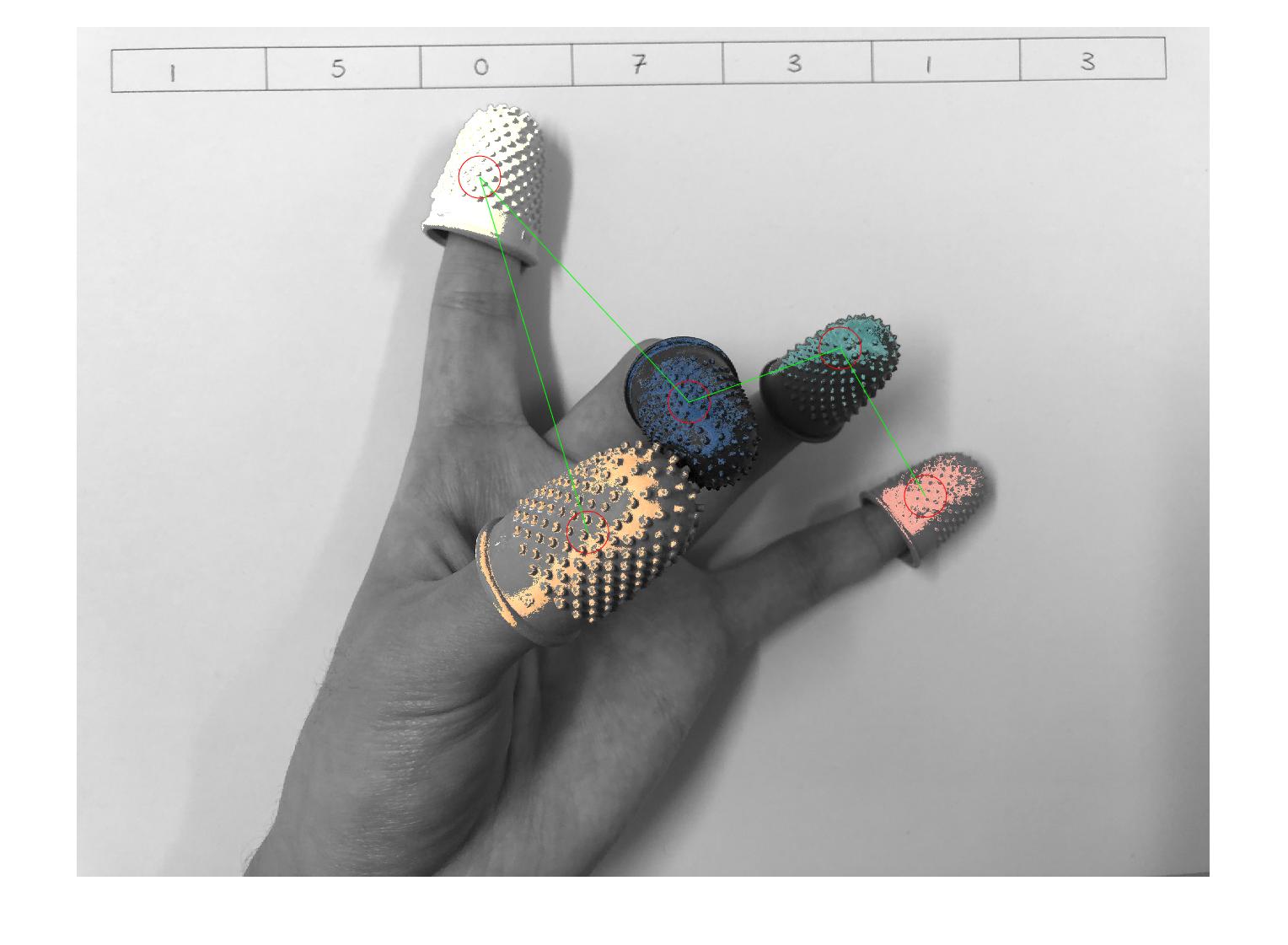
**This figure shows all the output commands back to the user corresponding to the individual gesture they upload to MATLAB.**

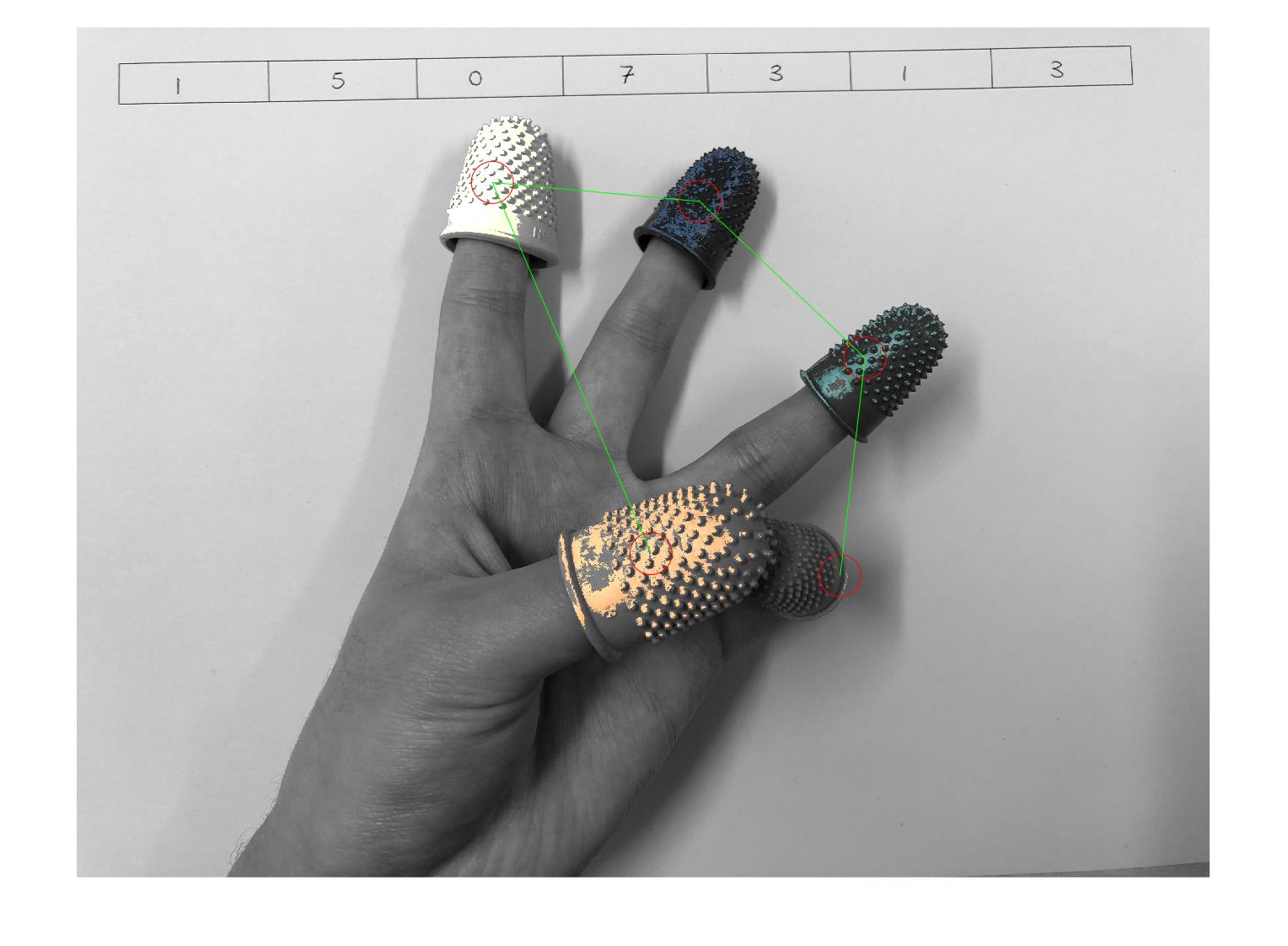
As this task was very much similar to what had to be done in Section A. I had to calculate the distance of the thumb to each other fingertip needed. From this, the smallest distance will tell me which gesture is being selected.

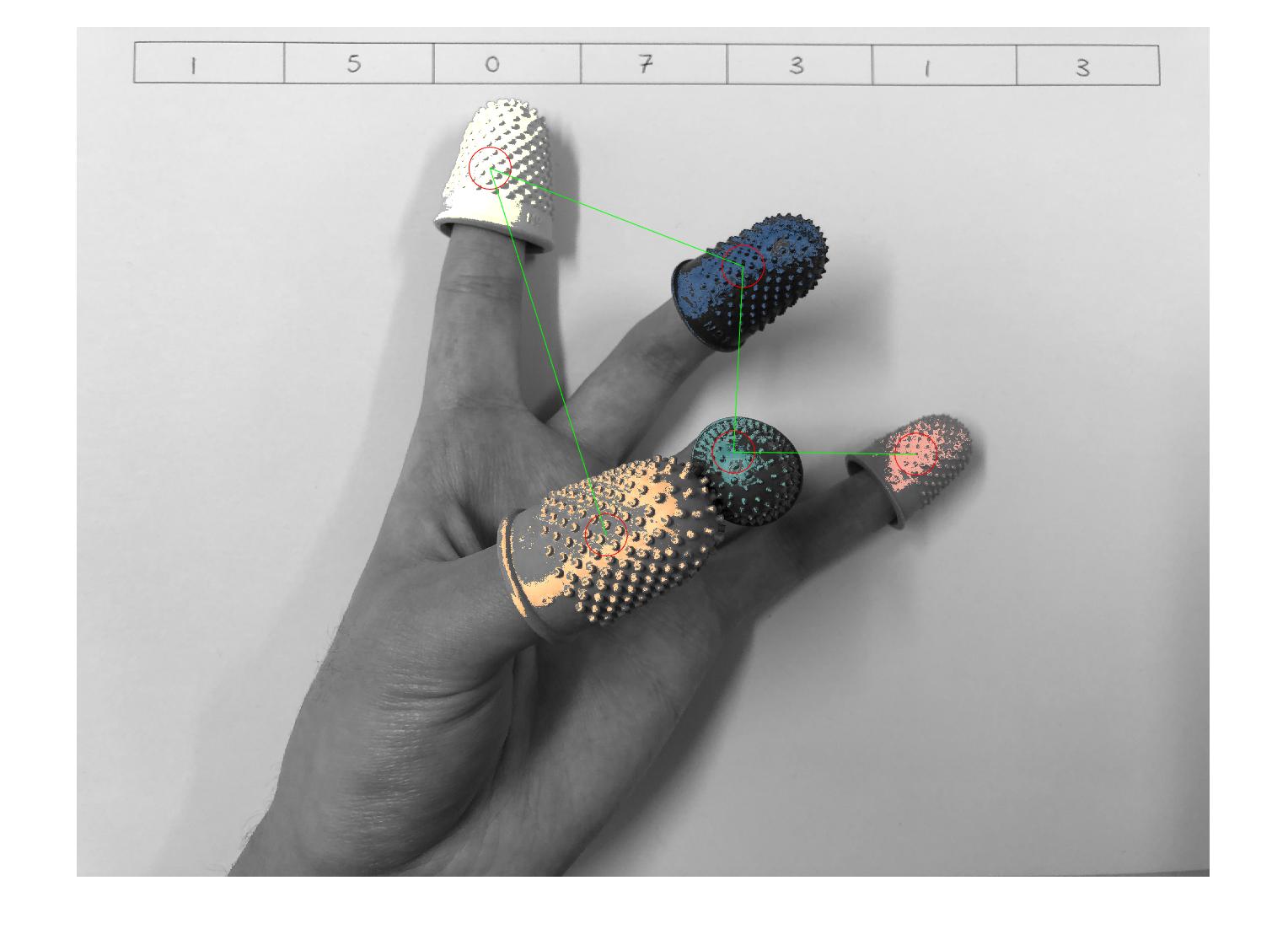
The approach was instead of finding the distance from each adjacent fingertip, I only had to find out the Euclidean distance from the thumb to each individual finger. Referring back to Section A, this was all the tweaking that needed to be done.

Using loops was not needed to retrieve my value if the code was implemented efficiently, in this case however I used an if statement several times to output which gesture was being loaded and I was unsuccessful displaying back that it was the palm down image. In addition to this, having to change the rgb thresholds again was very troublesome to identify all points for each picture. E.g Middle finger was not detected shown by a red circle in the thumb to middle finger image which was the main point. Though it showed up in the other four images.

**This figure is the thumb touching the index finger curved down into the palm with the identity centre of the pixel clusters annotated and joined centres with Euclidean-distance lines.**

**This figure is the thumb touching the middle finger curved down into the palm with the identity centre of the pixel clusters annotated and joined centres with Euclidean-distance lines.**

**This figure is the thumb touching the pinky finger curved down into the palm with the identity centre of the pixel clusters annotated and joined centres with Euclidean-distance lines.**

**This figure is the thumb touching the pinky finger curved down into the palm with the identity centre of the pixel clusters annotated and joined centres with Euclidean-distance lines.**

## Task B2

fileName = uigetfile();

HandImage = imread(fileName);

r = HandImage(:, :, 1);

g = HandImage(:, :, 2);

b = HandImage(:, :, 3);

%Displaying images side by side.

% figure;

% subplot(6,2,1);

% imshow(filledOrangeBinary);colormap gray;axis image;title('Binary Masks');

% subplot(6,2,2);

% imshow(orangeIsolated);colormap gray;axis image;title('Isolated Pixels');

%

% subplot(6,2,3);

% imshow(filledyellowBinary);

% subplot(6,2,4);

% imshow(yellowIsolated);

%

% subplot(6,2,5);

% imshow(filledblueBinary);

% subplot(6,2,6);

% imshow(blueIsolated);

%

% subplot(6,2,7);

% imshow(filledgreenBinary);

% subplot(6,2,8);

% imshow(greenIsolated);

%

% subplot(6,2,9);

% imshow(filledredBinary);

% subplot(6,2,10);

% imshow(redIsolated);

greyScaleHand = rgb2gray(HandImage);

catGreyScale = cat(3, greyScaleHand , greyScaleHand, greyScaleHand );

orangeBinary = r > 142 & r < 208 & g > 68 & g < 109 & b < 54;

orangeIsolated = HandImage;

orangeIsolated(repmat(~orangeBinary,[1 1 3])) = 0;

yellowBinary = r > 174 & r < 235 & g > 142 & g < 206 & b > 33 & b < 108;

yellowIsolated = HandImage;

yellowIsolated(repmat(~yellowBinary,[1 1 3])) = 0;

blueBinary = r > 5 & r < 30 & g > 40 & g < 60 & b > 70 & b < 170;

blueIsolated = HandImage;

blueIsolated(repmat(~blueBinary,[1 1 3])) = 0;

greenBinary = r > 0 & r < 50 & g > 70 & g < 100 & b > 70 & b < 105;

greenIsolated = HandImage;

greenIsolated(repmat(~greenBinary,[1 1 3])) = 0;

redBinary = r > 140 & r < 180 & g > 55 & g < 80 & b > 50 & b < 80;

redIsolated = HandImage;

redIsolated(repmat(~redBinary,[1 1 3])) = 0;

%Filling in of binary masks.

filledredBinary = imfill(redBinary,'holes');

filledyellowBinary = imfill(yellowBinary,'holes');

filledgreenBinary = imfill(greenBinary,'holes');

filledblueBinary = imfill(blueBinary,'holes');

filledorangeBinary = imfill(orangeBinary,'holes');

greyWithColour = imadd(catGreyScale, orangeIsolated);

greyWithColour = imadd(greyWithColour, redIsolated);

greyWithColour = imadd(greyWithColour, blueIsolated);

greyWithColour = imadd(greyWithColour, greenIsolated);

greyWithColour = imadd(greyWithColour, yellowIsolated);

imshow(greyWithColour);

[y,x] = find(filledorangeBinary);

Orangexmean = mean(x);

Orangeymean = mean(y);

hold on;

plot(Orangexmean, Orangeymean, 'ro', 'MarkerSize', 30);

[y,x] = find(filledgreenBinary);

Greenxmean = mean(x);

Greenymean = mean(y);

hold on;

plot(Greenxmean, Greenymean, 'ro', 'MarkerSize', 30);

[y,x] = find(filledblueBinary);

Bluexmean = mean(x);

Blueymean = mean(y);

hold on;

plot(Bluexmean, Blueymean, 'ro', 'MarkerSize', 30);

[y,x] = find(filledredBinary);

Redxmean = mean(x);

Redymean = mean(y);

hold on;

plot(Redxmean, Redymean, 'ro', 'MarkerSize', 30);

[y,x] = find(filledyellowBinary);

Yellowxmean = mean(x);

Yellowymean = mean(y);

hold on;

plot(Yellowxmean, Yellowymean, 'ro', 'MarkerSize', 30);

plot([Orangexmean, Yellowxmean] ,[Orangeymean ,Yellowymean], 'g')

plot([Yellowxmean, Bluexmean], [Yellowymean, Blueymean], 'g')

plot([Bluexmean, Greenxmean], [Blueymean, Greenymean], 'g')

plot([Greenxmean, Redxmean], [Greenymean ,Redymean], 'g')

orange2Yellow = pdist([Orangexmean, Orangeymean; Yellowxmean,Yellowymean], 'euclidean');

orange2Blue = pdist([Orangexmean, Orangeymean ; Bluexmean,Blueymean], 'euclidean');

orange2Red = pdist([Orangexmean, Orangeymean ; Redxmean,Redymean], 'euclidean');

orange2Green = pdist([Orangexmean, Orangeymean ; Greenxmean,Greenymean], 'euclidean');

distances =cat(1,orange2Yellow, orange2Blue, orange2Red, orange2Green);

%Displaying back to user output commands of each gesture.

if orange2Yellow<orange2Blue && orange2Yellow<orange2Red && orange2Yellow<orange2Green

disp('This is thumb to Index Finger.')

end

if orange2Blue<orange2Yellow && orange2Blue<orange2Red && orange2Blue<orange2Green

disp('This is thumb to Middle Finger.')

end

if orange2Green<orange2Yellow && orange2Green<orange2Red && orange2Green<orange2Blue

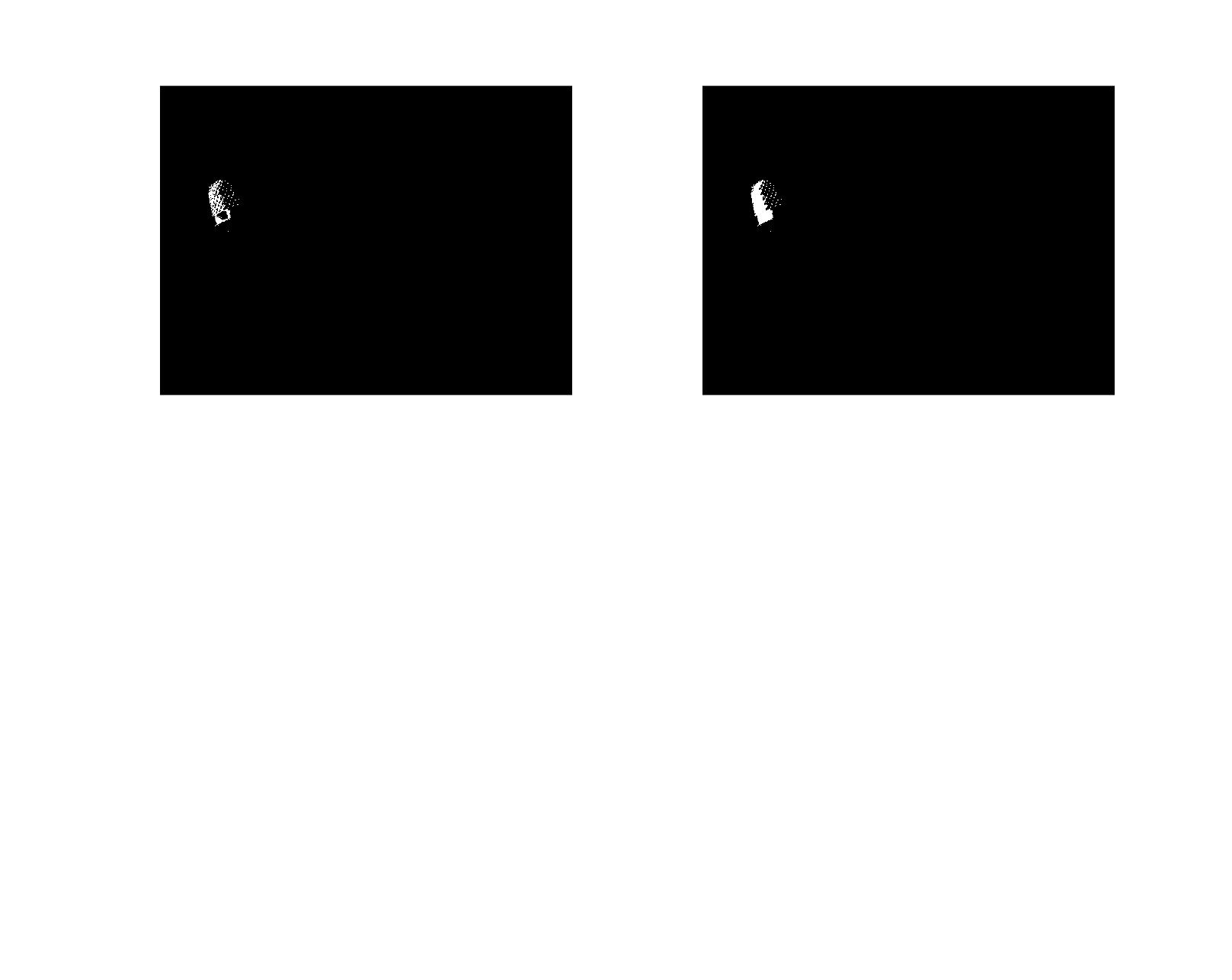
disp('This is thumb to Ring Finger.')

end

if orange2Red<orange2Yellow && orange2Red<orange2Blue && orange2Red<orange2Green

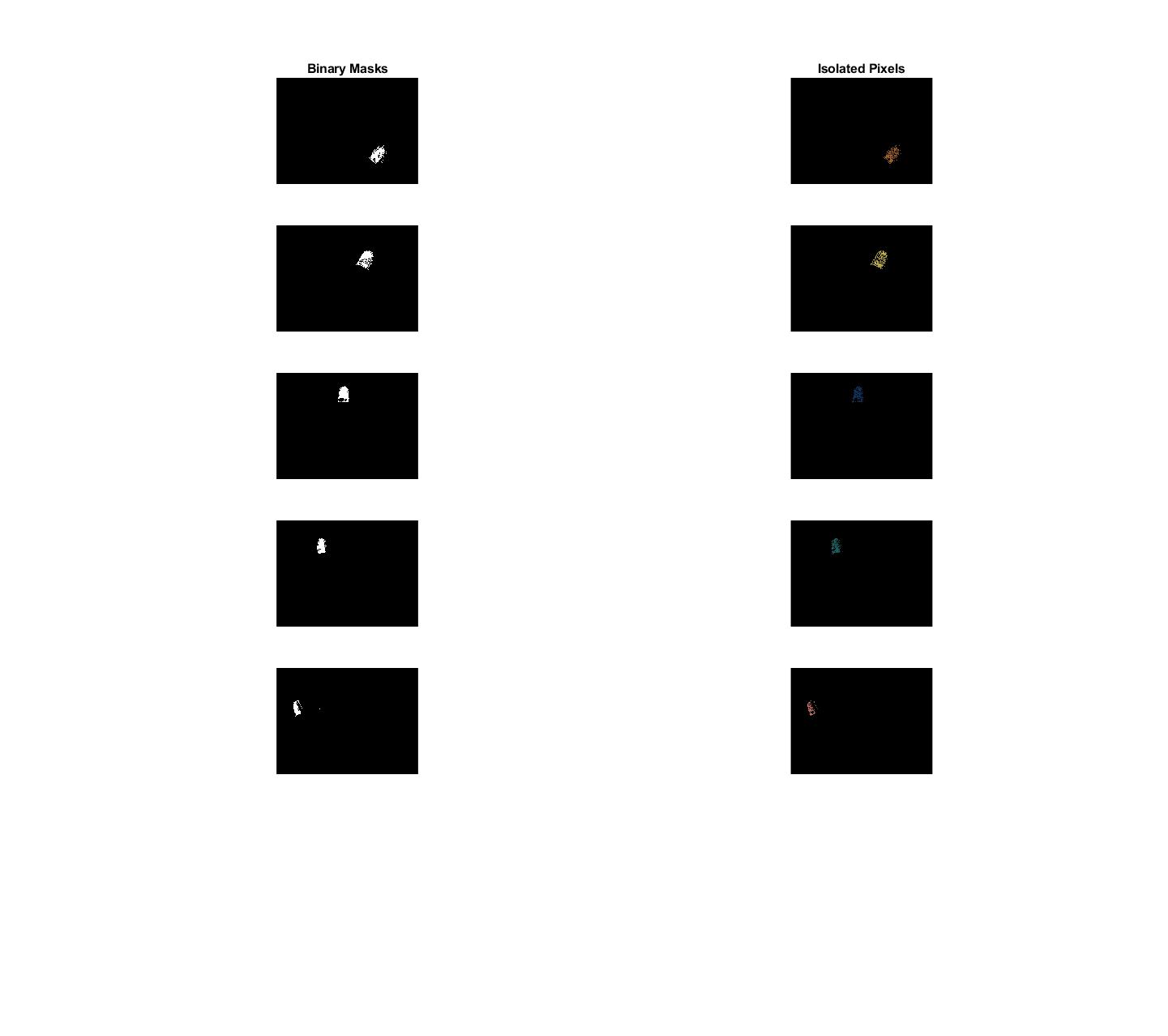
disp('This is thumb to Pinky Finger.')

end

Referring back to morphological operators back within MATLAB. The binary masks outputted from Task A3 for the pinky finger was severely contrasting to the other four fingertips. Within MATLAB I used a morphological operator which was called ‘imfill’ which is used to fill image regions and holes. This is an example of what the function was able to for the binary mask for the pinky finger.

**This figure shows the binary mask for the pinky finger and its image regions and holes filled out.**

After changing the variable names such as orangeBinary now to filledorangeBinary and so forth. I placed my code back into what I did for B1 and the average mean co-ordinates for x and y changed as expected. The classification of the gestures however it still remained the same as it did for B1. The program was still able to identify which gesture it was from the shortest Euclidean distance.

**This figure shows all the binary masks filled in for each finger and the isolated pixels.**