Optical Music Recognition: Progress Report

Ickiro Fujinaga, Bo Alphone, Bruce Pennycook, and Kharim Hogan

The Optical Music Recognition (OMR) project at McGill University is in progress since 1988 with the support of a research grant from the Social Sciences and Humanities Research Council of Canada and an equipment grant from the Apple Canada Foundation. Earlier documentation includes reports in proceedings from the First International Conference on Music Perception and Cognition, Kyoto, Japan 1989 and the ICMC, Columbus, Ohio, 1989. The computing environment has now been enhanced by the establishment of a local network with several NEXT computers.

The basic goals of the project is to design a system for recognition of musical notation which works well with a certain degree of user interaction and is reasonably fast and affordable. In its current state, the OMR system can be separated into three interdependent processes: the recognizer, the editor, and the learner. The function and interaction of these processes will be illustrated and demonstrated, and various problems and difficulties encountered during the design and testing of the system will be discussed.

The recognizer’s task, given a scanned image of music, is to locate and isolate graphic music symbols, and then to apply musical syntax in assembling and classifying them into musically meaningful categories. Classification is aided by a database of symbols and their features collected from previous sessions.

The output of the recognizer is displayed to a musically trained human operator for verification. A music editor is used by the operator to make any corrections. The final result is stored in a symbol database used by the classifier and the learner, and in a score database used for constructing an internal representation of the score suitable for library use or further processing in research applications or the preparation of new editions.

The learner’s task is to improve the performance of the classifier in terms of both speed and accuracy. This is achieved through the rearrangement of the symbol database, as well as through re-evaluation of the weights used in the classification scheme.

1. INTRODUCTION

The Optical Music Recognition (OMR) system functions by means of three interdependent processes: recognizer, editor, and learner (Figure 1). Operating on the scanned image of a musical score, the recognizer program locates, separates, and classifies music symbols interacting with a continuously growing database of symbol features. The output of the recognizer verified by a musically trained human operator and, if necessary, edited by means of a graphical score editor. The final result is stored in the database used by the classifier and the learner. The learner’s task is to improve the speed and accuracy of the recognizer.

![Figure 1. Overall process](image)
2. RECOGNIZER

The recognizer consists of four steps: Preprocessing, Segmentation, Feature Extraction and Classification. (See Figure 2.)

![Diagram of the recognizer process](image)

**Figure 2. Recognizer**

2.1 Preprocessing

**Location of staves** - In order to fully identify the musical symbols — in particular the pitches of the notes — and to simplify further processing by removing the staves, the staff lines must be located. This is done by means of projection technique: a projection is determined by the total number of black pixels that fall on a projection axis; thus, for example, a y-projection is the count of black pixels as the image is scanned horizontally. The height of staves on the page is estimated by auto-correlating the y-projection. With this information the position of the staves is determined; without it, beams and text, which may also show equally spaced peaks on the y-projection, will confuse the recognizer.

**Removal of staff lines** - Although the original design of the OMR system did not remove the staff lines, it became evident that the system started to deal with more complex scores, that the staff lines must be removed. The actual strategy is to remove everything but the staff lines. The result can then be XORed with the original image resulting in an image without staff lines:

1. The width of staff lines is approximated using y-projections.
2. Any vertical lines on the page which are taller than the staff line width is removed. The result may still contain slurs, ties, dynamic markings, and thin beams. (Figure 3 a & 3.b)
3. The image is scanned to remove anything that is situated between the staff lines. (Figure 3 c)
4. The resulting image is XORed to the original image (Figure 3 d)

**Separation of text and musical symbols** - Text, such as lyrics, and performance indications are separated from musical symbols. This will lessen the load on the symbol classifier which is not
designed to recognize text. A separate and specialized character recognition program can be used to recognize the text. Text is distinguished from musical symbols by using the typographical characteristics in a line of text, such as, characters having basically the same height and being placed side by side.

**Separation of systems** - Within a great deal of music, especially orchestra scores, two parts may be set so close horizontally that there is no blank line between the parts. Although the classifier can handle multiple-stave systems, as it needs to in keyboard music, it is most efficient when dealing with one staff. Therefore, whenever possible, scores are separated to have the least possible number of staves per system. This also allows for more efficient parallel processing (not yet implemented), since each staff can be dealt with independently of others.
2.2 Segmentation

The task of the segmenter is to isolate the desired symbols from the rest of the image. To determine the location of symbols, the $x$-projection is scanned for "segments" which are projection profiles separated by blanks. (Figure 4.) 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.

2.3 Feature Extraction

Features are sets of the measurable properties of a given symbol. The feature extraction phase measures these properties, producing a set of measurements called feature vector.

Feature calculation - The following features are currently used in the OMR system: width; height; area ($A_0$); and area of the bounding box ($A_b = \text{width} \times \text{height}$), which are normalized to staff width;
rectangularity: $A_0 / A_b$, which represents how well an object fills its bounding box; aspect ratio: width / height, which can distinguish slender objects from roughly square or circular objects; and normalized central moments.

**Normalized central moments** - Two-dimensional moments are used as shape descriptors in many pattern recognition tasks, including character recognition, aircraft and ship identification, and scene matching. Given a $M \times N$ matrix of a digital image, the moment of order $(p + q)$ is given by

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y) \quad p, q = 0, 1, 2, ...$$

where $f(x, y)$ is the pixel value at a point $(x, y)$.

The moments describe the image in the sense that a sufficiently large set of moments can regenerate the original image. (Teague 1980). The central moments are defined as (Gonzalez & Wintz 1987, 421)

$$u_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q f(x, y) \quad p, q = 0, 1, 2, ...$$

where

$$\bar{x} = m_{10} / m_{00}, \quad \bar{y} = m_{01} / m_{00}$$

are the coordinates of the centre of gravity of the image.

The normalized central moments are:

$$\eta_{pq} = \frac{u_{pq}}{\mu^{2+q}}$$

where

$$\gamma = \frac{(p + q)}{2} + 1, \quad p + q = 2, 3,...$$

These normalized central moments are invariant to the scaling and translation of an image.

### 2.4 Classification

The input to the classification phase is the feature vector and the output is the decision regarding the class to which the object belongs. OMR uses a classification technique called nearest-neighbour (Cover & Hart 1967) for this purpose.

**Nearest-neighbour (NN) classification** - NN is a classification scheme where distances between the feature vector of an unknown symbol and every feature vector of known symbols are calculated.
The symbol with the minimum distance is assigned as the symbol of the image. If \( N \) is the number of features used, then the distance between an unknown symbol and a known symbol is defined as:

\[
d_{ijk} = \sum_{k=1}^{N} w_k (f_{uk} - f_{ik})^2
\]

where \( w_k \) are the weights, \( f_{uk} \) are the features of the unknown symbol, and \( f_{ik} \) are the features of the known symbol. The values of the weights are determined by the learner described below. With this method, there are two ways to improve the accuracy. The first way is to collect more samples of each symbol and the second way is to find the optimal set of weights (so that a feature or a set of features with a strong discriminating power can be emphasized) which gives the most accurate results.

**Recursive separation of multiple symbols and classification** - The symbols assigned are actual music symbols, such as treble clef, quarter note, eighth rest, etc. There are two special predefined symbols, which are called `split_x` and `split_y`. These "symbols," when found, direct the recognizer to further segment the given symbol either horizontally or vertically, then go through the classification process again. This allows chords and beamed notes to be split up into note heads, stems, and beams. This is very important because the classification scheme cannot practically recognize near infinite configurations of chords and beamed notes. These two special symbols also allow two or more attached symbols, such as an accidental and a notehead, to be classified separately. In fact, these two are the only symbols that the recognizer knows at the beginning. All other symbols are learned.

3. **EDITOR**

The editing stage involves proofreading the output of the OMR system by comparing the original and the result via a music editor. Any corrections made by the human editor are noted and the database updated (modified) so that the learning system can improve its performance. It is important that the recognizer and the learner have the correct answers; else it will continue to make the same mistakes. This prevents the OMR from making similar errors in the future.

**Preparation for editing** - Once the scanning and the recognition phases are complete, a few special processes are performed to ease the task of displaying. The output of the recognizer is transcribed to a structure which contains the code assigned to the recognized symbol, the point size in both the \( x \) and \( y \) direction of the symbol and the \( x, y \) co-ordinates of the symbol’s bounding box. This information is then processed such that it can be written to a PostScript file. The end result is the reconstruction in musical notation of the recognizer’s output. Manual editing is then performed on the PostScript output with Glen Diner’s Notation software (Diner 1990). There are several steps involved in the rebuilding process. This phase of OMR assumes two types of symbols in musical notation which have been classified as constant and variable. Constant symbols include quarter-noteheads, quarter notes, half and whole notes, rests, fermata, all such symbols that do not vary in
shape. Variable symbols then are those including slurs, ties, beards, and any other symbol that is not fixed in shape. Interpretation of the constant symbols is straightforward in that the code is matched against those described by Adobe's Sonata Font. These symbols can then be transcribed directly to a PostScript file. The rest of the data consists of partial representations of whole symbols and require more complex processing before they can be written to the final PostScript file. Bar lines and staves must be calculated and redrawn to the specified size based on the data received by the recognizer. Sonata Font definitions of these therefore are not used. Beams, slurs and ties are split into segments by the recognizer and must be reconstructed to match the original scanned image. This process involves consideration of musical syntax, proximity to related symbols and local and global surroundings. The questions are: which pieces belong together, are there any gaps which must be filled in and what needs to be added or subtracted to produce accurate results. As some symbols are extremely close to others, the recognizer splits them into parts which makes the reconstruction more difficult as the data then represents many symbols that could all be related. An eighth note for example, although defined by the Sonata font as one symbol could be split up to include a flag, three stem segments and a quarter-notehead. The staves and ties also need to be specially processed because the curvature of the symbol must be specified to the editor. A thinning algorithm (Zhang and Suen 1984) is first applied to these symbols to reduce them to a thickness of one pixel after which another algorithm (Glassner 1990) is used to create Bezier curve control points.

4. LEARNER

The learning system not only builds up the database, it modifies itself to be more efficient and as well, by changing weights, it modifies the importance placed on properties of the features.

Updating the database - After the corrections are made by the human editor, the database must be updated, so that the same errors will not be repeated in the future. These involve changing the symbol code, addition (splitting one symbol to two or more symbols), and deletion (noise).

Recalculation of weights for best results - As noted, the distance used for classification is defined as the weighted sum of the difference between the features of two symbols. To determine the weights which will result in the most accurate classification, the distances are recalculated using a different set of weights. This is an extremely time-consuming task since the best set of weights can only be obtained by examining all possible combinations (Cover and Van Campenhout 1977). One of the ways to perform this task is to utilize the unused computer resources, the "sleep" cycle (Fujinaga 1989), possibly on multiple machines.
5. Conclusions

Although the entire OMR system is functional, there are many details that need to be studied and fine-tuned. One of the main concerns is in the recalculation of the weights, since this takes an excessive amount of time. There exist some non-exhaustive methods, such as branch and bound algorithm (Hayamoto et al. 1990), which may be useable. Another area of concern is the efficiency of the NN-classification scheme. The size of the database and the calculation time will continue to increase as more music is processed. There are several options that can be used to reduce the space and the time required, however this results in a lower degree of accuracy. Clearly, the more information is available, the more accurate the recognition, therefore, the ideal scenario restrict shortcuts to situations when resources are limited. This is in line with one of the goals of this project, which is to create a software that can adopt and evolve in different environments.

We would like thank Glen Diener for his contributions to our project. This research is partially supported by Social Sciences and Humanities Research Council of Canada and Apple Canada Education Foundation.

References


ICMC 73