## Assignment-1E4-Nachiketh-nxp251

## February 21, 2018

Question - Different ROC curves for feature detection.

For this I have calculated the right values and then computed the predicted values using the feature detector. I have plotted the ROC curve - True Positive Rate vs False Positives.

```
In [10]: import numpy as np
    import matplotlib.pyplot as plt
    import matplotlib.patches as patches

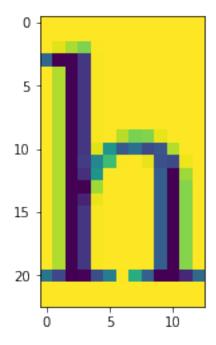
In [36]: image = plt.imread('../characters.png')
    image_gray = image[:,:,0]
    template = plt.imread('../template.png')
    template = template[:,:,0]

In [37]: image_height = image_gray.shape[0]
    image_width = image_gray.shape[1]
    template_height = template.shape[0]
    template_width = template.shape[1]

In [50]: right_values = [(321, 9), (448, 9), (517, 9), (19, 42), (64, 42), (18, 75), (412, 75)

In [39]: plt.imshow(image)
    plt.show()
```

does seem to work well in the face of these challenges is: the human visual system. It makes eminent sense, therefore, to attempt to understand the strategies this bio-100 logical system employs, as a first step towards eventually translating them into machine-based algorithms. With this objective in mind, we review here 19 important results 200 regarding face recognition by humans. While these observations do not constitute a coherent theory of face recognition in human vision (we simply do not have all the 300 pieces yet to construct such a theory), they do provide useful hints and constraints for one. We believe that for this reason, they are likely to be useful to computer vision. 400 researchers in guiding their ongoing efforts. Of course, the success of machine vision systems is not dependent on a slavish imitation of their biological counterparts. Insights 500 into the functioning of the latter serve primarily as potentially fruitful starting points for computational investigations. 600 100 200 300 400 500 600



```
(23, 13)
```

```
In [4]: def calcMetrics(true_values, predicted_values):
            count_correct = 0
            for each_value in predicted_values:
                if each_value in true_values:
                    count_correct += 1
            TP = count_correct
            FN = len(true_values) - count_correct
            FP = len(predicted_values) - count_correct
            return TP, FP, FN
In [5]: def threshold(sub_img, template):
            equals = 0
            total = 0
            for i in range(0, template.shape[0]):
                for j in range(0, template.shape[1]):
                    total += 1
                    if sub_img[i,j] == template[i,j]:
                        equals += 1
            return equals / float(total)
In [6]: def detect_feature(image_gray, template, limit):
            #fig, ax = plt.subplots(1)
            image_height = image_gray.shape[0]
            image_width = image_gray.shape[1]
            template_height = template.shape[0]
            template_width = template.shape[1]
            count = 0
            bottom_left_points = []
            for i in range(0, (image_height - template_height)):
                for j in range(0, (image_width - template_width)):
                    sub_img = image_gray[i:template_height+i, j:template_width+j]
                    threshold_val = threshold(sub_img, template)
                    if threshold_val > limit: #Adding a rectangle box
                        count += 1
                        bottom_left_x = j
                        bottom_left_y = i
                        bottom_left_points.append((bottom_left_x, bottom_left_y))
                        rectangle = patches.Rectangle((bottom_left_x,bottom_left_y), template_
                        ax.add_patch(rectangle)
            #ax.imshow(image)
            #plt.show()
            return count, bottom_left_points
In [71]: TPR_list = []
         FP_list = []
```

```
count, predicted_values = detect_feature(image_gray, template, 0.60)
         TP, FP, FN = calcMetrics(right_values, predicted_values)
         TPR_list.append(float(TP)/(TP + FN))
         FP_list.append(FP)
         print(TP, FP, FN)
38 100 0
In [72]: count, predicted_values = detect_feature(image_gray, template, 0.61)
         TP, FP, FN = calcMetrics(right values, predicted values)
         TPR_list.append(float(TP)/(TP + FN))
         FP_list.append(FP)
         print(TP, FP, FN)
38 92 0
In [73]: count, predicted_values = detect_feature(image_gray, template, 0.62)
         TP, FP, FN = calcMetrics(right_values, predicted_values)
         TPR_list.append(float(TP)/(TP + FN))
         FP_list.append(FP)
         print(TP, FP, FN)
38 9 0
In [74]: count, predicted_values = detect_feature(image_gray, template, 0.63)
         TP, FP, FN = calcMetrics(right_values, predicted_values)
         TPR_list.append(float(TP)/(TP + FN))
         FP_list.append(FP)
        print(TP, FP, FN)
38 8 0
In [75]: count, predicted_values = detect_feature(image_gray, template, 0.64)
         TP, FP, FN = calcMetrics(right_values, predicted_values)
         TPR_list.append(float(TP)/(TP + FN))
         FP_list.append(FP)
         print(TP, FP, FN)
38 0 0
In [76]: count, predicted_values = detect_feature(image_gray, template, 0.66)
         TP, FP, FN = calcMetrics(right_values, predicted_values)
         TPR_list.append(float(TP)/(TP + FN))
         FP_list.append(FP)
         print(TP, FP, FN)
```

