WITS UNIVERSITY
School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg

APPM4058A & COMS7238A

Project:

Counting Fingers

SM Curtis – 1657041

Problem Description

In this project, the number of fingers being held up on a colour image of a cropped hand need to be counted. The case where fingers are together and touch in the image does not need to be taken into account as input images. The output should be the number of fingers held up. The counting fingers problem can be used for different applications such as simply counting how many fingers are up on a hand (educational purposes) or even rock, paper, scissors (0 - rock, 5 - paper, 2 or 3 - scissors).

Input Images

The input images used in this project were taken from a hand gesture ASL dataset (https://www.kaggle.com/rraghav5600/hand-gesture-asl) that contains cropped images of hands on a black background. This project uses 75 images from this dataset to obtain a general and unbiased sample of hand images. The dataset contains five different people's hands holding up zero to five fingers as well as other types of hand gestures. From the images containing the general way in which people hold up zero to five fingers, two images were taken from each person's hand holding up the number of fingers zero to five. This results in 60 input images. The remaining 15 images come from images where the number of fingers held up is in a nonconventional way. The only images that were excluded from this dataset completely were hands where fingers were held up together. The method used in this project counts the correct number of fingers for all the input images.

Walk Through of Approach

The steps to solve this counting finger problem are as follows:

Do the following for each image in the dataset (loop through every image):

1. Get the name of the path to the image, img (1) to img (75) by updating the image number based on the current for loop number.

```
for i in range(1,76):

        name="fingers/img ("+str(i)+")"+".png" #path of fingers
```

2. Load the RGB image using this path.
3. Convert the RGB image to grayscale.

```
imgGray = rgb2gray(imgGray) #convert image from RGB to gray
```

4. Convert the grayscale image to a binary image using a threshold of 0 as the background of the cropped hand images is 0.
5. Create a square structuring element of size 10.
6. Perform binary opening on the image to remove any small noise around the hand while keeping the hand intact and the same size.

```
SE1=morphology.square(10)

imgBinary=morphology.binary_opening(imgBinaryRough,SE1).astype(int) #get rid of bac
kground noise
```

7. Create a disk structuring element of size 34.
8. Perform binary opening on the image using this structuring element to remove the fingers from the binary image (with no noise) to create a new (separate) image that only contains the palm.

```
SE2=morphology.disk(34)

imgPalm=morphology.binary_opening(imgBinary,SE2).astype(int) #removes fingers from
palm
```

9. Take the binary image with no noise and subtract the palm image away from it to leave behind only the fingers of the hand.

```
imgFingersRough=imgBinary-imgPalm # get rid of the palm section of the hand
```

10. Create a square structuring element of size 25.
11. Perform binary opening on the fingers image using this structuring element to remove any small noise left behind from the subtraction in step 9.

```
SE3=morphology.square(25)

imgFingers=morphology.binary_opening(imgFingersRough,SE3) #remove any small boarder
 forground pixels remaining from full hand
```

12. Display all images next to each other (RGB image, grayscale image, binary image, the binary image with no noise, palm image, full hand – palm image and final fingers image)
13. Perform connect component labelling on the fingers image with no noise.
14. Count the number of connected components excluding the background. This number is the number of fingers in the image. Print the number of fingers held up in the image.

```
num=measure.label(imgFingers) #labels the connected components

connectedComponents = np.unique(num) #find the connected component number

numFingers = np.count_nonzero(connectedComponents) #count the connected component n
umber excluding 0
```

Reasons for steps

- The input RGB image was converted to grayscale so that thresholding was able to be performed on the image.
- A threshold of 0 was used as the background of the image is black (0) so any pixel number other than 0 belong to the hand.
- Binary opening (erosion then dilation) with a small structuring element was used to remove any small pieces of noise surrounding the image while keeping the hand the same size. The noise could remain in the final fingers image and the connected component labelling with then give the wrong number of fingers held up so noise is removed here to avoid this.
- A binary opening is done again on that image with a larger disk structuring element as to remove the fingers from the image while leaving the same size palm behind. A disk structuring element was used as a finger is fairly round so binary opening removes the fingers best with a disk.
- You only want the fingers of the image to remain so subtracting the palm image from the binary hand image leaves these fingers behind but with surrounding small pieces of noise left behind from the palm.
- The small square structuring element was used with a binary opening to remove this noise left behind by the palm leaving only the number of fingers behind. Any noise that remains in the final fingers image will cause the connected component labelling to give the wrong number of fingers held up so noise is removed here to avoid this.
- Lastly, the labelling was done to count the number of fingers as these fingers are separate from each other and the result is a matrix of connected components. The unique method finds the connected component numbers from this labelling and stores them in a list, zero to n. These unique numbers are then counted (excluding 0 as 0 is background) to count the number of unique connected components, which is the number of fingers in the image.

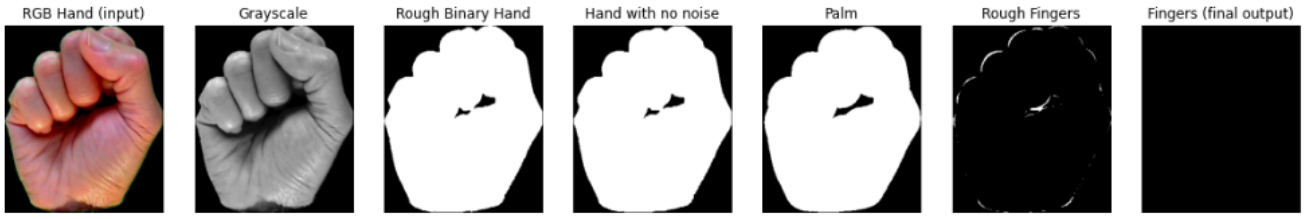
Built-in Methods Used

No algorithms from the DIP course were used.

The following built-in methods were used:

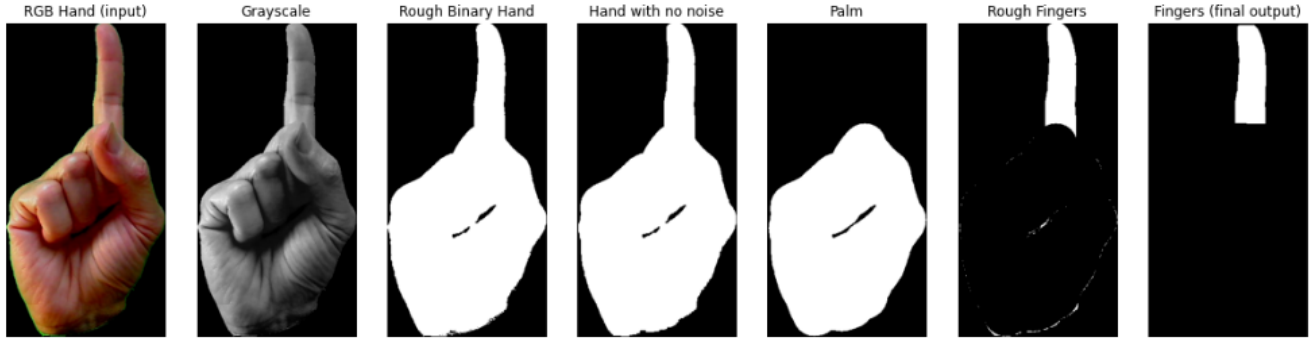
- rgb2gray – convert the RGB image to grayscale.
- morphology.square – create a square structuring element.
- morphology.disk – create a circle structuring element.
- morphology.binary_opening – performs binary opening on the image.
- measure.label – used for connected component labelling.
- np.unique – find the connected component numbers.
- np.count_nonzero – counts the number of connected components excluding 0 (number of fingers)

Examples of Input and Output



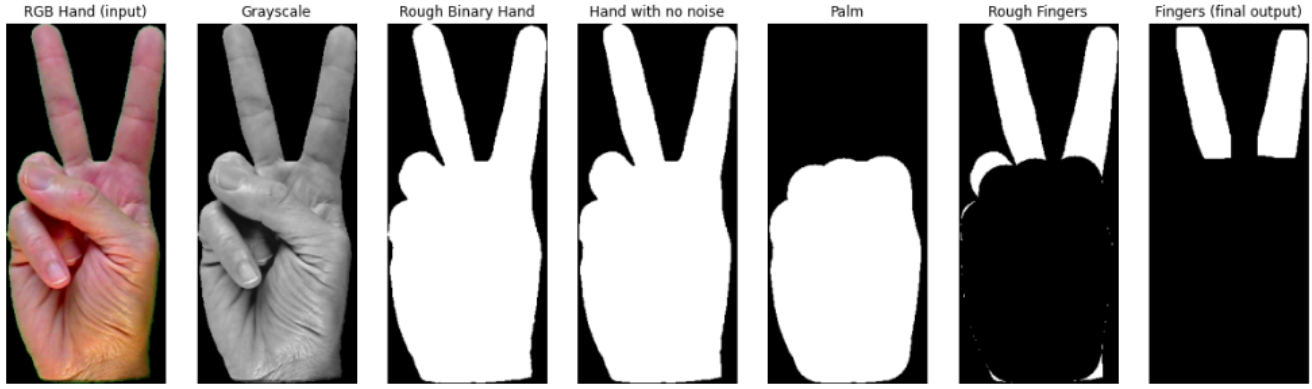

```
Name of image:  fingers/img (1).png
Number of Fingers:  0
```

Figure 1: Process of removing palm from fingers and counting fingers



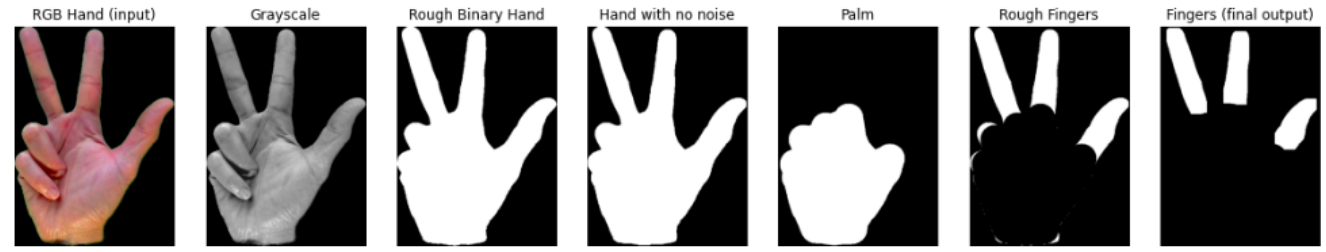

```
Name of image:  fingers/img (11).png
Number of Fingers:  1
```

Figure 2: Process of removing palm from fingers and counting fingers



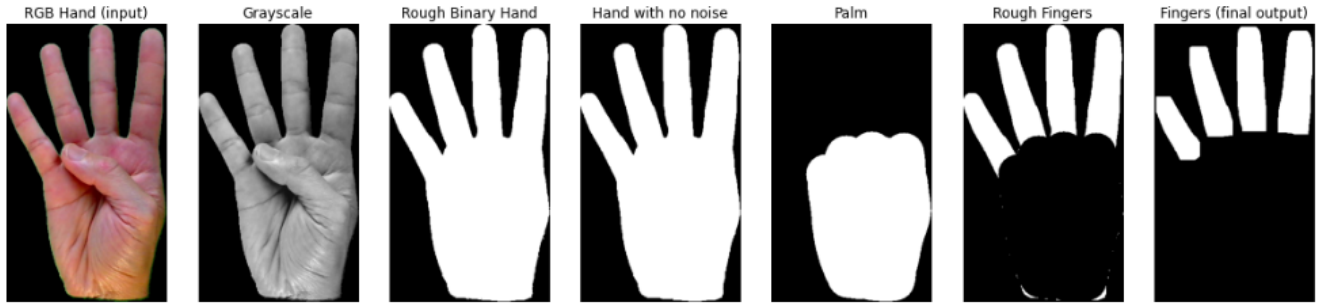

```
Name of image:  fingers/img (21).png
Number of Fingers:  2
```

Figure 3: Process of removing palm from fingers and counting fingers




```
Name of image:  fingers/img (31).png
Number of Fingers:  3
```

Figure 4: Process of removing palm from fingers and counting fingers

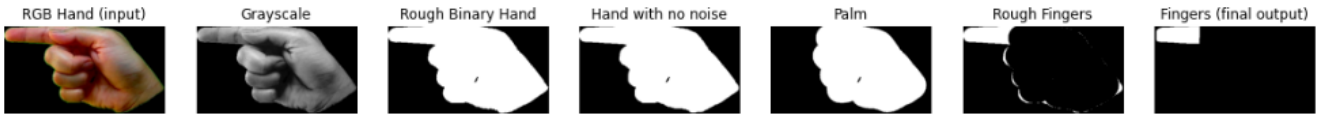



```
Name of image:  fingers/img (41).png
Number of Fingers:  4
```

Figure 5: Process of removing palm from fingers and counting fingers

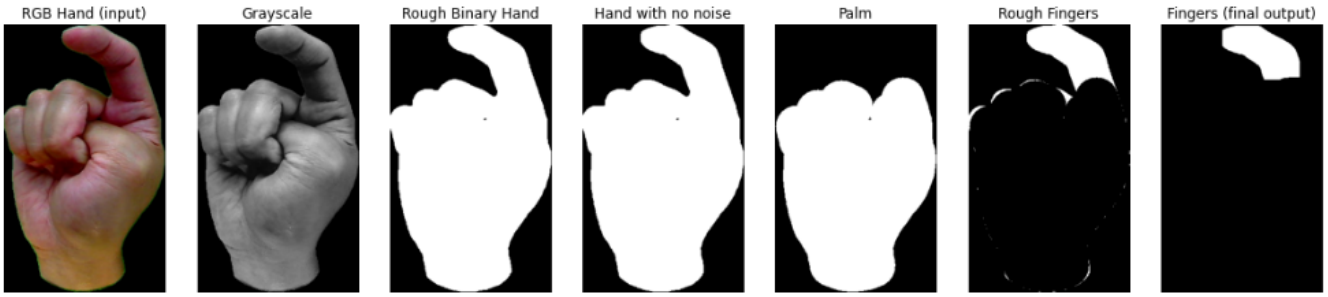

```
Name of image:  fingers/img (51).png
Number of Fingers:  5
```

Figure 6: Process of removing palm from fingers and counting fingers




```
Name of image:  fingers/img (63).png
Number of Fingers:  1
```

Figure 7: Process of removing palm from fingers and counting fingers




```
Name of image:  fingers/img (64).png
Number of Fingers:  1
```

Figure 8: Process of removing palm from fingers and counting fingers


```
Name of image:  fingers/img (65).png
Number of Fingers:  2
```

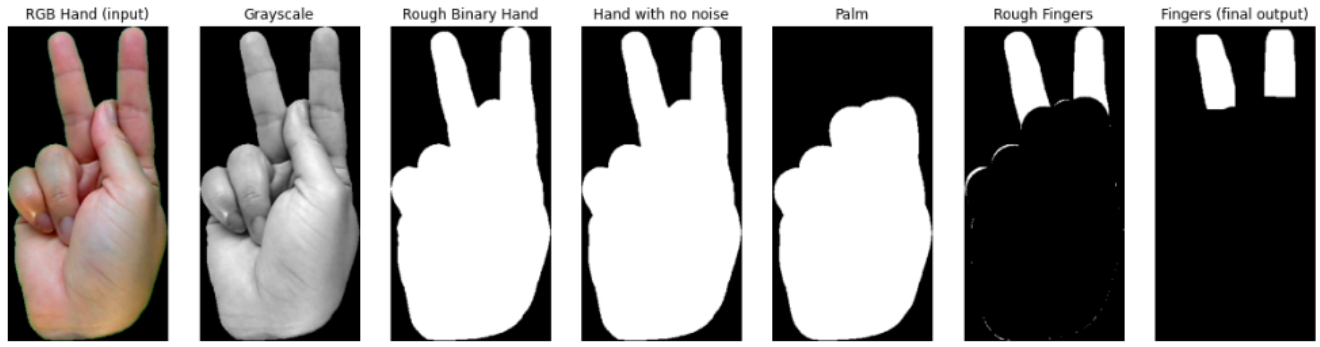
Figure 9: Process of removing palm from fingers and counting fingers


```
Name of image:  fingers/img (67).png
Number of Fingers:  2
```

Figure 10: Process of removing palm from fingers and counting fingers

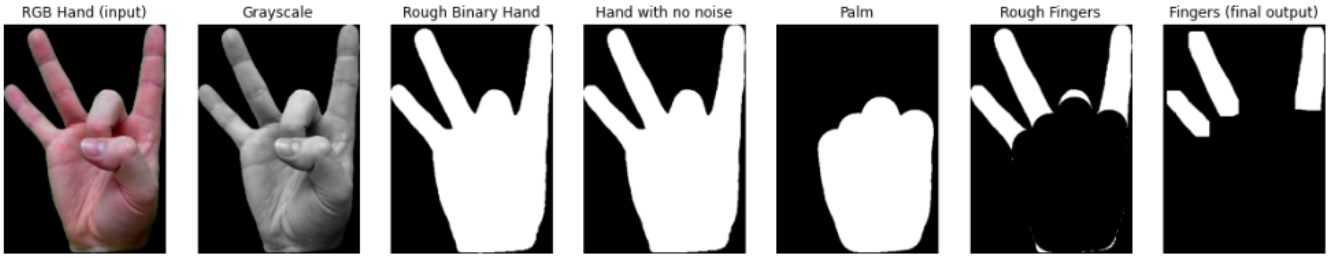

```
Name of image:  fingers/img (70).png
Number of Fingers:  2
```

Figure 11: Process of removing palm from fingers and counting fingers



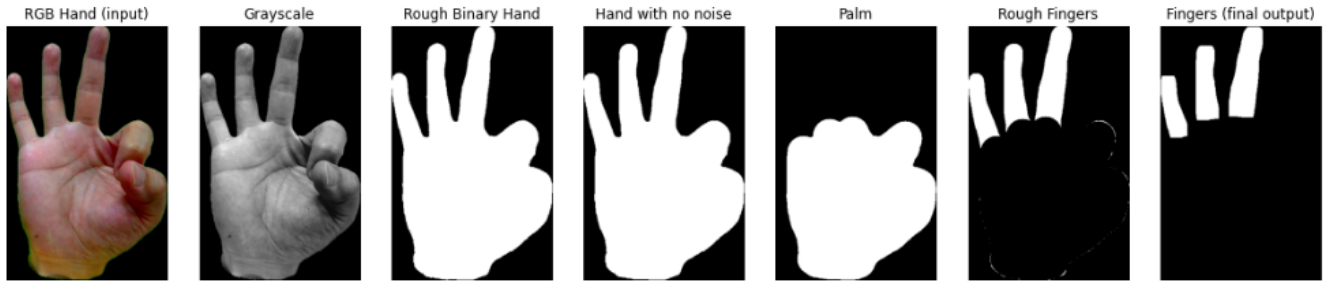

```
Name of image:  fingers/img (71).png
Number of Fingers:  2
```

Figure 12: Process of removing palm from fingers and counting fingers




```
Name of image:  fingers/img (73).png
Number of Fingers:  3
```

Figure 13: Process of removing palm from fingers and counting fingers




```
Name of image:  fingers/img (75).png
Number of Fingers:  3
```

Figure 14: Process of removing palm from fingers and counting fingers

Conclusion

The steps described in the Walk Through of Approach and Examples of Input and Output sections of this report shows how the method used works for all different types of input images and does not only work for certain images. This shows that the method used is generalised for multiple different examples and does not change for different input images to deal with them specifically as it works for all of them. The method is also correctly counting the number of fingers held up in all the input images.