Computer Vision
Assignment 1

Bardia Doosti
Jayalakshmi Sureshkumar

February 10, 2016

1 Optical Music Recognition (OMR)

Music OCR or optical music recognition (OMR) is the application of optical character recognition to interpret sheet music or printed scores into editable or playable form. Once captured digitally, the music can be saved in commonly used file formats, e.g. MIDI (for playback) and MusicXML (for page layout)[1].

In this project we get a piece of music notes and we should detect the notes and the pitches. For this goal we have implemented two algorithm for convolution matching, Sebel algorithm for edge detection, better algorithm just for horizontal lines (because in this project they are more important), Hough algorithm to detect the staves, a nearest neighbor algorithm for resizing the images and some algorithms for detect the pitches.

2 Convolution Function

In mathematics and, in particular, functional analysis, convolution is a mathematical operation on two functions f and g, producing a third function that is typically viewed as a modified version of one of the original functions, giving the integral of the pointwise multiplication of the two functions as a function of the amount that one of the original functions is translated [2].

If H is the input image and F is our filter

$$G = H * F$$

with this rule

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

We implemented this function in `convolve_general` function. The function `convolve_general` performs convolution given a filter and the image matrix. The function `convolve_general` takes a row and a column matrix, performs convolution first with the row matrix along the row of the image matrix, and then with the column matrix along the columns of the image matrix.

3 Separable kernel convolution

Separating the filter into row and column matrix, and then performing separable convolution considerably reduces computation. If the image is of a * b dimension and the filter is $m \times n$, the convolution process is $O(a \times b \times m \times n)$. This is reduced to $O(a \times b \times (m+n))$. For a filter which is not very small, this makes a huge difference as image processing can be an exhaustive process.

The function `convertToBinary` converts the representation of an image to binary. Given a threshold, values above and equal to the threshold are set to 0. This represents the close to white and white pixels. The values below the threshold are set to 1, representing black and close to black pixels.

The function `detectTemplate` detects the template. The parameters to the function are the image, the template, threshold for grey scale which is for converting the values of the image to binary, the percentage to set a threshold on the similarity score and the type of template which is used to set the color of the highlight region of the template. Playing around with various threshold, a threshold of 128 on the grey scale and a fraction of 0.89 on similarity score worked to get the best outcome for music1.png. This function uses the convertToBinary function to convert the input image and the template and uses the converted images to get the scores by calling the function similarityScores. Once we compute the score, the maximum score is found, and any pixel with score greater than a fraction of the maximum similarity score is identified as matching the template.

The code saves the output of this algorithms in `detected4.png` file. Figure 1 shows the notes and pitches detected by the convolution algorithm.

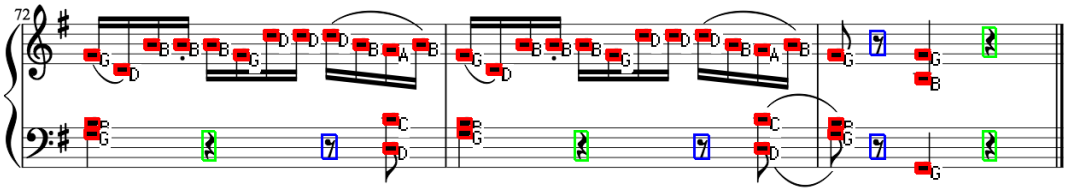

Figure 1: The notes and pitches detected by the convolution algorithm

4 Sobel Algorithm

The `sobel_gradient_filter convolves` the image with the sobel operators along x and y direction, and then gets the combined result of the two.

$$G = \sqrt{(G_x^2 + G_y^2)}$$

The function `similarityScores` calculates the hamming distances at each pixel between the region around the pixel and the given template. The image and the template given as a parameter to this function are in binary format. The pixel with the highest similarity score is most likely the best match to the template. This function returns a vector of symbols. These symbols are highlighted in the image by calling the `write_detection_image` function with the vector of symbols as a parameter.

The code saves the output of this algorithms in `detected5.png` file. Figure 2 shows the notes and pitches detected by the sobel algorithm.

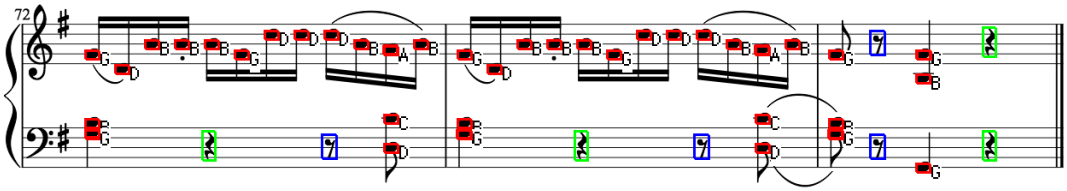

Figure 2: The notes and pitches detected by the sobel algorithm

5 Edge detection

Sobel algorithm will do the all the edge detection and our code will save the result in `edge.png` file. Figure 3 shows the edges detected by the sobel

algorithm.

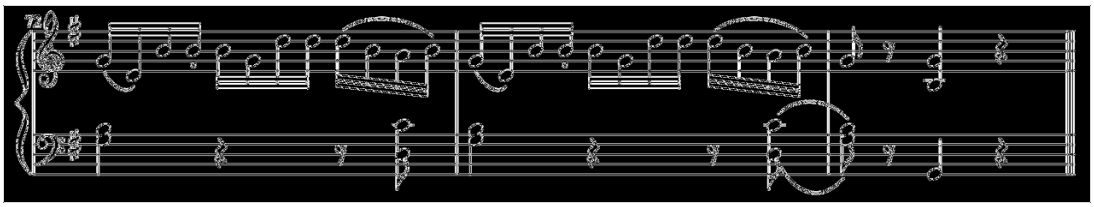

Figure 3: The edges detected by the sobel algorithm

The sobel algorithm is pretty good for edge detecting in a general problem, but in this problem as we see all the edges that we need are horizontal edges. So we implemented another algorithm for just detecting the horizontal edges. The code saves the output of this algorithms in `horizontaledges.png` file. Figure 4 shows the horizontal edges detected used in this code.

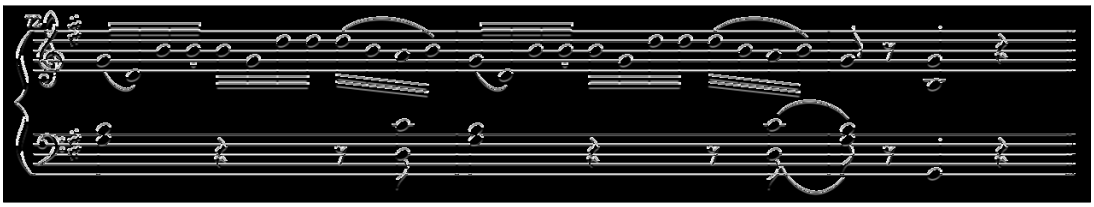

Figure 4: The horizontal edges detected used in this code

Also our code will save the detected staves in the `staves.png` file and show the staves with blue lines. Figure 4 shows the staves detected in the code.

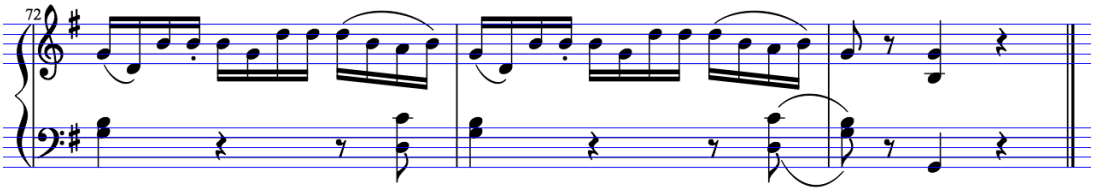

Figure 5: The staves detected in the code

6 Resizing the image

The scale of the input image may vary but our template is always same. To solve this problem we first detect the staves and then from distance of the staves we scale our templates. As we wanted our input and output image be the same size, we resized the templates instead of the input image.

For resizing the image we wrote two function `resizeImageScale` and `resizeImage` to resize the image. These functions simply resize the image using the nearest neighbor algorithm. It is a simple algorithm but it results good. Figure 6 shows actual image (center) and scaled version with scale factor 0.5 (left) and 2 (right).

Figure 6: Lena image scaled bigger and smaller with nearest neighbor algorithm

7 Future works

We did our best to write a code that will detect all the notes and pitches but because of being in short of time this is not perfect and many works can be done to improve this code.

7.1 Computation time

There are some place in code that can be computationally improved. The most important part is the convolution part. For this part we search all the image to find the template, but better algorithm is to search just on and between the staves. This will cause the algorithm much more faster and makes it more accurate.

References

[1] "Music OCR", Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 22 July 2004. Web. 10 Aug. 2004., *https://en.wikipedia.org/wiki/Music_OCR*

[2] "Convolution", Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 22 July 2004. Web. 10 Aug. 2004., *https://en.wikipedia.org/wiki/Convolution*