Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Music Score Reader Based on Image Processing and Generation of Music

Gajanan Jadhav, Rajendra Kulkarni, Saurabh Lende, Amit Mahalankar
Information Technology, University of Pune,
Maharashtra, India

Abstract--With the advances in mobile devices, we develop an optical music recognition system to interpret the printed sheet music captured by an Android phone and present the music through MIDI playback on the device. A music score reader which can read scanned music score and play the music out is the project using Image Processing techniques. By applying various image processing and detection techniques, the system is able to identify the pitch of the notes by spatial position and generate corresponding music. This system can be used by music learners in practicing unfamiliar pieces of music.

Keywords— music score, clef, image processing, android, MIDI etc.

I. INTRODUCTION

Music Score Reader is a system that can read scanned music score and play it out. A Music Score Reader on computer or mobile platform is a useful tool for many people, a soloist could have the computer play an accompaniment for rehearsal; a music editor could make corrections to an old edition using a music notation program. Other than those professional people in music, there is a large population of music lovers who have strong interest in singing/playing but cannot read music score which is the most common way of recording music. Optical music recognition consists of the interpretation of sheet music into playable form. It has been a field of interest since the late 1960's.In the past few years, mobile phones have quickly evolved into mini-computers with high-quality cameras and broadband data connectivity. As a result, we propose to integrate the optical music recognition technology into the mobile environment. Improvements of mobile cameras enable us to capture music notations directly with a mobile device, and a smartphone's media capability allows it to process and play the generated music file. Compared to its expensive non-portable counterparts, a portable mobile music recognition system would serve as a convenient and cost-efficient alternative for the average music enthusiasts.

II. SYSTEM ARCHITECTURE

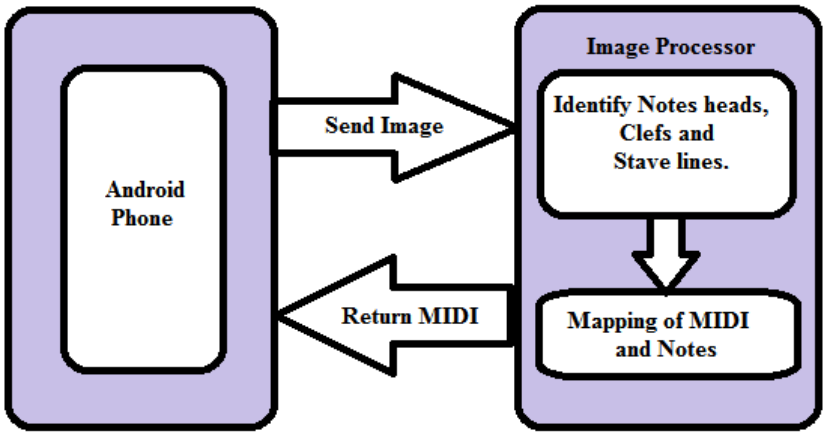

Fig. 1 System Architecture

III. ALGORITHM & FLOW PROCESS

A. Binarization

The input image in our data set is JPEG format. The color information is not important in the following part, so we firstly converted it to a grey level image. Then we reduced the gray level image to a binary image to differentiate the foreground and background. We use the popular method, Otsu's method, to perform the binarization. The algorithm

assumes that the image to be thresholded contains two classes of pixels or bi-modal histogram (e.g. foreground and background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal [5]. With a proper threshold, we obtained a binary image containing all the useful information, such as notes and staff lines, as its foreground. Binarized image is compressed after processing and stored on SD card.

B. Stave Parameter Location and Extraction

As most of the subsequent steps rely heavily on knowing the precise location of the staves, this is one of the most critical phases of the application architecture. If the stave detection algorithm erroneously detects a stave which isn't one, the subsequent modules will process that as being stave and the accuracy of the OMR system will be compromised. Similarly, determining the exact coordinates of the stave lines is important as this will be used in the midi generation module to determine the exact pitch of notes.

The first step in locating the stave(s) in the music score involves approximating the thickness of each stave line and the white space in between each stave line. Since the staves are the most predominant feature in a music score. By traversing the music score column by column, we can apply the RLE (Run Length Encoding) algorithm and record. the number of consecutive black and white pixel runs in two arrays. After applying the RLE algorithm, the arrays contain the number of times a certain amount of consecutive black or white pixels occurred. The peak in the array corresponds to the stave line thickness if the array is populated by runs of black pixels. The size of both arrays was chosen to contain 100 integers for the simple reason that a stave line and the distance between two stave lines is unlikely to exceed 100 pixels. The typical thickness of a stave line ranges anywhere from 3-6 pixels and the distance between two stave lines can range from 15 to 30 pixels. This of course depends on the typesetting of the music score. In order to detect and locate the position of the staves, we take the y-projection of the image and obtain a one-dimensional array whose contents is the summation of all black pixels along the y axis. The resulting array is then traversed and all local maximas are found. Having previously calculated the approximate thickness of a stave line and the distance between two stave lines, we can develop an algorithm that will scan the array [1].

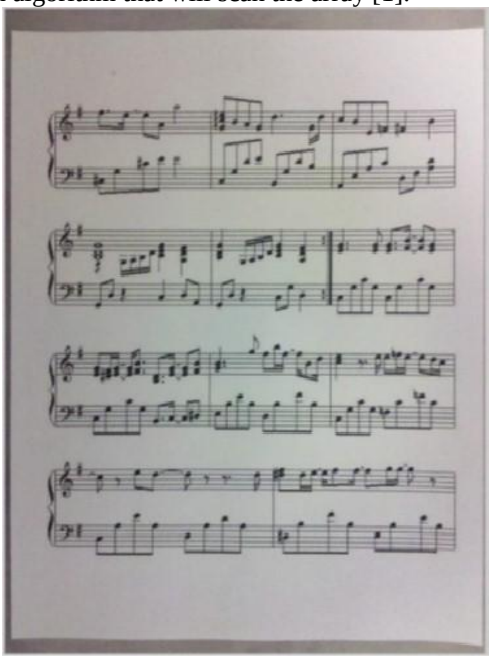

Fig 2. Orignal RGB Image

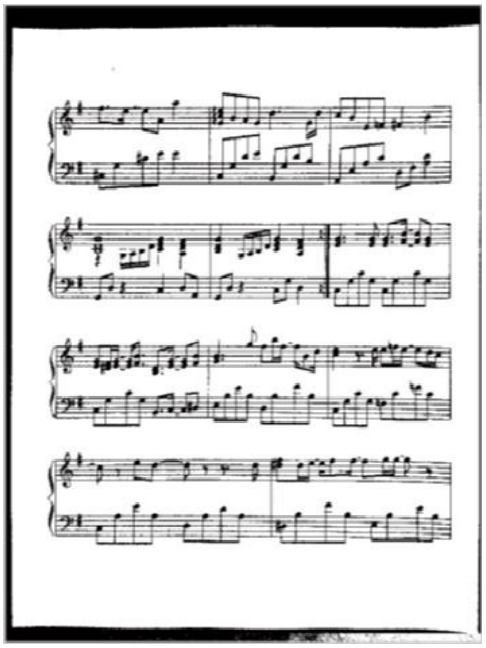

Fig 3.Binary Image with white Background

C. Segmentation

The segmentation module attempts to find groups of symbols, individual symbols, and individual note heads and labels them. Groups of symbols are defined as being groups of notes that are attached to one another with beams. The main purpose of this step is to reduce the search space of the image in order to reduce the processing time required for the subsequent phases. It is unnecessary to process areas of the image which we are certain contain no musical symbols. The method used to locate groups of symbols relies on the x-projection. The x-projection for each stave is calculated and stored in an array for post processing. We choose maximum stave line height here as we want to try to set a minimum threshold that will allow us to find empty staves. The array containing the x-projection is then scanned and as soon as we find values above the minimum threshold, we start a counter. The process then continues until a value below the minimum threshold is found. The counter then defines the width of the new segment that was found and its coordinates are stored in a data structure.

1) Locate Treble and Bass Clefs

Use of input data independence and serialization for exploring opportunities of morphism, and overloading; hence deriving the multi-threaded architecture SRS resulting in effective use of multicore CPU/ concurrent processing/ memory requirements and segmentation.

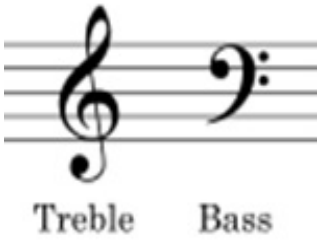

Fig 4. Treble and Bass Clefs

2) Note Head Detection

The purpose of this step is to detect and label any segments containing filled note heads. It does so by analysing the y-projection of the Level 1 segments. In the particular example shown below, we notice that the Level 1 segment contains a total of four note heads. Similarly, different musical notes are detected. We are concerned with finding the pitch and duration for each note. The pitch for each note is relatively simple to find as we already know their x and y coordinates from the note head detection module. We now need to determine the duration of the note and this is done based on which side of the note the stem is located. If the stem is located on the left of the note, we need to look at the glyph directly below the note and if the stem is on the right, we need to look at the segment to the top right of the node. This follows the way that music scores are written in general.

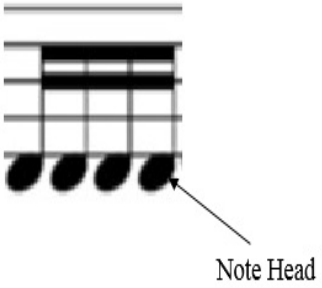

Fig 5. Note head detection

D. Generate MIDI files

With the help of a user defined library had to be compiled using the Android API, we generate the offline MIDI file by mapping the pitches of the detected notes to corresponding MIDI numbers based on a MIDI table. Each note in the MIDI format requires a number identifying its pitch, a start time and end time, as well as its volume.

Octave #	Note Numbers											
	C	C#	D	D#	E	F	F#	G	G#	A	A	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	222	123	124	125	126	127				

Fig 6. MIDI Table

E. Play the Music

Each pitch key is associated with a unique frequency. Fig 6 shows the detailed association relationship. By matching note with its MIDI file music is generated for whole sheet. Different instruments are also provided.

IV. CONCLUSIONS

In this project, we developed a music score reader which can read picture of music score, process it and play the music out. We binarized and resized image; identified the location of staff lines, used Image Processing techniques and MIDI files to play music. For clean standard score, we can achieve an good accuracy. This will automate the music score reading technique and will save time and work load.

REFERENCES

[1] Dingyi Li, Bing Han and Jia Ji, Music Score Reader Based on Morphological Image Processing,Department of Electrical Engineering Stanford University, Stanford, US.-2012-13

[2] Bainbridge, David, and Tim Bell. "The Challenge of Optical Music Recognition". Computers and the Humanities 35 (2001): 95-121.

[3] Steve Dai, Ye Tian, Chun-Wei Lee,"Optical Music Recognition and Playback on Mobile Devices", Department of Electrical Engineering Stanford University, Stanford, CA 94305,USA.2012-12.

[4] S. Inglis. "Music Image Compression". In Proceedings of the IEEE Data Compression Conference.Ed J. Storer and M. Cohn. Snowbird,Utah: IEEE Computer Society Press,1997, pp.208218,26

[5] Nobuyuki Otsu. "A threshold selection method from gray-level histograms". IEEE Trans. Sys.,Man., Cyber. 9 (1): 62-66.

