University of Canterbury

Department of Computer Science

A Complete Optical Music Recognition System: Looking to the future

David Bainbridge

November 17, 1994

Introduction

Reading music is something a child can learn, and once understood, it becomes such a natural process that it is no longer a conscious effort. If we were to dissect this 'natural process,' we might hypothesise that reading music is decomposed into two parts: the visual recognition of graphical shapes; and the application of our musical knowledge to derive its meaning. A computer paradigm that models this structure would be a *vision system* connected to a *knowledge base*.

Imagine an Optical Music Recognition (OMR) system where the user describes the simple graphical shapes found in music using a customised drawing package, and expresses the musical knowledge necessary to correctly interpret these simple graphical shapes, using a specially designed musical language. Such a system would capture the *essence* of reading music, forming a versatile foundation.

Music is rich in its diversity of notation. Some instruments include specialised markings in a score, for example bowing information for a violinist, and in extreme situations a score is presented using a substantially different notation style, for example guitar tablature. Moreover, music notation is evolving, so even if were possible to completely capture all the primitive shapes used in music today, the set would eventually become incomplete.

Such attributes emphasize the dynamic nature of the problem domain of OMR. The described system meets such demands by allowing the user to specify 'what makes up music'. Consequently the system is itself dynamic.

Let us now consider the proposed system in more detail, by studying the roles of the customised drawing package and specially designed musical language in turn.

A Customised Drawing Package

The drawing package encapsulates the concept of *primitive* shapes. In general, musical features are too complex to recognise in their entirety, it is therefore a common technique in OMR projects to decompose musical features into simpler geometric shapes that can be recognised. These shapes are known as primitives.

Where the drawing package differs from existing work is its flexibility. If a scanned image includes a new primitive, the user can simply request a 'New' drawing; specify the shape; 'Save' the drawing; and add the new filename to the set of primitive given to the OMR system at run-time. If a scanned image includes an existing primitive, but its appearance is different (for example, there is a significant difference in the appearance of note tails between publishers), the user can 'Load' the existing primitive; make suitable alterations to the shape; save the drawing using the 'Save as' option; and update the set of primitives accordingly.

From these two examples it can been seen that a library of primitive shapes would evolve naturally. In fact the *set* of primitives given to the OMR system at run-time would itself be stored in a file, thus a library of these files would reflect different groups of scanned images.

Existing OMR systems can be adapted to cope with changes in primitives, although the changes would have to be programmed into the system and then recompiled. This would require knowledge and source code that typically only the implementor of the system would have.

Many drawing packages permit the arbitrary scaling of the drawn shapes. Since music can be any size, arbitrary scaling would no longer be a convenient attribute for a musical primitive drawing package, but a necessity. In addition to this, when the primitive is drawn, the size of the particular example must be specified.

Freeform curves look to be an ideal choice for the drawing package [FvDFH90]. Not only would they be the natural choice when defining primitives such as a note tail and a bass clef, they are a super-set of Conic Sections. This means that all shapes (line, rectangle, ellipse, etc.) that the artwork package offers as drawing options can be generated using freeform curves. Such a homogeneous treatment of graphical shapes simplifies the implementation.

Perhaps the most attractive property of freeform curves is that they are affine under transformation. That is to say a freeform curve can be scaled, translated, and rotated by first transforming its control points and then generating the curve. Thus a transformed freeform curve is resolution independent.

A Specially Designed Music Language

The purpose of the music language would be to provide a frame-work for expressing the abstract knowledge we, as humans, have of music notation. A more descriptive title for the language would be a Musical Knowledge Expression Language. This distinguishes the requirements of the music language from the numerous music languages already in existence, whose purpose is only to capture the essence of a particular piece of music, be it either for audio play-back or graphical construction.

The use of a language also addresses the issue of the diversity of music notation, alluded to earlier in the report. If the language truly captures the essence of musical knowledge, then any musical notation system (past, present or future) could be described: mediaeval, tablature, figured bass, atonal percussion, etc.

Naturally the specification of musical knowledge using the language must form a dynamic component of the system. When the OMR system is activated not only would the set of primitives to use be specified, but which musical knowledge file to read would also be given. The specifications of dynamic primitives and dynamic music knowledge are, therefore, closely related, however it is beneficial to isolate them in this manner.

To understand a *handwritten* written score a human needs no additional musical knowledge. More time, however, might be spent deciding what shapes are present in the page, since this will not be as clear as a printed work. This is reflected in the proposed system, where only the redefinition of the set of primitives would be required to adapt the system to handwritten music. That is not to say it would be a trivial step; the decomposition of a handwritten page of music into primitives is a significant task.

Another situation where the separation of primitives and knowledge is useful, would be in the pro-

cessing of different classes of music. Scores for the recorder and the piano may use an identical set of primitives, however the legal combinations of these primitives for each instrument are different, since it is only possible for a piano to play a chord. Such differences can be exploited in the OMR system by developing different musical knowledge files, tailored for particular instruments. Consequently with this design, not only would a library of knowledge files develop, but a wider set of music notation could be recognised with little effort by 'pairing up' primitive sets and musical knowledge.

Two key mechanisms the language must provide are: a way to specify the legal configurations of primitives that constitute musical features; and a method for expressing the musical semantics of the composite musical features. For the first mechanism, it is clear what must be accomplished, however as it will be seen later in the report, the meaning of 'musical semantics' depends on the specified end point of an OMR system.

Legal Configurations of Musical Primitives

To informally describe legal configurations of musical primitives, terms such as 'relationships', 'constraints' and '2-dimensional layouts' could be used. A well designed language would incorporate such concepts in a natural manner. The following attributes are considered vital elements of the language.

- Object oriented (hierarchy particularly important).
- Sets and set operators, e.g. '*rests* = { *crotchet_rest*, *semibreve_rest* ... }.'
- Common spatial relationships, e.g. **right**, **left**, **above** and **below**.
- Restrictions, e.g. '*full_note_head* **left** *vertical_line* **within** *(20,40)*.'
- Constraints, e.g. 'rest **intersects** staff.'
- Boolean combination of expressions,
  e.g. '*bass_clef_curl* **intersects** *staff* && *bass_clef_curl* **left** *dot(2)*.'

A derivative of first order predicate calculus promises to capture these element.

Musical Semantics

The aim of OMR is to convert printed music into a versatile machine-readable format: applications range from audio play back to compositional analysis. Merely recognising the graphical features within a page of music is not sufficient. This is why the language must provide a mechanism for converting the graphically recognised features into a computer pliable form; in other words, specifying the musical semantics of the graphical features. Unfortunately there is not a standard music interchange format. ANSI have set up the 'Standard Music Representation Work Group', however there has been nothing forthcoming yet. In the meantime projects invent new formats or arbitrarily become committed to an existing format.

As a consequence of this, the end point for a complete OMR system is defined here to be, an internal data structure that can be traversed and files conforming to particular file format (MIDI, Lime, DARMS, etc.) written out. Clearly the internal data structure is a super-set of musical formats.

If the ANSI work group is finding it difficult to write a specification for a general music file representation, perhaps this is an unrealistic goal. Fortunately this is not the case. The structure of the internal music representation is defined within the musical knowledge file, but so too are the functions that traverse the internal data structure. At every stage, the dynamic design of the internal data structure is shadowed by the dynamic design of the traversal routines. When put together the two dynamic components 'cancel' each other out, producing a file conforming to a static specification. Working with internal data structures rather than files also eases the task, since the former method is free from the restrictive 1-dimensional regime inherent in the latter.

The following attributes are considered vital elements of the language for specifying musical semantics:

- Structural/procedural decomposition (objected oriented style).
- Abstract Data Type support.
- Functional qualities, such as 'map' and 'apply'.

Is Such a System Possible?

Work carried out has already proven that many of the key ideas to the proposed system are realistic [Bai91], [Bai94] and [Bai95]. Compelling arguments can be given, using only existing computer science knowledge and techniques, that the remaining concepts are attenable.

Dynamic primitives are fully realised in the current system. Geometric shapes specified in the drawing package can be directly imported into the primitive identification program, and the tolerances for matching can be specified dynamically. Currently a text editor is used to alter tolerances. Ideally this functionality should be incorporated into the drawing package, however this is a purely aesthetic issue.

In the implemented system, the legal configurations of musical primitives are defined dynamically using a specially designed language. Though significant musical knowledge can be expressed using the language, it is too simple, verbose at times, and ultimately limited. Qualities for an improved language have been presented earlier in this report. These anticipated requirements do not exceed the mechanisms offer by existing compiler techniques.

The existing matching algorithm is naive — a consequence of the simple language — and could not easily be extended to match the powerful descriptions of legal configurations expressed by the new language. Semantic Networks and Frame-Based Systems would permit sophisticated matching techniques that would equal the power of the proposed language. There is a substantial body of literature involving these two AI techniques. Work in computer vision is particularly relevant.

The idea of building an internal data structure representing the musical meaning of the graphically recognised shapes is implemented in the current system. Using this approach it was a simple task to

generate output files in the MIDI, Csound and Lime formats. The construction and traversal of the data structure, however, are currently 'hard-wired' into the system.

Moving the construction and traversal of the data structure into the musical language substantially increases the complexity of operational power the language must have. The language currently used is significantly simpler than this, hence it was inconsequential to develop an interpreter for it. The proposed language is a larger undertaking but has much in common with standard programming languages. It is therefore imagined that the new language would be 'interpreted' by translating it into an existing language. Whether the chosen target language is itself interpreted or compiled is of little consequence. In the latter case, the resultant file could be compiled and and dynamically linked to the OMR system; of course such detail would be hidden from the user.

A property of music that promises to simplify the building of the internal data structure is *time-threads*. Since printed music describes temporal information it is fundamentally 1-dimensional, and even though printed music is a complex 2-dimensional layout of musical features with intricate relationships, it is guaranteed that there is a concurrent 1-dimensional decomposition of primitives that represents the piece of music, otherwise the page of music is ambiguous. In practical terms this corresponds to an x-y ordering of fundamental time primitives (note heads and rests) that form a lattice data structure.

A 'paper and pencil' exercise using denotational semantics would aid the design of the language. It is intended that such work would be carried out before any irrecoverable decisions about the language are made.

Conclusion

Historically, different regions of the world developed music notation in isolation, however, Western Music Notation has emerged dominant — this is reflected in its alternative name, Common Music Notation (CMN). It forms the principal body of printed music in the world today and consequently is the primary interest in OMR work. The described system has been discussed primarily with CMN in mind, however it should be clear from the discussion that the design would not be restricted to CMN.

The system does, however, have its restrictions. Despite the primitive identification 'front end' of the system being applicable to more or less any Document Image Analysis problem [BBY92], the anticipated use of time-threads restricts the class of problem that can be solved with the system to 2-dimensional graphical notations representing temporal information. Unfortunately such a restriction excludes documents such as maps and technical drawings, however it would include dance notation. The restriction, therefore, is not seen as severe. There is still a large body of documents that would benefit from the existence of the system described in this report.

References

[Bai91] David Bainbridge. *Preliminary experiments in musical score recognition.* BEng thesis, Department of Computer Science, University of Edinburgh, The Kings Buildings, Mayfield Road, Edinburgh, UK, June 1991. Honours project.

[Bai94] David Bainbridge. *Optical Music Recognition: Progress Report 1.* Technical report, Department of Computer Science, University of Canterbury, NZ, March 1994.

[Bai95] David Bainbridge. *Optical Music Recognition: Progress Report 2.* Technical report, Department of Computer Science, University of Canterbury, NZ, January 1995.

[BBY92] H. S. Baird, H. Bunke, and K. Yamamoto, editors. *Structured Document Image Analysis.* Springer-Verlag, 1992.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley Publishing Company, second edition edition, 1990.

[Min86] Marvin Minsky. *The Society of the Mind.* Picador, 1986. (*General reading*).