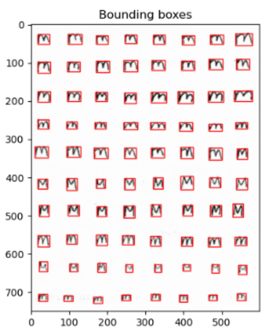
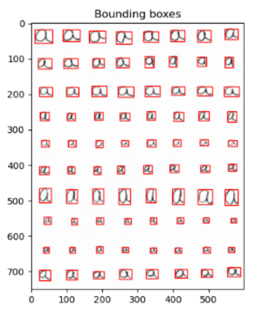
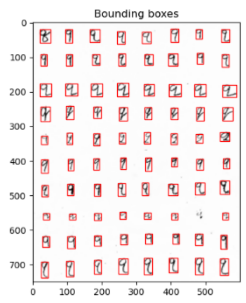
Qp13

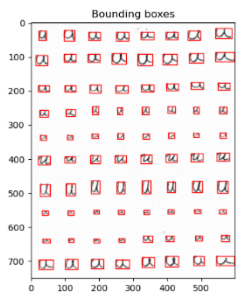
Qinyuan Pu

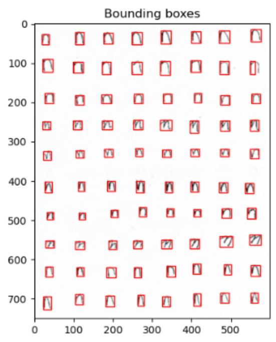
For all Training Images

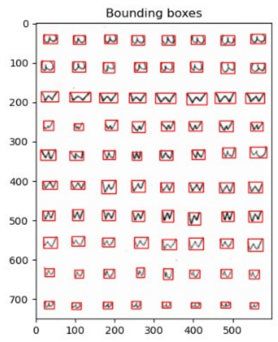


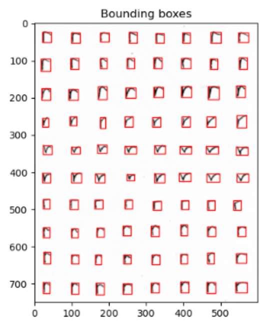


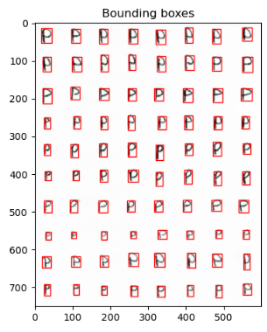


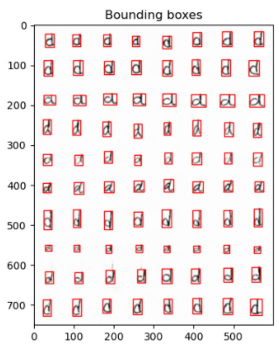


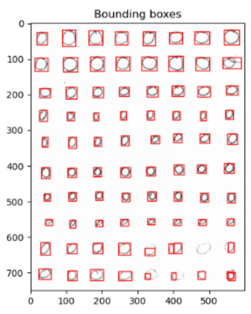




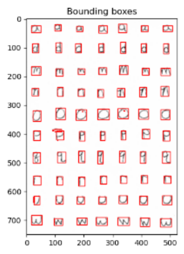








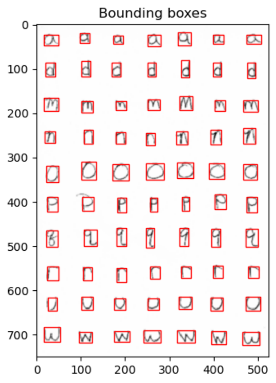
Test image(not enhanced):



Enhancements Attempt:

Problem: too many small fragments

Solution camp up: give the boxes the limit
Result: problem deleted

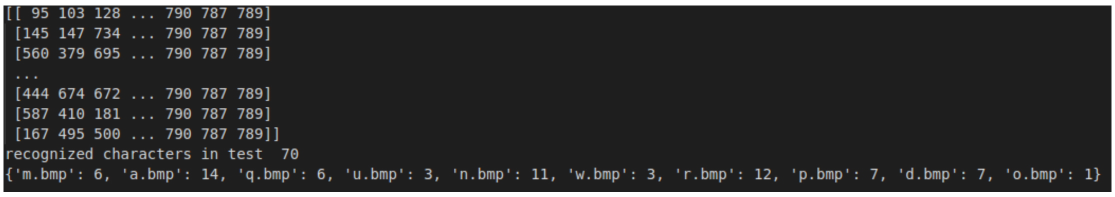


Recognition Rates and other Values

Binary Threshold value: 206
Number of components: 70
Recognition rate for test image before enhancement: 54%
Recognition rate after enhancement: 60%

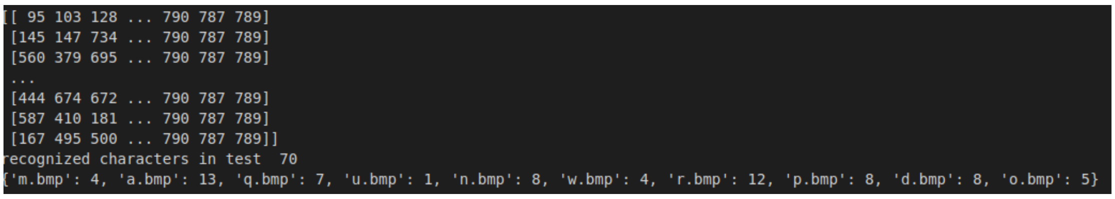
Problem: accuracy is too low
Possible reason: no enough comparison
Solution camp up: from the D_index, pick first 50 distance and find the most possible result for each character
Result: accuracy becomes a little bit higher

--*Original result: accuracy 54%*



```
[[ 95 103 128 ... 790 787 789]
 [145 147 734 ... 790 787 789]
 [560 379 695 ... 790 787 789]
 ...
 [444 674 672 ... 790 787 789]
 [587 410 181 ... 790 787 789]
 [167 495 500 ... 790 787 789]]
recognized characters in test  70
{'m.bmp': 6, 'a.bmp': 14, 'q.bmp': 6, 'u.bmp': 3, 'n.bmp': 11, 'w.bmp': 3, 'r.bmp': 12, 'p.bmp': 7, 'd.bmp': 7, 'o.bmp': 1}
```

--*Enhanced Result: accuracy 60%*



```
[[ 95 103 128 ... 790 787 789]
 [145 147 734 ... 790 787 789]
 [560 379 695 ... 790 787 789]
 ...
 [444 674 672 ... 790 787 789]
 [587 410 181 ... 790 787 789]
 [167 495 500 ... 790 787 789]]
recognized characters in test  70
{'m.bmp': 4, 'a.bmp': 13, 'q.bmp': 7, 'u.bmp': 1, 'n.bmp': 8, 'w.bmp': 4, 'r.bmp': 12, 'p.bmp': 8, 'd.bmp': 8, 'o.bmp': 5}
```

Tested/Used Features

First, I try to adjust the binary threshold hold to recognize some characters that is not obvious (for example, some characters from training image 'O.bmp').
Then I try to prevent the algorithm to recognize fragments as characters, I gives the boxes a minimum size.
Finally I try to enhance the accuracy of recognition, I try to use more samples to enhance my result. However, I find that it is not good to pick to much sample, the distance matrix between test image and training features are 70*810, theoretically I should pick up to 810 comparisons for each character, however, that leads to a bad result. After few testing, I find out 50 comparison for each character is the best for result.

About RunMyOCR.py

Following the instruction, while setting classes from mydict, it always fail and says KeyError: 'classes', since the test_gt_py3.pkl is given and I do not know whether should I modify it. I didn't use the way to test accuracy, I just count how many characters has been recognized and store it to the dictionary 'outputlist' in the test.py. About reading images, the train will automatically detect all images in the give path 'path1' in train.py, and extract its features for test.py use.

Code

train.py

```
import numpy as np
from sklearn.metrics import confusion_matrix
from scipy.spatial.distance import cdist
from skimage.measure import label, regionprops, moments, moments_central,
moments_normalized, moments_hu
from skimage import io, exposure
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import pickle
from multiprocessing import Pool
import os


Features=[]
path1 = "/home/acrdgg/Desktop/cs334_hw4/images/";

listing = os.listdir(path1);
#print(listing);
count = 0;
listcount = 0;
map_count = 0;
image_map = {};
image_namelist = {};


for file in listing:
	img = io.imread(path1+file);



	#print(img.shape);

	#io.imshow(img);
	#plt.title('original image');
	#io.show();

	# hist = exposure.histogram(img);
	# plt.bar(hist[1], hist[0]);
	# plt.title('histogram');
```

```
    # plt.show();

    th = 206
    img_binary = (img < th).astype(np.double);
    # io.imshow(img_binary);
    # plt.title('binary image');
    # io.show();

    img_label = label(img_binary, background = 0);
    # io.imshow(img_label);
    # plt.title('labeled image');
    # io.show();
    #print(np.amax(img_label));



    regions = regionprops(img_label);
    #io.imshow(img_binary);
    ax = plt.gca();

    fine_boxes = 0;
    hu_numbers = 0;
    for props in regions:
            minr, minc, maxr, maxc = props.bbox;
            if (np.abs(maxc-minc) < 10):
                continue
            roi = img_binary[minr:maxr, minc:maxc];
            m = moments(roi);
            cc = m[0,1] / m[0,0];
            cr = m[1,0] / m[0,0];
            mu = moments_central(roi, center=(cr,cc));
            nu = moments_normalized(mu);
            hu = moments_hu(nu);
            Features.append(hu);
            hu_numbers += 1;
            ax.add_patch(Rectangle((minc, minr), maxc - minc, maxr - minr, fill=False,
edgecolor='red', linewidth=1));
            fine_boxes += 1;

    count += fine_boxes;
    image_map[map_count] = count;
    image_namelist[map_count] = file;
    map_count += 1;
    # print("image:", file, "stop at", count, " fine boxes number: ", fine_boxes);
```

```
        # print("current features numbers: ", len(Features));
        ax.set_title('Bounding boxes');
        io.imshow(img)
        io.show()

#print(image_map);
# print("Features's Length: ",len(Features));

mean_std = {};
i = 0;

while i < len(Features):
        temp_mean = np.mean(Features[i]);
        temp_std = np.std(Features[i]);


        Features[i]-=temp_mean;
        Features[i]/=temp_std;

        mean_std[i] = [temp_mean,np.var(Features[i])];
        i += 1;

#print(Features);
#print(image_map)
#print(image_namelist)
#print(map_count)

Features



#distance maxtrix calculation
# D = cdist(Features,Features);
# D_index = np.argsort(D, axis=1);
# io.imshow(D);
# plt.title('Distance Matrix');
# print(D)
#np.set_printoptions(threshold=np.inf)
#print(D_index);
#print(D_index.shape)
# io.show();
```

test.py

```
from typing_extensions import final
import numpy as np
from sklearn.metrics import confusion_matrix
from scipy.spatial.distance import cdist
from skimage.measure import label, regionprops, moments, moments_central,
moments_normalized, moments_hu
from skimage import io, exposure
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import pickle
from multiprocessing import Pool
import os
from train import Features
from train import image_map
from train import image_namelist
from train import map_count

Features_t = []
count = 0
chara_count = {}

#read test img
img = io.imread('test.bmp')

#binarize
th = 206
img_binary = (img<th).astype(np.double)

#find each characters
img_label = label(img_binary, background=0)
regions = regionprops(img_label)
ax = plt.gca()

for props in regions:
    minr, minc, maxr, maxc = props.bbox
    if(np.abs(maxc-minc)<10 or np.abs(maxr-minr)<10):
        continue
    roi = img_binary[minr:maxr, minc:maxc];
    m = moments(roi);
    cc = m[0,1] / m[0,0];
    cr = m[1,0] / m[0,0];
    mu = moments_central(roi, center=(cr,cc));
```

```
        nu = moments_normalized(mu);
        hu = moments_hu(nu);
        Features_t.append(hu);
        ax.add_patch(Rectangle((minc,    minr),    maxc    -    minc,    maxr    -    minr,    fill=False,
edgecolor='red', linewidth=1));

ax.set_title('Bounding boxes')
io.imshow(img)
io.show()


D = cdist(Features_t, Features)
#io.imshow(D)
D_index = np.argsort(D, axis=1)
plt.title('Distance Matrix')

print(image_map)
print(image_namelist)
print(D_index)
# print(D_index.shape)
# print(D_index.shape[0])

#initialize
ini_count = 0
while ini_count < map_count:
        chara_count[ini_count] = 0
        ini_count += 1

print("recognized characters in test ", D_index.shape[0])
#recog - standard
# rec_count = 0

# while rec_count < D_index.shape[0]:
#         temppos = 0
#         while D_index[rec_count][0] > image_map[temppos]:
#                 temppos += 1
#         chara_count[temppos] += 1
#         rec_count += 1

#recog - enhancememt
rec_count = 0
enhance_output = {}
while rec_count < D_index.shape[0]:
        hori_count = 0
        #initialize temp hashmap
        ini2_count = 0
        while ini2_count < map_count:
                enhance_output[ini2_count] = 0
                ini2_count += 1
        #start counting
        while hori_count < 49:
                temppos = 0
                while D_index[rec_count][hori_count] > image_map[temppos]:
                        temppos += 1
                enhance_output[temppos] += 1
                hori_count += 1
        finalpos = max(enhance_output, key=enhance_output.get)
        chara_count[finalpos] += 1
        rec_count += 1


#finalize
final_count = 0
outputlist = {}
while final_count < map_count:
        outputlist[image_namelist[final_count]] = chara_count[final_count]
        final_count += 1


print(outputlist)

#io.show()
Outputlist
```

RunMyOCR.py

```
from typing_extensions import final
import numpy as np
from sklearn.metrics import confusion_matrix
from scipy.spatial.distance import cdist
from skimage.measure import label, regionprops, moments, moments_central, moments_normalized, moments_hu
from skimage import io, exposure
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import pickle
from multiprocessing import Pool
import os
from test import outputlist

pkl_file = open('test_gt_py3.pkl', 'rb')
mydict = pickle.load(pkl_file)
pkl_file.close()
classes = mydict['classes']
locations = mydict['locations']

print(outputlist)
```