B657 Assignment 1: Image Processing and Recognition Basics

Submitted By:
Thijs Benschop (thijbens@indiana.edu)
Debasis Dwivedy(ddwivedy@iu.edu)
Bipra De(bde@indiana.edu)

Instructions on running the code:

In the terminal, go inside the project directory and run the below 2 commands in chronological order.

1. make
2. ./omr <input_image> <template1> <template2> <template3>

Note: Since we have used **opencv** for image scaling, there has been a change in the make file.

```
omr : a1.cpp SImage.h SImageIO.h DTwoDimArray.h
        g++ -O3 -o omr a1.cpp -I . -lpng `pkg-config --cflags --libs opencv`
```

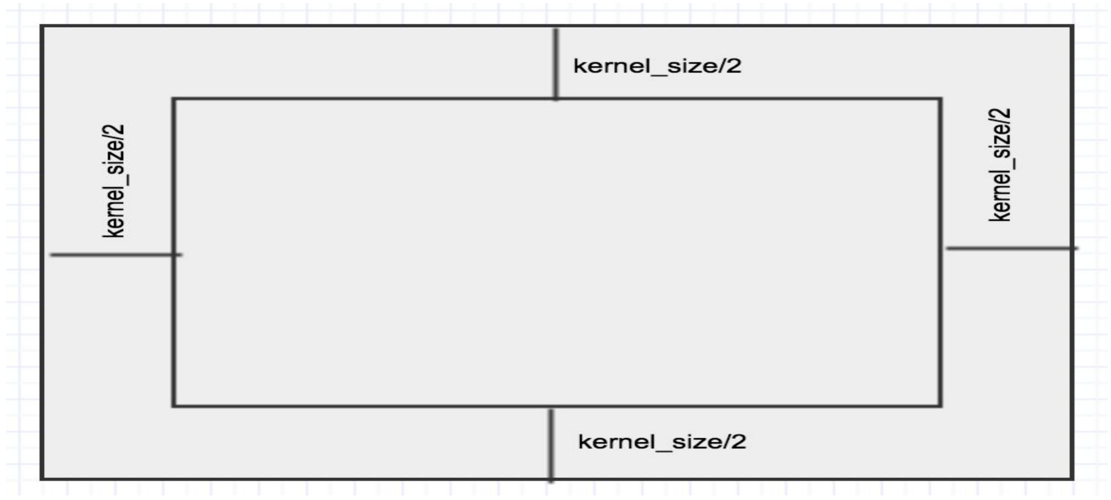
Q2 and Q3.

- A NxN box kernel was used for convolution for Q2.We experimented with different sizes of kernel i.e. N starting from 3 to 10
- For Q3, we used 1xN row vector and Nx1 column vector to perform two convolution passes and varied the size of the kernel from 3 to 10.

Boundary case : When the Kernel doesn't completely overlap with the image during convolution at the image edges

Approach to handle this :
We tried 2 approaches:

1. Zero padding the kernel regions not overlapping with the image
2. We ignored performing convolution on (**kernel_size/2**) number of pixels from all the four sides of the original image. Thus, the boundary case was handled as the kernel never went outside the image area during convolution operation.



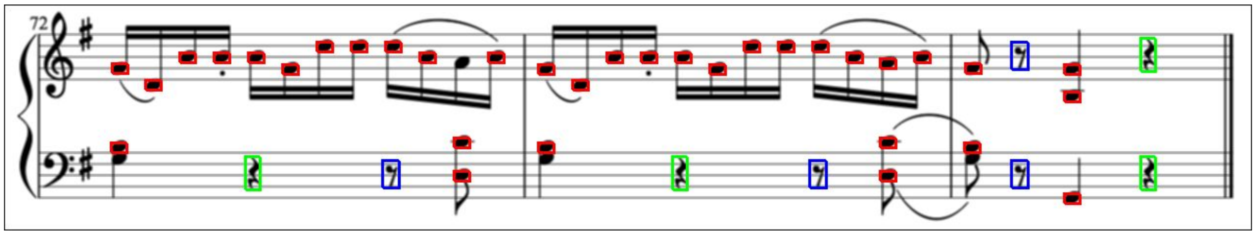
Formula Used for convolution:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

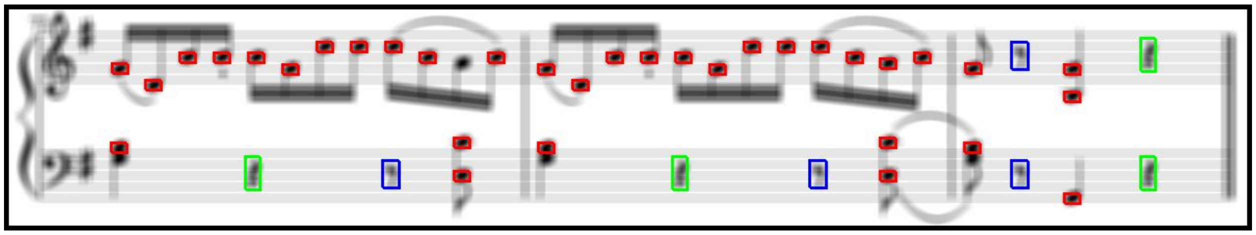
Observations made : As we increased the size of the kernel, the blurring effect increased.

Output for music1.png:

Kernel size=3



Kernel size=10



Q4.

1. We first converted the input image and the template into binary image using 151 as pixel value threshold for assigning 0 to a pixel

```
if(image[i][j]<=151){
        output_image[i][j]=0;
}
else{
        output_image[i][j]=1;
}
```

2. We applied the hamming distance formula as our similarity function by performing convolution on the image to match an image area with a template.The result was stored in a SDoublePlane type matrix named “hamming_matrix”

3. Now, to find the top left coordinate of a detected symbol,we computed the *max* cell value in hamming_matrix. The coordinate of all those pixels whose value is greater than 91 percent of this max value becomes the coordinate of a detected symbol which is then pushed to the symbols vector.

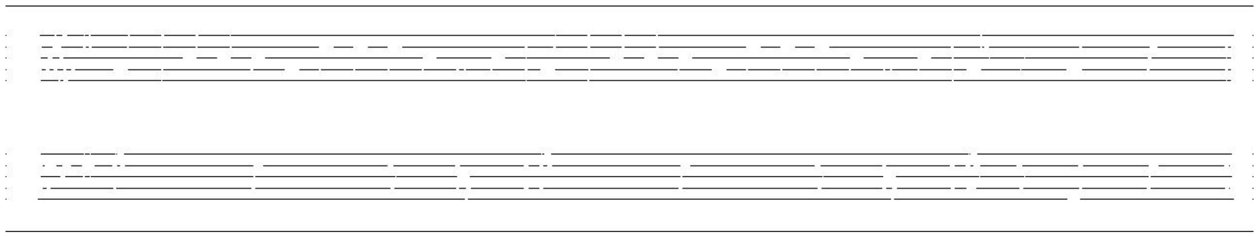
Note : We discussed this third step on a high level with Suhas.

4. To detect the staff lines, we used the below kernel and performed convolution on the input image.
This kernel was able to detect the horizontal lines in the image properly for music1.png but gave a poor performance for other images especially music3.png.

[Source : http://homepages.inf.ed.ac.uk/rbf/HIPR2/linedet.htm]

-1	-1	-1
2	2	2
-1	-1	-1

To accurately detect only the staff lines, we first changed the pixel values of the convoluted image to 0 if the value at that pixel was less than -460.Else they were set to -255. After many trials this threshold was found to give a decent accuracy. This results in a black and white image containing the horizontal lines as shown below.



5. Then we iterated over the above image and started counting the number of pixels in a row whose value is 0 i.e. all the black pixels.If the the total number of black pixels in a row is greater than *20 percent* of the total number of pixels present in a row in the original image, then the row is declared as a staff line and the row number is assigned to a vector named “indices” storing such staff lines’ row coordinates. This *20 percent* threshold was selected after several experimental runs.

Note: For the convolutions performed in this step, the boundary case of non-overlapping kernel parts with the image was handled in the same way as done for Q2 and Q3 mentioned above.

6. Now, to compute the pitch of a note head, its centre coordinate is computed and compared against the staff line coordinates computed in the previous step and is then assigned an appropriate pitch based on the rules specified in the *compute_pitch* method. This method has been made generic enough to compute the pitch when the image has more than 10 staff lines like in music3.png and rach.png.

Observation for Q4: Blurring the input image gives us poor results for staff line detection which in turn affected the accuracy of the symbols’ pitch.This is because blurring being a high pass filter smoothes all the high frequency regions of the image like the edges. Thus, line detection accuracy deteriorated.

Q5.

1. We first converted the input image and the template into binary image using taking the highest_value_of_pixel/5 as pixel value threshold for assigning 0 to a pixel

```
if(image[i][j]> highest_value_of_pixel/5){
        output_image[i][j]=255;
}
else{
        output_image[i][j]=0;
}
```

2. Sobel filter has been implemented to detect the edges of the input and template image. Before applying sobel filter the image is blurred using a 21X21 gaussian kernel to reduce the noise if any. The output of the sobel filter is passed as the input to the Cross correlation function. The function definition are mentioned below:-

```
edge_input_image=output_edge_detector(input_image);
edge_template_image1=output_edge_detector(template1_image);
edge_template_image2=output_edge_detector(template2_image);
edge_template_image3=output_edge_detector(template3_image);
```

Cross Correlation:-

```
FIT(edge_input_image,edge_template_image1,symbols,NOTEHEAD);
```

FIT has the method **closet_edge_pixel** to calculate D and Gamma for the input image.

3.The edge detected image is converted to binary by making all 255 to 1 and keeping 0 as zero. The cross correlation formula was given to us in our assignment PDF.

4. We applied cross correlation to the input image with the template. The point where the image correlates gets a lower value compared to other pixel points.

5.The distance matrix D(I,J) is represented in the "**closet_edge_pixel**" method where the closest distance between the corresponding pixels with its edges is found.

6. The image template is then cross correlated with D and gives out the output which has the lowest pixel values where they correlate and higher otherwise.

7. The threshold to denote the value is set to be as maximum_output_pixel/17 for which we get the detected symbols.

8. Boundary condition has been implemented by Zero-Padding Technique.

Observations for Q5
It was taking a long time to produce the desired output i.e., Q(n^4) as we use 4 loops to get the solution.An optimization to reduce this time was later implemented to reduce the run time.

```
if(i>=closest_edge_output.rows()-template_binary_image.rows()||j>=
closest_edge_output.cols() - template_binary_image.cols()) {
        output[i][j] = 999999999;
        continue;
        }
```

Q6. Staff Line Detection

The technique we used for implementing the Hough Transform on this problem turned out to be very specific to this task in particular. As suggested in the assignment, the two dimensions of the transform were votes as to 1. The row of the top staff line in each section, and 2. The scaling of the corresponding staff. This was performed in a few steps:

1. Traverse the image, one pixel at a time. When a black pixel was found, we checked the pixels to the right and the left to ensure that it was a potential parallel line.
2. When a candidate pixel was found, we considered the pixels in the column directly below it. 4 candidate black pixels were found, and we checked that each was separated from the one above by a white pixel.
3. After a set of 5 candidate staff pixels was found, we checked if the separation between them was consistent, on the hypothesis that staff lines are roughly parallel and equally spaced. There was some error allowed here (+/- 5 px).
4. If a pixel had passed both of these tests, a vote was cast in the Hough voting space for this pixel's row, with a scale vote of (top px)-(bottom px)/4, being the average whitespace between lines found.
5. After the voting was completed, noise was removed in three steps. The first was thresholding, and candidates less than double the average voting score were removed.
6. Secondly, non-maximal suppression is performed, with each remaining candidate being compared against its top and bottom neighbors and only those lines greater than both of these will be retained.
7. The final step is also particular to this problem, with the candidates traversed from top to bottom, and when a line was added to the final answer, it removed any remaining candidates below that would fall within its staff line (row+(scale*5))

After these pruning steps, detection was accurate on all of the test data, and provided us will valuable information that was used for scaling in part 7, as well as pitch tagging in part 4 (as simple convolution line detectors were insufficient for finding staff lines).

Q7. Best effort detection

For part 7, we used the parts of the previous sections that worked the best for symbol detection, and enhanced with information from staff line detection. The main enhancement was to scale the template images to the found staff sizes, which provided much better results on the outlier images. The scaled template images were also convolved with a gaussian kernel,as the input image had also been convolved with the same. Other than this improvement, we could have performed more filtering, but much of the assignment effort was spent on the first sections of the assignment. Non-maximal suppression should also have been added to the symbol detection, which would have allowed for more sensitivity.

Evaluation Results:

Music1.png

```
Mean average precision for filled_note = 1.000000
Mean average precision for eighth_rest = 0.342113
Mean average precision for quarter_rest = 0.000000
OVERALL Mean average precision = 0.447371
---
Fraction of correct note_head detections with correct pitch = 0.853659
```

Music4.png

```
Mean average precision for filled_note = 0.834878
Mean average precision for eighth_rest = 0.126695
OVERALL Mean average precision = 0.480786
---
Fraction of correct note_head detections with correct pitch = 0.646552
```