MEF UNIVERSITY

PROJECT 1

CHARACTER RECOGNITION USING MOMENTS

COMP 204 - Programming Studio

Instructor: Prof.Dr. Muhittin Gökmen

Rabia Kodal-041701018

Date: 01.03.2020

Abstract

Machine learning methods are used to accurately predict optical characters and in this project it is aimed to be recognized with deep learning and high accuracy and high speed to recognize the characters in the images.

Introduction

In this project, we will develop a program in Python to recognize the characters in an image.It is a widespread technology to recognise text inside images, such as scanned documents and photos.This technology is used to convert virtually any kind of images containing written text (typed, handwritten or printed) into machine-readable text data.Probably the most well known use case for OCR(Optical Character Recognition) is converting printed paper documents into machine-readable text documents. Once a scanned paper document went through OCR processing, the text of the document can be edited with word processors like Microsoft Word or Google Docs.OCR applications are often used in the following areas:

- Document Processing
- Desktop Publishing
- Order Processing
- Staff Records Management
- Check Processing
- Payment Processing
- Retirement Fund Processing

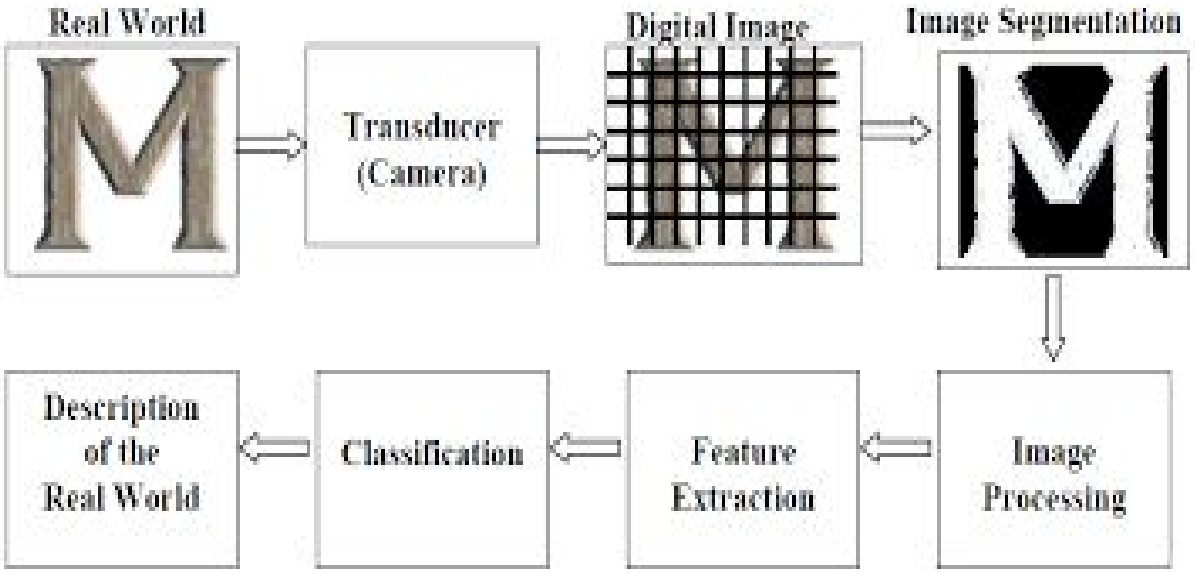
To avoid complexity, it is necessary to do these step by step.

Action items

1. Reading image
2. Character detection
3. Feature extraction
4. Classification
5. Test results
6. GUI
7. Report

Description of the project

In this project, we are developing a program that recognizes characters. The character recognition system in the study consists of sections such as preprocessing the image file, finding the lines, extracting the characters in the lines and classifying the characters as shown in the figure.

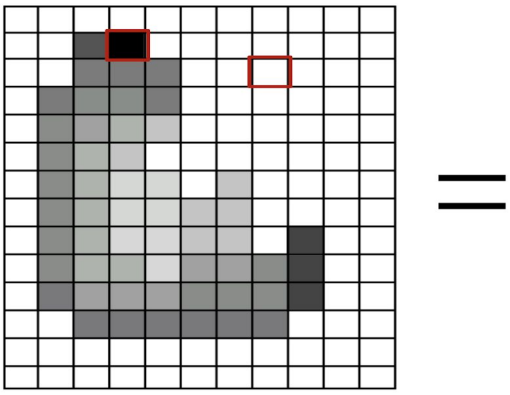


1)Reading an image file

Firstly we get an image file in * png * jpg format as input. We can't use any libraries other than PIL and numpy for this project. As an example, you can use an image such as

What is an image?

A grid (matrix) of intensity values



=

255	255	255	255	255	255	255	255	255	255	255
255	255	20	0	255	255	255	255	255	255	255
255	255	75	75	255	255	255	255	255	255	255
255	75	95	95	75	255	255	255	255	255	255
255	96	127	145	175	255	255	255	255	255	255
255	127	145	175	175	175	255	255	255	255	255
255	127	145	200	200	175	175	95	255	255	255
255	127	145	200	200	175	175	95	47	255	255
255	127	145	145	175	127	127	95	47	255	255
255	74	127	127	127	95	95	95	47	255	255
255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255

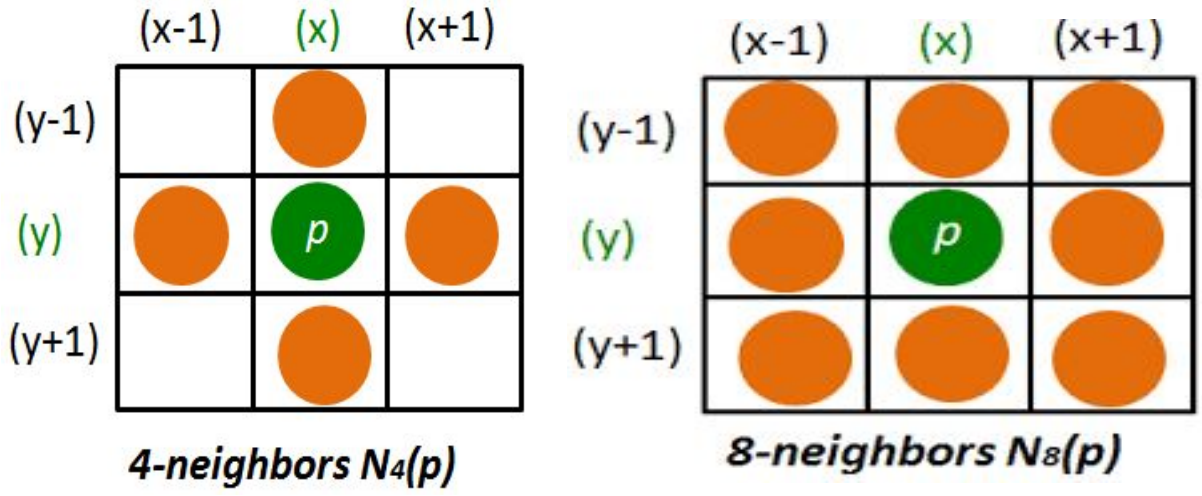
0 = black; 255 = white

```python
def readPILimg(): #read image file using PIL
    img = Image.open('/Users/gokmen/Dropbox/vision-python/images/brick-house.png')
    img.show()
    img_gray = color2gray(img)
    img_gray.show()
    #img_gray.save('/Users/gokmen/Dropbox/vision-python/images/brick-house-gs','png')
    #new_img = img.resize((256,256))
    #new_img.show()
    return img_gray
```

We can use the code above to read the image.

2)Character Detection

First we will apply 8 linked drop coloring algorithm. After labeling the 8 linked components with the drop coloring algorithm, we will find the bounding box of each character.



4- neighbors

For a pixel p (x,y) shown in green circle, N4(p) is the set of 4-neighbors which share a face (or edge of the pixel) with p. They are four in number and are shown by orange circles.

8 –neighbors:

For a pixel p (x, y) shown in green circle, N8(p) is the set of neighbours which share a face or a vertex/corner. There are 8 such neighbours and they are shown by orange circles.

Neighbors of a Pixel

- A pixel p at coordinates (x,y) has four *horizontal* and *vertical* neighbors whose coordinates are given by:
  (x+1,y), (x-1, y), (x, y+1), (x,y-1)

	(x, y-1)	
(x-1, y)	*P (x,y)*	(x+1, y)
	(x, y+1)	

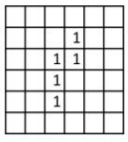
This set of pixels, called the 4-*neighbors* or p, is denoted by $N_4(p)$.
Each pixel is one unit distance from (x,y) and some of the neighbors of p lie outside the digital image if (x,y) is on the border of the image.

- Why we apply 8 linked drop coloring algorithm?

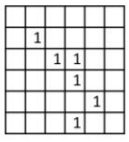
  Characters can cross diagonally, for example A. So we also have to connect with cross points.

Connected Components

4-connected component



8-connected component



Neighbors to consider during labeling

4-connected component

8-connected component

3)Feature Extraction

Crop and Resize

For every rectangle surrounding a character, we turn every rectangle into a square.Then we bring the square to a regular size (21,21). For example:

Moments

We calculate the Hu moment immutable coefficients and save them as the Hu properties of this character. Then we calculate the Zernike Moment coefficients and save it as the Zernike properties of this character.

Hu Moment

Hu moments are translation, scale, and rotation invariant.

normalized central moments

$$H_1 = \eta_{20} + \eta_{02}$$
$$H_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$
$$H_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$
$$H_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$H_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$
$$H_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$
$$H_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

R Moment

Liu (2008) proposed ten R moments which can improve the scale invariability of Hu moments.

Hu moments

$$= \frac{\sqrt{H_3}}{\sqrt{|H_5|}} \qquad R_2 = \frac{H_1 + \sqrt{H_2}}{H_1 - \sqrt{H_2}} \qquad R_3 = \frac{\sqrt{H_3}}{\sqrt{H_4}} \qquad R_4$$

$$R_5 = \frac{}{\sqrt{|H_5|}} \qquad R_6 = \frac{|H_6|}{H_1 \cdot H_3} \qquad R_7 = \frac{|H_6|}{H_1 \cdot \sqrt{|H_5|}}$$

$$R_8 = \frac{|H_6|}{H_3 \cdot \sqrt{H_2}} \qquad R_9 = \frac{|H_6|}{\sqrt{H_2 \cdot |H_5|}} \qquad R_{10} = \frac{|H_5|}{H_3 \cdot H_4}$$

Zernike Moments Features

$$ZR_{nm} = \frac{n+1}{\pi}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)R_{nm}(\rho_{ij})\cos(m\theta_{ij})\Delta_{xi}\Delta_{yj}$$

$$ZI_{nm} = -\frac{n+1}{\pi}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)R_{nm}(\rho_{ij})\sin(m\theta_{ij})\Delta_{xi}\Delta_{yj}$$

$$Z_{nm} = \sqrt{ZR_{nm}{}^2 + ZI_{nm}{}^2}$$

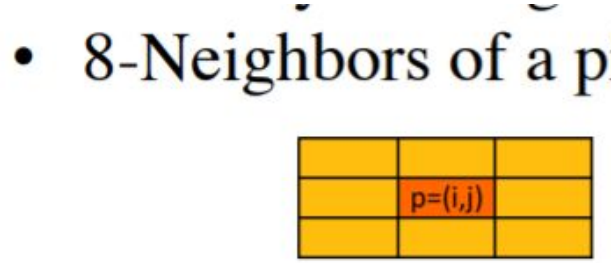
Features: $Z_{11}, Z_{22}, Z_{31}, Z_{33}, Z_{42}, Z_{44}, Z_{51}, Z_{53}, Z_{55}, Z_{62}, Z_{64}, Z_{66}$

Description of my solution

Blob coloring 8 connected:

```
k = 0
for i in range(nrow):
    for j in range(ncol):
        im[i][j] = max_label - 1
for i in range(1, nrow - 1):
    for j in range(1, ncol - 1):
        c = bim[i][j]
        l = bim[i][j - 1]
        u = bim[i - 1][j]
        label_u = im[i - 1][j]
        label_l = im[i][j - 1]
        label_ul = im[i - 1][j - 1]
        label_ur = im[i - 1][j + 1]
        im[i][j] = max_label - 1
        if c == ONE:
            min_label = min(label_u, label_l, label_ul, label_ur)
            if min_label == max_label - 1:
```

We define separate labels for upper, left, upper left, upper right.Because we need to go to the cross neighbors to be able to read the characters fully.




```
if min_label == max_label - 1:
    k += 1
    im[i][j] = k
else:
    im[i][j] = min_label
    if min_label != label_u and label_u != max_label - 1:
        update_array(a, min_label, label_u)

    if min_label != label_l and label_l != max_label - 1:
        update_array(a, min_label, label_l)

    if min_label != label_ur and label_ur != max_label - 1:
        update_array(a, min_label, label_ur)

    if min_label != label_ul and label_ul != max_label - 1:
        update_array(a, min_label, label_ul)
```

Find Rectangle

```
def find_rectangle(im):
    nrow = im.shape[0]
    ncol = im.shape[1]

    k_max = 0
    for i in range(nrow):
        for j in range(ncol):
            if im[i][j] == 10000 - 1:
                continue
            myLabel = im[i][j]
            if myLabel > k_max:
                k_max = myLabel

    mini = np.full(k_max + 1, np.inf)
    minj = np.full(k_max + 1, np.inf)
    maxi = np.zeros(k_max + 1)
    maxj = np.zeros(k_max + 1)
```

At this stage we need to find the minimum and maximum points for the rectangle

```
label_set = set()
for i in range(nrow):
    for j in range(ncol):
        if im[i][j] == 10000 - 1:
            continue
        myLabel = im[i][j]
        label_set.add(myLabel)
        if i < mini[myLabel]:
            mini[myLabel] = i
        if i > maxi[myLabel]:
            maxi[myLabel] = i
        if j < minj[myLabel]:
            minj[myLabel] = j
        if j > maxj[myLabel]:
            maxj[myLabel] = j
return mini, minj, maxi, maxj, label_set,  # output
```

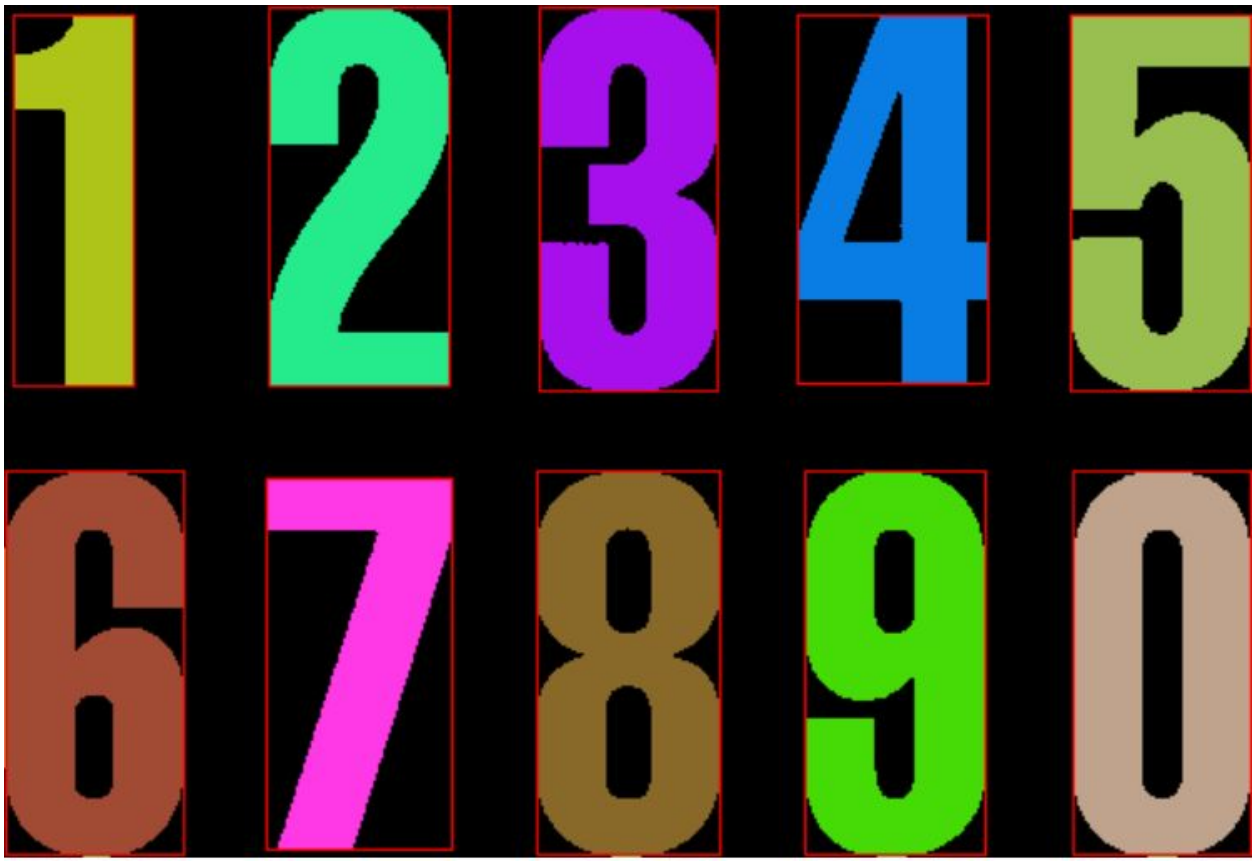
Drawing Rectangle(Crop and Resize)

```
mini, minj, maxi, maxj, label_set = find_rectangle(labels)
print(mini)
new_img2 = np2PIL_color(label)
for x in range(len(mini)):
    if (mini[x] < 1000):
        shape = [(minj[x], mini[x]), (maxj[x], maxi[x])]
        rect = ImageDraw.Draw(new_img2)
        rect.rectangle(shape, outline="red")
new_img2.show()


for x in range(len(mini)):
    if (mini[x] < 1000):
        cropped = new_img2.crop((minj[x], mini[x],maxj[x], maxi[x]))
        cropped = cropped.resize((21, 21))
        cropped.show()
```

At this stage, we create our rectangle using the ImageDraw.Draw (img) function.We also crop the image to 21 to 21 square size.

OUTPUT



 cropped version

List of achievements (what did i learn during the project)

- ❖ Reading image
- ❖ How to convert the image to gray and binary
- ❖ How to do 8-connected Blob Coloring and why it is needed
- ❖ How to find the minimum and maximum points of characters

❖ Rectangular plotting and croping functions

❖ Many extra functions

1. Image.open:opens and identifies the given image file.
2. img.convert('L'):converts the image to grayscale image.
3. img.show():Displays the image.This method is mainly intended for debugging purposes.
4. numpy.asarray(): function is used when we want to convert input to an array.
5. Threshold: thresholding is a technique, which is the assignment of pixel values in relation to the threshold value provided and etc..

References

- Rouse M, OCR (optical character recognition)

  https://searchcontentmanagement.techtarget.com/definition/OCR-optical-character-recognition

- https://edtech.engineering.utoronto.ca/object/2d-image-digital-representation
- Koyun A., Afşin E.,2D Optical Character Recognition Based on Deep Learning

  https://dergipark.org.tr/en/download/article-file/346880