Assignment 1
CSCI B657 - Computer Vision

prmurali-sriksrin-anikgaik-a1

Prashanth Kumar Murali, Srikanth Srinivas Holavanahalli, Aniket Gaikwad

Files Present and how to run the code:

a1.cpp - Contains the main function and the core logic of the assignment including the convolutions and edge detection.
essentials.h - Contains the essential functions which were provided with the assignment like write_detection and overlay_rectangle
extra_manipulations.h - Contains all the image manipulation functions created by us.
hough.h - Contains the code for hough transformation.
a1_test.cpp - For experimentation purposes we tried to match the templates of individual notes to the ones in the actual image by rescaling. It didn't work with good accuracy so we have reserved its inclusion.

Extra Images:
Images of all musical notes(in Notes Template A.png-G.png), Plotted Hough space image (hough_space.png)

Running Instructions:

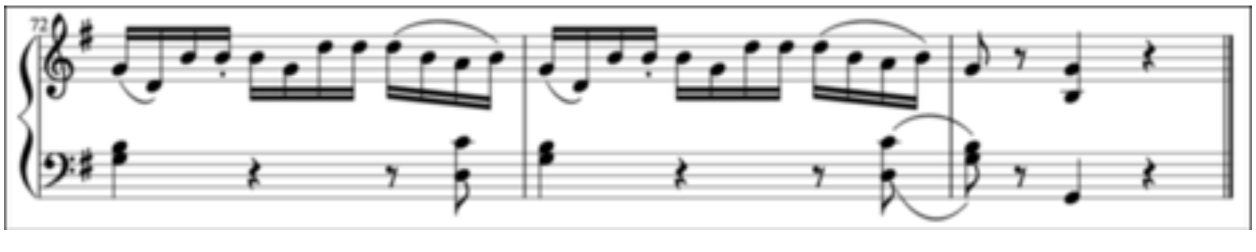
```
$ make
$ ./omr <image name>
```

The above commands will run and produce results for all the questions.

Question 2:

For handling the edges in the image we added extra rows and columns, with the pixel values equaling last row and column of the image. We find the number of rows and columns in the image and the templates. Then we find the Lowest Common Multiple of these values and match the image dimensions to that by appending pixels to the end. The convolution handles filters with odd dimensions (3X3 5X5 etc.).

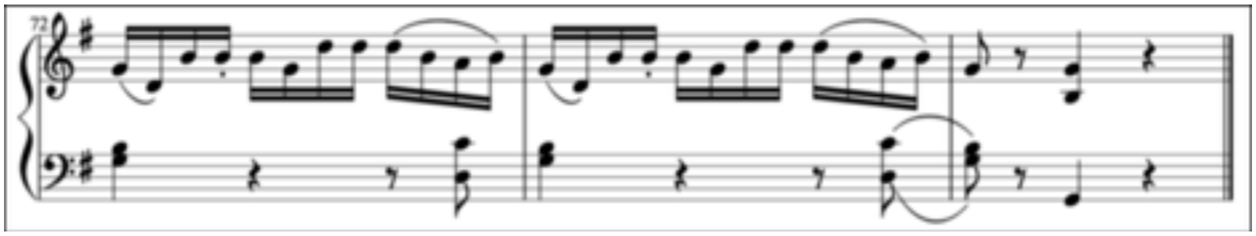
Images:



Question 3:

The image boundaries are handled the same way as in question number 2. It is a general separable convolution implementation by convolving with the row first and then the column kernel. This method handles for only odd dimensioned kernels (1X3, 1X5 etc.).

Images:



Question 4:

While calculating the hamming distance, we modified the given formula by replacing (1-I)*(1-T) (where I is the image, T is the template) by (256-I)*(256-T) for normailization purpose. The intuition behind this change is that by subtracting both image and the template by 256 we are inverting the image pixels instead of converting them to binary by assuming some random magic threshold. We obtained better results by this approach.

We convoluted the image and the template based on the hamming distance formula and find the hamming distance matrix for the whole image. To get a threshold value, we convoluted the given template with itself which results in the matrix having non zero value at location (0,0) and 0 at rest of the locations. We used 90% of location (0,0) value as the threshold. This is because when we have the exact match between the same image, we get the highest score.

We first identify the staff lines by the find_lines function which will loop through the whole image look for the rows with black pixels grater than 80% (optimal) or any threshold value specified. We can change the threshold by passing it as a parameter to the find_lines function.

Then we find the ledger line difference from the identified lines. We set smart thresholds to eliminate false values and get only the difference between adjacent ledger lines. We take an average of the first 4 differences (for the first 5 lines) and mark that as the ledger line difference for that image. Assumption is that the file will maintain this difference throughout. We have identified the ledger line difference of music1 image as 11.0 and since the templates are based on this image we rescale the templates to the target image dimensions by the following formula:

rescale_ratio = ledger_line_difference - 11.0/11.0
image_ratio = template_cols/template_rows

new_rows = template_rows * rescale_ratio
new_cols = template_cols * image_ratio * rescale_ratio

Then we fill the extra pixels by replicating the previous values to it. We do this uniformly by taking the modulo of the increase with the number actual and inserting pixels in equal intervals. We then pass these templates through the image to find the matching points.

Based on the co-ordinates of the matching points and the lines we map the pitch by the rules followed from this image:

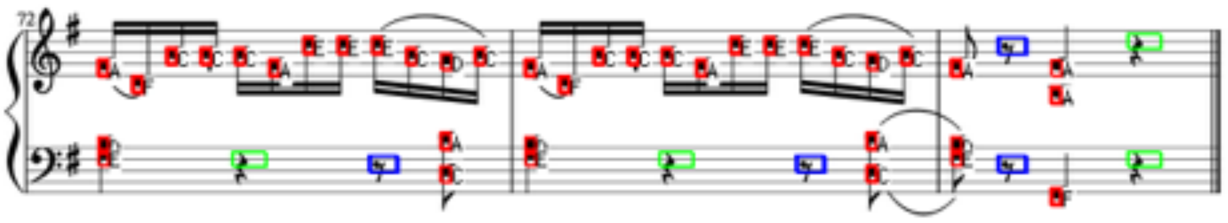
http://thumbs.dreamstime.com/z/music-note-names-notes-bass-treble-clef-isolated-white-50399154.jpg

Algorithm:

1. Read image and templates
2. Modify image to handle edge cases
3. Find lines and ledger differences
4. Rescale the template to map to the ratio of the input image
5. For each template (3 in our case) generate hamming score and get the matching pixels
6. Provide for pitch detection with line co-ordinates and matching template co-ordinates.
7. Print the matches as an image.

In about 98% of confidence we get a result of 0.99826 faction of correct pitch detection from the a1_eval.py. We have not given scores4.png as it was the hamming score distances with values of the order 10^6 instead we have given the scores as scores4.txt which has all the values.

Images:



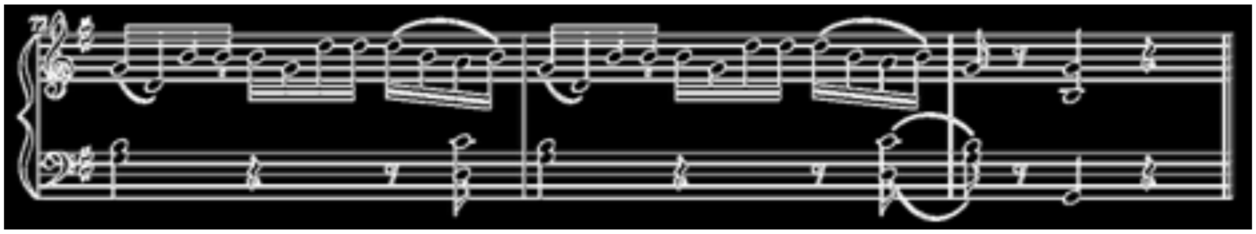
Question 5:

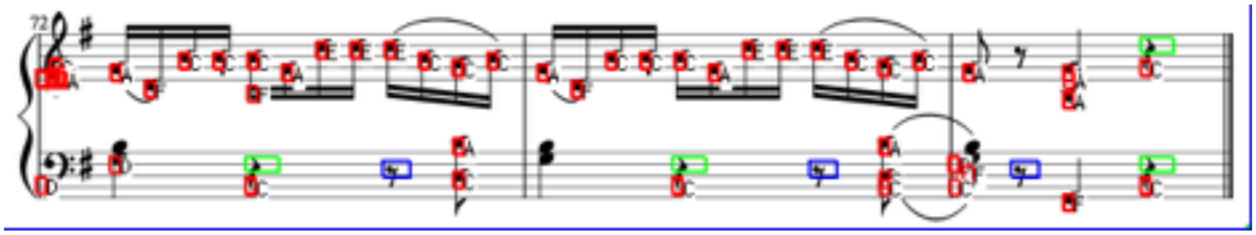
Algorithm:

1. Apply sobel filter over input image and template using convolution.
2. Find edge pixel distance for all the pixels in filtered input image.
3. The output would be 0 for edges and distance to the nearest edge for the non edge pixels (Distance is Euclidean).
4. Compute the scores of the distance matrix and the template window with threshold 1000.
5. Find the matches (points lesser than 1000) and provide for plotting.

6. Pitch detection is carried out by the previous method.

Images:



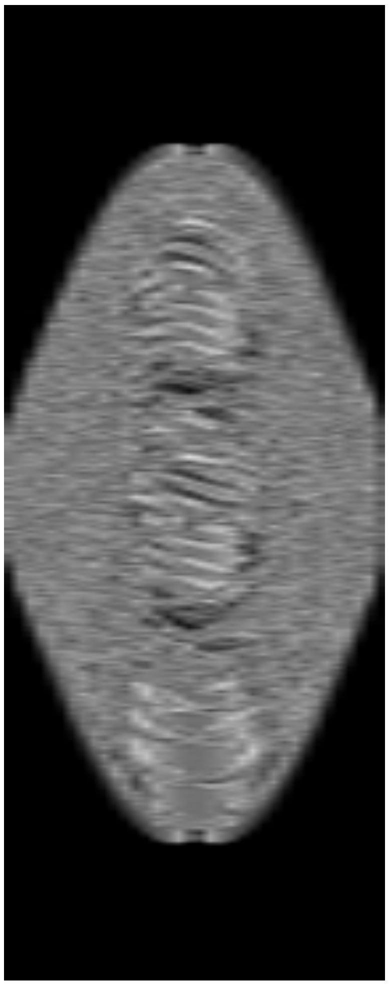


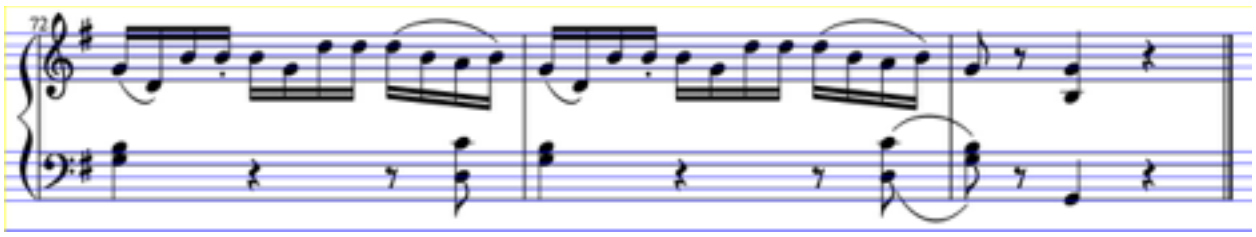
Question 6 [1]:

Algorithm:

1. Filter the image by sobel gradient and pass it for hough transformation.
2. Compute the accumulator size by the image dimensions and accumulator width is 180 (theta).
3. For all pixels above 200.0 (Well illuminated or actual points) increment on the accumulator the corresponding values (Forms the line based on x*cos(theta) + y*sin(theta))
4. Plot the hough_space as an image.
5. Then based on accumulator value above 600 (threshold) we find the x and y value in the image space.
6. We restrict to only the staff lines as follows:
    - y co-ordinates do not vary by more than 2 pixels (horizontal lines).
    - The start y co-ordinate must be more than 5 pixels away from the previous detection.
    - The y co-ordinates do not vary from the previous detection by more than 50 pixels or this is the first staff line of the series.
7. Then we plot the line by Bresenham's line drawing algorithm [2]

Images:



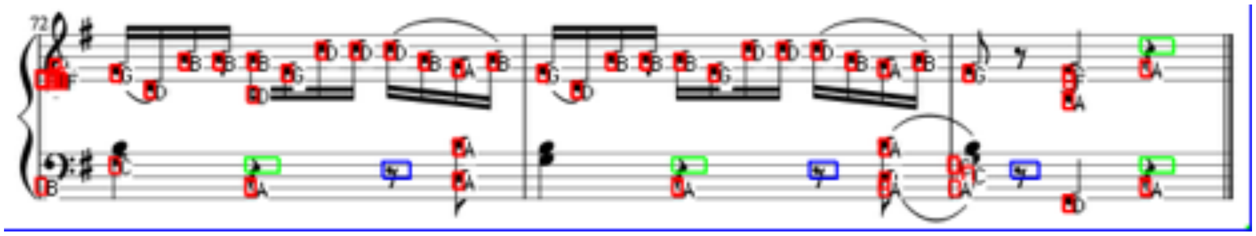


Question 7:

a) Loaded.
b) Answer as above.
c) Rescaling is done for all questions.
d) All notes detected.

Images:

Extra methods:



We implemented a simple image filter (image_filter extra_manipulations.h) which will take a difference of 2 images and present it as a different image. We also implemented a Gaussian function (gaussian extra_manipulations.h) which will generate a gaussian filter of given dimensions. Using this we did a difference of gaussian filter (somewhat similar to the one mentioned in SIFT) to get the outline image. We were able to produce good results with music1 but we were not able to consolidate the edges well in music3 and images which have a lot of noise.

For question 4 we tried to map the templates of actual notes with the hamming distance, but did not achieve expected results. You can find that implementation in a1_test.cpp.

Rescale_template is a native image rescaling function we built. It maintains the ratio without much loss in quality of image.

References:

[1] http://www.keymolen.com/2013/05/hough-transformation-c-implementation.html
[2] https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm