JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.6, Special Issue.10, Nov 2018 E-ISSN: 2347-2693

Optical Music Recognition using Image Processing and Machine Learning

Prince Mathew[1*], Rahul Vijayakumar[2], Aju Tom Kuriakose[3], Jesmy Sunny[4], Ramani Bai V[5]

[1]School of Mathematics and Computer Science, Indian Institute of Technology, Goa, India
[2]School of Mathematics and Computer Science, Heriot-Watt University, Dubai, UAE
[3]Department of Computer Science and Engineering, Rajiv Gandhi Institute of Technology, Kerala, India
The School of Computer Studies, Lambton College, Ontario, Canada
[5]Department of Computer Science and Engineering, Vidya Academy of Science and Technology, Kerala, India

[*]*Corresponding Author: princemathew07@gmail.com, Tel.: +91-94477-64066*

Available online at: www.ijcseonline.org

Abstract— The ability to understand music score is a basic requirement for learning music. This paper proposes a mathematical method to find the pitch of a musical note from digital image of sheet music and a classification-based method for detecting the duration of a music note. In a sheet music, the horizontal direction can be associated with the notes starting time, whilst the vertical direction can be associated with pitch. The symbols used for a note represents its duration. Music scores sometimes need to be transposed or slightly modified, having the score in a digital format greatly reduces the time and effort required to do these. In this paper, we make use of techniques such as Run Length Encoding (RLE), Horizontal projection and Vertical Projection (X & Y projections) for Segmentation and attribute extraction. For note recognition, a classifier based system is used which returns the duration of the given input symbol. The pitch, duration and position of notes are finally given as input to a midi generation module, which generates a MIDI file corresponding to the given input music notation. There are several other applications to Optical Music Recognition (OMR) systems. Converting music scores in Braille code for the blind is yet another application of an OMR system.

Keywords—Optical Music Recognition, Image Processing, RLE, Classification, Machine Learning

I. Introduction

This paper proposes a mathematical method to find the pitch of a musical note from digital image of sheet music. In contrast to Optical Character Recognition (OCR) systems, OMR systems are subject to greater complexities as music notation is represented in a two-dimensional space.

The first step in this music transcription application is to remove the noises in the input image and then to detect the thickness of stave lines and the spacing between stave lines. For this we perform an initial noise reduction and black and white conversion of the input image and then perform Run Length Encoding (RLE) algorithm which produces a histogram considering consecutive black and white pixels. The next phase involves detection of stave lines by taking y-projection of the input image. The initial segmentation phase[2] helps in obtaining horizontally segmented staves. The next step in the segmentation process involves division of staves into blocks.

Note head detection is then performed with the help of RLE algorithm and y-projection. Then a segmentation phase is done to vertically segment the image to produce individual symbols or glyphs. Also, for glyph recognition, the image is normalized to a 5x5 feature vector. Pitch of each of the notes present are calculated mathematically using attributes such as position of notehead, stave line thickness and distance between stave lines. The duration corresponding to a symbol is obtained after the image classification process. The pitch, duration and position are then given to a MIDI generation module to produce the final output.

Although there are several commercial applications for optical music transcription, they don't always perform accurately and those giving fairly good output is of very high cost. This was the motivation for this paper.
We investigated the research that has been conducted in the past to gain a better understanding in this field [7]. This work has its' roots from O3MR, a tool dedicated for printed music sheet recognition by the Department of System and Informatics of the University of Florence, and OpenOMR[1] projects. The proposed method will offer a low-cost method with good accuracy for detection of music from digital image of sheet music.
Rest of the paper is organized as follows, Section I contains the introduction of optical music transcription, Section II explains the methodology, Section III describes results and discussion, Section VI concludes research work with future directions.

II. METHODOLOGY

There are mainly three steps for recognizing a musical note from a picture of sheet. Initially we will have to perform stave detection to find the stave lines, calculate the average thickness of stave lines and distance between them. The second step is segmentation and note head detection. Then we will have to do the actual pitch detection. Each of these processes involved are explained below in detail. A sample input image is shown in Fig. 1.

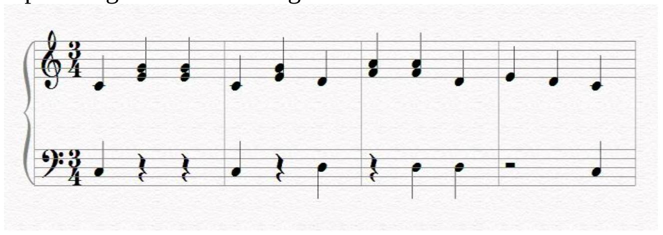

Figure 1. A sample input image

A. Pre-Processing

1) Black & White conversion: Before processing the image for detection of the pitch, we convert the input image into black & white format. For this process Red, Green and Blue values of each pixel are added. If it is 0, it means that the pixel is black. So a threshold value '*thres*' is set. Now if the sum of RGB values of a pixel is greater than *thres*, then that pixel is converted into white. Else, it is converted into black. The result of pre-processing on the sample input image in Fig.1 is shown in Fig. 2.

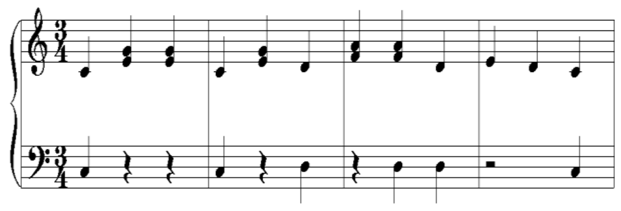

Figure 2. Image after noise reduction and black and white conversion

Skew correction, if required, must also be done during this pre-processing stage. Techniques involving Fast Fourier Transforms (FFT) and windowed FFT are considered as effective in providing skew correction and hence in removing rotational errors.

B. Techniques Employed

1) X-Projection: The X-Projection is the projection of an image onto the X-axis. The result is a vector whose ith component is the sum of all black pixels in the ith column of the image. This is done by adding all black pixels of the column to an array. X projection was used in segmentation and note head detection.

2) Y-Projection: The Y-Projection is the projection of an image onto the Y-axis. The result is a vector whose ith component is the sum of all black pixels in the ith row of the image. This is done by adding all black pixels of the row to an array. Y-Projection is used in stave line detection. The Y-Projection of the sample input image in Fig. 1, after pre-processing, is given in Fig. 3.

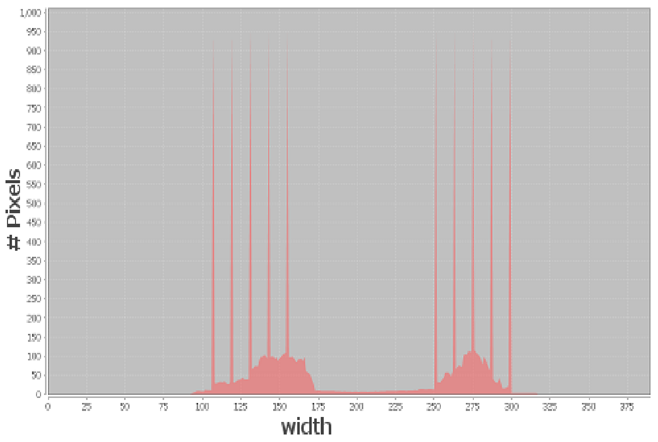

Figure 3. Y-Projection of a sample input image

3) Run Length Encoding (RLE): The RLE algorithm works by counting the number of consecutive pixels with a given colour when performing a scan through the image in a given direction starting from a given position. RLE is typically used in lossless image compression algorithms.

C. Stave Detection

Initially, we find the average thickness of each stave line and the average distance between them. For this we find the number of consecutive black and white pixels vertically which occurs the most number of times using RLE algorithm. The number of consecutive white pixels which occurs the most number of times, gives the distance between stave lines (*dstave*) and that of black pixels gives the thickness of stave lines (*tstave*).

Now for detecting each stave we will have to find a stave threshold, which is the maximum value of Y-Projection of the image. Now, if a line has more than 55% to 88% of stave threshold, depending on the thickness of black pixels, then we expect a stave line to be present there and group each set of five stave lines together into a stave. The initial image is now segmented into separate staves and each stave is processed separately.

Here each stave will have 75% of the stave width above and below the stave or till the top or bottom of the input image (for first and last staves respectively), whichever is the smallest, in its block. This is the level 0 segmentation. The level 0 segmentation on an input image is depicted in Fig. 4.

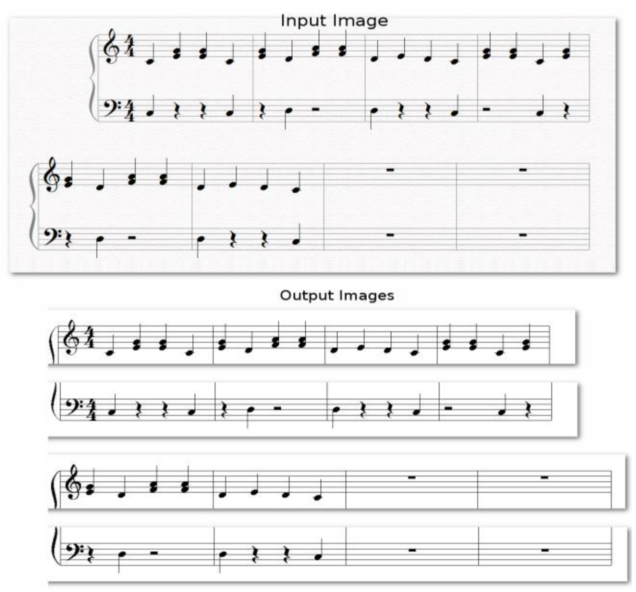

Figure 4. A sample input image cut into several blocks with respect to staves

D. Block division

In the next step we take the X-Projection of each stave separately and check if there is something present other than just stave lines using a threshold value. This is the first level of segmentation. Here we set the threshold as $5*tstave+slack$. Here, *slack* accounts for the minor noises in the input.

After this we divide the stave into blocks, where each blocks X-Projection exceeds the threshold value. The block division is shown in Fig. 5.

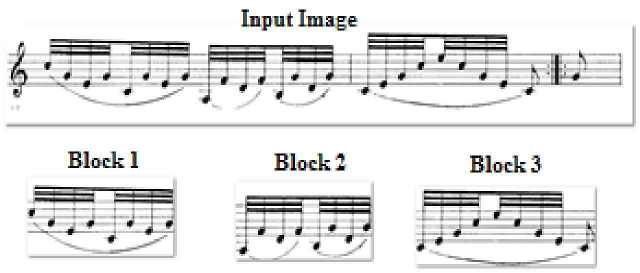

Figure 5. Dividing Staves into Blocks

E. Note segmentation

In the next step we calculate the x-coordinate of each note head, both empty and filled note heads, in each block and finds the coordinates to segment each block into independent modules. For this, we take a copy of each block. Here for detecting empty noteheads, we check the image of each block length wise using the RLE (Run Length Encoding) technique to find continuous white pixels, whose number is less than (*dstave-slack*) and replaces those with black pixels.

If there are any empty noteheads, then all of them will be filled when this step is completed. Now we just need to check for filled noteheads to find the y-coordinate of each notehead.

For this, we check the modified image for continues black pixels vertically, whose number lies between $(dstave-slack)-3*tstave$ & $(dstave-slack)*4+3*tstave$. Everything which doesn't lie between these values are set to white. This process is then repeated horizontally, where we check for continuous black pixels whose number lies between $(dstave-slack)-3*tstave$ & $(dstave-slack)*2$. Everything which doesn't lie between these values are also set to white.

Now the image contains only the noteheads as depicted in Fig. 6. We then find the positions of each of these noteheads. This is later used to divide each block into individual segments containing a single note or a chord based on the positions obtained.

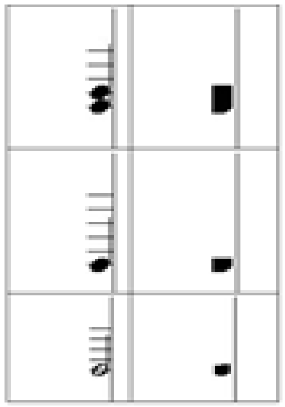

Figure 6. Filling noteheads

F. Pitch Detection

The next step is the actual pitch detection process. Here, we initially find the number of noteheads in each segment and check whether it is a single note or more than one note (chords) and then save the midpoint of each notehead into an array. Let '*beg*' indicate the start position (first staveline's position) of the stave, '*thic*' indicate the stave line thickness and '*dist*' indicate the sum of staveline thickness and distance between stavelines. Here S1, S2, S3, ... are midpoint of stavelines and D1, D2, D3, ... are mid points between two stavelines as shown in Fig. 7.

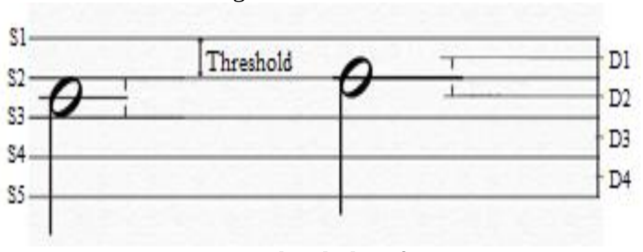

Figure 7. Notehead identification

The S_i and D*i* values are calculated according to Table 1 and are used to calculate the position of the noteheads with respect to the stave and hence is used to determine the pitch of a note.

Table 1. Detection of Noteheads On and Off Stavelines

Note on staveline	**Position Calculated**	**Note between stavelines**	**Position Calculated**
S3’	S2’-*dist*	D3’	(S3’+S2’)/2
S2’	S1’-*dist*	D2’	(S2’+S1’)/2
S1’	S0-*dist*	D1’	(S1’+S0)/2
S0	S1-*dist*	D0	(S0+S1)/2
S1	*beg*+*thick*/2	D1	(S1+S2)/2
S2	S1+*dist*	D2	(S2+S3)/2
S3	S2+*dist*	D3	(S3+S4)/2
S4	S3+*dist*	D4	(S4+S5)/2
S5	S4+*dist*	D5	(S5+S6)/2
S6	S5+*dist*	D6	(S6+S7)/2
S7	S6+*dist*	D7	(S7+S8)/2
S8	S7+*dist*	D8	S8+*dist*/2

While selecting a scale, by specifying the number of flats or sharps, the midi values[5] of the notes given correspondingly in the Table 2 will be increased or decreased by *one* depending on whether the note becomes sharp or flat. While doing transpose (+1) the pitch of each note is increased by a semitone. That is, midi value of every note is increased by 1.

Table 2. Change in Pitch on Scale Selection

No. of Sharp	**Notes that becomes Sharp**	**No. of Flats**	**Notes that becomes Flat**
1	F	1	B
2	F C	2	B E
3	F C G	3	B E A
4	F C G D	4	B E A D
5	F C G D A	5	B E A D G
6	F C G D A E	6	B E A D G C
7	F C G D A E B	7	B E A D G C F

G. Duration Estimation

Estimation of duration is done by identifying the symbol used to represent the music note. Each glyph obtained after segmentation will be normalized to a 5x5 feature vector as shown in Fig. 8 and the corresponding RGB values will be taken.

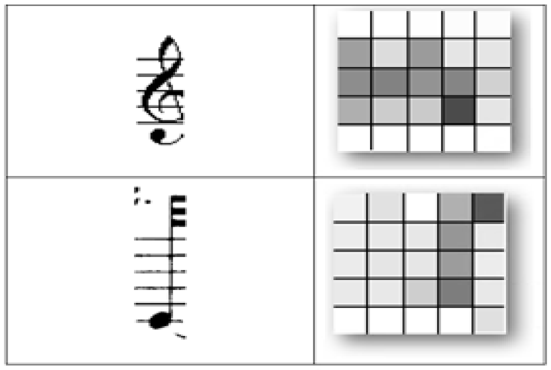

Figure 8. Images before and after normalization

A training dataset is created from a set of normalized images with their corresponding RGB values and class labels. A model is then built using the Random Forest Classifier available with Weka[3], which shows a prediction accuracy of about 83.87, when evaluated using 10-fold cross validation. This model is saved and used for classification of music notation symbols.

The glyphs after segmentation is passed to the compare image module, where the image is normalized to 5*5 matrix and corresponding RGB values is taken. These 25 values are then stored to a temporary ARFF[9] file. The header of the ARFF file is given below.

```
@RELATION notation

@ATTRIBUTE arr1     REAL
@ATTRIBUTE arr2     REAL
@ATTRIBUTE arr3     REAL
@ATTRIBUTE arr4     REAL
@ATTRIBUTE arr5     REAL
@ATTRIBUTE arr6     REAL
@ATTRIBUTE arr7     REAL
@ATTRIBUTE arr8     REAL
@ATTRIBUTE arr9     REAL
@ATTRIBUTE arr10 REAL
@ATTRIBUTE arr11 REAL
@ATTRIBUTE arr12 REAL
@ATTRIBUTE arr13 REAL
@ATTRIBUTE arr14 REAL
@ATTRIBUTE arr15 REAL
@ATTRIBUTE arr16 REAL
@ATTRIBUTE arr17 REAL
@ATTRIBUTE arr18 REAL
@ATTRIBUTE arr19 REAL
@ATTRIBUTE arr20 REAL
@ATTRIBUTE arr21 REAL
@ATTRIBUTE arr22 REAL
@ATTRIBUTE arr23 REAL
@ATTRIBUTE arr24 REAL
@ATTRIBUTE arr25 REAL
@ATTRIBUTE class     {Notation-treble,Notation-
bass,Notation-minim,Notation-crotchet,Notation-
quaver,Notation-semibreve,Notation-breve,Notation-
crotchetrest,Notation-scale,Notation-breverest}
```

The classification is done immediately on this data to recognize the duration of a note[8] and the result is returned to the comparison module. This process is repeated for all the glyphs sequentially.

The duration, pitch and the starting time of all the notes are then passed to a midi generation module which generates a midi file corresponding to the input sheet music.

III. RESULTS AND DISCUSSION

An effective method for mathematically calculating the pitch of musical note using attributes such as stave line thickness and distance between stave lines is found. The various levels of segmentation on a sample input is depicted in Fig. 9.

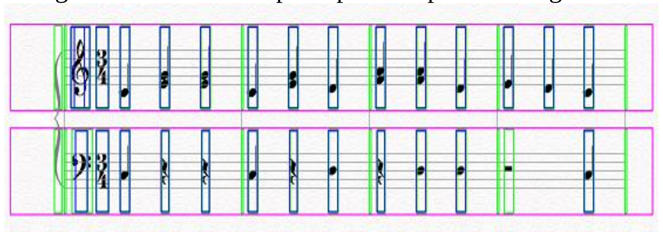

Figure 9. Representation of various levels of segmentation of an input image

However, The segmented music note is finally given as input to a classification module, which detects the corresponding symbol, thereby detecting the duration of that note. Accuracy of classification when performed using various classification algorithms were evaluated and is shown in Table 3. A model created using the *Random Forest* algorithm which shows the greatest accuracy was finally selected for the classification of the input glyph. The pitch, duration and position are then given to the midi generation module[6] to produce the final output.

Table 3. Accuracy of sample dataset on various Classifiers

Classifier Used	**Accuracy Obtained**
KStar	32.25
ZeroR	35.48
ZeroR	35.48
DecisionStump	41.93
AdaBoostM1	41.93
NaiveBayes	50.32
OneR	54.83
Decision Table	60.00
REPTree	67.74
JRip	69.67
BayesNet	71.61
SimpleLogistic	72.90
MultilayerPerceptron	72.90
PART	73.54
RandomTree	75.48
Bagging	75.48
IBk	76.12
RandomForest	83.87

Classification accuracy can be further improved by selection of important attributes, which contributes more towards the classification process, which can be identified by using methods such as Shannon's Entropy[4] calculation. This can help in reducing the time taken for classification and model building, dimensionality reduction and hence reduces the effect of overfitting in the resultant model. Use of better classification algorithms and training the model using various features extracted from the input images can also further improve the performance of the model.

IV. CONCLUSION AND FUTURE SCOPE

This paper proposes a segmentation-based approach for pitch and duration detection of music notes from printed music score. Various processes such a stave thickness calculation, stave detection, block division, empty note head filling, note segmentation and image classification were performed for extracting pitch and duration information from the music score.

The proposed system can be further enhanced by incorporating one or more of the following features into the existing system. This includes detection of accidental sharp and flats and scale detection, detection of time signature and bars and their utilization to obtain quantised and synchronized outputs, detection of dotted notes, dynamics and accents, multi-threading to improve the performance of the detection process and use of better classification algorithms and image processing methods for note detection. These changes will help us to create a low-cost music recognition engine, capable of detecting music with good accuracy.

ACKNOWLEDGMENT

This work was supported by Saintgits College of Engineering, Kottayam, Kerala. We thank our colleagues and faculty members from Saintgits who provided insight and expertise that greatly assisted in the creation of this paper.
We are grateful to the Department of Computer Science and Engineering, Saintgits College of Engineering and all the faculty members for providing us with the facilities, valuable suggestions, helpful supervision and encouragement which led to the successful completion of this work.

REFERENCES

[1] A. F. Desaedeleer, "*Reading sheet music – openomr*", Imperial College London,(University of London), http://sourceforge.net/projects/openomr/.

[2] R. J. Baugh Earl Gose, "*Pattern Recognition and Image Analysis*", In Pattern Recognition and Image Analysis, volume **1**, **2011**.

[3] M. Hall, E. Frank , G. Holmes , B. Pfahringer , P. Reutemann , H. Ian : "*The WEKA Data Mining Software: An Update*". SIGKDD Explorations 11, **2009**.

[4] C.E. Shannon: "*A Mathematical Theory of Communication*". The Bell System Technical Journal 27, **379–423**, **623–656**, July, October **1948**.

[5] Online. "*Note names, MIDI numbers and frequencies*". http://www.phys.unsw.edu.au/jw/notes.html, June **2005**.

[6] Online. "*Midiutil - A Python interface for writing multi-track MIDI Files*". https://code.google.com/p/midiutil/, December **2013**.

[7] Online. "*PythonInMusic*". https://wiki.python.org/moin/PythonInMusic,December **2013**.

[8] Online. "*Note value*". http://en.wikipedia.org/wiki/Note_value, March **2014**.

[9] Online. "*Attribute-Relation File Format (ARFF)"* https://www.cs.waikato.ac.nz/ml/weka/arff.html, November **2008.**

Authors Profile

Mr. Prince Mathew pursed Bachelor of Technology from Mahatma Gandhi University, Kottayam, Kerala in 2014 and Master of Technology from University of Kerala, Thiruvananthapuram, Kerala in year 2016. He is currently working as Junior Research Fellow in Indian Institute of Technology, Goa since 2018. His main research work focuses on Machine Learning, Formal Methods and Software Verification. He has 1 year of industrial experience and 1 year of research experience.

Mr. Rahul Vijayakumar pursed Bachelor of Technology from Mahatma Gandhi University, Kottayam, Kerala in 2014 and Msc in Information Technology- Software Systems from Heriot-Watt University, Scotland in year 2017. He is working as Software Engineer in Cycloides Technologies since 2018. His research work focuses on VR/ AR / Hybrid mobile app development technologies.

Mr. Aju Tom Kuriakose pursed Bachelor of Technology from Mahatma Gandhi University, Kottayam, Kerala in 2014 and Master of Technology from APJ Abdul Kalam Technological University, Kerala in year 2017. He is currently working as Systems Engineer in Infosys Ltd. He has a publication in IJARCCE and it is also available online. His main research work focuses on Data Mining, Information Retrieval and Computational Linguistics.

Ms. Jesmy Sunny pursued Bachelor of Technology from Mahatma Gandhi University, Kottayam, Kerala in 2014 and Post Graduated in Computer Software and Database Development from Lambton College, Sarnia, Canada in 2018. She is currently working as business analyst in Bell, Canada since 2018. She has 2 years of industrial experience.

Dr. Ramani Bai V pursued PhD from Anna University, Chennai in 2012. She is currently working as Professor and Head of Department of Computer Science Engineering, Vidya Academy of Science and Technology, Thrissur, Kerala. She is also leading the Vidya Centre for Artificial Intelligence and Research in the same institution. Her research areas include Machine Learning, Data Science and Analytics. She has 2 decades of academic teaching experience.