# Computer Vision Library for Western Music Sheet Notations

Sanka Rasnayaka
Department of Computer Science and Engineering
University of Moratuwa
Sri Lanka
sanka.11@cse.mrt.ac.lk

***Abstract*—This paper discusses a computer vision system to detect western music notations from images. The developed library will take in images of western music sheet notation and identify the key features necessary to extract the notes. The images will go through several pre-processing stages and then using straight line detection techniques the staff and notes will be detected. The paper will discuss the algorithms used and developed to achieve this. Finally the paper will present the accuracy measures in the developed system for different types of images.**

***Keywords—computer vision; notation; detection***

## I. Introduction

Western music notation scheme is a highly structured and methodical notation scheme which uses five horizontal lines (the staff) to place notations according to their pitch. This is illustrated in the Fig. 1.

With the high use of technology in music, there are many software which could be used to generate, edit or play western music sheet notations. However a vision system which could detect sheet notations and recognize notes was lacking, making vast number of printed sheet notations unusable for digital processing without manually entering them into digital formats.

All the existing developments focus on creating western music notations and there are no popular solutions for extracting data from an existing western music notation in a computer understandable way.

The library discussed in this paper is intended to extract key features of the western music notation scheme using standard image processing and computer vision techniques. These features would be used to identify the notations in the sheet music.

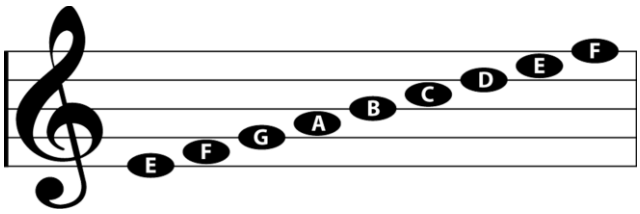


Fig. 1. Western Music Notation Scheme.

The paper will first discuss the related work in the fields of western music and the computer vision libraries selected for the implementation. Next the implementation details of the algorithms and the results obtained by the developed algorithms is presented.

## II. Related Work

There are several standards for representing western music notations, like musicXML [1] which is a digital sheet music interchange and distribution format. Also there are specifications like SMuFL [2] which provides a standard way of mapping the thousands of musical symbols required by conventional music notation into the Private Use Area in Unicode's Basic Multilingual Plane [9] for a single font.

Also there are open source applications which give features like creating sheet music, playing them and composing music. For an example Musescore [3] is a free and open source software used for music composing and notation creating. Musescore supports exporting musicXML files to save sheet music that have been created.

In order to create a vision library for detecting western music notations currently available computer vision libraries were searched. OpenCV [4] is an open source library which provides many features. Therefore OpenCV was used in this project. Since the library was created in java the JavaCV [5] wrapper for OpenCV was used for the implementation.

## III. Design and Implementation

The system flow is shown in Fig. 2.

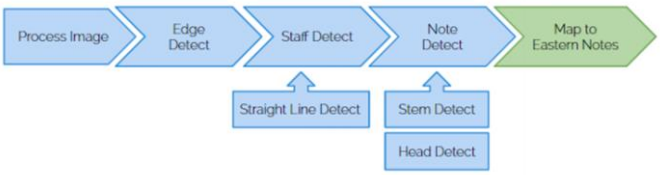


Fig. 2. System flow.

### A. Processing Image

At the initial stage several image processing algorithms were run on the images in order to standardize them for the detection algorithms. Some such operations were; converting the images to binary images, scaling the image, rotating to make sure the staff is horizontal and smoothing the image to reduce noise effects.

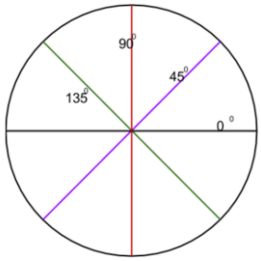


Fig. 3. Chosen angles.

### B. *Edge Detection*

The Canny edge detection algorithm [6] was implemented for detecting edges in the image. This algorithm can be explained in the following four steps:

1. Applying a gaussian mask: A mask matrix is used to average the values of neighbouring pixels.
2. Sobel operator: The gradients on the horizontal and vertical directions were calculated for each pixel by applying a sobel operator. The equations (1) and (2) show the two masks that were used for this.

$$xMask = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (1)$$

$$yMask = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (2)$$

At each pixel the mask was applied using its neighbors and the gradient values in $x$ direction ($G_x$) and $y$ direction ($G_y$) were calculated.

3. Calculate edge directions: The angle and the strength for each pixel was then calculated with the following equations:

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (3)$$

$$Strength = \sqrt{{G_x}^2 + {G_y}^2} \quad (4)$$

4. Trace edges: The adjacent pixels for each pixel would be traced along the directions and edges will be detected when directions match. For the current application, angles were reduced to 4 ranges shown in Fig. 3, and the tracing was done only using the immediate adjacent pixels.

Table 1 shows how this tracing happens from i(x,y)'th pixel to the next.

TABLE I. TRACING DIRECTIONS

| Angle | Next Position |
|---|---|
| 0° | (x + 1, y) |
| 45° | (x + 1, y + 1) |
| 90° | (x, y + 1) |
| 135° | (x − 1, y − 1) |

Fig. 4 shows the sample image used to illustrate intermediate steps in this paper, and Fig. 5 shows the result after applying the canny edge detection algorithm explained above

### C. *Staff Detection*

The edge detection output is fed to the staff detect algorithm, which would use the openCV cvHoughLines2 [7][8] for straight line detection with a tolerance in order to make sure the same lines are not selected multiple times due to its thickness.

The OpenCV implementation uses the polar coordinate system in representing the points. Using this system the Hough Transform in OpenCV keeps track of the intersection between curves of every point in the image. If the number of intersections is above some threshold, then it declares it as a line with the parameters $(\theta, r_\theta)$ of the intersection point.

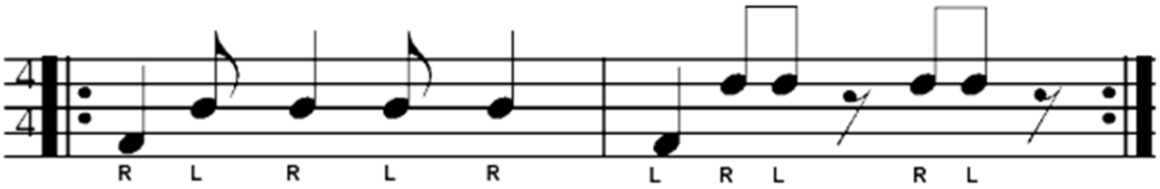


Fig. 4. Sample Western Music Notation.

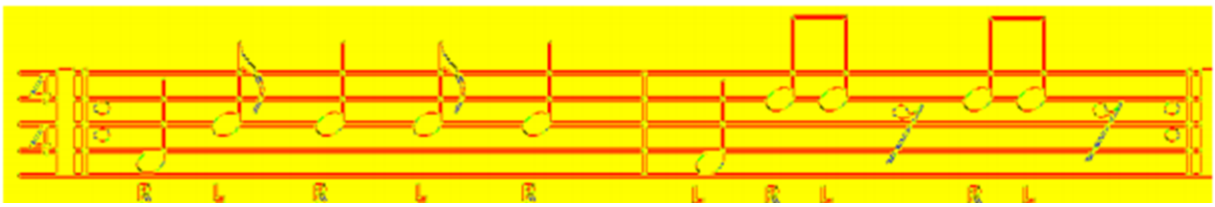

Fig. 5. After Canny Edge Detection.

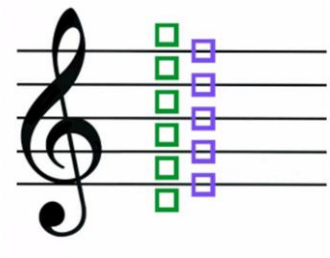

Fig. 6. Head positions.

After that the length of the lines is used to filter out within the selected lines and to identify the staff of the notation. The vertical position of each staff line is the only necessary feature for the next steps. The result after staff detection is shown in Fig. 7.

### D. Note Detection

*First the stems of the notes were detected using the same approach to detect vertical lines. After that the heads of the notes were detected using the relative position of the heads with respect to the stem and the staff as shown in Fig. 6.*

*The level of white vs. the level of black in each of these locations were compared to select the position of the heads of the notes. Fig. 8 shows the result obtained by this step.*

### E. Applications

There are many applications which could use this vision library for western music notations. A proof of concept was developed to demonstrate this, which is indicated using green color in the Fig. 2.

Here the identified Western Music notations will be converted into the Eastern Music notation scheme.

Some other applications would be to export the notation in a standard format like musicXML which then can be used in already available software like museScore to play back, make changes and use in compositions.

## IV. Evaluation and Results

### A. Accuracy

The final detection gave diverse results depending on the quality of the input image. When considering digital images with high pixel density (like in Fig. 4) the accuracy of the algorithm was about 95% - 100%.

But when testing with scanned images as shown in Fig. 9 the accuracy went down to 50% ~ 60%.

Another issue encountered was that the low pixel density images would lead to overlapping of features which made the detection of the staff impossible as illustrated in the Fig. 10.

However for images scanned or digital with high pixel density the algorithms were able to detect the notes at the said 95% - 100% accuracy.

### B. Performance

The time complexity of the algorithms used were analyzed to get an understanding of the performance of the library. Let's assume the input image is of n x m pixel size.

#### 1) Edge Detection

The masking operation will require O(*nm*) to iterate through the pixels and mask them.

Calculation of the angles and tracing through them will also require O(*nm*) time.

Therefore the Edge detection task is of time O(*nm*).

#### 2) Staff Detection

After the line detection algorithm is run if number of line segments identified is l, the complexity of identifying the staff from these lines will be O(l). This will always be less than the complexity of the cvHoughLines2 complexity.

#### 3) Note Detection

The note detection is similar to the staff detection, after the OpenCV line detection it will only need constant time to detect the position of the head of the note.

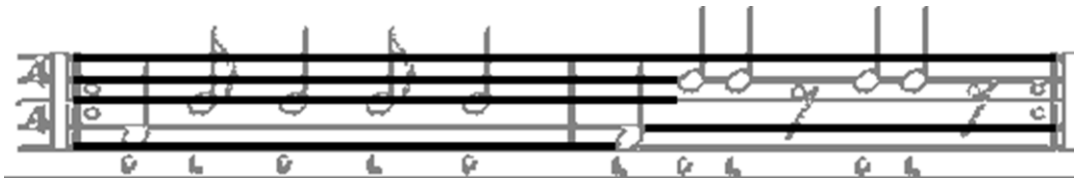

Fig. 7. After Staff Detection.

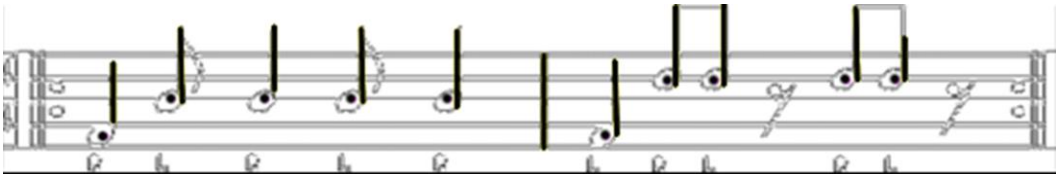

Fig. 8. After Note Detection.

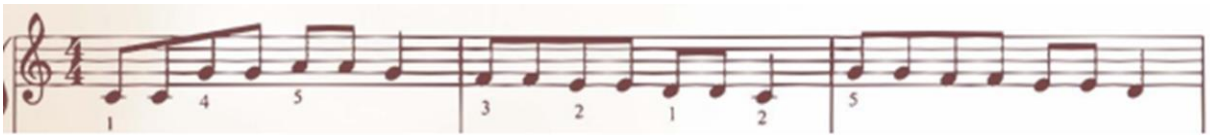

Fig. 9. Low pixel scanned image.

Fig. 10. Erroneous Staff Detection.

## V. Conclusion and Future Work

This library gives basic functionality needed for Western music notation detection. Yet it requires many further improvements to be used in practical applications. Detecting treble/bass clefs, sharp/flat signs and the timing notations are needed in order to enhance usability of this library.

Also to make the library more robust more pre-processing could be done on the images like enhancing the images to improve the accuracy for low quality images. Compensating for skewed images and distorted images is also a pre-processing step which would be added.

Having a complete library for detecting western music notations could lead to many interesting applications and it can be integrated with many existing standards and software.

## References


[1] MusicXML, "MusicXML for Exchanging Digital Sheet Music", 2015. [Online]. Available: http://www.musicxml.com/. [Accessed: 04 Jul 2015].

[2] SMuFL, "SMuFL | Standard Music Font Layout", 2013. [Online]. Available: http://www.smufl.org/. [Accessed: 04 Jul 2015].

[3] musescore.org, "MuseScore | Free music composition and notation software", 2015. [Online]. Available: https://musescore.org/. [Accessed: 04 Jul 2015].

[4] opencv.org, "OpenCV | OpenCV", 2015. [Online]. Available: http://opencv.org/. [Accessed: 04 Jul 2015].

[5] Code.google.com, "javacv - Java interface to OpenCV and more - Google Project Hosting", 2015. [Online]. Available: https://code.google.com/p/javacv/. [Accessed: 04 Jul 2015].

[6] J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, 1986.

[7] R. Duda and P. Hart, "Use of the Hough transformation to detect lines and curves in pictures", *Commun. ACM*, vol. 15, no. 1, pp. 11-15, 1972.

[8] dcs.opencv.org, "Hough Line Transform — OpenCV 2.4.11.0 documentation", 2015. [Online]. Available: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html. [Accessed: 04- Jul- 2015].

[9] Unicode.org, (2015). *Roadmap to the BMP*. [Online] Available: http://unicode.org/roadmaps/bmp/ [Accessed: 10 Sep. 2015].