Machine Perception (COMP 3007)

Assignment Semester 2, 2017

Student ID: 18249833

Name: Xhien Yi Tan (Xavier)

# 1.Overview

This assignment uses Python programming language and OpenCV. The goal is to find the barcode numbers and uses barcode as a hint where the numbers are.

The steps consist of:

1. Barcode number detection and localisation
2. Barcode digit extraction
3. Barcode number recognition
4. Complete perception pipeline

Please note that before running any shell script, please use this command “chmod u+x task1.sh” to allow the shell script to be executed. Otherwise, the error Permission Denied will show out. In task2.sh and task3.sh, I will create one more folder called barcode5 in case there are five images for input.

# 2.Methodology

Task 1 – Barcode number detection and localisation:

Main methodologies for this task are Sobel and different types of Morphology.

* First by reading images from directory.
* Get the image name for saving purposes. E.g., img1.jpg will get me 1.
* Start to process the image by scaling and rotation if needed.
* If an image is scaled and it can’t be further process, my program will catch the exception and then rotate this image until it can be further process.
* If everything is good after image processing, convert it to gray scale.
* Use Sobel to get horizontal and vertical gradients.
* Use this gradient to filter out the noise then apply binary threshold.
* This threshold will result in any pixel the gradient image that is not greater than 225 is set to 0 which is black, otherwise, the pixel is set to 255 which is white.
* Then perform a series of dilation and erosion to find the suitable thresholded image.
* This will leave me with a barcode area, then I crop this part out.
* After getting this area, I then apply binary and otsu thresh and also a series closing and erosion. The reason of doing this is to refine the barcode area so that I can find the contour and draw a rectangle box more precisely around the barcode area.
* After getting the barcode area, then I crop this area by a specific percentage so that I get the number area.

Task 2 - Barcode digit extraction:

* First by reading images from directory.
* Get the image name for saving purposes. E.g., barcode1.png will get me 1.
* Convert the image into a gray scale then apply adaptiveThreshold.
* After that find contours using RETR\_EXTERNAL which draws a rectangle box around each digit and it will also prevent drawing a smaller box inside the digit itself.
* In order to prevent overlapping the existing rectangle, I have set some constraints on height and weight.
* Then I will store each cropped digit into the directory and also store the name of the digit (e.g. d01.png) into a dictionary along with its x value.
* After cropping all digits, then I will sort them according to their x values, this will make them to have the same order as the barcode itself.

Task 3 – Barcode number recognition:

* Before running this task, I have trained my model using supervised learning. The data I used for training are the digits extracted from task 2. I stored the pixel value for each digit into an array then append it to a file.
* Load my predictive model which is the file from previous step.
* Initialise KNearest to train my model.
* Read images from directory then make it into gray scale and apply adaptiveThreshold
* Steps here are similar to task 2, find contours and then draw a rectangle box around each image.
* Then resize the image into 10x10 and convert into matrix float point.
* Then use KNearest with k = 1 to classify the digit.
* Finally, store the classified digit into a text file.

Task 4 – Complete perception pipeline:

A combination of task 1, 2 and 3, then save the output as a single text file instead different text files.

# 3.Evaluation on the Validation Sets

Task 1:

My task 1 only works on the “easy” barcode at the moment. It only works correctly on at least one image on validation sets. However, it works 80% correctly on the training sets provided by Sonny.

The results for task 1:

Screen%20Shot%202017-10-12%20at%201.57.14%20pm.png

Figure 1: Result of img1.jpg

Screen%20Shot%202017-10-12%20at%201.58.42%20pm.png

Figure 2: Result of img2.jpg

Screen%20Shot%202017-10-12%20at%201.58.56%20pm.png

Figure 3: Result of img3.jpg

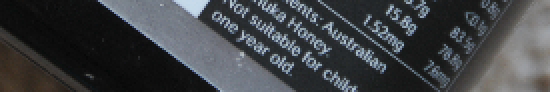


Figure 4: Result of img4.jpg

Screen%20Shot%202017-10-12%20at%201.56.59%20pm.png

Figure 5: Result of img5.jpg

Task 2:

My task 2 works well on the validation sets. It segments the digits correctly.

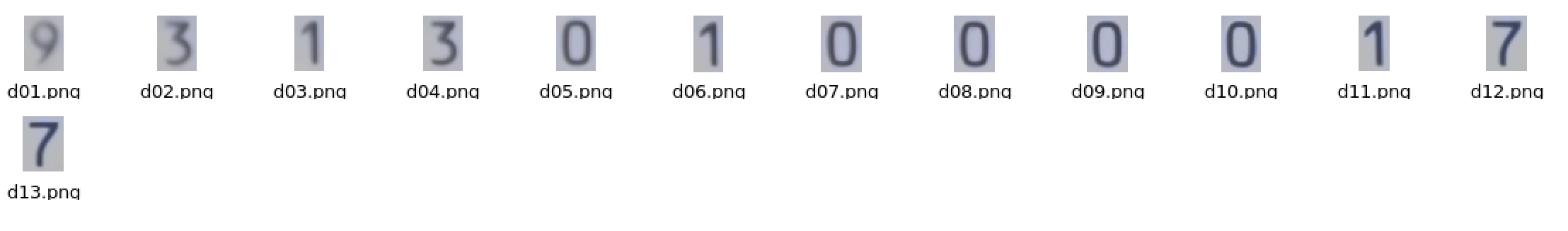


Figure 6: Result of barcode1.png

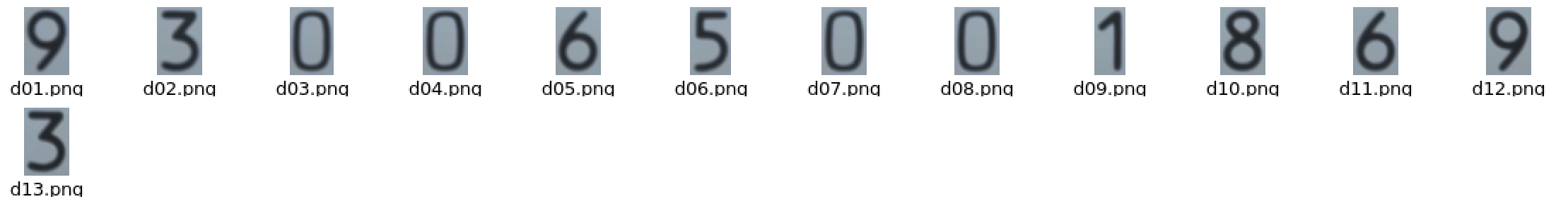


Figure 7: Result of barcode2.png

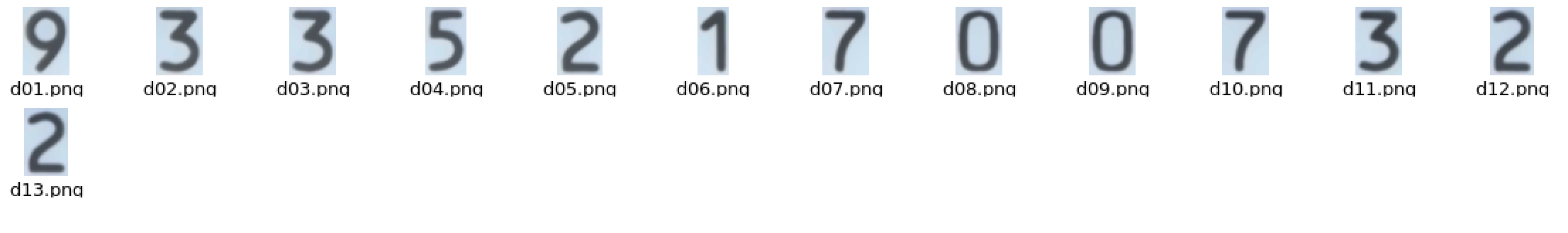


Figure 8: Result of barcode3.png

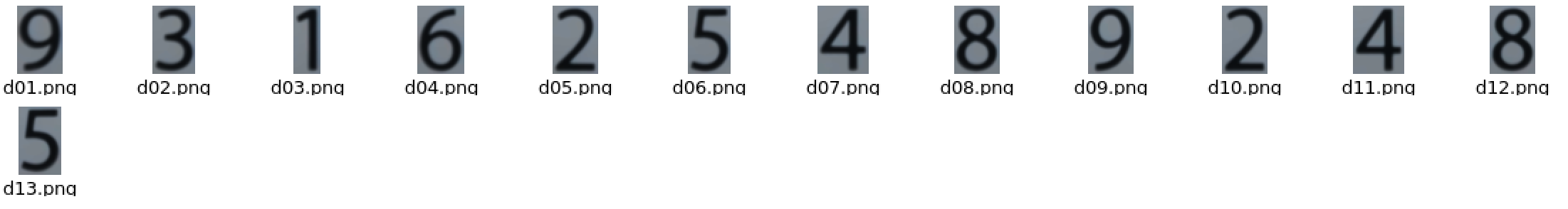


Figure 9: barcode4.png

Task 3:

My task 3 has an accuracy of 80%. Some digits are still misclassified. Because my task 3 outputs in txt file, I will not provide any screenshots for this, instead I will provide the result in words.

For each digit in barcode1.png:

The expected output: 9313010000177

My program’s output: 0313010000107

For each digit in barcode2.png:

The expected output: 9300650018693

My program’s output: 9300650018693

For each digit in barcode3.png:

The expected output: 9335217007322

My program’s output: 9335817207422

For each digit in barcode4.png:

The expected output: 9316254892485

My program’s output: 9414254892485

Task 4:

For figure 1: my program produces 901625989241

For figure 2: my program produces 3377

For figure 3: my program produces nothing

For figure 4: my program produces 9444

For figure 5: my program produces 9300350010939

# 5.References

Barcode Detection:

<https://www.pyimagesearch.com/2014/11/24/detecting-barcodes-images-python-opencv/>

Barcode number training and recognition:

<https://stackoverflow.com/questions/9413216/simple-digit-recognition-ocr-in-opencv-python>