Optical Music Recognition

Live camera recognition of handwritten musical notes

Department of Electrical Engineering, Ben-Gurion University
Barak Ben-Dayan ; Ilai Giloh

ABSTRACT

Optical Music Recognition (*OMR*), or the process of automatically identifying and digitizing musical notation sheets, is a well-known and thoroughly studied problem in the field of image processing. However, to this date, a solution that provides satisfactory results is yet to be found, especially when dealing with handwritten notation.

In this project, we have built a simple OMR system, which recognizes musical notes from a live video camera, as the composer writes them on a blank sheet. The system then plays the recognized music, as it would have heard on a piano.

The offered system enables music composers to get an immediate audio feedback on-the-fly, without being dependent on dedicated hardware. The composer would be able to work on his musical creation at any environment, needing nothing more than a mobile device with a video camera.

Keywords- OMR; Notes; Handwritten; Sheet music; Beam; Staff; Machine learning; Computer vision; Image processing.

1. INTRODUCTION

The field of automatically identifying handwritten symbols introduces great challenges, due to the high variance of how different people may draw/write the same symbol.
Furthermore, the procedure of correctly identifying musical notation enfolds different, perhaps more complex, challenges than text recognition.
In text recognition, all of the needed information resides in identifying each ink-stain as one of 22 symbols "letters", in a horizontal continues set ("word"). In musical notation however, the musical information resides both in recognizing the different symbol, their position, orientation, and mutual relations. No musical information may be satisfactory when estimated without obtaining all of the above.

On top of that, more difficulties arise when dealing with live video stream from a free-to-move camera, in a natural environment. In order to obtain the musical data, one must first establish the sheet's orientation in the 3D space, eliminate lightning effects and pre-process the written information into a digitized (usually binary) form.

In this project, we aim to achieve a robust system that can deal with multiple users with no need for user introduction (i.e. user-specific learning process). Our main focus is on making the detection and recognition process as robust and independent as possible, at the expense of taking some permissive assumptions for the video camera processing.

In this paper, we survey our method of solution to the presented problem initially from a bird's view and then describe each part separately in section 2. We than summarize experiments performed and the results obtained in section 3. We end by making conclusions about our method and suggesting further directions.

2. OUR SOLUTION

Figure 2-1 describes the general building blocks of the recognition process, with chronological order of appearance. Video frames are captured in (1) and processed in (2) to form black and white images, than staves (staff, pl.) are found in the image. Each staff is separated in (3) and undergoes the rest of the process as a complete and independent unit. Staff lines are detected and being removed in (4). Each beam is separated into stem units and all stems on the staff are separated into note candidates in (5). Each note candidate is tested and classified, and its geometrical data are extracted in (6). These data are sent to machine-learning algorithm for classification of note's head type and flag/beam type which determine its duration in (7). Other signs on the staff (not representing notes) are found and classified in (8). The data extracted from the staff arrive to decision algorithm to derive their musical meaning in (9). The musical interpretation is than sent for playback and demonstration in (10).

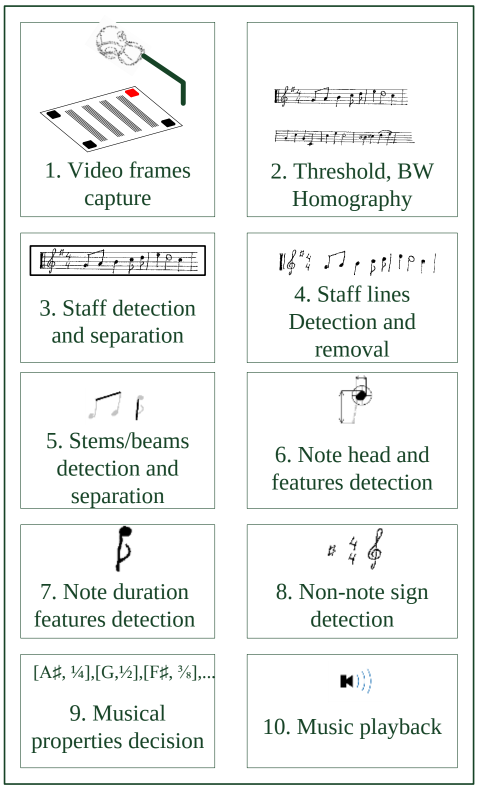
1. Video frames capture
2. Threshold, BW Homography
3. Staff detection and separation
4. Staff lines Detection and removal
5. Stems/beams detection and separation
6. Note head and features detection
7. Note duration features detection
8. Non-note sign detection
[A♯, ¼],[G,½],[F♯, ⅜],...
9. Musical properties decision
10. Music playback

Figure 2-1 - system high-level design

1. Video frames capture

The video module is responsible for capturing a clear, straight image of the notation sheet at every given time, when possible.
The video module relies on dedicated finder patterns (also found on QR codes) located at the corners of the music sheet. The Lower-right pattern is colored red to establish conclusive orientation (see figure 2-2). The video module offers robustness to sudden shadows, light changes, and slight movements by assuming orientation is maintained during a short interruption to the corner recognition.

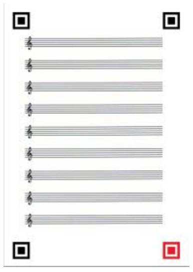

Figure 2-2 - the notes page

2. Image pre-processing

Every frame captured by the video camera is adjusted by adaptive filtering to minimize data loss in the binarization process.
The frame is then converted into a binary, black and white format.

3. Staff detection and separation

The staff has the unique property of being constructed with lots of lines which intersect with each other. This can be interpreted as a closed (with internal area) connected object. There may be staff objects which are not connected to the staff; however the centroid of the staff is well within it. Thus we fill the image's holes in order to create the mass objects and list the connected objects in the image. The staves should obviously be the largest connected objects in the image, and thus may be extracted and split in the middle line between the staves, thus keeping the out-of-staff objects within the staff image.

4. Staff lines detection and removal

In this phase, each staff is searched for the staff lines in order to locate them with a single-pixel precision. This is crucial for their removal to be clean (because everything that's left on the image is than being recognized by other phases). Figure 2-3 describes the process in detail starting from a given staff image in figure 2-3 (a). In figure 2-3 (b), Canny edge transform is calculated on the image. Since a staff has five staff lines, there should be ten edges in a column wherever there is nothing but staff lines (we'll refer to those columns as '10 row' columns). By experimenting with various images with distinct features, we concluded that in every staff that doesn't contain Legato (Slur or Tie - arc between the notes), "Octave up" symbol ( ) and alike, there are many '10 row' columns distributed all over the staff. We only need several clear columns per measure, and we assume that the staff doesn't contain any of the above mentioned signs and so it is fair to assume the abundance of clear columns in a staff. Since a '10 row' column can contain edge points not on the edge of a staff line, a maximum vote algorithm determines which '10 row' columns contain pure staff line edges (we refer to those columns as 'clear columns'). Clear columns are marked with green dots on figure 2-3 (c). In order to trace the staff lines in non-clear columns, a tracing algorithm traces the edge between two non-adjacent clear columns and tries to match the edge points to a straight line between the edge points of the non-clear column range. In case of a difference, the algorithm moves along the straight line and thus maintaining the straight line. Figure 2-3 (d) shows the traced staff line points in red along with the clear column points in green.

5. Stems/beams detection and separation

Figure 2-4 displays the view of the staff after staff lines removal. In order to detect the stems, we sum the pixels on each column. Since stems are not vertical straight lines, and since the note heads also have mass, we dynamically create a high-pass filter (with length relative to the *space- width* – the width (in pixels) of the space between two adjacent staff lines). This causes note stems and other features to show as peaks (local maxima) in the filtered sum. This is represented in the bottom histogram and the yellow solid vertical lines in figure 2-5. Then, contiguous ranges of nonzero columns are found (two ranges closer than 0.3 space-widths are considered to be a single range in order to overcome loss of pixels due to staff line removal) and the notes are separated into *beam-units*. Each range's start and end points are shown in figure 2-5 as green and red broken lines, respectively.

Contiguous regions wider than three times the space-width, which contain more than one stem, are considered to be a beam notes group, such as the one shown in figure 2-6 (a). An enclosing quadrangle passing through the diagonal edges (diagonal because the beam lines can be in relatively steep descending / ascending angles) of the image is found (red quadrangle in figure 2-6 (a)) and the line which passes through its horizontal center is calculated (green line in the same figure), splitting the beam unit into an *upper-part* and a *lower-part*. By calculation of each part's contiguous ranges, the algorithm determines the orientation of the beam's notes (upwards or downwards) and the frame of each note (or any other symbol) in the beam unit. Figure 2-6 (b), (c) and (d) show the cutting of the beam notes as well as their features detection, which is described next.

6. Note head and features detection

The note head features detection process extracts data about the note's head type (full / empty), head position, whether this is or isn't a note, note's orientation (up/down), note's ledger lines (little 'staff lines' beyond the staff's five lines to help locate out-of-staff notes' pitch – such as those for the two left notes of figure 2-5)

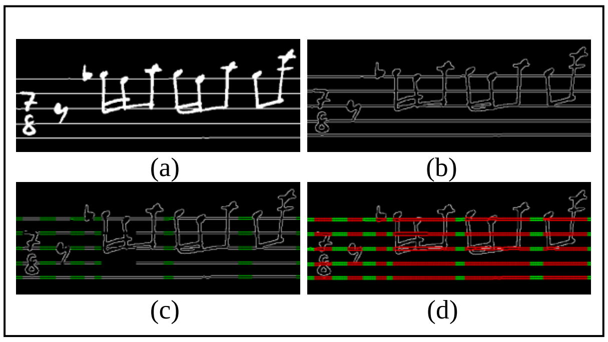

Figure 2-3 - Staff line detection

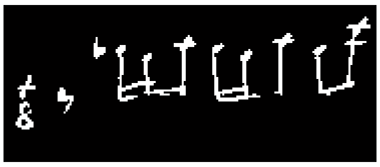

Figure 2-4 - Staff lines removal

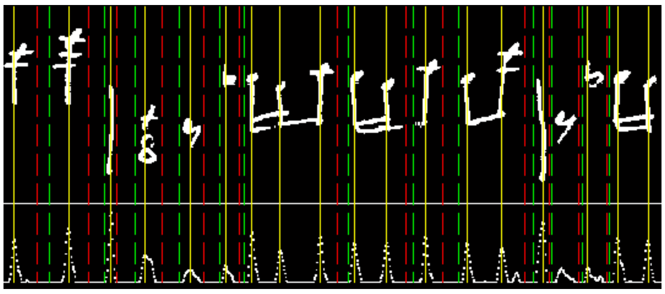

Figure 2-5 - Stems detection

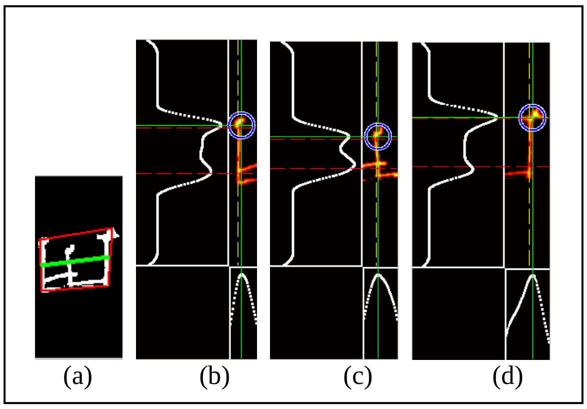

Figure 2-6 - Beam notes separation and detection

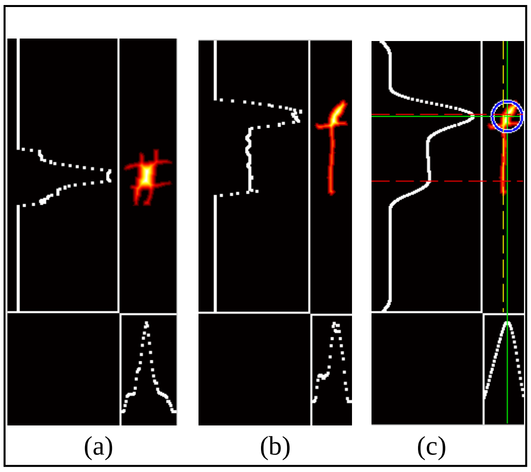

Figure 2-7 - note head detection process

among other data. A distances transform (calculating the distance in pixels to the nearest black spot) is calculated on the note's frame and is summed with a weighting expression as described in expression (1)

$$(1)\quad density = \log_W \left(\sum W^{(dist)} \right)$$

where (assuming that the calculation is for example in y-direction distance density- *YDD*) *density* is the amount of the y distance density on a given row, *W* is the width of the frame and *dist* is the value of the distance transform on a certain point on that row. The summation is performed over the entire row. In this way, we emphasize the highest distance pixels (which correspond to the note's head center) over a row of low valued pixels (which correspond to ledger lines and

interferences). For the X distance density (*XDD*), this calculation is similar, summing over the columns instead of the rows. The distance densities of a sharp sign and a note are shown in figure 2-7 (a) and (b). Since the densities are noisy and the peaks can spread over a region of up to one space-width, the densities are than filtered with a triangular filter of length space-width. The result is the white lines shown in figure 2-7 (c). The peak point of the filtered XDD and the YDD give the coordinates of the head's center (in beam-contained notes, we search only the part (upper/lower) which does not contain the beam and thus avoiding errors in cases where the beam line is brighter than the note head, which occasionally occurs). In case that the shape is too short (shorter than twice the space-width) it is not considered to be a note. This applies also to shapes in which the note's center is too far from the edge of the stem line. Figure 2-7 (c) shows much of the data extracted about a note – the solid horizontal and vertical green lines show the head center's X- and Y-coordinates, respectively. The horizontal broken red lines show the area which mostly contain the note's unique signature, the broken yellow vertical line shows the note's stem mean x coordinate. The blue circle shows the note head's area with diagonal of one space-width length.
Ledger lines are detected using a mass density (summation in the Y-direction on the pixels). The outcome has sharp peaks on the y coordinates of the ledger lines. Currently this feature is not yet complete and therefore is still subject to change.

7. Note duration feature detection

After the notes were identified and separated from other symbols, they are sent to a classification mechanism to establish the note's duration, which resides in the notes "shape", regardless of its position.
Each note is cropped and resized to an 80x20 image, which is "flattened" to form a 1600-fold feature vector. These vectors are then fed to a Machine Learning system that utilizes one-vs.-all Logistic Regression classification, trained in advance to establish notes' durations.

An additional solution based on Neural Network was tested, and proven to achieve higher accuracy at the task of recognizing newly introduced notes - at the expense of being computationally slower.
At this point in time, we have neglected this solution, deciding to favor speed over accuracy due to the live-video nature of the system.

In order to achieve a robust, user independent classification system, there was a need to train the machine learning algorithm with a rich database. The original DB of ~800 notes was obtained from [1]. This might be a good place to mention our gratitude to Dr. Rabelo who has provided us with a part of her DB. We have then built an offline sub-system that processes unlabeled handwritten musical sheets, both from [2] and from our own camera. The system identifies and separates notes and musical symbols, and sends them to a manual classification GUI.

8. Non-note symbols detection

After the notes have been identified and classified, they are removed from the image, leaving an image sparsely allocated with musical symbols.
All remaining shapes on the image (bigger than a certain threshold) are presumed at this stage to be musical symbols. Each one is being recorded for its position, cropped and sent to an additional Machine Learning classifier to be classified as one of nine known musical symbols, or otherwise discarded.
We should mention that musical notation includes more than 60 different symbols (in addition to notes) – however this system only classifies the nine most fundamental and commonly used ones.

9. Musical properties decision

Once all data about the staff lines, notes, and musical symbols were acquired, they are sent to an integration and evaluation module. This module integrates and evaluates information such as mutual position and orientation of all detected objects in the frame, and builds a complete representation of the musical information, which is completely independent from the optical data.

10. Music playback

Currently, music playback feature is implemented using the machine's speaker directly. An array of musical notes (each containing type (note/silence), pitch and duration data) is received and is translated into a sine wave with matching frequency, where A (La) first octave has frequency of 440Hz, and each half-tone corresponds to a frequency multiplication of $\sqrt[12]{2}$, because the pitch-frequency scale is a logarithmic scale. The note's sine wave amplitude decreases over time to simulate a real instrument and its phase is kept continuous in order to avoid abrupt changes which are not pleasant to the ear. The phase continuum is implemented as a steady-state response (the next note) and a transient response (the transition between the notes which die out over time). The whole sine wave is than fed to the speaker for playback. We plan to change this feature to use MIDI sound instead.

3. EXPERIMENTS AND RESULTS

The system as a whole is yet to be experimented and tested, since at the time writing those lines some new features and algorithms are being developed to help mitigate certain issues.

Current problems/issues include whole note recognition (currently unsupported), graphical shapes splitting due to the staff line detection and removal (causing recognition percentage to lower), inaccurate or coarse thresholding algorithm causing a relatively high pixel error rate (for our needs), detection failure of close shapes (closer than 0.3 space-widths), inaccurate detection of beams and stems, inability to detect staff lines when ties/slurs are present on the staff among other issues and bugs.

The experiments we conducted were 'whitebox' tests, on each module separately. In order to work with as many samples and diverse hand-writing styles as possible, we wrote many notes of different types by hand, as well as used Dr. Rebelo's rich DB to test our ability of recognition. Some of the aspects we paid attention to include the width and separation of the staff lines, the shape of the note heads (anywhere from near-perfect circles to very narrow ellipses), stem lines angle, staff lines angle, closeness of objects to one another, flags and beams effect on note recognition etc.

At most times, the system's parts work at good recognition percentages, it detects the notes quite accurately. Following is a case study of an experiment in which the system typically fails to recognize properly. Our effort, along with system final integration, is to mitigate such issues before the final version of the system is presented.

Experiment - detecting a beam line which is at a shallow angle adjacent to a staff line. Figure 3-1 demonstrates a case in which there is a beam unit which intersects the staff lines in a shallow angle (almost parallel to the staff line). In the magnified image, one can see that the line stepper algorithm (the red dots) which attempts to follow the staff line and to estimate it when the slope is too high, incorrectly follows the beam line, and not continuing straight just as the staff line does. This error causes much of the beam line to be removed in the staff line removal phase, which in turn causes the beam detection and note detection phases to classify the right hand note as a single note (one which doesn't belong to any stuff) and since it has no flag on its stem is will be categorized as a quarter-duration note, rather than an eighth note, which it actually is.

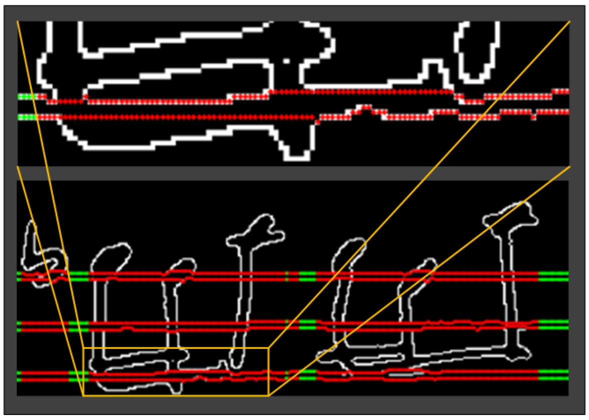

Figure 3-1 - Staff detection failure

4. CONCLUSIONS, FUTURE IDEAS

We conclude now, in the final stages of this project, that indeed quality OMR processes are complex and hide a lot of difficult issues to mitigate in each and every part of it. We had to use our creativity extensively, and still use it in order to find new solutions to old and new problems. We can certainly state that there is much more to this project, and that we have written more algorithms and designs than we could implement at such a short time. The algorithms that were still not implemented are documented for future trials by us or others who wish to get closer to a stable and robust OMR system. Future extensions may include support of ties, slurs, articulation (stacatto, tenuto), text, dynamics (Forte, Piano, Pianissimo etc.) and lots more music.

5. REFERENCES

[1] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, Jaime S. Cardoso, "Optical music recognition: state-of-the-art and open issues," *International Journal of Multimedia Information Retrieval,* 2012.

[2] Alicia Fornés, Anjan Dutta, Albert Gordo, Josep Lladós, "CVC-MUSCIMA: A Ground-truth of Handwritten Music Score Images for Writer Identification and Staff Removal," *International Journal on Document Analysis and Recognition,* 2012.

[3] A. Rebelo, G. Capela, Jaime S. Cardoso, "Optical recognition of music symbols, A comparative study," *International Journal on Document Analysis and Recognition,* 2009.

[4] A. NG, "Standford CS229 - practical Machine Learnning course (by coursera.org)," 2012. [Online]. [Accessed 2012].