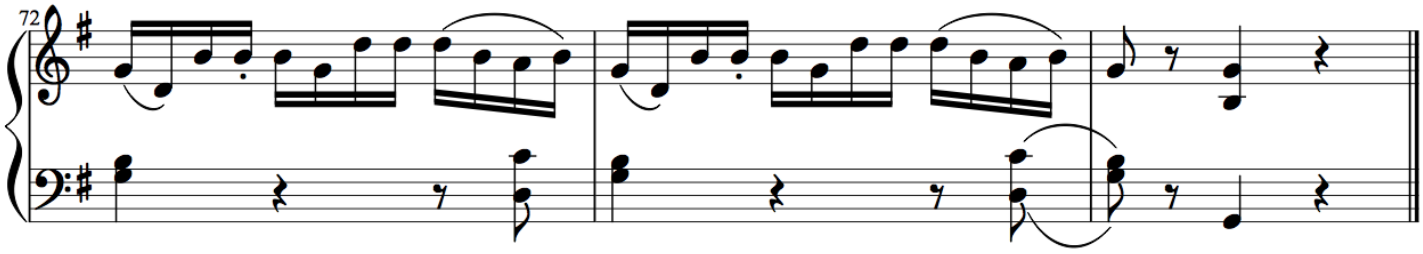
B657 Computer Vision

Image Processing and Recognition Basics

For the report, we have attached images that we obtained when music1.png is passed as the input.



The report states the assumptions we made while working on the assignment, the difficulties we faced, the output we got and the right way to run our program.

Ayoub Lassoued

Manikandan Murugesan

Pradeep Kumar Ravilla

Assignment Report 1

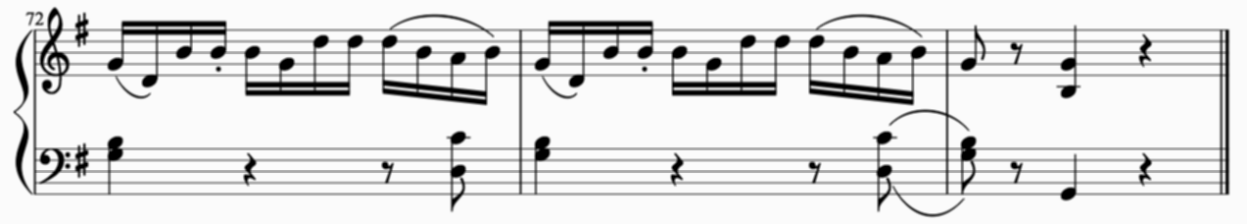
B657 Computer Vision

Convolution

We wrote the naive implementation of the convolution code. Our code iterates through each pixel in the image and convolves against the filter.

For the filter, we tried the mean filter which was given in the skeleton code. But through repetitive testing, we found that Gaussian filter was better and we used a 3x3 Gaussian filter to convolve the image and reduce the noise.

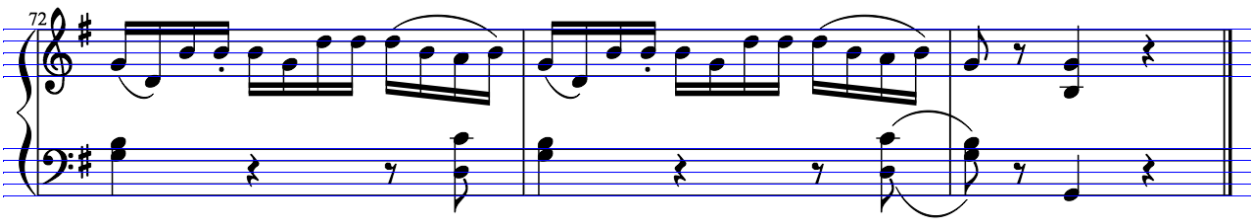
For the boundary conditions, we used a padding of white pixels around the original image to work with. So this gives us the following image:



This reduces the noise in the image to an extent. We also made the convolution function in a generic way to take in parameters that we pass and return the correct output to reuse it for other applications.

Staff Detection

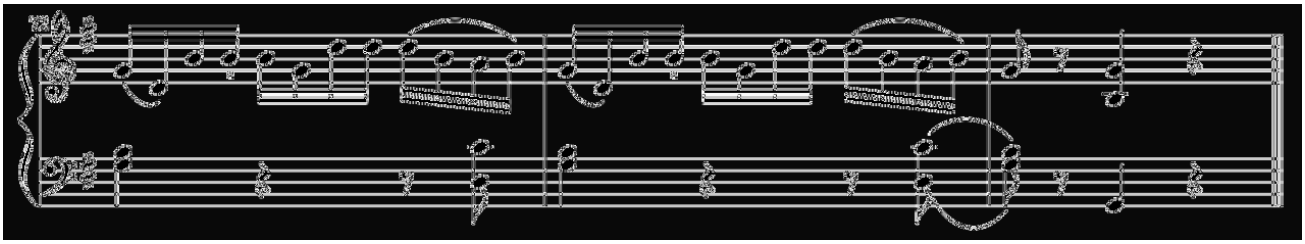
We detect each staff line using Hough Transformation. We store them in a new class we created. Each adjacent line is equidistant. We mark the staff lines on the image. The output can be found as follows:



Template Matching

We implemented the template matching with the given formula.

The output for the template matching score is given by the following image:

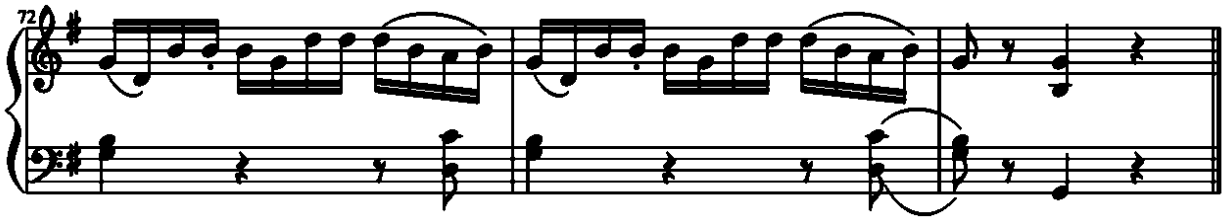


The above image shows the score. Once this is done, it finds the notes and adds it to symbols which is printed in the end as part of a text file as well as marked on the image.

Template Matching after Edge Detection

To achieve this step, we took the image convolved with Gaussian filter and then converted it into a binary image.

This we achieved by assuming a threshold value after trial and error method. The image that we got is :



Once we did this, we convolved the image with Sobel filter and found the edges.

We observed that when we use the values for the Sobel filter as follows:

$$H_x^S = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}$$

$$H_Y^S = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

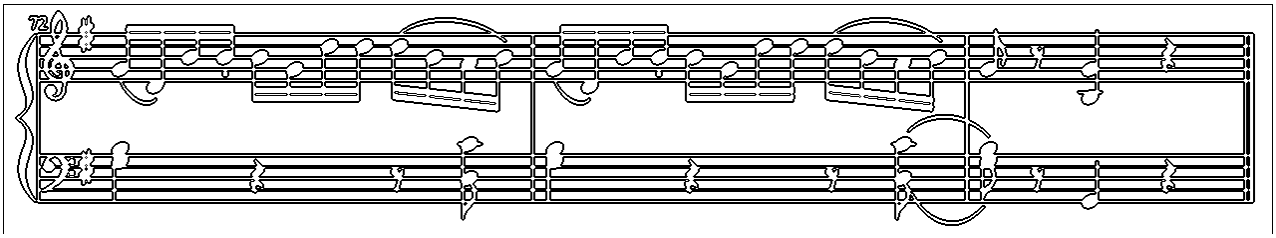
This we do to normalize the Sobel filter

$H_x^S = (1/32) \times H_x^S$

$H_Y^S = (1/32) \times H_Y^S$

Using the above filter showed that it is better than using the traditional Sobel filter and also Canny Edge detection technique.

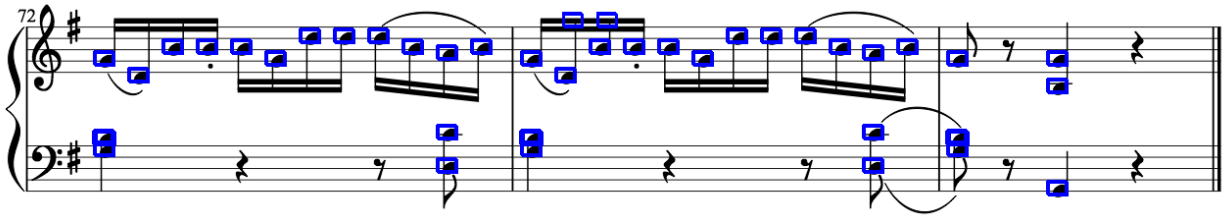
The image we got from the filter is given as follows:



This gives the proper edge map we expect. Now we find the edge map for the templates. The edge maps for the templates are given as follows:

Now we run template matching across edge maps. We find the symbols and pass it to a function to mark the symbols on the image.

Our template matching for template1.png yields the following results:



As we can see from the image, even though it finds all the notes, it detects some false positives too which can be avoided.

Time taken for the program to run

For the convolution to run, our program takes around 175 ms

The template matching using step 4 takes 400 ms to run

The template matching after edge detection using step 5 takes around 28900 ms to run

Our assumptions and Simplifications

For the boundary conditions, we set white pixels around the image and then convolve the image with the kernel so that the boundaries are also considered.

This could have been done better by using reflection. This could be done as an improvement.

Even though our program's speed drastically improved for the step 5 after ignoring edge points for D(i,j) calculation, we feel that it could still be improved. Further research on this might yield a good solution.