

21st December 2018

Image Processing Module: MOD002643

SID: 1535380

Contents

[Section A 2](#_Toc533157706)

[1 2](#_Toc533157707)

[2 4](#_Toc533157708)

[3 7](#_Toc533157709)

[4 11](#_Toc533157710)

[5 14](#_Toc533157711)

[6 18](#_Toc533157712)

[Section B 24](#_Toc533157713)

[1 24](#_Toc533157714)

[References 36](#_Toc533157715)

# 

# Section A

## 1

clear all;

close all;

clc;

%reads the colour image to 'RGBHand'

RGBHand = imread('hand.bmp');

%splits RGBHand into R,G and B

R = RGBHand(:,:,1);

G = RGBHand(:,:,2);

B = RGBHand(:,:,3);

[M,N]=size(R);

%converts to greyscale using the lightness approach

for x = 1: M

for y = 1:N

values=[R(x,y),G(x,y),B(x,y)];

lightness= max(values)+min(values);

lightness=lightness/2;

GreyHand(x,y,1)=lightness;

GreyHand(x,y,2)=lightness;

GreyHand(x,y,3)=lightness;

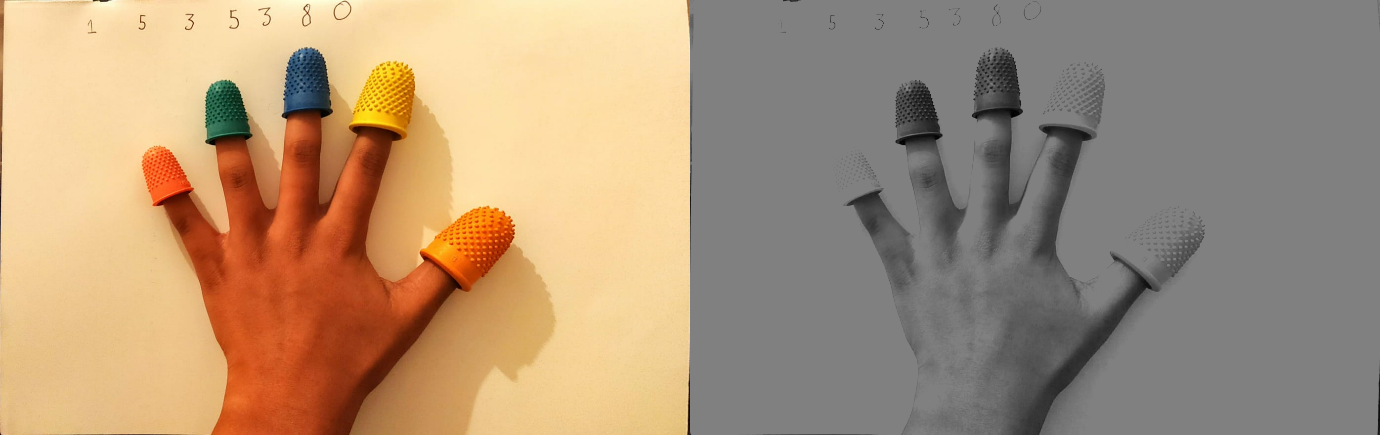
end

end

%displays the colour image and the greyscale image together

imshowpair(RGBHand,GreyHand,'montage');

figure 1: Output from question 1 (Left: colour image of hand, Right: greyscale image of hand)



Lightness: Eq 1.1

In order to convert from a colour image (RGB) to greyscale image, the lightness (Eq 1.1) approach was used as can be seen in the code. The code is fully functional and does what is required, however it does use a ‘for’ loop, in order for the result to be stored into a ‘bmp’ file. If the images could have been saved together into a file regardless off the array sizes, there would have been a preferrence to use the luminosity approach as this would have yielded a sharper image. Using the lightness approach has caused the greyscale image to have a higher exposure.

## 2

clear all;

close all;

clc;

RGBHand = imread('hand.bmp');

R = RGBHand(:,:,1);

G = RGBHand(:,:,2);

B = RGBHand(:,:,3);

GreyHand = ((0.21\*R)+(0.72\*G)+(0.07\*B));

figure1 = figure;

subplot(1,2,1); %RGB and greyscale histograms can display next to eachother

R\_count = imhist(R); %creates an array of counts of values 0 to 255

[a,b]=size(R); % gets the size of the x-axis and y-axis

R\_count=R\_count/(a\*b);% normalises the counts

%calculates the mean of R

R\_mean=sum(R);

R\_mean = sum(R\_mean);

R\_mean = R\_mean/(a\*b);

R\_max=max(max(R)); %calculates the maximum value of R

R\_min=min(min(R)); %calculates the minimum value of R

%calculates mean, max and min of G

G\_mean=sum(G);

G\_mean = sum(G\_mean);

G\_mean = G\_mean/(a\*b);

G\_max=max(max(G));

G\_min=min(min(G));

%calculates the mean, max and min of B

B\_mean=sum(B);

B\_mean = sum(B\_mean);

B\_mean = B\_mean/(a\*b);

B\_max=max(max(B));

B\_min=min(min(B));

% calculates the normalised frequencies of G

G\_count = imhist(G);

[a,b]=size(G);

G\_count=G\_count/(a\*b);

%calculates normalised frequencies of B

B\_count = imhist(B);

[a,b]=size(B);

B\_count=B\_count/(a\*b);

%plots the R,G,B histograms on same grid

plot(R\_count,'r');

hold on

plot(G\_count,'g');

plot(B\_count,'b');

%plots the mean of R, G and B on the grid

y = ylim;

plot([R\_mean R\_mean],[y(1) y(2)],'--r')

plot([G\_mean G\_mean],[y(1) y(2)],'--g')

plot([B\_mean B\_mean],[y(1) y(2)],'--b')

hold off

xlabel('Luminance','FontSize',14);

ylabel('Normalized Count','FontSize',14);

subplot(1,2,2) %RGB and greyscale histograms can display next to eachother

%calcuates the normaliased frequencies of the greyscale image

Grey\_count = imhist(GreyHand);

[a,b]=size(GreyHand);

Grey\_count=Grey\_count/(a\*b);

%calculates the mean, max and min of teh greyscale image

Grey\_mean=sum(GreyHand);

Grey\_mean = sum(Grey\_mean);

Grey\_mean = Grey\_mean/(a\*b);

Grey\_max=max(max(GreyHand));

Grey\_min=min(min(GreyHand));

%plots the greyscale histogram

plot(Grey\_count);

hold on

%plots the mean of teh greyscale image on the same grid

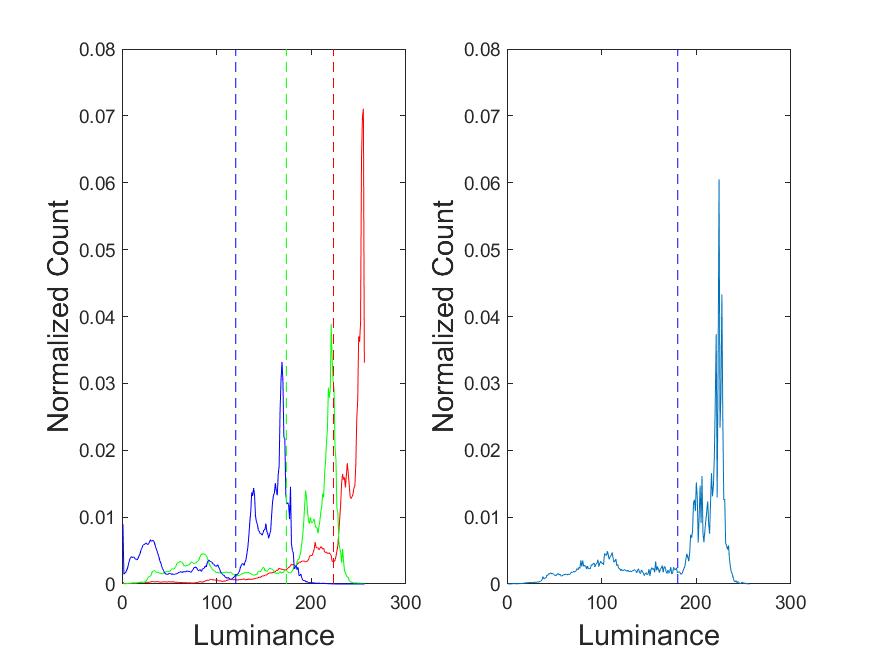
plot([Grey\_mean Grey\_mean],[y(1) y(2)],'--b');

xlabel('Luminance','FontSize',14);

ylabel('Normalized Count','FontSize',14);

saveas(figure1, 'A2.jpg');

figure 2: output from A2 (Left: normalised histogram of R,G and B with displayed means , R: normalised histogram of greyscale with displayed mean)



R\_max = 255 , R\_min = 0

G\_max= 255, G\_Min = 0

B\_max = 232 B\_Min = 0

Grey\_max=254, Grey\_min = 1

Luminosity: Eq 2.1

Arithmetic Mean Eq 2.2

The luminosity (Eq 2.1) approach to convert to greyscale was used for this question, so that the input does not have a histogram for an image with a high exposure. The code is fully functional. There are no ‘for’ loops needed for this question, which makes the code more quicker to run. The output displays the two histograms (one for colours R,G and B and another for greyscale) next to each other, displaying the normalised count and the frequencies. The axis of the histograms are labelled appropriately and the means of each histogram counts are displayed correctly. The information in the image is clear to read and understand. The arithmetic mean (Eq 2.2) was used in order to place the mean indicators on the histograms.

## 3

clear all;

close all;

clc;

RGBHand = imread('hand.bmp');

R = RGBHand(:,:,1);

G = RGBHand(:,:,2);

B = RGBHand(:,:,3);

[a,b]=size(R);

RGBH1=RGBHand; %assigns RGBHand to RGBH1

%looks at every value in RGBH1

for x = 1 : a

for y = 1 : b

if R(x,y)>235 && G(x,y)<80 && B(x,y)<50 %checks for thresholds

%if within thresholds this value in little appears white

little(x,y,1)=255;

little(x,y,2)=255;

little(x,y,3)=255;

else

%if not within thresholds this value in little appears black

little(x,y,1)=0;

little(x,y,2)=0;

little(x,y,3)=0;

%the values in RGBH1 appear black

RGBH1(x,y,1)=0;

RGBH1(x,y,2)=0;

RGBH1(x,y,3)=0;

end % end if

end % end for

end % end for

RGBH2 = RGBHand; %assigns RGBHand to RGBH2

%looks at every value in RGBH2

for x = 1 : a

for y = 1: b

if G(x,y)>100 && R(x,y)<60 && B(x,y)<100 %checks for thresholds

%if within thresholds this value in ring appears white

ring(x,y,1)=255;

ring(x,y,2)=255;

ring(x,y,3)=255;

else

%if not within thresholds this value in ring appears black

ring(x,y,1)=0;

ring(x,y,2)=0;

ring(x,y,3)=0;

%the values in RGBH2 appear black

RGBH2(x,y,1)=0;

RGBH2(x,y,2)=0;

RGBH2(x,y,3)=0;

end % end if

end %end for

end %end for

RGBH3 = RGBHand; %assigns RGBHand to RGBH3

%looks at every value in RGBH3

for x = 1 : a

for y = 1: b

if B(x,y)>90 && R(x,y)<150 && G(x,y)<110 %checks for thresholds

%if within thresholds this value in middle appears white

middle(x,y,1)=255;

middle(x,y,2)=255;

middle(x,y,3)=255;

else

%if not within thresholds this value in middle appears black

middle(x,y,1)=0;

middle(x,y,2)=0;

middle(x,y,3)=0;

%the values in RGBH3 appear black

RGBH3(x,y,1)=0;

RGBH3(x,y,2)=0;

RGBH3(x,y,3)=0;

end %end if

end %end for

end %end for

RGBH4 = RGBHand; %assigns RGBHand to RGBH4

%looks at every value in RGBH4

for x = 1 : a

for y = 1: b

if R(x,y)>240 && G(x,y)>200 && B(x,y)<100 %checks for thresholds

%if within thresholds this value in index appears white

index(x,y,1)=255;

index(x,y,2)=255;

index(x,y,3)=255;

else

%if not within thresholds this value in index appears black

index(x,y,1)=0;

index(x,y,2)=0;

index(x,y,3)=0;

%the values in RGBH4 appear black

RGBH4(x,y,1)=0;

RGBH4(x,y,2)=0;

RGBH4(x,y,3)=0;

end %end if

end %end for

end %end for

RGBH5 = RGBHand; %assigns RGBHand to RGBH5

%looks at every value in RGBH5

for x = 1 : a

for y = 1: b

%checks for thresholds

if R(x,y)>240 && G(x,y)>100 && G(x,y)<170 && B(x,y)<20

%if within thresholds this value in thumb appears white

thumb(x,y,1)=255;

thumb(x,y,2)=255;

thumb(x,y,3)=255;

else

%if not within thresholds this value in thumb appears black

thumb(x,y,1)=0;

thumb(x,y,2)=0;

thumb(x,y,3)=0;

%the values in RGBH5 appear black

RGBH5(x,y,1)=0;

RGBH5(x,y,2)=0;

RGBH5(x,y,3)=0;

end % end if

end % end for

end %end for

a = figure;

%displays the values of 10 images all together.

imagesc([little RGBH1; ring RGBH2; middle RGBH3; index RGBH4; thumb RGBH5]);

axis image;

axis off;

saveas(a,'A3.jpg');

Figure 3: Output from A3 (1-Left: binarized image of isolated little finger tip,1-Right:colour image of isolated little finger tip,2-Left:binarized image of isolate ring finger tip,2-Right:colour image of isolated ring finger tip,3-Left:binary of isolated middle finger tip,3-Right:colour of isolated middle finger tip, 4-Left: binary of isolated index finger tip, 4-Right:colour of isolated index finger tip,5-Left:binary of isolated thumb finger tip,5-Right:colour of isolated thumb finger tip

thumb : R(x,y)>240,High Pass(Eq 3.2) G(x,y)>100 && G(x,y)<170 , Band pass(Eq 3.3) B(x,y)<20, Low pass(Eq 3.1)

index: R(x,y)>240,High pass (Eq 3.2) G(x,y)>200, High pass (Eq 3.2) B(x,y)<100, Low Pass (Eq 3.1)

middle: B(x,y)>90, High Pass (Eq 3.2) R(x,y)<150, Low pass (Eq 3.1) G(x,y)<110, Low Pass (Eq 3.1)

ring: G(x,y)>100, High pass (Eq 3.2) R(x,y)<60, Low pass (Eq 3.1) B(x,y)<100, Low pass (Eq 3.1)

little: R(x,y)>235, High pass (Eq 3.2) G(x,y)<80, Low pass (Eq 3.1) B(x,y)<50, Low pass (Eq 3.1)

Low Pass: Eq 3.1

High Pass: Eq 3.2

Band-Pass: Eq 3.3

This code is fully functional, however it uses many ‘for’ loops to look at every value in an array. The output displays ten images, consisting of a binary image of the isolated fingertip and the coloured image of the isolated fingertip for each finger correctly. Due to the lighting in the image being ‘yellow’ and other factors such as the skin tone of the hand, the isolation became slightly less accurate. For example, the skin tone appeared to be a close colour to the red and orange fingertips. This could have yielded better results if the fingertips were a truer version of the colours and/ or other colours were used instead of orange. The lighting used when taking the image could have also been a white light, this could help when results could be skewed due to colours being similar.

## 4

clear all;

close all;

clc;

RGBHand = imread('hand.bmp');

R = RGBHand(:,:,1);

G = RGBHand(:,:,2);

B = RGBHand(:,:,3);

[a,b]=size(R);

littleSize=0; %initialisizes little cardinality value

for x = 1 : a

for y = 1 : b

if R(x,y)>235 && G(x,y)<80 && B(x,y)<50

little(x,y)=1;

littleSize=littleSize+1; %finds cardinality of 'little' cluster

else

little(x,y)=0;

end %end if

end %end for

end %end for

ringSize=0; %initialises cardinality value

for x = 1 : a

for y = 1: b

if G(x,y)>100 && R(x,y)<60 && B(x,y)<100

ring(x,y)=1;

ringSize=ringSize+1; %finds cardinality of ring cluster

else

ring(x,y)=0;

end %end if

end %end for

end %end for

middleSize=0; %initializes cardinality value

for x = 1 : a

for y = 1: b

if B(x,y)>90 && R(x,y)<150 && G(x,y)<110

middle(x,y)=1;

middleSize=middleSize+1; %finds cardinality of middle cluster

else

middle(x,y)=0;

end %end if

end %end for

end %end for

indexSize=0; %initializes cardinlity value

for x = 1 : a

for y = 1: b

if R(x,y)>240 && G(x,y)>200 && B(x,y)<100

index(x,y)=1;

indexSize=indexSize+1; %finds cardinality of index

else

index(x,y)=0;

end %end if

end %end for

end %end for

thumbSize=0; %initializes cardinality value

for x = 1 : a

for y = 1: b

if R(x,y)>240 && G(x,y)>100 && G(x,y)<170 && B(x,y)<20

thumb(x,y)=1;

thumbSize=thumbSize+1; %finds cardinality of thumb cluster

else

thumb(x,y)=0;

end %end if

end %end for

end %end for

%adds all the clusters into a mask

mask=little+ring+middle+index+thumb;

for x=1:a

for y=1:b

if mask(x,y)==0

%converts the area outside the clusters into greyscale using

%lightness

z = [RGBHand(x,y,1),RGBHand(x,y,2),RGBHand(x,y,3)];

z=max(z)+min(z);

z=z/2;

RGBHand(x,y,1)=z;

RGBHand(x,y,2)=z;

RGBHand(x,y,3)=z;

end %end if

end %end for

end %end for

%displays the coloured clusters and greyscale background image

imshow(RGBHand)

imwrite(RGBHand,'A4.bmp');



Figure 4: output from A4 ( coloured finger tips and greyscale background and hand)

Cardinalities:

Little: 688

Ring: 1702

Middle: 4689

Index: 3614

Thumb: 6036

The code is fully functional, however, not very efficient, as it uses many ‘for’ loops. The output is accurate, the image displayed is each fingertip isolated so only the fingertips are in colour and everything else (the hand and the background) is in greyscale. As mentioned above, this can be improved my using different coloured fingertips as some of the fingertips are a close colour to the skin tone of the hand and/ or a white light could have been used. The lightness approach (Eq 1.1) was used to convert the non – fingertip to greyscale.

## 5

clear all;

close all;

clc;

RGBHand = imread('hand.bmp');

R = RGBHand(:,:,1);

G = RGBHand(:,:,2);

B = RGBHand(:,:,3);

[a,b]=size(R);

littleSize=0;

for x = 1 : a

for y = 1 : b

if R(x,y)>235 && G(x,y)<80 && B(x,y)<50

little(x,y)=1;

littleSize=littleSize+1;

else

little(x,y)=0;

end

end

end

ringSize=0;

for x = 1 : a

for y = 1: b

if G(x,y)>100 && R(x,y)<60 && B(x,y)<100

ring(x,y)=1;

ringSize=ringSize+1;

else

ring(x,y)=0;

end

end

end

middleSize=0;

for x = 1 : a

for y = 1: b

if B(x,y)>90 && R(x,y)<150 && G(x,y)<110

middle(x,y)=1;

middleSize=middleSize+1;

else

middle(x,y)=0;

end

end

end

indexSize=0;

for x = 1 : a

for y = 1: b

if R(x,y)>240 && G(x,y)>200 && B(x,y)<100

index(x,y)=1;

indexSize=indexSize+1;

else

index(x,y)=0;

end

end

end

thumbSize=0;

for x = 1 : a

for y = 1: b

if R(x,y)>240 && G(x,y)>100 && G(x,y)<170 && B(x,y)<20

thumb(x,y)=1;

thumbSize=thumbSize+1;

else

thumb(x,y)=0;

end

end

end

[posY,posX]=find(little); %finds the positions of the values in little

littleX=round(sum(posX)/littleSize); %calculates average X position

littleY=round(sum(posY)/littleSize); %calculates average Y position

[posY,posX]=find(ring); %finds the positions of the values in ring

ringX=round(sum(posX)/ringSize); %calculates average X position

ringY=round(sum(posY)/ringSize); %calculates average Y position

[posY,posX]=find(middle); %finds the positions of the values in middle

middleX=round(sum(posX)/middleSize); %calculates average X position

middleY=round(sum(posY)/middleSize); %calculates average Y position

[posY,posX]=find(index); %finds the positions of the values in index

indexX=round(sum(posX)/indexSize); %calculates average X position

indexY=round(sum(posY)/indexSize); %calculates average Y position

[posY,posX]=find(thumb); %finds the positions of the values in thumb

thumbX=round(sum(posX)/thumbSize); %calculates average X position

thumbY=round(sum(posY)/thumbSize); %calculates average Y position

%inserts red circles at average pixel

RGBHand=insertShape(RGBHand,'circle',[littleX littleY 10],'Color','red','LineWidth', 3);

RGBHand=insertShape(RGBHand,'circle',[ringX ringY 10],'Color','red','LineWidth', 3);

RGBHand=insertShape(RGBHand,'circle',[middleX middleY 10],'Color','red','LineWidth', 3);

RGBHand=insertShape(RGBHand,'circle',[indexX indexY 10],'Color','red','LineWidth', 3);

RGBHand=insertShape(RGBHand,'circle',[thumbX thumbY 10],'Color','red','LineWidth', 3);

% displays the image with red circles and green lines according to the average

image = figure;

imagesc(RGBHand);

axis image;

axis off;

%inserts green lines between the fingers

line([littleX ringX],[littleY ringY],'Color','green');

line([ringX middleX],[ringY middleY],'Color','green');

line([middleX indexX],[middleY indexY],'Color','green');

line([indexX thumbX],[indexY thumbY],'Color','green');

%calculates the distance between the fingers

little\_to\_ring=round(sqrt((littleX-ringX)^2 + (littleY-ringY)^2));

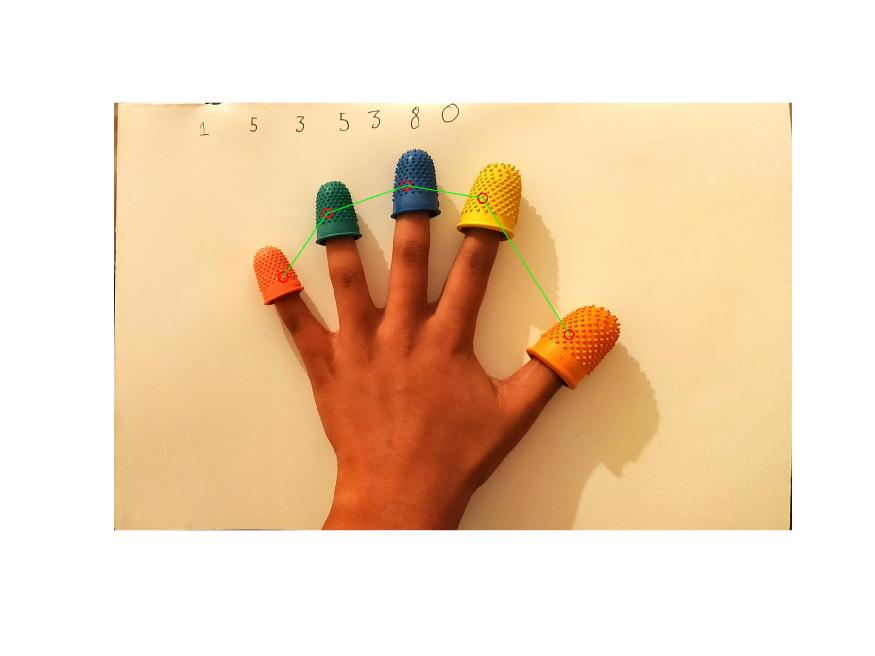
ring\_to\_middle=round(sqrt((ringX-middleX)^2 + (ringY-middleY)^2));

middle\_to\_index=round(sqrt((middleX-indexX)^2 + (middleY-indexY)^2));

index\_to\_thumb=round(sqrt((indexX-thumbX)^2 + (indexY-thumbY)^2));

saveas(image,'A5.jpg');

figure 5: output from A5 ( colour image of average locations of each finger tip indicated by red circles and green lines connected by averages.



Average Coordinates:

Little = (349,359)

Ring = (439,228)

Middle = (603,171)

Index = (760,197)

Thumb = (938,478)

Distances Between:

Little and Ring = 159

Ring and Middle = 174

Middle and Index = 159

Index and Thumb = 333

Euclidean Distance Eq 5.1

Mean Eq 5.2

The code is fully functional, by which it means that it outputs an image that displays the average pixels in red circles and green lines connecting the averages. However, it uses ‘for’ loops which lowers the efficiency of the code. The output is displayed in the correct format. The distances and the average coordinates are reported accurately. Using the Euclidean distance ( Eq 5.1) allows the output distances to be accurate. The mean (Eq 5.2) is used to calculate the average distances.

## 6

clear all;

close all;

clc;

RGBHand = imread('hand.bmp');

R = RGBHand(:,:,1);

G = RGBHand(:,:,2);

B = RGBHand(:,:,3);

[a,b]=size(R);

for x = 1 : a

for y = 1 : b

if R(x,y)>235 && G(x,y)<80 && B(x,y)<50

little(x,y)=1;

else

little(x,y)=0;

end

end

end

for x = 1 : a

for y = 1: b

if G(x,y)>100 && R(x,y)<60 && B(x,y)<100

ring(x,y)=1;

else

ring(x,y)=0;

end

end

end

for x = 1 : a

for y = 1: b

if B(x,y)>90 && R(x,y)<150 && G(x,y)<110

middle(x,y)=1;

else

middle(x,y)=0;

end

end

end

for x = 1 : a

for y = 1: b

if R(x,y)>240 && G(x,y)>200 && B(x,y)<100

index(x,y)=1;

else

index(x,y)=0;

end

end

end

for x = 1 : a

for y = 1: b

if R(x,y)>240 && G(x,y)>100 && G(x,y)<170 && B(x,y)<20

thumb(x,y)=1;

else

thumb(x,y)=0;

end

end

end

mask=little+ring+middle+index+thumb;

for x = 1:a

for y=1:b

%these thresholds allow to isolate only the skin

if mask(x,y)==1 || R(x,y)>220 || G(x,y)>100 || B(x,y)>50 || R(x,y)<80 || G(x,y)<20

Mask2(x,y)=0;

else

Mask2(x,y)=1;

end %end if

end %end for

end %end for

%(van der Linde,2017)

kx = [+1 0;

0 -1];

ky = [ 0 +1;

-1 0];

W = size(kx,1);

Lx = zeros(a,b);

Ly = zeros(a,b);

for i=1+floor(W/2):a-floor(W/2)

for j=1+floor(W/2):b-floor(W/2)

window = Mask2(i-floor(W/2):i+floor(W/2)-1,j-floor(W/2):j+floor(W/2)-1);

outputValue = sum(sum( (double(window).\*fliplr(flipud(kx))) ));

Lx(i,j) = outputValue;

outputValue = sum(sum( (double(window).\*fliplr(flipud(ky))) ));

Ly(i,j) = outputValue;

end %end for

end%end for

rob = sqrt(Lx.^2+Ly.^2); %result of Robert's

% (van der Linde,2017)

kx = [-1 0 1;- 1 0 1;- 1 0 1];

ky = [-1 -1 -1;0 0 0;1. 1 1];

W = size(kx,1);

Lx = zeros(a,b);

Ly = zeros(a,b);

for i=1+floor(W/2):a-floor(W/2)

for j=1+floor(W/2):b-floor(W/2)

window = Mask2(i-floor(W/2):i+floor(W/2),j-floor(W/2):j+floor(W/2));

outputValue = sum(sum( (double(window).\*fliplr(flipud(kx))) ));

Lx(i,j) = outputValue;

outputValue = sum(sum( (double(window).\*fliplr(flipud(ky))) ));

Ly(i,j) = outputValue;

end %end for

end %end for

prew = sqrt(Lx.^2+Ly.^2); %result of Prewitt

%(van der Linde,2017)

kx = [-1 0 1;- 2 0 2;- 1 0 1];

ky = [-1 -2 -1;0 0 0;1. 2 1];

W = size(kx,1);

Lx = zeros(a,b);

Ly = zeros(a,b);

for i=1+floor(W/2):a-floor(W/2)

for j=1+floor(W/2):b-floor(W/2)

window = Mask2(i-floor(W/2):i+floor(W/2),j-floor(W/2):j+floor(W/2));

outputValue = sum(sum( (double(window).\*fliplr(flipud(kx))) ));

Lx(i,j) = outputValue;

outputValue = sum(sum( (double(window).\*fliplr(flipud(ky))) ));

Ly(i,j) = outputValue;

end %end for

end %end for

sob = sqrt(Lx.^2+Ly.^2); %result of Sobel

%(van der Linde,2017)

k = [0 -1 0;- 1 4 -1;0 -1 0];

W = size(k,1);

can = Mask2; % Will leave border unconvolved

for i=1+floor(W/2):a-floor(W/2)

for j=1+floor(W/2):b-floor(W/2)

window = Mask2(i-floor(W/2):i+floor(W/2),j-floor(W/2):j+floor(W/2));

outputValue = sum(sum( (double(window).\*k) ));

can(i,j) = uint8(outputValue);

end

end

%creates a binary mask to use below for activecontour

for x = 1:a

for y = 1:b

if rob(x,y)>0

mask(x,y)=true;

else

mask(x,y)=false;

end %end if

end %end for

end %end for

%calculates the boundries for Robert's

ro = activecontour(rob, mask,20,'edge');

visboundaries(ro,'Color','r');

%creates a binary mask to use below for activecontour

for x = 1:a

for y = 1:b

if prew(x,y)>0

mask(x,y)=true;

else

mask(x,y)=false;

end %end if

end %end for

end %end for

%calculates the boundries for Prewitt

pr = activecontour(prew, mask,20,'edge');

visboundaries(pr,'Color','r');

%creates a binary mask to use below for activecontour

for x = 1:a

for y = 1:b

if sob(x,y)>0

mask(x,y)=true;

else

mask(x,y)=false;

end %end if

end %end for

end %end for

%calculates the boundries for Sobel

so = activecontour(sob, mask,20,'edge');

visboundaries(so,'Color','r');

%creates a binary mask to use below for activecontour

for x = 1:a

for y = 1:b

if can(x,y)>0

mask(x,y)=true;

else

mask(x,y)=false;

end %end if

end %end for

end %end for

%calculates the boundries for Canny

ca = activecontour(can, mask,20,'edge');

visboundaries(ca,'Color','r');

Robert = RGBHand;

%uses boundries calcuated from Roberts and inputs it into image

for x=1:a

for y=1:b

if ro(x,y) == true

Robert(x,y,1)=255;

Robert(x,y,2)=0;

Robert(x,y,3)=0;

end %end if

end %end for

end %end for

Prewitt = RGBHand;

%uses boundries calcuated from Prewitt and inputs it into image

for x=1:a

for y=1:b

if pr(x,y) == true

Prewitt(x,y,1)=255;

Prewitt(x,y,2)=0;

Prewitt(x,y,3)=0;

end %end if

end %end for

end %end for

Sobel = RGBHand;

%uses boundries calcuated from Sobel and inputs it into image

for x=1:a

for y=1:b

if so(x,y) == true

Sobel(x,y,1)=255;

Sobel(x,y,2)=0;

Sobel(x,y,3)=0;

end %end if

end %end for

end %end for

Canny = RGBHand;

%uses boundries calcuated from Canny and inputs it into image

for x=1:a

for y=1:b

if ca(x,y) == true

Canny(x,y,1)=255;

Canny(x,y,2)=0;

Canny(x,y,3)=0;

end %end if

end %end for

end %end for

%Displays all 4 images.

image = figure;

imagesc([Robert Prewitt ;Sobel Canny]);

axis image;

axis off;

saveas(image,'A6.jpg');



Figure 6:output from A6 (top-Left:red trace from Robert’s filter, top-Right: red trace from Prewitt’s filter, bottom-Left: red trace from Sobel’s filter, bottom-Right:red trace from Canny filter)

Robert’s Kernal: Eq 6.1

Prewitt’s Kernal: Eq 6.2

Sobel’s Kernal Eq 6.3

Canny Kernal Eq 6.4

The code is not fully functional, as it does not show the results of the Canny filter (Eq 6.4). However, when we compare the remaining other results, the Sobel filter (Eq 6.3) gives the best results, as the outline is the thickest. The output is given in the correct format and the remaining three filters give the accurate result.

# Section B

## 1

clear all;

close all;

clc;

%read into variables

RGBH1=imread('hand1.jpeg');

RGBH2=imread('hand2.jpeg');

RGBH3=imread('hand3.jpeg');

RGBH4=imread('hand4.jpeg');

RGBH5=imread('hand5.jpeg');

%declare the M and N

[M,N]= size(RGBH1(:,:,1));

%isolates the little finger tip

littleSize=0;

for x = 1 : M

for y = 1 : N

if RGBH1(x,y,1)>235 && RGBH1(x,y,2)<70 && RGBH1(x,y,3)<50

little1(x,y)=1;

littleSize=littleSize+1; %gets cardinality of little

else

little1(x,y)=0;

end%end if

end%end for

end%end for

%isolates the ring finger tip

ringSize=0;

for x = 1 : M

for y = 1: N

if RGBH1(x,y,2)>50 && RGBH1(x,y,1)<40 && RGBH1(x,y,3)<80

ring1(x,y)=1;

ringSize=ringSize+1;%gets cardinality of ring

else

ring1(x,y)=0;

end%end if

end%end for

end%end for

middleSize=0;

%isolates middle finger tip

for x = 1 : M

for y = 1: N

if RGBH1(x,y,3)>75 && RGBH1(x,y,1)<30 && RGBH1(x,y,3)<110

middle1(x,y)=1;

middleSize=middleSize+1;%gets cardinality of middle

else

middle1(x,y)=0;

end%end if

end%end for

end%end for

indexSize=0;

%isolates index finger tip

for x = 1 : M

for y = 1: N

if RGBH1(x,y,1)>230 && RGBH1(x,y,2)>150 && RGBH1(x,y,3)<110

index1(x,y)=1;

indexSize=indexSize+1;%gets cardinality of index

else

index1(x,y)=0;

end%end if

end%end for

end%end for

thumbSize=0;

%isolates thumb finger tip

for x = 1 : M

for y = 1: N

if RGBH1(x,y,1)>230 && RGBH1(x,y,2)>75 && RGBH1(x,y,2)<150 && RGBH1(x,y,3)<15

thumb1(x,y)=1;

thumbSize=thumbSize+1;%gets cardinality of thumb

else

thumb1(x,y)=0;

end%end if

end%end for

end%end for

%gets average X and Y of little

[posY,posX]=find(little1);

little\_X=round(sum(posX)/littleSize);

little\_Y=round(sum(posY)/littleSize);

%gets average X and Y of ring

[posY,posX]=find(ring1);

ring\_X=round(sum(posX)/ringSize);

ring\_Y=round(sum(posY)/ringSize);

%gets average X and Y of middle

[posY,posX]=find(middle1);

middle\_X=round(sum(posX)/middleSize);

middle\_Y=round(sum(posY)/middleSize);

%gets average X and Y of Index

[posY,posX]=find(index1);

index\_X=round(sum(posX)/indexSize);

index\_Y=round(sum(posY)/indexSize);

%gets average X and Y of Thumb

[posY,posX]=find(thumb1);

thumb\_X=round(sum(posX)/thumbSize);

thumb\_Y=round(sum(posY)/thumbSize);

%gets distances between thumb and other fingers

thumb\_to\_index1=round(sqrt((thumb\_X-index\_X)^2 + (thumb\_Y-index\_Y)^2));

thumb\_to\_middle1=round(sqrt((thumb\_X-middle\_X)^2 + (thumb\_Y-middle\_Y)^2));

thumb\_to\_ring1=round(sqrt((thumb\_X-ring\_X)^2 + (thumb\_Y-ring\_Y)^2));

thumb\_to\_little1=round(sqrt((thumb\_X-little\_X)^2 + (thumb\_Y-little\_Y)^2));

%repeats for teh second image

[M,N]= size(RGBH2(:,:,1));

little1=0;

littleSize=0;

for x = 1 : M

for y = 1 : N

if RGBH2(x,y,1)>235 && RGBH2(x,y,2)<70 && RGBH2(x,y,3)<50

little1(x,y)=1;

littleSize=littleSize+1;

else

little1(x,y)=0;

end

end

end

ring1=0;

ringSize=0;

for x = 1 : M

for y = 1: N

if RGBH2(x,y,2)>50 && RGBH2(x,y,1)<40 && RGBH2(x,y,3)<80

ring1(x,y)=1;

ringSize=ringSize+1;

else

ring1(x,y)=0;

end

end

end

middle1=0;

middleSize=0;

for x = 1 : M

for y = 1: N

if RGBH2(x,y,3)>75 && RGBH2(x,y,1)<30 && RGBH2(x,y,3)<110

middle1(x,y)=1;

middleSize=middleSize+1;

else

middle1(x,y)=0;

end

end

end

index1=0;

indexSize=0;

for x = 1 : M

for y = 1: N

if RGBH2(x,y,1)>230 && RGBH2(x,y,2)>150 && RGBH2(x,y,3)<110

index1(x,y)=1;

indexSize=indexSize+1;

else

index1(x,y)=0;

end

end

end

thumb1=0;

thumbSize=0;

for x = 1 : M

for y = 1: N

if RGBH2(x,y,1)>230 && RGBH2(x,y,2)>75 && RGBH2(x,y,2)<150 && RGBH2(x,y,3)<15

thumb1(x,y)=1;

thumbSize=thumbSize+1;

else

thumb1(x,y)=0;

end

end

end

[posY,posX]=find(little1);

little\_X=round(sum(posX)/littleSize);

little\_Y=round(sum(posY)/littleSize);

[posY,posX]=find(ring1);

ring\_X=round(sum(posX)/ringSize);

ring\_Y=round(sum(posY)/ringSize);

[posY,posX]=find(middle1);

middle\_X=round(sum(posX)/middleSize);

middle\_Y=round(sum(posY)/middleSize);

[posY,posX]=find(index1);

index\_X=round(sum(posX)/indexSize);

index\_Y=round(sum(posY)/indexSize);

[posY,posX]=find(thumb1);

thumb\_X=round(sum(posX)/thumbSize);

thumb\_Y=round(sum(posY)/thumbSize);

thumb\_to\_middle2=(round(sqrt((thumb\_X-middle\_X)^2 + (thumb\_Y-middle\_Y)^2)));

thumb\_to\_ring2=(round(sqrt((thumb\_X-ring\_X)^2 + (thumb\_Y-ring\_Y)^2)));

thumb\_to\_little2=(round(sqrt((thumb\_X-little\_X)^2 + (thumb\_Y-little\_Y)^2)));

%repeats for 3rd image

[M,N]= size(RGBH3(:,:,1));

little1=0;

littleSize=0;

for x = 1 : M

for y = 1 : N

if RGBH3(x,y,1)>235 && RGBH3(x,y,2)<70 && RGBH3(x,y,3)<50

little1(x,y)=1;

littleSize=littleSize+1;

else

little1(x,y)=0;

end

end

end

ring1=0;

ringSize=0;

for x = 1 : M

for y = 1: N

if RGBH3(x,y,2)>50 && RGBH3(x,y,1)<40 && RGBH3(x,y,3)<80

ring1(x,y)=1;

ringSize=ringSize+1;

else

ring1(x,y)=0;

end

end

end

middle1=0;

middleSize=0;

for x = 1 : M

for y = 1: N

if RGBH3(x,y,3)>75 && RGBH3(x,y,1)<30 && RGBH3(x,y,3)<110

middle1(x,y)=1;

middleSize=middleSize+1;

else

middle1(x,y)=0;

end

end

end

index1=0;

indexSize=0;

for x = 1 : M

for y = 1: N

if RGBH3(x,y,1)>230 && RGBH3(x,y,2)>150 && RGBH3(x,y,3)<110

index1(x,y)=1;

indexSize=indexSize+1;

else

index1(x,y)=0;

end

end

end

thumb1=0;

thumbSize=0;

for x = 1 : M

for y = 1: N

if RGBH3(x,y,1)>230 && RGBH3(x,y,2)>75 && RGBH3(x,y,2)<150 && RGBH3(x,y,3)<15

thumb1(x,y)=1;

thumbSize=thumbSize+1;

else

thumb1(x,y)=0;

end

end

end

[posY,posX]=find(little1);

little\_X=round(sum(posX)/littleSize);

little\_Y=round(sum(posY)/littleSize);

[posY,posX]=find(ring1);

ring\_X=round(sum(posX)/ringSize);

ring\_Y=round(sum(posY)/ringSize);

[posY,posX]=find(middle1);

middle\_X=round(sum(posX)/middleSize);

middle\_Y=round(sum(posY)/middleSize);

[posY,posX]=find(index1);

index\_X=round(sum(posX)/indexSize);

index\_Y=round(sum(posY)/indexSize);

[posY,posX]=find(thumb1);

thumb\_X=round(sum(posX)/thumbSize);

thumb\_Y=round(sum(posY)/thumbSize);

thumb\_to\_index3=round(sqrt((thumb\_X-index\_X)^2+(thumb\_Y-index\_Y)^2));

thumb\_to\_ring3=round(sqrt((thumb\_X-ring\_X)^2 + (thumb\_Y-ring\_Y)^2));

thumb\_to\_little3=round(sqrt((thumb\_X-little\_X)^2 + (thumb\_Y-little\_Y)^2));

%repeats for 4th image

[M,N]= size(RGBH4(:,:,1));

little1=0;

littleSize=0;

for x = 1 : M

for y = 1 : N

if RGBH4(x,y,1)>235 && RGBH4(x,y,2)<70 && RGBH4(x,y,3)<50

little1(x,y)=1;

littleSize=littleSize+1;

else

little1(x,y)=0;

end

end

end

ring1=0;

ringSize=0;

for x = 1 : M

for y = 1: N

if RGBH4(x,y,2)>50 && RGBH4(x,y,1)<40 && RGBH4(x,y,3)<80

ring1(x,y)=1;

ringSize=ringSize+1;

else

ring1(x,y)=0;

end

end

end

middle1=0;

middleSize=0;

for x = 1 : M

for y = 1: N

if RGBH4(x,y,3)>75 && RGBH4(x,y,1)<30 && RGBH4(x,y,3)<110

middle1(x,y)=1;

middleSize=middleSize+1;

else

middle1(x,y)=0;

end

end

end

index1=0;

indexSize=0;

for x = 1 : M

for y = 1: N

if RGBH4(x,y,1)>230 && RGBH4(x,y,2)>150 && RGBH4(x,y,3)<110

index1(x,y)=1;

indexSize=indexSize+1;

else

index1(x,y)=0;

end

end

end

thumb1=0;

thumbSize=0;

for x = 1 : M

for y = 1: N

if RGBH4(x,y,1)>230 && RGBH4(x,y,2)>75 && RGBH4(x,y,2)<150 && RGBH4(x,y,3)<15

thumb1(x,y)=1;

thumbSize=thumbSize+1;

else

thumb1(x,y)=0;

end

end

end

[posY,posX]=find(little1);

little\_X=round(sum(posX)/littleSize);

little\_Y=round(sum(posY)/littleSize);

[posY,posX]=find(ring1);

ring\_X=round(sum(posX)/ringSize);

ring\_Y=round(sum(posY)/ringSize);

[posY,posX]=find(middle1);

middle\_X=round(sum(posX)/middleSize);

middle\_Y=round(sum(posY)/middleSize);

[posY,posX]=find(index1);

index\_X=round(sum(posX)/indexSize);

index\_Y=round(sum(posY)/indexSize);

[posY,posX]=find(thumb1);

thumb\_X=round(sum(posX)/thumbSize);

thumb\_Y=round(sum(posY)/thumbSize);

thumb\_to\_index4=round(sqrt((thumb\_X-index\_X)^2 + (thumb\_Y-index\_Y)^2));

thumb\_to\_middle4=round(sqrt((thumb\_X-middle\_X)^2 + (thumb\_Y-middle\_Y)^2));

thumb\_to\_little4=round(sqrt((thumb\_X-little\_X)^2 + (thumb\_Y-little\_Y)^2));

%repeats for 5th image

[M,N]= size(RGBH5(:,:,1));

little1=0;

littleSize=0;

for x = 1 : M

for y = 1 : N

if RGBH5(x,y,1)>235 && RGBH5(x,y,2)<70 && RGBH5(x,y,3)<50

little1(x,y)=1;

littleSize=littleSize+1;

else

little1(x,y)=0;

end

end

end

ring1=0;

ringSize=0;

for x = 1 : M

for y = 1: N

if RGBH5(x,y,2)>50 && RGBH5(x,y,1)<40 && RGBH5(x,y,3)<80

ring1(x,y)=1;

ringSize=ringSize+1;

else

ring1(x,y)=0;

end

end

end

middle1=0;

middleSize=0;

for x = 1 : M

for y = 1: N

if RGBH5(x,y,3)>75 && RGBH5(x,y,1)<30 && RGBH5(x,y,3)<110

middle1(x,y)=1;

middleSize=middleSize+1;

else

middle1(x,y)=0;

end

end

end

index1=0;

indexSize=0;

for x = 1 : M

for y = 1: N

if RGBH5(x,y,1)>230 && RGBH5(x,y,2)>150 && RGBH5(x,y,3)<110

index1(x,y)=1;

indexSize=indexSize+1;

else

index1(x,y)=0;

end

end

end

thumb1=0;

thumbSize=0;

for x = 1 : M

for y = 1: N

if RGBH5(x,y,1)>230 && RGBH5(x,y,2)>75 && RGBH5(x,y,2)<150 && RGBH5(x,y,3)<15

thumb1(x,y)=1;

thumbSize=thumbSize+1;

else

thumb1(x,y)=0;

end

end

end

[posY,posX]=find(little1);

little\_X=round(sum(posX)/littleSize);

little\_Y=round(sum(posY)/littleSize);

[posY,posX]=find(ring1);

ring\_X=round(sum(posX)/ringSize);

ring\_Y=round(sum(posY)/ringSize);

[posY,posX]=find(middle1);

middle\_X=round(sum(posX)/middleSize);

middle\_Y=round(sum(posY)/middleSize);

[posY,posX]=find(index1);

index\_X=round(sum(posX)/indexSize);

index\_Y=round(sum(posY)/indexSize);

[posY,posX]=find(thumb1);

thumb\_X=round(sum(posX)/thumbSize);

thumb\_Y=round(sum(posY)/thumbSize);

thumb\_to\_index5=round(sqrt((thumb\_X-index\_X)^2 + (thumb\_Y-index\_Y)^2));

thumb\_to\_middle5=round(sqrt((thumb\_X-middle\_X)^2 + (thumb\_Y-middle\_Y)^2));

thumb\_to\_ring5=round(sqrt((thumb\_X-ring\_X)^2 + (thumb\_Y-ring\_Y)^2));

%saves distances to array of distances between thumb and other fingers

indexs=[thumb\_to\_index1,thumb\_to\_index3,thumb\_to\_index4,thumb\_to\_index5];

middles=[thumb\_to\_middle1,thumb\_to\_middle2,thumb\_to\_middle4,thumb\_to\_middle5];

rings=[thumb\_to\_ring1,thumb\_to\_ring2,thumb\_to\_ring3,thumb\_to\_ring5];

littles=[thumb\_to\_little1,thumb\_to\_little2,thumb\_to\_little3,thumb\_to\_little4];

%prompts the user to indicate which image they would like to test

prompt='Please enter which image you would like to test (1,2,3,4,5): ';

i=input(prompt); %reads input

% selects the image to test

if i == 1

test=RGBH1;

end

if i == 2

test=RGBH2;

end

if i== 3

test=RGBH3;

end

if i==4

test=RGBH4;

end

if i ==5

test=RGBH5;

end

%gets the cardinality and insolates little finger tip in test

[m,n]= size(test(:,:,1));

littleTestSize=0;

for x = 1 : m

for y = 1 : n

if test(x,y,1)>235 && test(x,y,2)<70 && test(x,y,3)<50

littleTest(x,y)=1;

littleTestSize=littleTestSize+1;

else

littleTest(x,y)=0;

end

end

end

%gets the cardinality and insolates ring finger tip in test

ringTestSize=0;

for x = 1 : m

for y = 1: n

if test(x,y,2)>50 && test(x,y,1)<40 && test(x,y,3)<80

ringTest(x,y)=1;

ringTestSize=ringTestSize+1;

else

ringTest(x,y)=0;

end

end

end

%gets the cardinality and insolates middle finger tip in test

middleTestSize=0;

for x = 1 : m

for y = 1: n

if test(x,y,3)>75 && test(x,y,1)<30 && test(x,y,3)<110

middleTest(x,y)=1;

middleTestSize=middleTestSize+1;

else

middleTest(x,y)=0;

end

end

end

%gets the cardinality and insolates index finger tip in test

indexTestSize=0;

for x = 1 : m

for y = 1: n

if test(x,y,1)>230 && test(x,y,2)>150 && test(x,y,3)<110

indexTest(x,y)=1;

indexTestSize=indexTestSize+1;

else

indexTest(x,y)=0;

end

end

end

%gets the cardinality and insolates thumb finger tip in test

thumbTestSize=0;

for x = 1 : m

for y = 1: n

if test(x,y,1)>230 && test(x,y,2)>75 && test(x,y,2)<150 && test(x,y,3)<15

thumbTest(x,y)=1;

thumbTestSize=thumbTestSize+1;

else

thumbTest(x,y)=0;

end

end

end

%gets average X and Y of little from test

[posY,posX]=find(littleTest);

littleT\_X=round(sum(posX)/littleTestSize);

littleT\_Y=round(sum(posY)/littleTestSize);

%gets average X and Y of ring from test

[posY,posX]=find(ringTest);

ringT\_X=round(sum(posX)/ringTestSize);

ringT\_Y=round(sum(posY)/ringTestSize);

%gets average X and Y of middle from test

[posY,posX]=find(middleTest);

middleT\_X=round(sum(posX)/middleTestSize);

middleT\_Y=round(sum(posY)/middleTestSize);

%gets average X and Y of index from test

[posY,posX]=find(indexTest);

indexT\_X=round(sum(posX)/indexTestSize);

indexT\_Y=round(sum(posY)/indexTestSize);

%gets average X and Y of thumb from test

[posY,posX]=find(thumbTest);

thumbT\_X=round(sum(posX)/thumbTestSize);

thumbT\_Y=round(sum(posY)/thumbTestSize);

%gets test distances between thumb and other fingers

index=round(sqrt((thumbT\_X-indexT\_X)^2 + (thumbT\_Y-indexT\_Y)^2));

middle=round(sqrt((thumbT\_X-middleT\_X)^2 + (thumbT\_Y-middleT\_Y)^2));

ring=round(sqrt((thumbT\_X-ringT\_X)^2 + (thumbT\_Y-ringT\_Y)^2));

little=round(sqrt((thumbT\_X-littleT\_X)^2 + (thumbT\_Y-littleT\_Y)^2));

%determines the shape of the hand

if index<min(indexs)

prompt='The thumb is touching the index finger.';

i = input(prompt);

elseif middle<min(middles)

prompt='The thumb is touching the middle finger.';

i = input(prompt);

elseif ring<min(rings)

prompt='The thumb is touching the ring finger.';

i = input(prompt);

elseif little<min(littles)

prompt='The thumb is touching the little finger.';

i = input(prompt);

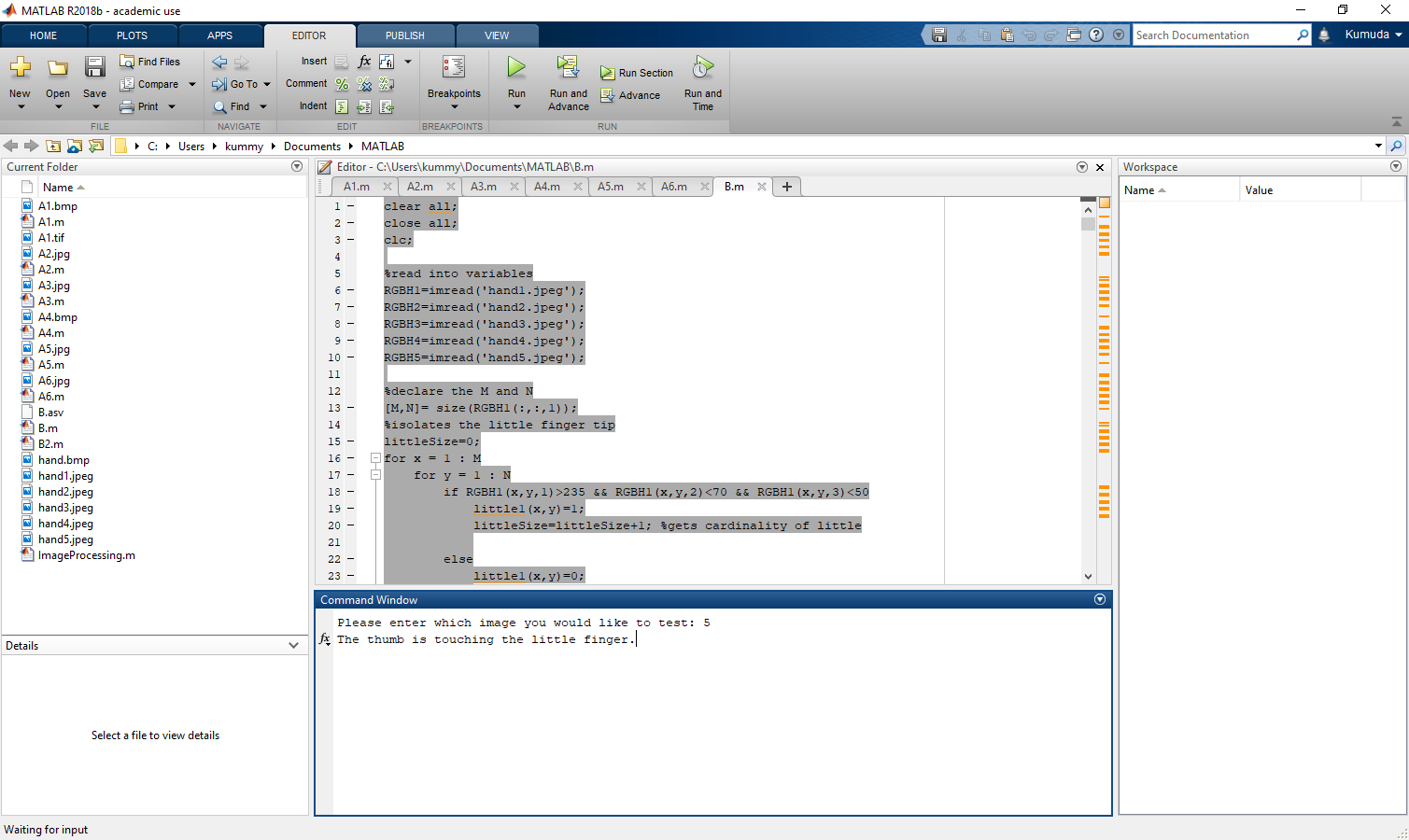
else

prompt='The hand is flat.';

i = input(prompt);

end

figure 7: output from B1



The code is fully functional, by which it means the code can detect the shapes the hand is in dependent on what which test is chosen. The average X and Y is calculated by using thresholding to isolate the finger tips. The average(Eq 5.1) X and Ys for each finger tip is then used to calculate the distance (Eq 5.2) between the thumbs and the other fingers. The same is then done on the test image chosen and the distances are compared to the ‘trained’ distances, if it is less than the trained distances, this indicates the two fingers are touching. This allows for the shape of the hand to be determined. The output is a simple text output that tells the user the shape of the hand. This particular functionality can be improved by adding more training data, so that the testing can become more accurate when testing various images and shapes. The thresholding can be improved by using a white light when producing images so the data is not skewed due to how a colour shows up under certain lights and using other coloured fingertips so skin tones do not clash with the finger tips.

# References

* Ian Van Der Linde (2017), laplacianLinearFilter [.m].