Introduction to Recognition Systems: Character Recognition Using Moments

Efe Ertekin
Student Number: 041701006
Department of Computer Engineering

Mef University
02/03/2020

Abstract

This project encourages students to understand the basis of recognition systems. Also pushes the students to understand already written codes that create the foundation of this project. There are many ways to use recognition systems in the world. For instance, governments use face recognition systems to keep their society safe. With this project, students can be aware of what happens behind these systems.

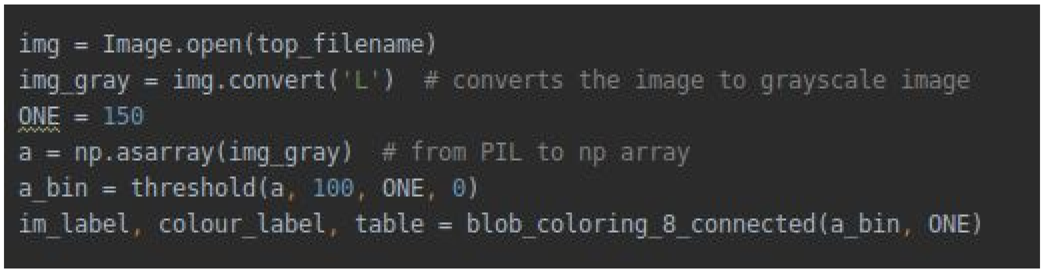
This paper gives information about 8-blob colouring algorithm, calculation of moments, where they are used and comparisons of these moments to make right guesses. Also provides graphical user interfaces' basic ideas.

It is hoped this project will inform students about the basis of recognition systems and give them a chance to take another step into this topic.

Description of the Project

Reading and binarization of image

We take a picture in a computer environment, first convert it to gray format and then return it to binary form. After that, make it usable to make array implementations. Our goal to do this is to work with 0's and 1's. 0s represent the background, while 1s represent parts of numbers or letters.



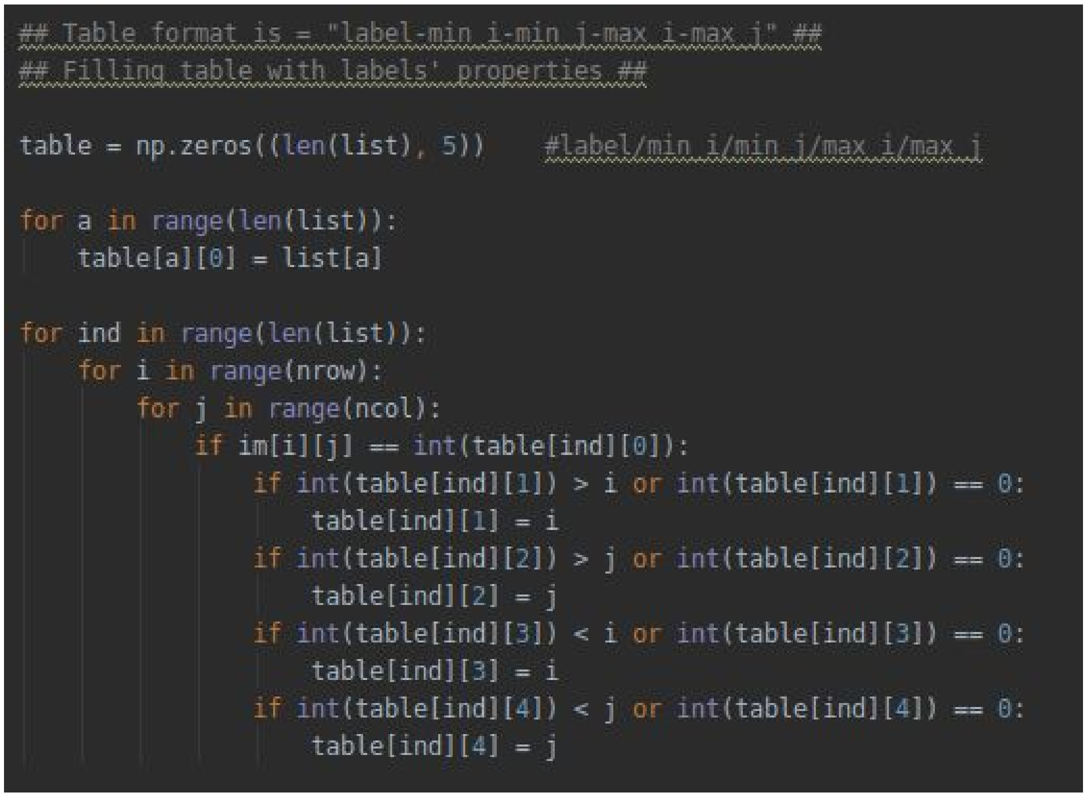
```
img = Image.open(top_filename)
img_gray = img.convert('L')  # converts the image to grayscale image
ONE = 150
a = np.asarray(img_gray)  # from PIL to np array
a_bin = threshold(a, 100, ONE, 0)
im_label, colour_label, table = blob_coloring_8_connected(a_bin, ONE)
```

Fig1. img -> grayscale image -> np array

Design and implement character detection

To do the character determination, we first take the image array consisting of 0's and 1's. We get the result of the 8-blob-coloring's labeling process on this array. Each character is assigned a different label (number). Thus, it is possible to determine how many different characters are in the picture.

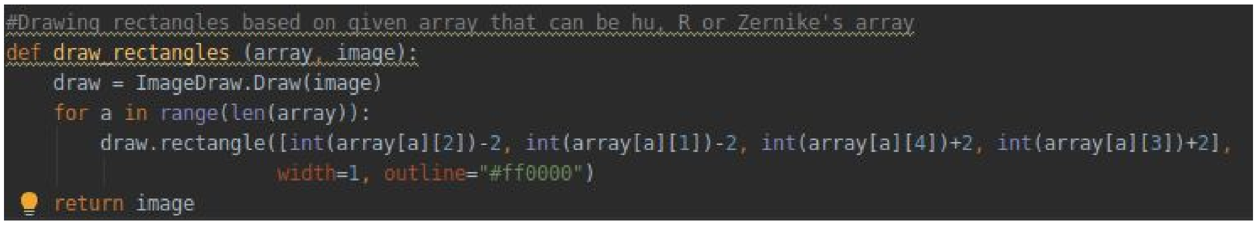
To find where these characters are in the picture, we scan for specific labels using the for loop. While doing this scan, we also update the max and min values of that label to an array. The values of max and min represent the pixel numbers of the picture.



```
## Table format is = "label-min i-min j-max i-max j" ##
## Filling table with labels' properties ##

table = np.zeros((len(list), 5))    #label/min i/min j/max i/max j

for a in range(len(list)):
    table[a][0] = list[a]

for ind in range(len(list)):
    for i in range(nrow):
        for j in range(ncol):
            if im[i][j] == int(table[ind][0]):
                if int(table[ind][1]) > i or int(table[ind][1]) == 0:
                    table[ind][1] = i
                if int(table[ind][2]) > j or int(table[ind][2]) == 0:
                    table[ind][2] = j
                if int(table[ind][3]) < i or int(table[ind][3]) == 0:
                    table[ind][3] = i
                if int(table[ind][4]) < j or int(table[ind][4]) == 0:
                    table[ind][4] = j
```

Fig2. Array that keeps max and min values for labels

We use the array we hold the max and min numbers for each label, and draw a rectangle around the characters with the Pillow library.

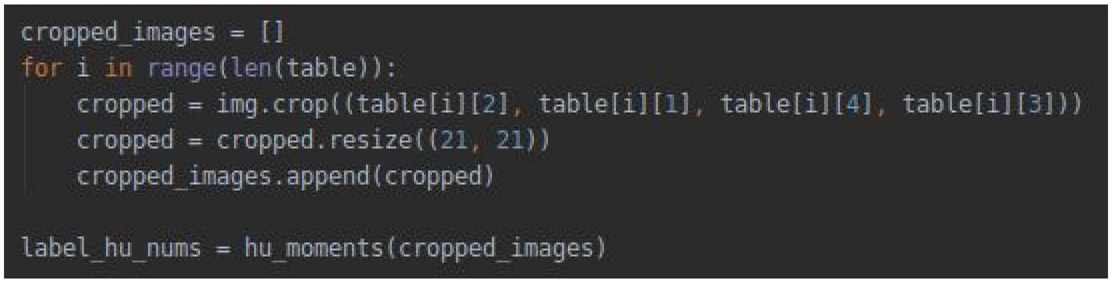


```python
#Drawing rectangles based on given array that can be hu, R or Zernike's array
def draw_rectangles (array, image):
    draw = ImageDraw.Draw(image)
    for a in range(len(array)):
        draw.rectangle([int(array[a][2])-2, int(array[a][1])-2, int(array[a][4])+2, int(array[a][3])+2],
                       width=1, outline="#ff0000")
    return image
```

Fig3. Drawing Rectangles Function

Design and implement feature extraction and recognition

In this section, we need to use the characters we have determined separately. To do this, we take each rectangle and turn it into a photo. We make the dimensions of this 21x21. We export these corped pictures as inputs to the function we make calculations. These functions are HU Moments, R Moments and Zernike Moments, respectively. The output of these functions is an array with the moments of those images.

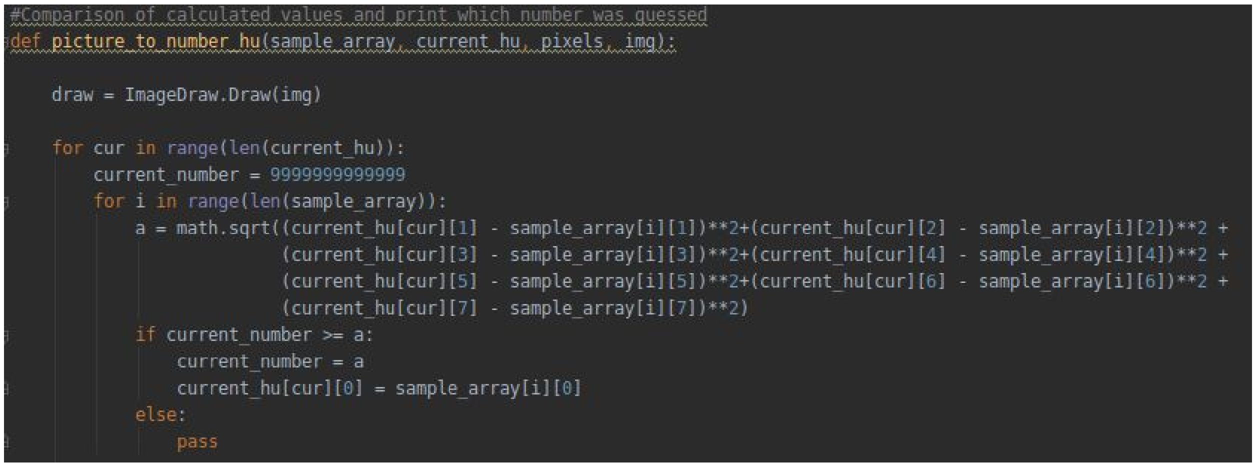


```python
cropped_images = []
for i in range(len(table)):
    cropped = img.crop((table[i][2], table[i][1], table[i][4], table[i][3]))
    cropped = cropped.resize((21, 21))
    cropped_images.append(cropped)

label_hu_nums = hu_moments(cropped_images)
```

Fig4. Creating cropped(21x21) images to calculate moments

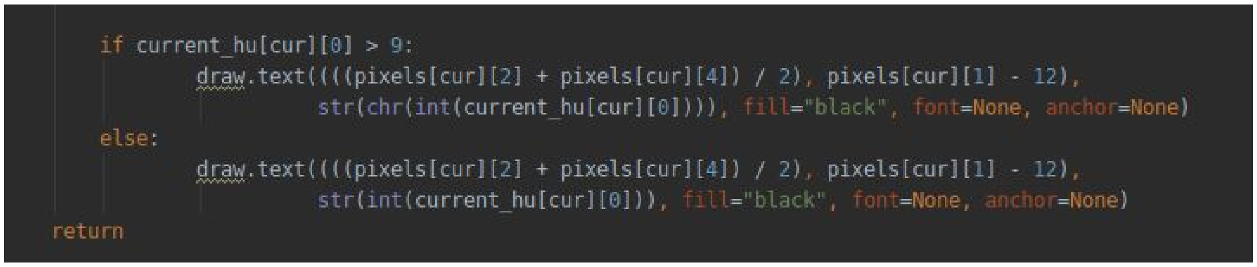
We take the array containing the moments of each picture and use it to create a database. For example, we find a picture of the 0's and throw it where we hold the 0's. The first column of this initially created array was left empty. Because the first column is designed to take a value according to whatever number it is.

Finally, using the distance formula, a loop is entered until the difference between the characters is at its lowest. When the lowest value is found, the character is written on the place where that character is located. We do this with ImageDraw, Pillow.



```
#Comparison of calculated values and print which number was guessed
def picture_to_number_hu(sample_array, current_hu, pixels, img):

    draw = ImageDraw.Draw(img)

    for cur in range(len(current_hu)):
        current_number = 9999999999999
        for i in range(len(sample_array)):
            a = math.sqrt((current_hu[cur][1] - sample_array[i][1])**2+(current_hu[cur][2] - sample_array[i][2])**2 +
                          (current_hu[cur][3] - sample_array[i][3])**2+(current_hu[cur][4] - sample_array[i][4])**2 +
                          (current_hu[cur][5] - sample_array[i][5])**2+(current_hu[cur][6] - sample_array[i][6])**2 +
                          (current_hu[cur][7] - sample_array[i][7])**2)
            if current_number >= a:
                current_number = a
                current_hu[cur][0] = sample_array[i][0]
            else:
                pass
```

Fig5. Comparison of Hu and naming



```
        if current_hu[cur][0] > 9:
                draw.text((((pixels[cur][2] + pixels[cur][4]) / 2), pixels[cur][1] - 12),
                          str(chr(int(current_hu[cur][0]))), fill="black", font=None, anchor=None)
        else:
                draw.text((((pixels[cur][2] + pixels[cur][4]) / 2), pixels[cur][1] - 12),
                          str(int(current_hu[cur][0])), fill="black", font=None, anchor=None)
    return
```

Fig6. Writing the predicted value

Design and implement GUI

The program needed an interface to make it easier to use. It was intended to be created with the help of the Tkinter library. It opens a window to choose a picture, it offers 2 different options to operate on the picture. One is testing, the other is training. While testing, comparing and printing which result was reached; Training plays a role in enlarging the database.

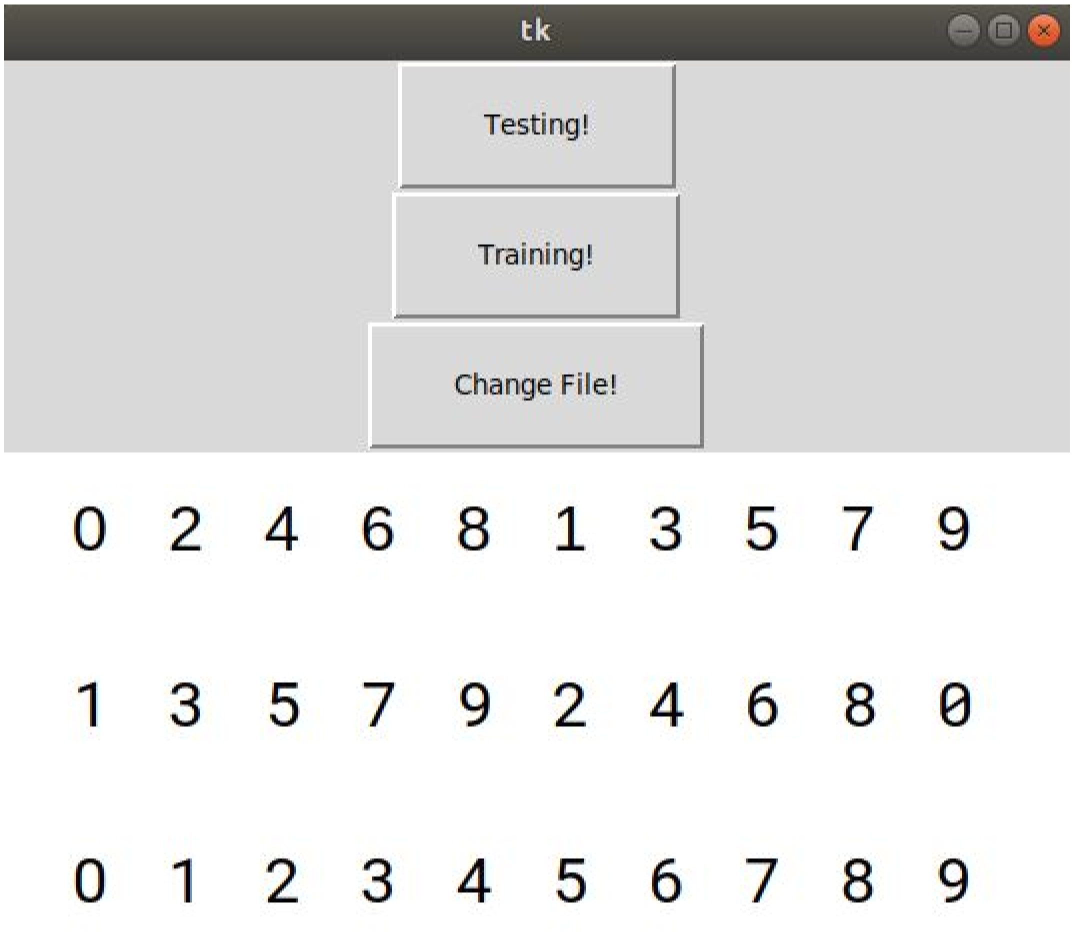


Fig7. GUI's main page

Description of Solutions

For the solution, I started by converting 4-blob-coloring to 8. For this, I added 2 new variables to get the function to control the upper cross pixels and put them in the for loop. With help of this, they now started to represent the same character in cross-pixels.

After the 8-blob-coloring is now working properly, I made the results of the labeling process available. To do this, I removed the label of the background in the picture. Then I went around the labels and pixels one by one and calculated which label stands in which pixels. While doing this, I updated the maximum and minimum pixels of the labels. I kept them in an array and saved them to draw a rectangle.**(Fig2.)**

I used the array that holds the maximum and minimum values that I filled before to draw the rectangles. So I knew which coordinates, that is, pixels, should I draw rectangles.**(Fig3.)**

Then, in order to calculate moments, I had to adjust all the characters to a certain format. Using the maximum and minimum coordinates I knew before, I turned the characters into individual pictures. I have created a standard in these paintings by adjusting their dimensions to 21x21. Then I gathered these 21x21 pictures in a list. Moment calculating functions started to operate using this list.**(Fig4.)**

I made the necessary calculations using the formulas given by the teacher of the course. I had 1 problem while passing these formulas into code. Is ZeroDivisionError. I used try-except to solve it. It was not fully resolved, but it played a role in continuing transactions. I just faced this problem for Zernike (I do not count the R moment in the calculation of the letters to A-Z.).

I kept the moments in the array. I deliberately left Array's first column empty. Because while writing the database, I wrote down which number of moment calculations belong to that part.**(Fig6.)** So a structure similar to this occurred: Number-HU1-HU2 -...-HU7.

While training, I determined which number these moments belonged and started to keep them all in the txt file. This txt file is what determines the rules of the game. Test and Training cannot proceed without a txt file.

After that, I proceeded to compare and guess the character. To do this, I created a function for every different moment process. Based on the distance formula, I started making comparisons with the values in my txt file. By keeping the number with the lowest distance, I placed it in the column at the very beginning of that array. For example: current_hu [x] [0] = txt [y] [0]. **(Fig5.-Fig6.)**

Finally, I took a step to execute these processes in an interface. I used the tkinter library to do this. First of all, I opened the file selection screen. The user was able to choose which picture he would like to process.**(Fig8.)**

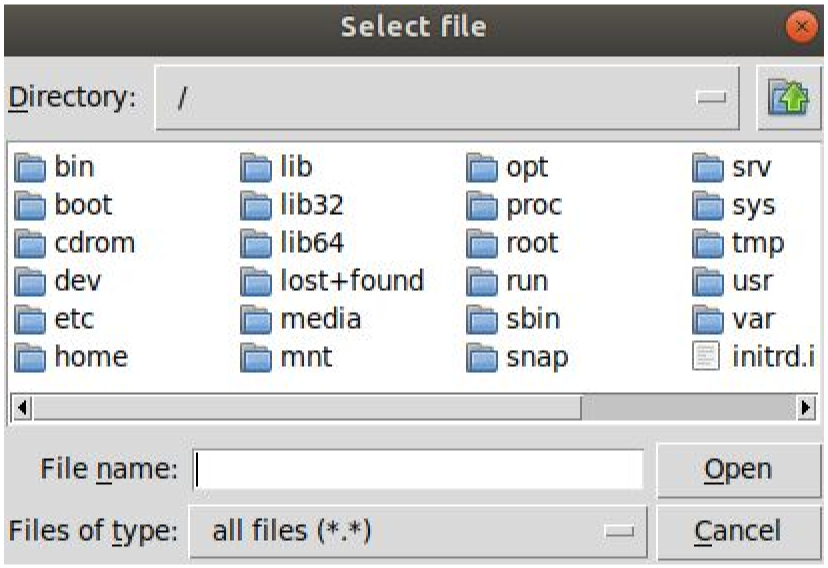


Fig8. Select File Window

The selected picture and the keys of things that can be done with that picture came to the screen. Options include file selection, training and testing buttons.**(Fig7.)**

If you click the Testing button, you will see 3 functions that can guess the characters in the picture. Once you choose one, a pop-up message will appear as soon as the process is over and it will appear on the results screen.**(Fig9.)**

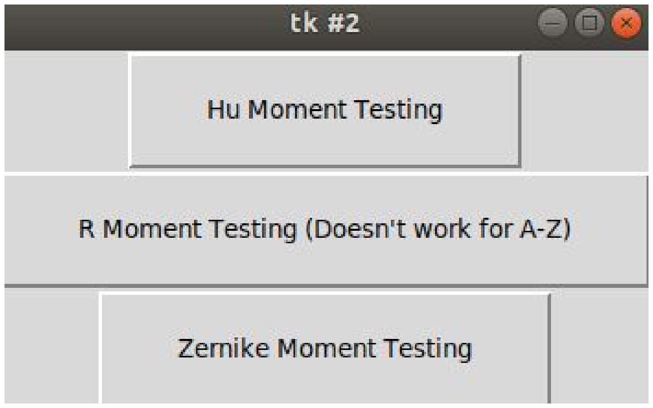


Fig9. Testing Window

If you choose the training button, you will get a little more options. You will see 2 different methods to train numbers and methods to train capital letters. The most important thing you should know if you are interested in train the numbers, if you are using a 1 digit train, you should not start the process without typing which digit you train in the top box. Otherwise, you can save it to your database without knowing which number it is.**(Fig10.)**

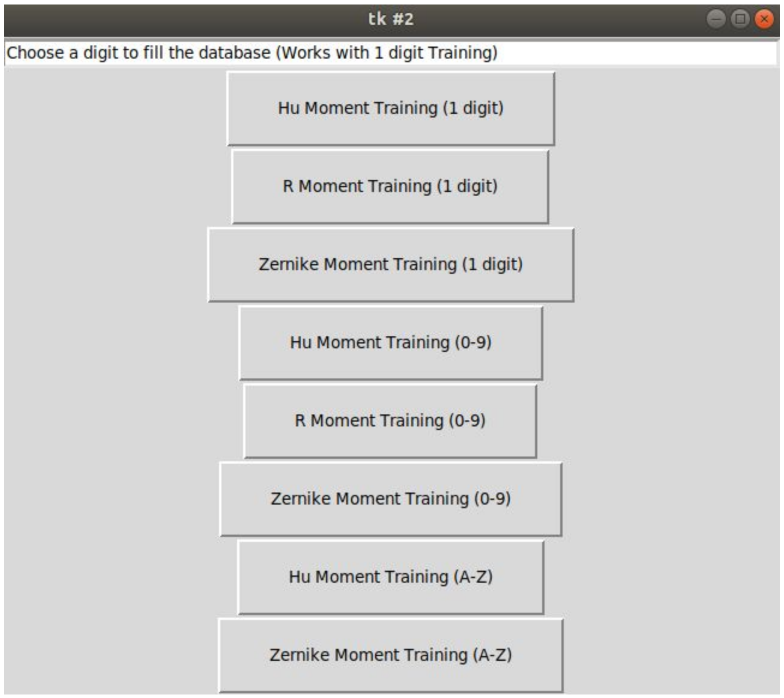
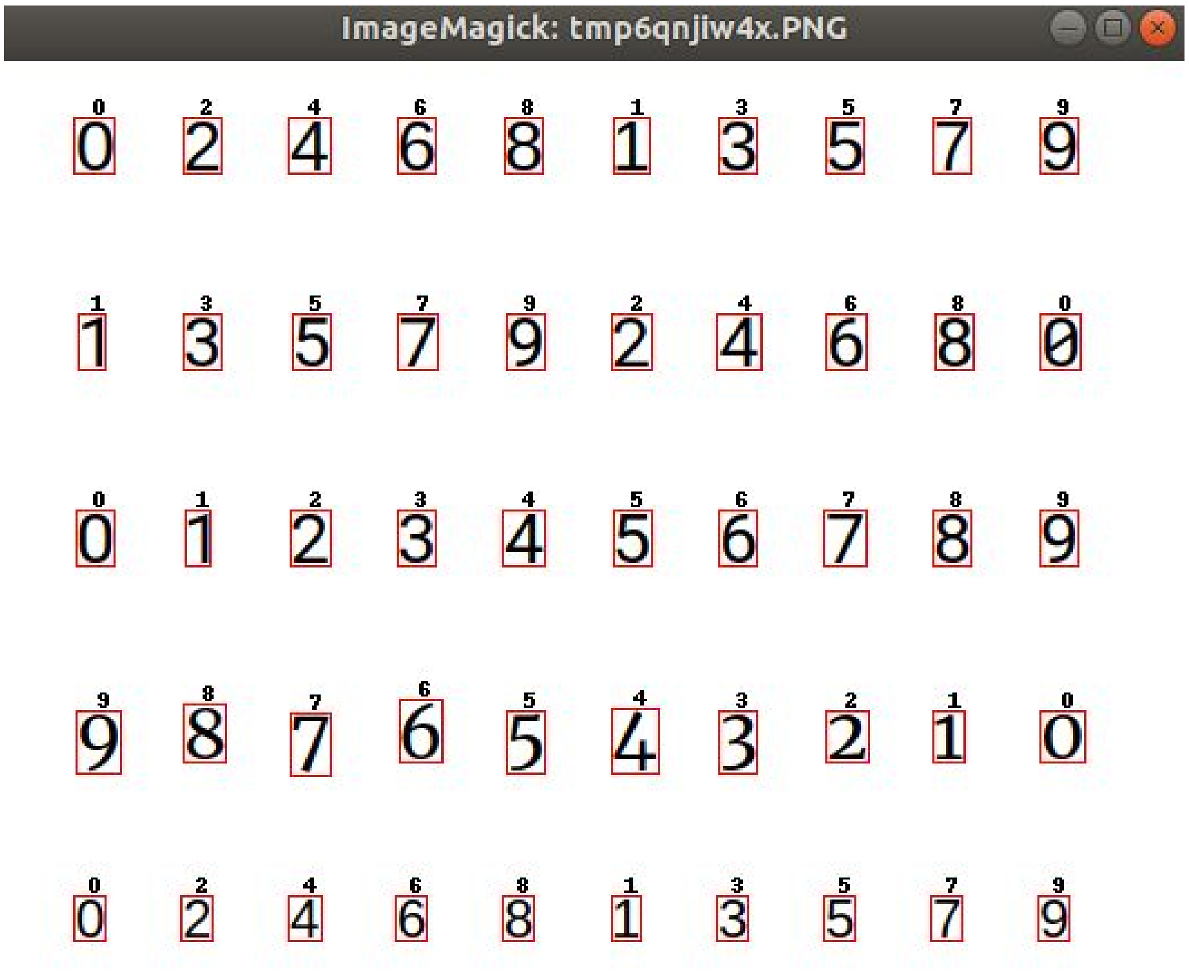

tk #2
Choose a digit to fill the database (Works with 1 digit Training)
Hu Moment Training (1 digit)
R Moment Training (1 digit)
Zernike Moment Training (1 digit)
Hu Moment Training (0-9)
R Moment Training (0-9)
Zernike Moment Training (0-9)
Hu Moment Training (A-Z)
Zernike Moment Training (A-Z)

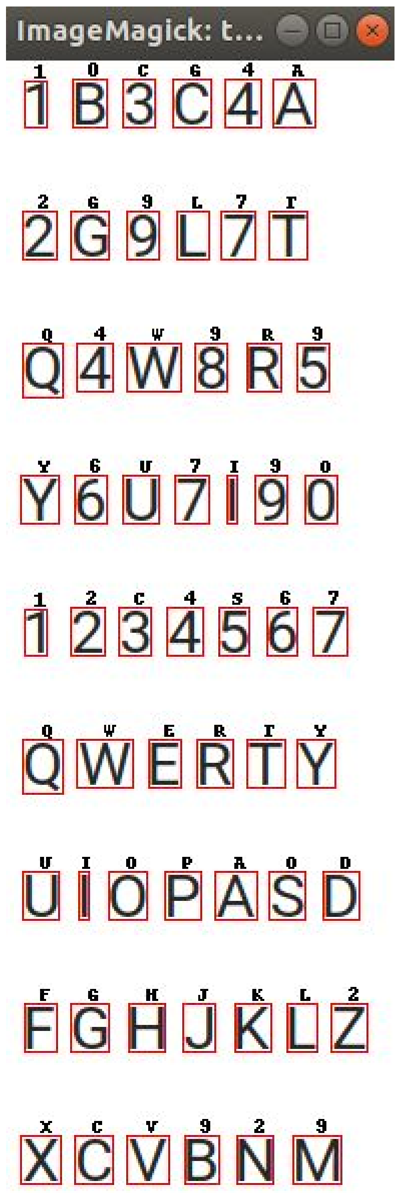
Fig10. Training Window

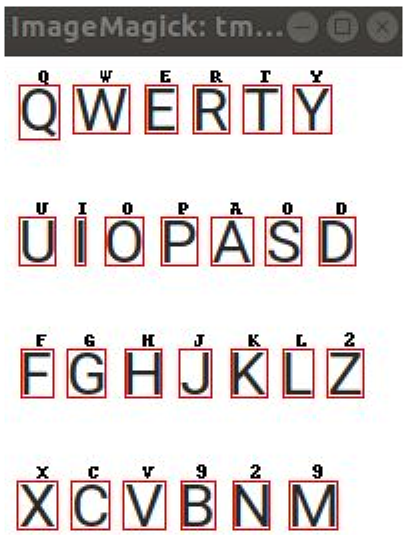
Sample Results

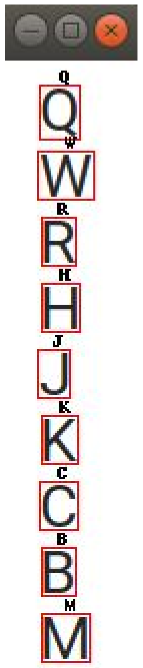
HU MOMENTS



%100 Success for this picture.

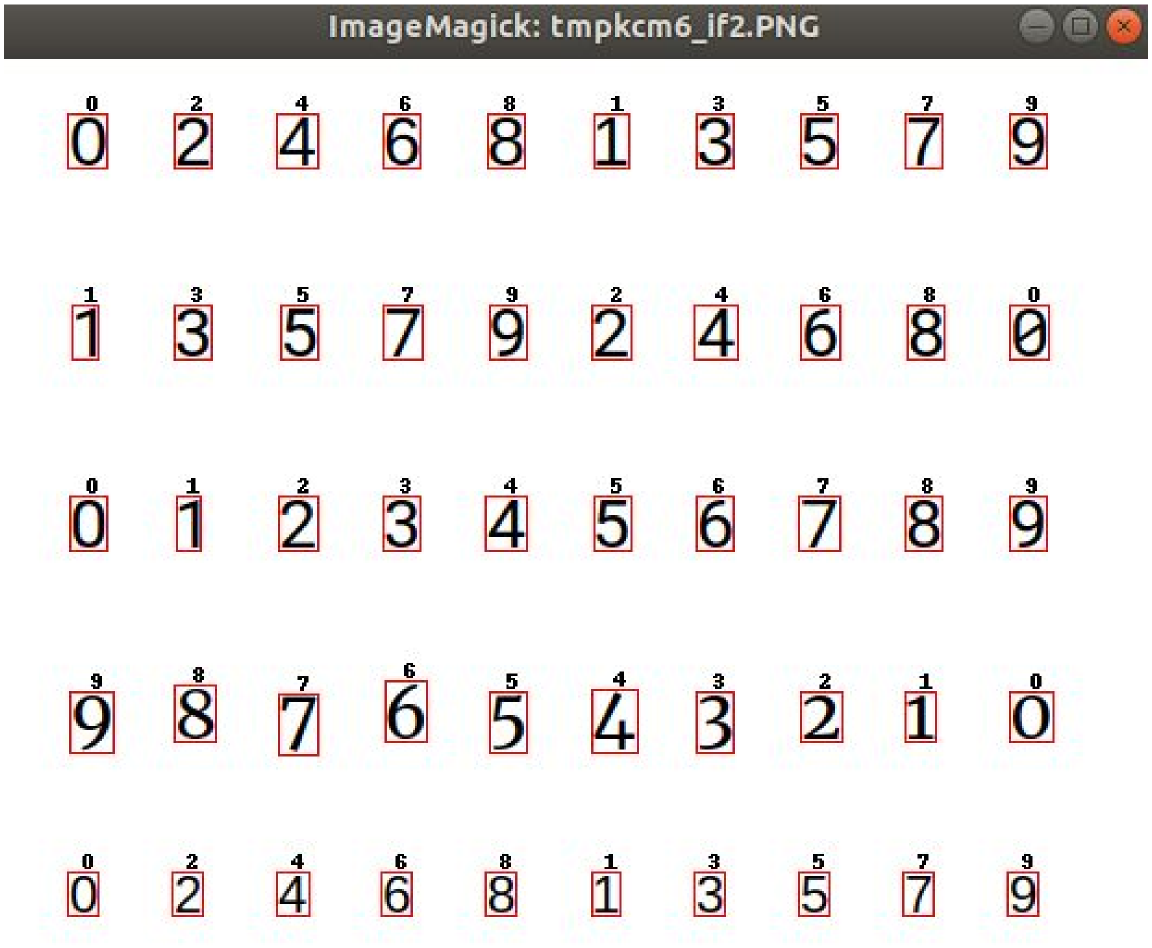






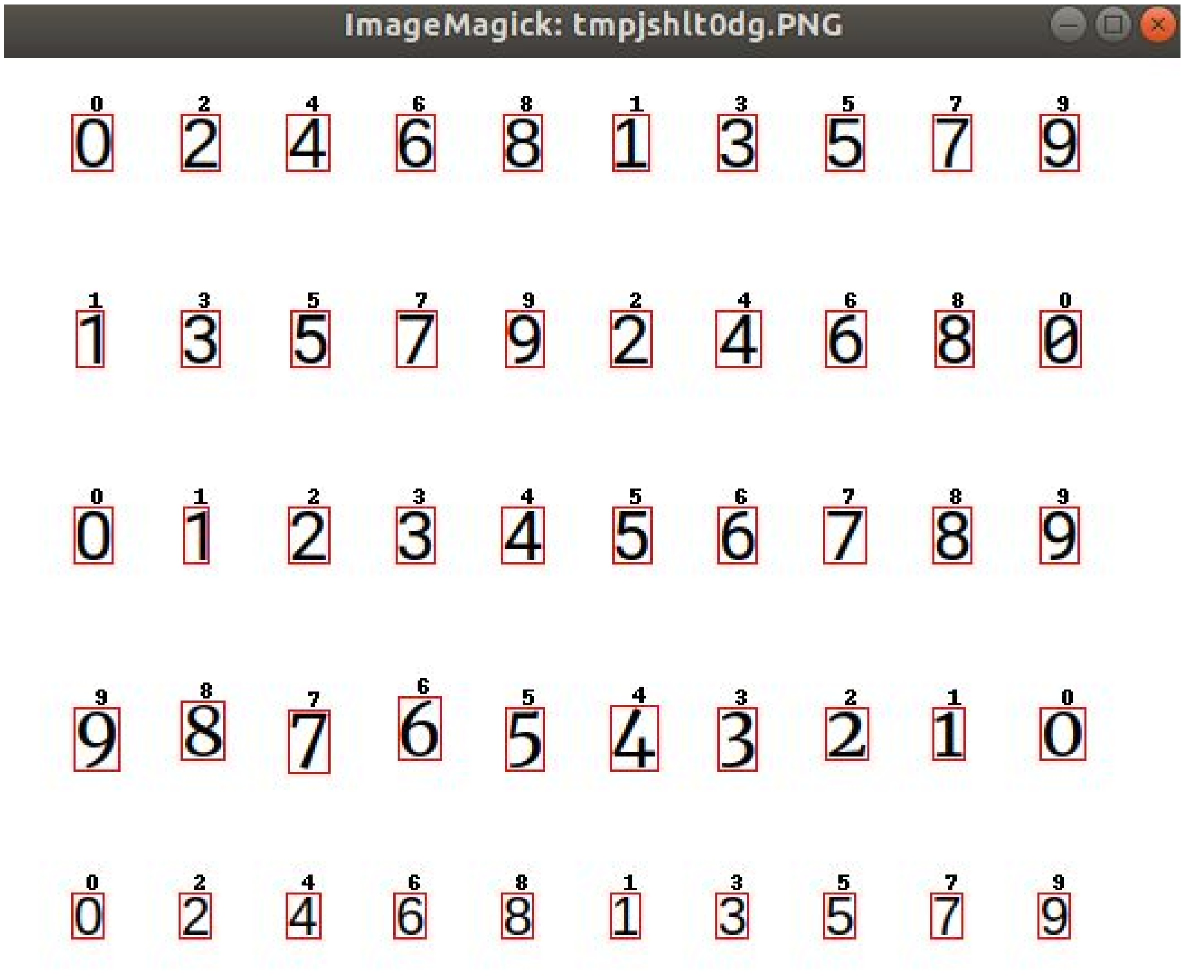
45/58 = %77 **20/26 = %76.9** **%100**

R MOMENTS

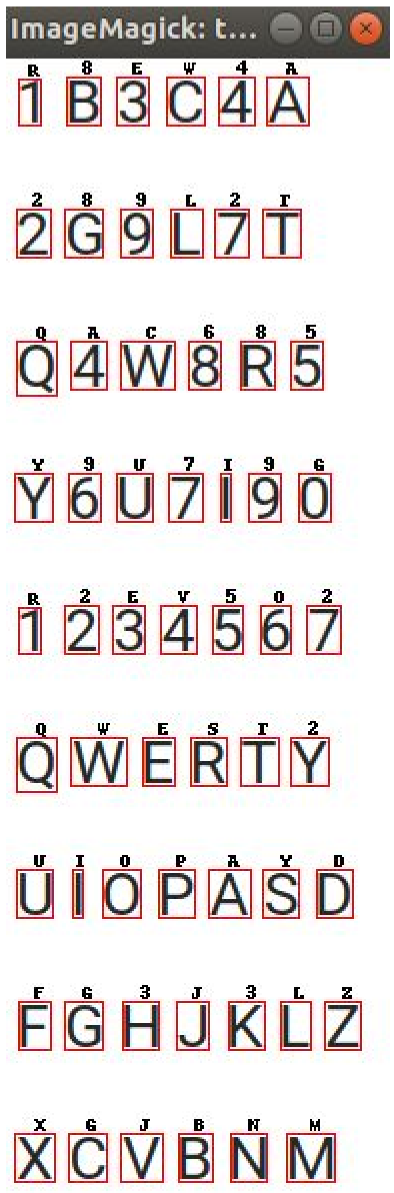


%100 Success for this picture.

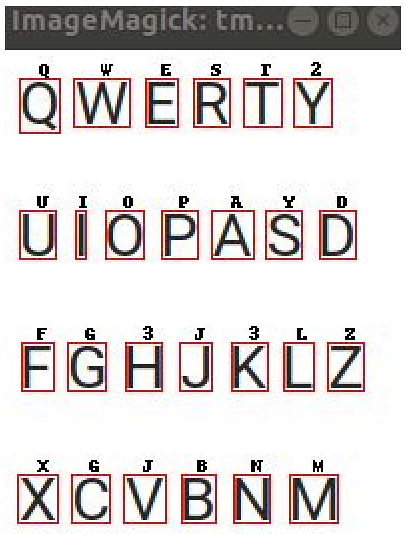
ZERNIKE MOMENTS



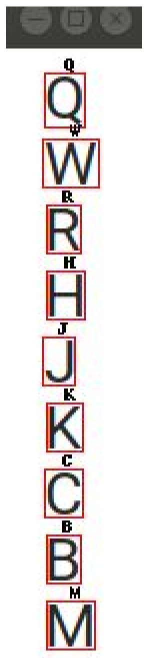
%100 Success for this picture.



34/58 = %58



19/26 = %73



%100

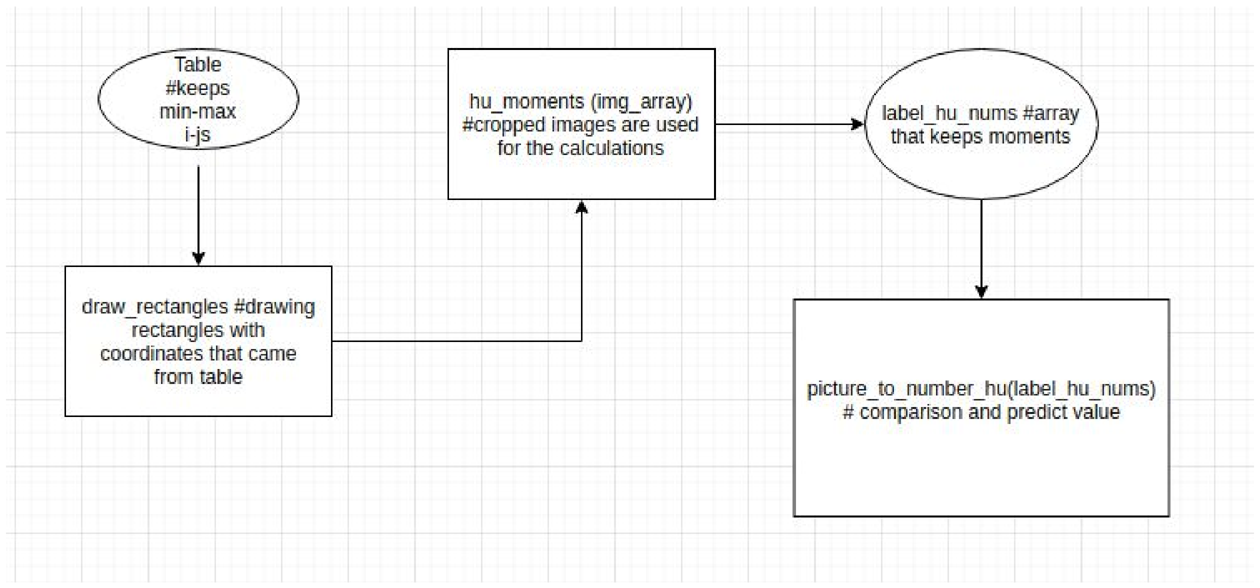


Table #keeps min-max i-js
draw_rectangles #drawing rectangles with coordinates that came from table
hu_moments (img_array) #cropped images are used for the calculations
label_hu_nums #array that keeps moments
picture_to_number_hu(label_hu_nums) # comparison and predict value

UML DIAGRAM

List of Achievements

The benefits of this project to students:

- They got used to Python syntax.
- They learned the working principles of numpy arrays.
- They learned to process the picture with coding.
- Their familiarity with the Pillow library has increased.
- They basically learned how to create the interface.
- They had ideas about recognition systems.
- They learned about file manipulation and how they are used.

References

[1] https://docs.python.org/2/library/tkinter.html

[2] https://pillow.readthedocs.io/en/stable/

[3] https://pillow.readthedocs.io/en/stable/reference/Image.html

[4] https://pillow.readthedocs.io/en/stable/reference/ImageDraw.html

[5] http://cs231n.github.io/python-numpy-tutorial/

[6] https://www.geeksforgeeks.org/python-numpy/

[7] https://docs.scipy.org/doc/numpy/reference/routines.io.html

[8] https://docs.scipy.org/doc/numpy/reference/generated/numpy.load.html

[9] https://en.wikipedia.org/wiki/Zernike_polynomials

[10]https://mef.blackboard.com/webapps/blackboard/execute/displayLearningUnit?course_id=_6991_1&content_id=_245444_1&framesetWrapped=true