Interactive Optical Music Recognition
Ichiro Fujinaga, Bo Alphonce, Bruce Pennycook
Faculty of Music
McGill University

Glendon Diener
Center for Computer Research in Music and Acoustics
Stanford University

Abstract

In this paper, we describe an optical music recognition system currently implemented on the NeXT computer. Although targeted at western staff-based music notation, the system imposes few other limitations as to the type or complexity of the scores which it recognizes. Good results have been obtained with everything from professionally-engraved scores to hand-written manuscripts.

The system comprises three interdependent processes: a recognizer, an editor, and a learner. Operating on the scanned image of a musical score, the recognizer locates, separates, and classifies symbols into musically meaningful categories. This classification is aided by a database of symbols and their features learned in previous sessions.

Output of the recognizer is corrected by a musically trained human operator using a music notation editor. In addition to supplying visual feedback of the recognition, the editor is capable of high-quality, real-time audio output. Editorial corrections made by the operator are passed to the learner. In order to improve the speed and accuracy of subsequent recognition operations, the learner reevaluates recognition data and relearns it into the database of symbols.

1. Introduction

The goal of McGill University's Optical Music Recognition (OMR) project is to provide the music research community with a fast, affordable music recognition system. Since its beginnings in 1988, the project has employed a variety of computer platforms in its research, including IBM AT’s, Mac II’s (AUX), and Sun workstations. More recently, the system has been ported to the NeXT computer, and integrated with Notation—a public-domain music notation editor under development at Stanford University.

The OMR system functions by means of three interdependent processes: a recognizer, an editor, and a learner.

Operating on the scanned image of a musical score, the recognizer locates, separates, and classifies notational elements into musically meaningful categories. This classification is aided by a database of symbols and their features collected from previous sessions. Using the Notation system, the recognizer's output is visually and audibly verified by a musically trained human operator. Editorial corrections are made as necessary, and these corrections are fed back into the database through the learner. The learner improves the speed and accuracy of future recognition operations by rearranging the database and by recalculating a set of weights used in the classification scheme.

2. Recognizer

The recognizer consists of four phases: preprocessing, segmentation, feature extraction, and classification.
Preprocessing  In this phase, staves are identified and staff lines are removed by means of a projection technique. Projections are calculated by summing the number of black pixels that fall along a given projection axis.

To illustrate, a y-projection is the total number of black pixels encountered as the image is scanned along a horizontal line.

![Projection Example](image_url)

By examining the y-projections, both globally and locally, staff lines can be located and then removed.

Wherever possible, textual material such as titles and lyrics are removed using a connected component analysis, which basically groups together pixels that are connected to each other.

Segmentation  The segmentation phase isolates the desired symbols from the rest of the image using a combination of the projection technique and connected component analysis. To determine the location of symbols, the x-projection is scanned for segments which are separated by white space.

![Segmentation Example](image_url)

For each x-projection segment, the y-projection is scanned for y-projection segments. This process is repeated until separation is no longer possible, at which point the resulting rectangle is sent to the feature extractor for further processing.

Feature Extraction  Features are quantifiable shape descriptions of a given symbol. The feature extraction phase calculates these properties, producing a set of measurements called a feature vector.

The OMR system currently calculates the following features: width, height, area, area of the bounding box (width × height), rectangularity (a measure of how well the object fills its bounding box), aspect ratio (width / height—distinguishes slender, square, and circular objects), and normalized central moments (a more detailed numerical description of the shape [Pujolíng et al., 1999]).

Classification  The classification phase uses a k nearest-neighbor (k-NN) classification technique to determine the class of a given symbol on the basis of its feature vector.

In k-NN classification, a measure of the distance between an unclassified symbol and previously classified symbols is calculated as the weighted sum of the difference between their feature vectors. That class represented by the majority of k closest neighbors is then assigned to the unclassified symbol.

Typically, such classes are actual music symbols such as treble clefs, quarter notes, eighth rests, and so on. These are, however, two special predefined symbols called split_x and split_y which, when identified, direct the recognizer to further segment a given symbol either horizontally or vertically. This results in inefficient recognition of its near infinite configurations of chords and beamed notes by splitting them up into note heads, stems, and beams. Recently, a special routine in specifically locate stems has been added to speed up the splitting process.

3. Editor  The editing stage involves manually correcting the output of the OMR system by comparing the original score with the recognized one using Notation. Although not implemented yet, our goal is that when corrections are made, the learner (see Section 4.) will be notified to update the database of classified symbols.

Preprocessing  Once the scanning and the recognition phases are complete, a few special processes are performed to exit the editing task.

The recognizer spins many musical symbols into more fundamental components. An eighth-note, for example, will have been separated into its notehead, stem, and flag components, while a beam complex will be separated into noteheads, stems, and several small beam segments. These symbols are grouped to identify more complete
musical symbols before they are sent to the editor. Slurs and ties also require special processing, for the curvature of these symbols must be specified to the editor. A thinning algorithm reduces their thickness to one pixel, and then a Bezier curve description is calculated.

The resulting data is a list of records consisting of a unique code for each recognized symbol together with its size and the x,y co-ordinates of its origin.

Notation Notation [Dienert 1990, 1992] is a highly interactive system for editing and manipulating music notation. In the system, notational elements are represented as objects in an object-oriented, visual programming language. These objects, called glyphs, display a user-specifiable graphics image. Any image which can be rendered in Postscript can be displayed.

Notation represents two-dimensional musical scores as three-dimensional constructs made of piles of these glyphs. As the following figure shows, compositions are created by piling note glyphs on staves, staff glyphs on systems, system glyphs on pages, and so on. The musical fragment is followed by a special view in which all glyphs are drawn opaque to reveal their pile structure.

Such piles are simply the graphical analog of an underlaying tree structure, and glyphs communicate with each other by sending and responding to messages passed along the branches of this tree. To illustrate, whenever a note glyph receives a play message, it sends a message to the staff glyph it is piled upon, asking that glyph to respond with its pitch value. The staff glyph answers after first determining what kind of clef has been piled upon it. The note glyph then uses this information to schedule its performance with the NeXT computer's on-board digital signal processing facilities.

Notation processes the list of records it receives from the OMR system in three passes. The first pass simply computes a bounding box for the entire page. On the second pass, staff records are extracted, and corresponding system and staff glyphs are created and piled onto the page. Finally, on the third pass, glyphs for the remaining records are created and piled onto the staves. The resulting representation is then ready to be viewed and edited interactively. Furthermore, the representation can be played (though playing is currently limited to one part per staff), furnishing an audible description of the recognition system's results.

4. Learner

The primary goal of the learner is to improve the accuracy of recognition. Enhancing the efficiency of the recognition is a secondary goal.

In our view, the efficiency of recognition is less crucial than its accuracy, for these reasons:

- The recognition task can be performed without human intervention through background processing and, if necessary, on multiple computers.
- Both the number of features used and the number of symbols in the database have practical upper bounds. Suppose, for example, that a subset of 10 features are selected from a total of 20. We estimate that 100,000 symbols would occupy 4 megabytes of computer memory. Using more features would not be practical on today's desktop computers from a storage viewpoint. (Note that this figure refers to the on-line database: a full database containing a much larger number of symbols and a full set of features can be stored on hard disk).
- If we assume these upper bounds, an entry-level workstation such as NeXT can perform the classification tasks in the order of a few minutes, depending on the complexity of the music. If higher recognition speed is required, one may use a more powerful computer.

Improving the accuracy of recognition. Determining which weights will result in the most accurate classification is an extremely compute-intensive task, for the best set can only be obtained by examining all possible combinations. Fortunately, the task can be performed both through background processing and by using idle resources of workstations on a network. The exhaustive search for set weights, however, remains intractable (testing with five different values of weights for all fea-
utors would take several thousand years! Consequently, we are implementing a two-stage method for calculating the set of weights.

In this method, we first determine the accuracy of recognition using all features but allowing only binary weights. (i.e., either we use a feature or we don't). From the result we pick the best N (e.g., 10) features. Then we re-evaluate the accuracy using the N best features and M (e.g., 3) different weights. The values for M and N can be adjusted manually to meet the available time and hardware resources.

In order to reduce the selection time, several types of selection procedures are being investigated, including the branch and bound method [Foroutan & Sklansky 1987; Hayamato et al. 1990] and the application of generic algorithms method [Siedlecki & Sklansky 1989] which claims three to four orders of magnitude reduction in calculation time.

Other areas for improvement under investigation include changing the value of k in the k-NN scheme, and the use of different types of distance measures.

Improving the speed of recognition There are three principle areas where the efficiency of the classification can be improved: reduction of the number of samples, reduction of the number of features, and use of more efficient distance measures.

Two well-known methods for reducing the size of the database are editing and condensing. Editing involves discarding those samples which are obviously misclassified by a subset of all samples [Wilson 1972; Devijver & Kittler 1980]. Condensing involves identifying a subset of samples which would still correctly classify the complete set [Par 1968].

5. Future developments

The system described in this paper is capable of recognizing a variety of musical styles. Before multi-voice-per-staff textures can be played, however, methods need to be found which can reliably assign notes to individual parts.

To date, our testing has focussed primarily on professionally-engraved scores, although preliminary results from recognition of hand-written manuscripts seem promising.

With the basic components of the recognition system in place, we are investigating ways to integrate more tightly the user-interface components of the various recognition stages. We hope to produce a system which can be operated by users with only modest computer skills.

To be practical, the system needs to be fast. Through an emphasis on code optimization together with greater use of distributed processing techniques, we hope to greatly improve the speed of the recognition process.

Finally, as more and more recognized material begins to accumulate, we are confronted with the questions of what database and communication technologies to adopt in order best to share this material with the music research community.

Acknowledgments

This research is partially supported by the Social Sciences and Humanities Research Council of Canada.

References


Dumer, G. R. 1992. A visual programming language for music Notation. (In these proceedings).


