B657: Computer Vision
Assignment 1: Image Processing and Recognition Basics

Tousif Ahmed(touahmed)

February 12, 2016

1. Instruction to run the code

The following command is sufficient to run part 7:

```
make
./omr <image_name.png>
```

However, I also created separate commands to test part2- part6:

```
make
./omr <image_name.png> <part_no>
```

The part no should be p2, p3, p4, p5, p6 to test the corresponding parts. For p2 and p3 I manually set the filter size to 5. If no part_no is given then the code will automatically run part 7.

2. Design Decisions

1. I have tested both in linux and Mac machine. It works fine.
2. Implementing part 2 was straight-forward. I reused some of my code from CSCI-B 659 course that I took in Spring 2015.
3. I assume we could not do Dynamic Programming here. I used this function to test Sobel operator.
4. I changed the distance metric. I guess this is a similarity metric. I converted it to dissimilarity metric. That was easier to control the threshold. The dissimilarity metric can be calculated by simply subtracting the similarity value from the size of image or $m * n - f_T^I(i,j)$

   I spend so much time to convert the equation to general convolution but could not do it. I would be happy to know if there is any way.
5. Implementing sobel operator was straight forward. However, implementing the distance function was difficult. I tried the naive approach, which was excruciatingly slow. Then, I tried k-d tree but after spending some time I realized that k-d tree was not accurate. Later, after exploring several topics I figured there is an easy way to calculate the distance map in quadratic time. I used Chamfer Method to calculate the distance. Template matching part was straight forward.
6. Implementing Hough Transform was also challenging. I made several design decisions to detect the staves. First of all, as the lines in the staves are perfectly horizontal, I checked for lines only on horizontal direction.Also, instead of finding all the edges I only looked for horizontal edges which means I used Sobel only in vertical direction. After performing gradient operation, I used non maximum suppression to identify the dominating pixels. Then, in the horizontal direction I just calculated the number of pixels that vote for that vertical position.

I did not need to incorporate the space dimensions, as generally staves are really long so by limiting the threshold I was able to detect lines for several images. However, it did not work for really noisy image.

7. Overall, this part was more challenging. After doing Hough Transform, I stored the indexes of the lines in vector. I tried to detect the notes by checking the indices of the staves, however due to round up error it did not work. The algorithms performs well with a clean image, however, it did not work well with other test images. My rescaling procedure is really naive, for that reason the algorithm did not work well.

   In my code, the general template matching algorithm (without edge map) worked well.

3. Evaluation

The code was able to detect notes only in music1,music2 and music3 file.

In Music 1, total notes detected = 42, Accurace=91.5% In Music 2 test image, Total correct notes detected =11, Accuracy=14.47% In Music 3 test image, Total correct notes detected =0, Accuracy=0% In Music 4 test image, Total correct notes detected =15, Accuracy≈10%