Optical Recognition

A Survey of Current Work

An Interactive System

Recognition Problems

The Issue of Practicality

Optical Recognition of Musical Notation: A Survey of Current Work

Optical recognition of music—the machine acquisition of intelligent information through the correct interpretation of an electronically scanned image—is widely viewed as an important area of technical research. It is also an immensely difficult one. This explains why it was not until this year that the first commercial program appeared. Although research on OMR can be traced back to the work of David Prerau at M.I.T. more than 20 years ago, a large number of research efforts have been launched and nurtured over the past five years. Much of the literature they have generated is found in publications on image processing. The groundswell of recent interest is affirmed by the establishment in 1993 of two electronic discussions of optical recognition.[1] Some practitioners claim that the craft of optical recognition stands approximately where software for music notation stood ten years ago.

Vis-à-vis the additive process of composing musical notation, the general principle employed in most systems for OMR is one of subtraction. Removal of selected graphical elements up to the point where residual objects can be identified is the intermediate goal. Objects that can be isolated are much more likely to be identified correctly than those that cannot. More so than in music printing, the steps in the recognition process are unpredictable in number and nature.

First CCARH Survey

Numerous projects in this field have been reported in *Computing in Musicology* since 1987 [a list of previous reports is given on pp. 117-8]. Believing that it was time to introduce a systematic survey of the numerous and diverse efforts that are known to exist, we distributed a survey concerned with general aims and accomplishments to 36 OMR developers. One measure of the difficulty of the undertaking is that only six groups re-

[1] News about research related to optical recognition in general is circulated by Karl Tombre, INRIA Lorraine / CRIN-CNRS, Post: Batiment LORIA, BP 239, 54506 Vandoeuvre CEDEX, France, or 615 rue du jardin botanique, BP 101, 54602 Villers CEDEX, France; tel.: +33 83/59.20.71; fax: +33 83/41.30.79; e-mail: *Karl.Tombre@loria.fr*. Martin Roth [*cf.* p. 145] maintains an electronic discussion [*omr@ips.id.ethz.ch*] of recognition of musical notation.

turned detailed reports on their current work. Some researchers reported briefly that they did not feel that their work was far enough progressed to warrant an official report. A few regarded some of the information sought as proprietary.

For our part, we discovered that constructing a survey for optical recognition is a daunting task. Although some of our questions were intended to facilitate comparison, we learned that it is difficult, and at the present time probably unwise, to make them. The reasons why this is so are themselves instructive, particularly for potential users of optical recognition software. They are indicated in the following pages.

Approaches and Problem Types: A General Introduction

There are myriad approaches to optical recognition. Each one is guided by a different view of the most significant obstacles to recognition and the most efficient available means of addressing general problems of optical recognition.

1. Conceptual Issues

An overriding problem of interpretation is that of distinguishing foreground from background. We normally think of a musical score as a two-dimensional object, but it is actually interpreted cognitively as one of three dimensions. The third dimension consists of non-sounding cues to interpretation, such as staves and systems. For this reason, many systems begin with the removal of staff lines, which form a complicating visual background, in order better to reveal the foreground of notes, stems, and beams.

The adage that what is easy for human beings is difficult for machines is nowhere more true than in optical recognition. White notes are harder for most systems to identify than black notes because they contain very little visual matter. Contextual information is problematic: a dot of prolongation and a dot of articulation (staccato) cannot be distinguished by shape or size; they must be distinguished contextually. When it comes to interpreting the images captured, recognition programs must do a lot of the same bookkeeping required in music printing programs to interpret items correctly. Pitch interpretation is dependent on the clef in use, duration may be dependent on metrical signature, and so forth.

2. The Basic Process

The main steps in the process of music recognition are (a) capture of the image as a bitmap, (b) editing of the image to facilitate correct interpretation, (c) conversion of bitmap to a code (such as MIDI, *DARMS*, or *SCORE*) representing musical information, and (d) translation of the code to an application producing sound or printed musical notation.

3. Factors Relating to the Work Environment

The competence of the result and the ease with which it is obtained can be influenced by many factors—(1) the hardware, operating system, and software environment in use, (2) the graphic quality of the material scanned, (3) the complexity of the music, (4) the format to which conversion is sought, and (5) the efficiency of the applications program used to complete the recreation of the music.

(1) For programs implemented on the PC, acquisition times on a 486 will be significantly faster than those on a 386 but not necessarily the same from machine to machine, on which hardware components and operating setups may vary. Machine speeds have a significant effect on program performance. Programs implemented on some Unix workstations may have some inherent advantages conferred by specific resources for image capture and editing. The workstations themselves normally have much larger screens and higher screen resolutions than PCs or Macs, traits that are valuable in the editing of images.

(2) Many prototype programs of the past were designed to work only with single parts on one staff. Most current programs concentrate on music for not-too-large instrumental ensembles; many exclude keyboard music because of its often complex textures. We know of no program that attempts to capture text underlay in vocal music. Most do not attempt to capture other text elements, such as tempo indications. Users of recognition software must expect to supplement the material captured automatically with additional information if they wish to create performing materials.

(3) Intelligent scanning programs of the present time usually do not attempt to replicate completely the image of the original. Instead they try to capture selected attributes of musical information. Sound-out programs attempt to capture pitch and duration. Depending on the repertory selected, print-out programs may need to capture a significantly greater number of elements of information—slurs, stems, beams, articulation marks, dynamics, and ornaments.

(4) The time required to correct scanning errors will depend not only on the level of accuracy of the automatically acquired material but also on the nature and efficiency of the applications program into which the material is read, not to mention the user's facility in using it. While support for notation output is inherently more demanding than that for sound output, most programs intended to produce printed output from scanned material support only a finite number of the additional elements—stems and beams, perhaps, but not slurs. The number of attributes supported characteristically increases as the program grows in overall competence.

(5) There are no uniform measures of efficiency in the evaluation of applications programs. Some relevant issues are raised in the article "How Practical is Optical Music Recognition as an Input Method?" (pp. 159-166).

Problem Categories

Those involved in the development of programs for recognition of music tend to make decisions about all of the above items at the outset of their work. They spend much of their subsequent research time trying to address problems presented by the graphical image itself. These can be grouped into three categories—visual surface problems, object recognition problems, and problems of music representation.

1. Visual surface problems

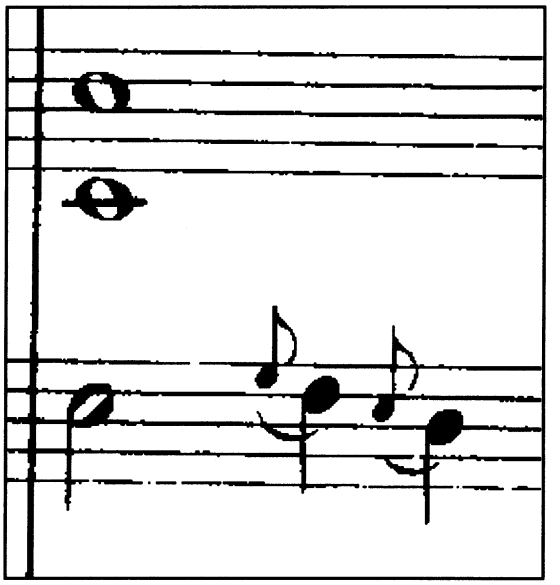


Figure 1. Surface imperfections: skewing and ambiguous positioning (uppermost note).

Most visual surface problems result from imperfections in the printing. Given that the optical acquisition of works printed in recent decades is likely to violate copyright, scanning researchers often select materials from the nineteenth century for training materials. This is the case with all of the illustrations in this section, which come from the Breitkopf und Härtel edition of Mozart's Symphony in G Major, K. 114, published in 1879. The erratic typography shown in our enlargements is characteristic of a great deal of printed music that is legally available for scanning.

Some common imperfections are rotation of staves so that staff lines are not exactly parallel with the edge of the page, variability in staff line thickness, and incorrect positioning (Figure 1). However trivial they may seem to the human eye, these irregularities can be quite debilitating to recognition software, which can usually accept small variations but only within clearly set limits. Respondents to our survey reported acceptance limits of skewed images within the range of 5° to 10°. The image in Figure 1 is the only example shown here that has not been rotationally corrected by at least 1°.

Other problems of the visual surface fall into two categories—missing information and superfluous information. Staff lines and leger lines that are not continuous (as in Figure 2) as well as objects that are incompletely drawn (*e.g.*, the half note in Figure 3) or incompletely filled (*e.g.*, notes on the downbeat in Figure 4) are all familiar problems of insufficiency in the visual image itself. In all of these examples small variations in notehead size and placement relative to stems and staff lines can also be detected.

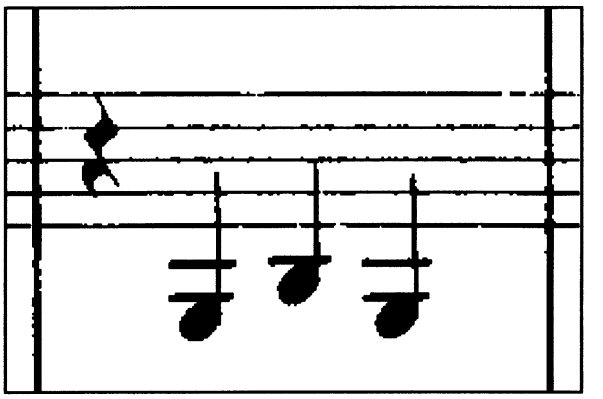


Figure 2. Surface imperfections: note the broken staff line at the top right and the variable width of both staff- and barlines.

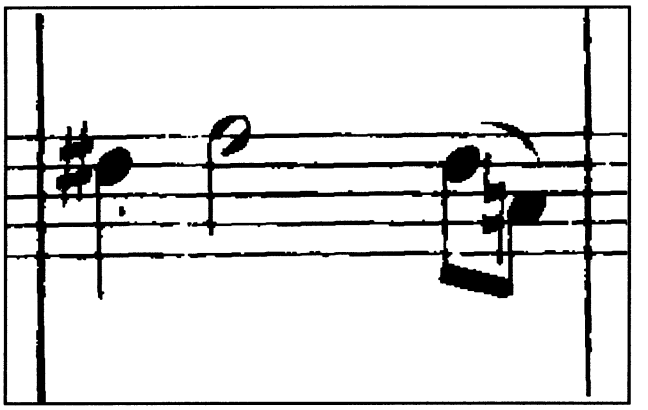


Figure 3. Insufficient information: the half note and the natural sign both lack closure. Compare the hypothetical white space in the half note with the actual white space bordered by the stem, the notehead, and the contingent flag in the tied octaves of Figure 4.

The problems caused by superfluous information, which cannot be filtered by recognition software unless anticipated, are less likely to be evident to users. The most common kind of superfluous information is dirt (Figure 5).

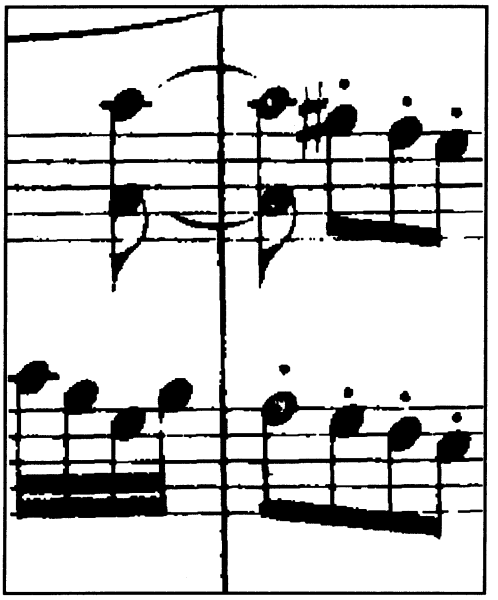


Figure 4. Flawed information: the eighth notes on the first beat are incompletely filled. Note the variable distance between the staccato dots and the notes to which they pertain.

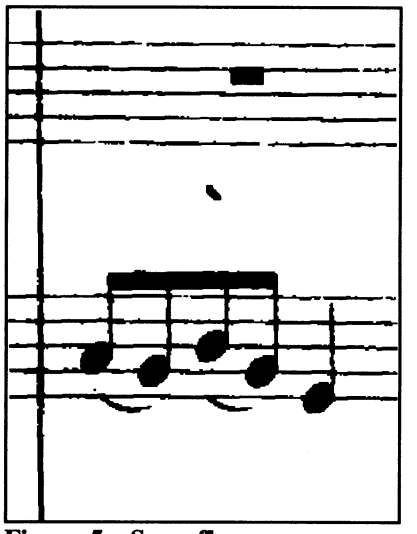


Figure 5. Superfluous information: dirt.

Depending on where it occurs in relation to other objects, dirt can be misinterpreted to be almost anything. If one were attempting to read the blob in Figure 5, one might be tempted by its placement to consider it a dynamics mark of some kind.

Programs can also create superfluous information through the misrecognition of objects. In one of the ensuing examples (5a on p. 132), dynamics markings were interpreted as whole notes. Errors of this kind are common in the development stage, since the effort to be selective about which musical attributes to capture provides scope for a category of errors that would not exist if all elements could be reliably captured.

2. Object Recognition

Optical recognition software in general utilizes a wide range of diverse approaches. One approach is to bound specified areas in a hypothetical box and make inferences from box sizes and shapes. If one were always scanning material from the same typographical source, such as a particular press that always worked with the same font and the same size staves, one might attempt to make templates for the various objects and match captured objects to the templates. Most situations are not so simple, and various methods developed in artificial intelligence are employed.

Problems of object recognition vary somewhat with the nature of the approach. Some approaches are better suited to large features, such as slurs and beams, others to small features, such as stems and flags. Reliable interpretation for replication of printed material will obviously require equal competence in the treatment of both ranges. This is the kind of success that remains elusive.

Problems that are common to most approaches include inconsistencies of size, shape, and presentation as well as those of superimposition. In all of these categories some confusing situations conform to the accepted grammar of musical notation and others occur by accident.

A. INCONSISTENCIES OF SIZE

An intended inconsistency of size occurs in the case of a grace note. A program that tries to identify objects by shape only might have trouble distinguishing a grace note or a cue note from an ordinary note. It can be filtered out by its size, providing that the surround area is not too noisy or too dirty. However, when one filters by size, then the erratic nature of page composition can muddle the result. In older scores that are, from a copyright perspective, appropriate for scanning, stem lengths and beam widths are likely to be variable (Figures 6a and 6b).

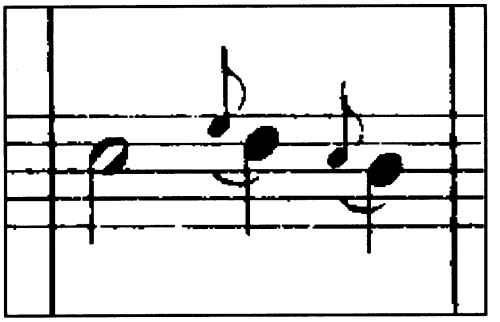


Figure 6a. Compare the stem lengths in this passage with those in Ex. 6b.

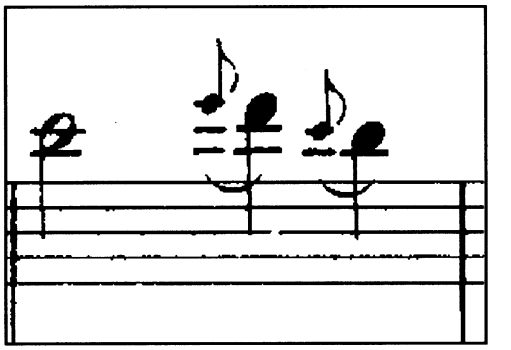


Figure 6b. Compare the stem lengths with those of Ex. 6a.

B. INCONSISTENCIES OF SHAPE AND PRESENTATION

Some intended instances of inconsistency in presentation would be the note centered on the staff line vs. the note centered on the space. When multiple beams cross multiple staff lines, many kinds of unpredictable shapes can result, especially since the angle of tilt in the beams is variable and the overlay created will depend on the vertical placement of the associated notes. This creates an unfortunate but accepted area of difficulty for visual interpretation. However, when such objects as rests and fermatas are erratically aligned in a score, a recognition program must search a relatively large area in order to

locate and identify them. Clef signs, meter signatures, and quarter-note rests were frequently cited as problems because of wide variations in graphic design.

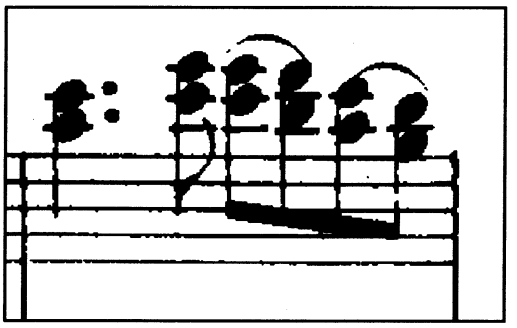


Figure 7. Superimposition: slurs touch noteheads. Note also that the flag of the first eighth note crosses a leger line.

C. Contiguity and Superimposition

Even when the quality of the original material is superb, the contiguous placement or Superimposition of objects presents serious obstacles to recognition. Slurs are especially troublesome. A slur crossing a stem (Figures 1, 6a, 6b) may be tolerated, but a slur touching a note (Figure 7) or crossing a dynamic marking creates apparent objects that will not be found in a graphic lexicon.

3. Problems of Music Representation

A generation of researchers have explored the anomalies of representing Western art music in common music notation without exhausting all the aberrations or describing them systematically. It is not the task of optical recognition research to do so, but since recognition software produces files for the reconstruction of musical scores and sounds, it is inevitable that from time to time problems of musical representation will need to be addressed. Global problems such as the realization of grace notes and the separation of tracks in keyboard works may not interfere often if the intended output is specifically for sound or for notation. Local problems are more likely to be troublesome. In Figure 8 we show one—an instance in which multiple objects (notes of diverse durations) share common parts (here stems) in an unlikely way.

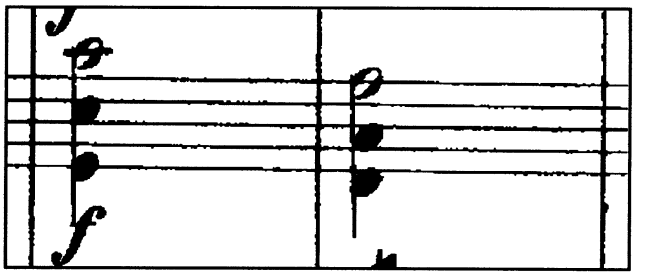


Figure 8. Issues in music representation: half notes and quarter notes share common stems.

Such examples interfere with a grammatical approach to object recognition, since in its graphic presentation the underlying logic of notation is beset on every side with exceptions of an unpredictable nature. Additionally, recognition programs may be confused by objects not intended for capture. One developer cited guitar chords as an example: they are hard to ignore because of parallel lines and black circles.

4. Partial Solutions

Approaches to optical recognition have varied widely over the years. Early efforts at text recognition often used *template-matching*. Since this depends on matching specific fonts at exact sizes, its value is severely limited in handling typographically diverse materials. Recognition by *geometrical analysis* of graphic features such as lines, angles, orientations, and curvatures of scanned objects is based on algorithmic matching. This approach overcomes the problems of exactness encountered in template-matching but elicits new areas of confusion in attempting to distinguish, for example, between b's and h's, l's and 1's, or O's and 0's.

The optical scanning of music cannot build entirely on these approaches, because the symbols used in musical notation are so much more numerous, their visual grammar so much more complex, and their meanings sometimes determined by contextual clues, as the previous figures have demonstrated. One technique is that of bounding groups of objects, such as a series of eighth notes connected to a common beam, and defining the contents hierarchically from most to least comprehensive. The demarcated area is called a *bounding box*. Researches have explored foreground-background separation (*i.e.*, removing the staff lines to expose the notes), background enhancement, tests of identity by rotation and/or slanting of objects, and a host of other image-processing techniques. In one enterprise, additional staff lines are hypothetically projected to assess the height and depth of objects above and below the staff. In another, the staff grid is not inspected along the whole of its horizontal length; instead the program sets the clef at the start of each line and assumes that it does not change along the way. This has obvious pitfalls for accuracy but saves processing time. Most enduring projects use a combination of methods to attend to the diverse problems encountered.

Some projects reported in recent issues of *Computing in Musicology* include the following:

- 1987: Bernard Mont-Reynaud, Stanford University; Unix environment.
- 1987, 1989, 1990, 1991: Nicholas Carter*, University of Surrey, UK—programming on Unix workstations and PCs; output to *SCORE* [notation program]. Now licensed to Coda Music Technology.
- 1987, 1990, 1991: Alastair Clarke, University of Wales, Cardiff, UK—programming on PC.
- 1987, 1990: Waseda University, Tokyo—one aspect of the WABOT musical robotics enterprise; limited number of symbols attempted; *SMX* [code] output
- 1990, 1991: William McGee*, Paul Merkley, University of Ottawa—programming on PC; original focus on syntactically simple medieval music [but subsequently broadened to common music notation]; output in *DARMS* [code] to *The Note Processor* [notation program]

- 1990: Dimitris Giannelos, ERATTO-CNRS, Paris—programming on Macintosh; focus on Greek Orthodox music; output to *Euterpe* [notation program]
- 1991: Ichiro Fujinaga, McGill University, Montréal—programming on Unix workstations and PC; output to *Nutation* [notation program for the NeXT]. Now licensed to A-R Editions.

*Starred contributors are among the respondents whose work is represented in the ensuing pages.

Seven Current Projects

The seven detailed responses we received concern three programs running on Unix machines and three on PCs. These can be briefly identified as follows:

Program	**Respondent**	**Platform**	**Output format(s)**	**Application**
AMSR	Kia-Chuan Ng	Unix	MIDI	sound
MidiScan	Christopher Newell	MS-DOS	MIDI	sound
MusicReader	William McGee	MS-DOS	*DARMS*, MIDI	printing, sound
NoteScan	Cindy Grande	Macintosh, MS-DOS	*NoteScan* *NoteScan*	printing, sound
OMR	Martin Roth	Unix	*Lipsia*	printing
SAM	Elizabeth Botha	MS-DOS	*MOD*	sound
SightReader	Nicholas Carter	Unix	*SCORE*	printing

Since the difficulty of optical recognition varies tremendously with the kind of music scanned, our questionnaire listed ten kinds of pieces that would produce different problems. Respondents were asked to rate the difficulty of each. Their responses are indicated in Table 1. Note that the two examples we circulated—by Handel and Clementi respectively—represent a single printed part from an orchestral score and a simple printed score for piano. Respondents were also asked to rate the difficulty of capturing various kinds of musical objects. The results are indicated in Table 2. Note that most participants did not attempt to recognize all the object types in the set pieces.

A one-page description of each project follows these two tables of general information. In team efforts, the name of the respondent is indicated by an asterisk (*). Bibliographical citations are given with annotations at the end of the article. Then the set pieces are discussed and performance results are given.

Test sources	AMSR	Midi-Scan	Music-Reader	Note-Scan	OMR	SAM	Sight-Reader
Printed parts/band or orchestra	easy	somewhat easy	easy	tested	tested	being tested	tested
Printed parts/early music	—	—	—	—	—	—	tested
Printed lead sheets	—	easy	—	—	tested	—	tested
Printed scores/chamber music	somewhat easy	—	somewhat easy	tested	—	being tested	tested
Printed scores/orchestral	somewhat difficult	somewhat easy	somewhat easy	tested	—	being tested	tested
Printed scores/piano vocal (popular/folk)	easy	easy	somewhat difficult	tested	tested	easy	tested
Printed scores/choral	moderate	somewhat easy	somewhat easy	—	—	—	tested
Facsimiles/early printed music	—	—	—	—	—	—	—
Manuscripts/recent	difficult	somewhat easy	—	—	—	—	—
Manuscripts/early	—	—	difficult	—	—	—	—
Maximum number of parts in score	no maximum	16	9	no maximum	not relevant	not stated	not stated

Table 1. Repertories attempted in the testing of optical recognition software. Respondents were asked to indicate which ones had been attempted and to rate the results in achieving accurate recognition on a five-point scale ranging from easy to difficult. The *SightReader* response did not provide separate ratings but said that the relative difficulty depends on the musical content of the pages and the quality of the printing, a position that undoubtedly applies to the other programs as well.

Feature	AMSR	Midi-Scan	Music-Reader	Note-Scan	OMR	SAM	Sight-Reader
Noteheads (black)	1	1	1	1	2	X	X
Noteheads (non-contiguous) in two-note chords	1	1	1	2	3	X	X
Noteheads (non-contiguous) in three-note chords	2	1	1	2	3	X	X
Stems	1	1	1	1	2	X	X
Stem directions	1	1	1	1	4	—	X
Clef signs	2	1	2	5	4	X	X
Time signatures	—	1	5	5	—	—	—
Key signatures	3	1	2	1	4	X	X
Accidentals	3	1	2	2	3	X	X
Gracenotes	—	—	3	—	—	—	—
Ornaments	—	—	3	—	—	—	—
Dynamics letters (*p*, *f*)	—	—	—	—	—	—	—
Articulation marks (staccato, etc.)	1	—	1	—	4	X	X
Tempo words	—	—	—	—	—	—	—
Text underlay	—	—	—	—	—	—	—
Crescendos, diminuendos	4	—	—	—	—	X	X
Beams	2	1	1	2	3	X	X
Braces	—	—	—	—	—	—	X
Brackets	—	—	—	—	—	—	X
Barlines (part)	1	1	1	2	2	X	X
Barlines (score)	2	1	1	2	—	X	X
Ties	2	1	3	X	—	—	X
Slurs	2	—	3	—	—	—	X
Tuplet numerals	—	—	—	—	—	—	—
Basso Continuo figuration	—	—	—	—	—	—	—
Guitar tablature	—	—	—	—	—	—	X

Table 2. Musical features supported by optical recognition software. Respondents were asked to score the competence of their programs in general in recognizing each feature on a sliding scale in which 1 = very easy to recognize and 5 = very difficult to recognize. Some respondents merely indicate which features are supported; these responses are represented by an X.

Automatic Music Score Recogniser (AMSR)
Developers: Kia-Chuan Ng* and Roger D. Boyle
Location: The University of Leeds, England
Hardware/operating system: Unix (Silicon Graphics Indigo)
Scanning equipment: Hewlett Packard ScanJet Plus
Scanning resolution: 300 d.p.i.
Export format: Standard MIDI files
Intended applications: sound, printing, analysis
Anticipated date of availability: end of 1994

AMSR, which is being developed on a Unix workstation, is a graphics-to-sound translation program. It utilizes a screen display of the scanned bit-mapped image. There is no need to provide a screen display of the reconstructed score, since it does not aim to support a full range of music printing features. In consequence, however, there is currently no support for screen editing of the information. No information about the two sample pieces distributed was included in the response. However, the respondents report having tested their program on extracts from piano sonatas by Beethoven and Mozart and solo violin parts from Bach's sonatas and partitas.

The overall approach is described as one employing a method which directly reverses the process of music writing. Whereas a composer would normally write a notehead followed by a stem and/or a beam and lastly other markings such as slurs and ties, this group would first pick out long and thin features such as slurs and ties, followed by beams, then stems. In this way the complicated compound features are broken up into a lower graphical level of primitives before recognition.

Currently an effort is being made to use feedback from the *Recogniser* to control the segmentation into musical primitives automatically. Further information is available in the respondents' article on segmentation [see References].

Further information: Kia-Chuan Ng, Division of Artificial Intelligence, School of Computer Studies, The University of Leeds, Leeds LS2 9JT, England, UK; tel.: +44 532/336798; fax: 532/335468; e-mail: *kia@scs.leeds.ac.uk*. Roger D. Boyle: *roger@scs.leeds.ac.uk*.

MidiScan
Respondent: Christopher Newell
Location: Musitek, Ojai, California
Hardware/operating system: MS-DOS/Windows 3.1 (286, 386, 486, PS II compatible)
Scanning equipment: any scanner capable of producing TIFF files can be used
Scanning resolution: 300 d.p.i.
Export format: Standard MIDI files, readable on the Mac, Atari, and Amiga as well as the originating DOS machine
Intended applications: sound via MIDI
Anticipated date of availability: available by direct mail from Musitek and through music merchants since May 1993

MidiScan, the first commercial program for music recognition, is a self-contained system for converting printed information to Standard MIDI files. [It does not attempt to recognize objects not represented in a Standard MIDI file.] Major parts of the program have been written by Wladyslav Homenda (CPZH, Warsaw, Poland).

Users need a hand-held or fullpage scanner to capture each page of the score as well as imaging software that is capable of auto-switching and outputs TIFF files. The TIFF files are loaded and processed sequentially by MIDISCAN. User intervention is required only for screen editing. The bit-mapped image which it displays can be edited on the screen with the Music Notation Object Recognition (MNOD) graphical editor.

MidiScan quotes an approximate recognition time of 5 minutes per page of music.

Further information: Christopher Newell, Musitek Music Recognition Technologies, 410 Bryant Circle, Ste. K, Ojai, CA 93023-4209; tel.: (805) 646-8051; fax: (805) 646-8099; no e-mail address provided. [A review of *MidiScan* by Martin Roth appeared in the November 3 (1993) issue of Roth's electronic discussion of optical music recognition.]

MusicReader
Developers: William F. McGee* and Paul Merkley
Location: Ontario, Canada [originated at University of Ottawa]
Hardware/operating system: MS-DOS (386/16 MHrz)
Scanning equipment: Hewlett Packard ScanJet
Scanning resolution: 300 d.p.i.
Export formats: Standard MIDI, General MIDI, DARMS
Intended applications: sound, printing, analysis
Anticipated date of availability: January 1994 (beta-test version currently available)

MusicReader, on which an fuller description was separately submitted, provides display of a bit-mapped image. It produces both *DARMS* and MIDI files that can be read and edited using the *Note Processor*, a notation program for the PC. The reconstructed part or score can be displayed and edited using the *Note Processor*. *DARMS* data can also be used for analytical applications. *MusicReader*'s Standard and General MIDI files can also be used by MIDI notation programs and by MIDI hardware including synthesizers and tone generators. The MIDI files produced provide realizations for trills, mordents, turns, and arpeggios.

MusicReader software also offers an alternative input option to the user without a scanner, whereby a MIDI keyboard is used for pitch acquisition and a computer keyboard supplies duration. Output to both *DARMS* and MIDI is possible.

MusicReader has established a service bureau and offers scanning, with output in either format, at $5/page. Hardcopy is available at an additional $5/page.

Further information: William F. McGee, 73 Crystal Beach Drive, Nepean, Ontario, Canada K2H SN3; tel.: 613/828-9130; fax: 828-9130; e-mail: *mcgee@citr.ee.mcgill.ca*. Paul Merkley, Department of Music, University of Ottawa, Ottawa, Canada K1N 6N5; tel.: (613) 564-9239; fax: (613) 564-5643; ; e-mail: *merkley@acadvm1.uottawa.ca*. An extensive description of this system is provided in the Developer's Report on pp. 146-51.

NoteScan™
Respondent: Cindy Grande*
Location: Grande Software, Inc., Seattle, WA
Hardware/operating system: Macintosh IIci (25MHz); conversion to MS-DOS
Scanning equipment: Apple One Scanner
Scanning resolution: 200 and 300 d.p.i.
Export format: *NoteScan*
Intended applications: printing, sound, analysis
Anticipated date of availability: demo disk available; to be offered as enhancements to two notation programs—*Nightingale* and *Music Printer Plus*—by Temporal Acuity Products in Spring 1994

NoteScan converts recognized information to an intermediate file format (*NoteScan* NTIF), from which it may be converted to a wide variety of proprietary formats used by commercial notation programs. Any scanner producing TIFF files may be used. Existing commercial agreements for its use are non-exclusive.

The Macintosh version produces bitmapped images on the screen. These may be edited using programs such as Adobe *Photoshop*. The reconstructed score may be displayed, enlarged, reduced, and edited in music applications programs such as *Music Printer Plus* and *Nightingale*.

Cindy Grande is the developer of the recognizer logic. Charles Rose wrote the *Nightingale* interface. Gary Barber wrote the MS-DOS adaptation and the interface to *Music Printer Plus*.

Further information: Cindy Grande, President, Grande Software, Inc., 19004 37th Avenue South, Seattle, WA 98188; tel.: (206) 439-9828; fax: (206) 824-2612. Temporal Acuity Products, Inc., is located as 300 120th Avenue N.E., Bldg. 1, Bellevue, WA 98005; tel.: (800) 426-2673. NoteScan is a trademark of Grande Software, Inc.

Optical Music Recognition (OMR)
Respondent: Martin Roth
Location: Eidgenössische Technische Hochschule, Zurich, Switzerland
Hardware/operating system: Unix (Sun, NeXT)
Scanning equipment: various
Scanning resolution: 200 and 300 d.p.i.
Export format: *Lipsia*
Intended applications: printing
Anticipated date of availability: undertaken as thesis project; future plans pending

Development of the *OMR* program was initiated as a thesis project in engineering and computer science (1993) and is not directly comparable with other programs listed here. It is intended to support automatic recognition of music fonts. The program, in *C*, concentrates on such features as a bounding box and estimates of weight and center of gravity. It does not attempt to supply semantic information needed for interpretation of captured symbols. It has been designed to work with *Lipsia*, a notation editor developed several years ago at ETH by Giovanni Müller [see *CM 1987*, Illustration #44, p. 71].

OMR is oriented toward object acquisition. Bit-mapped images may be edited using the workstation utilities *X-loadimage* and *X-view*. A semantic lexicon is used to determine placement by *Lipsia*; *OMR* has no semantic constraints. A reconstructed score may be shown using Display postscript on a NeXT workstation. Development of *Lipsia* has now ceased, and the future path of *OMR*'s development is uncertain. The program is unusual in handling an arbitrary number of staves.

Further Information: Martin Roth, ETH Zurich, IPS - RZ F16, Steinstrasse 58, CH-8003 Zurich, Switzerland; tel.: +10 1/256 55 68; 1/463 13 61; fax: 1/261 04 68; e-mail: *roth@ips.id.ethz.ch*. Roth moderates an electronic discussion of optical music recognition (*omr@ips.ed.ethz.ch)* and maintains an electronic bibliography in several formats (details given on p. 145).

Score Analyzing Maestro (SAM)
Developers: Elizabeth C. Botha* and students (Karl Geggus, Johan Boot)
Location: University of Pretoria, South Africa
Hardware/operating system: MS-DOS (486/33)
Scanning equipment: Hewlett Packard ScanJet
Scanning resolution: 300 d.p.i.
Export format: MOD (Amiga) format [supported by Soundblaster card]
Intended applications: sound, conventional music notation, Braille music notation, analysis, educational software
Anticipated date of availability: first half of 1994

SAM is a self-contained acquisition program for the PC oriented entirely toward sound output. It does not provide for screen display and editing of the reconstructed score. Pitch and duration are captured and plans are underway to implement volume level and change parameters in output files from the recognition of dynamics indicators in scanned scores. Support for Standard MIDI files and General MIDI is planned.

The developers intend to make the program available electronically by FTP. For current information via the Internet send the command:

At mainframe prompt:	**ftp** ***ftp.ee.up.ac.za***
Logon:	**anonymous**
To change directories:	**cd** ***/u/ftp/pub/musiek***
To prepare for a binary transfer:	**binary**

Retrieve (**get**) the file *readme.tex*. The documentation files are updated as changes in the program are implemented.

Further information: Dr. Elizabeth C. Botha, Dept. of Electrical and Electronic Engineering, University of Pretoria, Pretoria 000 2, South Africa; tel.: +27 12/420-2981; fax: 12/437837; e-mail: *botha@ford.ee.up.ac.za*.

SightReader
Respondent: Nicholas P. Carter
Location: University of Surrey, Guildford, England
Hardware/operating system: Unix (Sun, Hewlett Packard, NeXT) for development; MS-DOS (486) for output
Scanning equipment: Hewlett Packard ScanJet (or other scanner producing TIFF files)
Scanning resolution: 300 d.p.i.
Export format: *SCORE* input files and *SightReader* files
Intended applications: sound, notation, analysis, electronic distribution
Anticipated date of availability: undetermined; licensed to Coda Music Technology

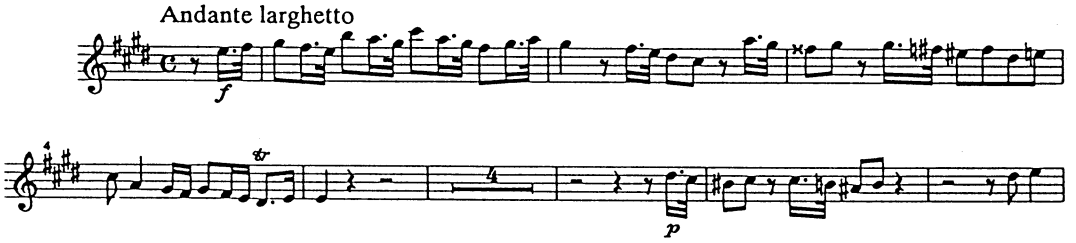
SightReader has been developed on Unix workstations and makes use of separate utilities, for example to provide a scrollable screen view of bit-mapped images converted to TIFF files. The bitmap can be cropped and scaled. The image can be reduced or enlarged. *SightReader* exports files to the input format used by *SCORE*, a notation program for the PC. Parts of *SightReader* are now running on a PC.

In separate experiments, *SightReader* was used to explore the capture of information from late Renaissance part books containing white mensural notation [see "Segmentation" in the Bibliography] and with the reconstruction of an early twentieth-century score [see "Walton"].

Further information: Dr. Nicholas P. Carter, Department of Physics, University of Surrey, Guildford, Surrey GU2 5XH, England, UK; tel.: +44 483/300800; fax: 483/300803; e-mail: *N.Carter@ph.surrey.ac.uk*. Also see the Developer's Report on pp. 152-58.

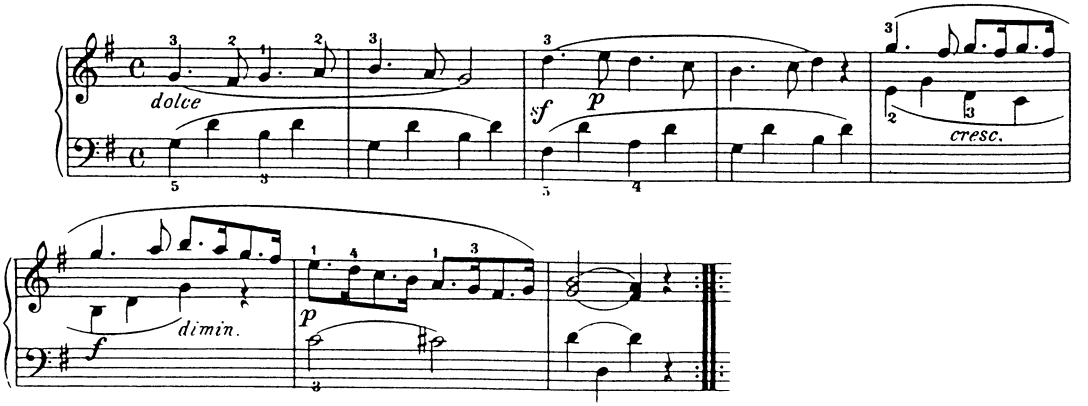
Set Pieces

In the interests of creating a common focus for discussion, two musical examples were distributed with the questionnaire. The first was a violin part from Handel's opera *Radamisto*.



Example 1. Excerpt from Handel's *Radamisto*.

This example contains beamed eighth, sixteenth, and thirty-second notes, as well as single quarter and eighth notes with a key signature involving four sharps and numerous accidentals of various kinds. It also has half, quarter, and eighth rests. In addition it includes a tempo indication, dynamics markings, a multibar abbreviation, a measure number, and a trill sign. This example was based on a recent print generated by computer at CCARH.



Example 2. Excerpt from a Clementi sonatina in G Major.

The second example was an eight-bar excerpt from a well known piano sonatina by Clementi. It contains half, quarter, eighth, and sixteenth notes (some with dots and/or beams), quarter rests, and repeat signs. It is mainly different from the first example in that (a) it is in a two-stave score, (b) it contains numerous slurs, and (3) it contains fingering numbers over many notes. This example was published in the early twentieth century by conventional means.

Respondents were given the option of scanning either or both of these examples or an item of their own choice. They were asked to record the time required to go, in four steps, from the original material to its complete, accurate reconstruction. Machine speeds were not uniformly reported. Respondents were asked to compute a statistical result representing the accuracy of the automatically captured data. In the absence of a commonly accepted way of making such a computation, they were also asked to explain their own means and to suggest a way of providing a uniform measure of competence.

Test Reports and Results

Respondents' reports provide little basis for comparison not only because of differences in hardware, operating systems, and software environments but also because some respondents did not provide precise numbers or scanned material other than what was sent. One group, *AMSR*, did not provide scanning times for this survey.

MidiScan

MidiScan [see description on p. 122] reported the following scanning times for the Handel example:

Operation	Handel
Input time	0:12
Image processing time	0:20
Screen correction time	0:30
Output time	0:20
Total elapsed time	1:22

Table 3. *MidiScan*: Scanning, processing, and output time in minutes and seconds. Output was to MIDI files. Processing was done on a 486.

Note that since a MIDI file is the objective, this scan could exclude the dynamics marks, trill signs, and tempo words ("Andante larghetto") and that no printed notation is produced by *MidiScan*.

MusicReader

MusicReader's tests (Table 4) were run on a 386/16 MHz PC.

Operation	Handel	Clementi
Input time	0:16	0:22
Image processing time	4:16	8:50
Screen correction time	3:05	6:28
Output time	0:38	0:30
Total elapsed time	8:15	16:10

Table 4. *MusicReader*: scanning time in minutes and seconds. "Screen correction" represents time spent editing *DARMS* code. Output time represents *DARMS*-to-MIDI conversion and audio playback. Processing was done on a 386.

Since other DOS contributors offered results based on 486 performance, we asked *MusicReader* for an estimate of times that would be achieved if the tests had been performed on a 486. These were roughly calculated to be one third of the values shown in Table 4. However, it would not necessarily follow that screen correction time, the most substantial part of the process, would fall by this much, since it is dependent largely on manual skills.

```
I1 !G !K4# !MC,00@Andante larghetto$ 5RE ((8S.D,VF (9TD))) /
I1 (30ED (9S.D (8TD))) (32ED (31S.D (30TD)))
   (33ED (31S.D (30TD))) (9ED (30S.D (31TD))) /
I1 30QD 5RE ((9S.D (8TD))) (7ED 6ED) 5RE ((31S.D (30TD))) /
I1 (9##ED 30ED) 4RE ((30S.D (9*#TD))) (8#ED 9ED 7ED 8*ED) /
I1 !G !K4# 6ED 4QU ((3SU 2SU)) (3EU (2SU 1SU)) (0E.U,0T (1SU)) /
I1 1QU 5RQ 5RH /
I1 R4W /
I1 5RH 5RQ 5RE ((7S.D,VP (6TD))) /
I1 (5#ED 6ED) 5RE ((6S.D (5*TD))) (4#EU 5EU) 5RQ /
I1 5RH 4RE 7ED 8QD /
```

Example 3. *DARMS* code for the Handel example produced by *MusicReader*.

Since *MusicReader* supports both a print code (*DARMS*; cf. Example 3) and a sound code (MIDI), it must support a broader range of objects than if it supported only one output format. Therefore in the screen editing phase for the Handel example *MusicReader* would be obliged to add codes for dynamics and tempo words; in the Clementi example the dynamics and fingering numbers would need to be added. The commentary on *MusicReader* is on p. 123. A substantial description is given independently on pp. 146-51.

NoteScan

NoteScan, unknown to us when our survey was distributed, was located shortly before we went to press. In lieu of the distributed examples, its respondent provided results of other timed tests; accuracy rates above 90% were claimed. These tests were all run on a Macintosh IIci/25MHz with output to both the *NoteScan* file format and to *Nightingale*. Times reported below (in minutes and seconds) do not include editing:

1. Bach: Two-Part Invention in G Minor, BWV 782, Bars 1-10, scanned from a typeset original of excellent quality at 200 d.p.i. 3:30

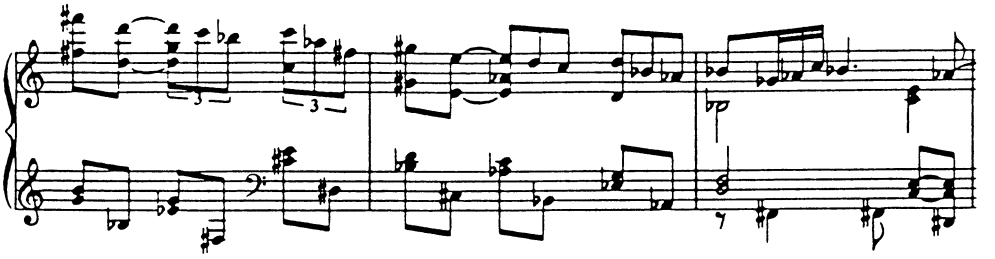
2. Beethoven: Piano Sonata Op. 81a ("Les Adieux"), First Movement, Bars 146-175, scanned from a typeset original of excellent quality at 200 d.p.i. 4:10

3. Haydn: (unidentified) String Quartet in F, final 15 bars of a fast movement, scanned from a miniature score at 300 d.p.i. 5:20

4. "Feeling Good" [piano/vocal with guitar chords], Bars 1-10 [excluding text and guitar chords]; scanned from a computer-set arrangement [*Finale*] at 300 d.p.i. 3:10

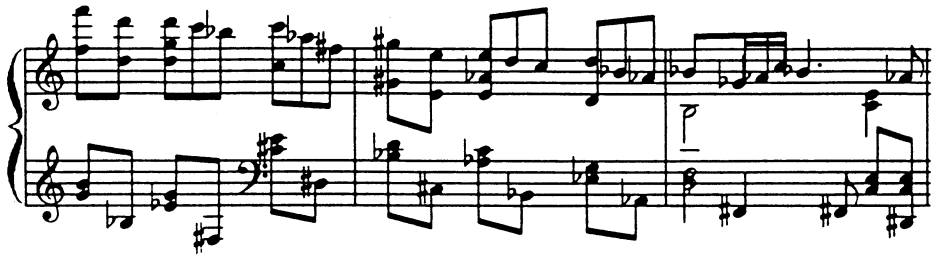
5. Leo Ornstein: Piano Sonata No. 9, 20-bar excerpt with complex voicing, numerous accidentals, *etc.*; scanned from typeset original at 300 d.p.i. 7:30

Examples 4a, b, and c show a three-bar passage from this excerpt:

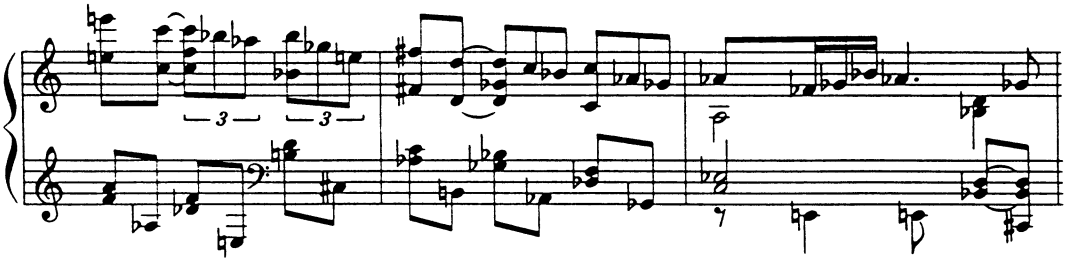


© 1990 Leo Ornstein. Used by permission of Poon Hill Press, Woodside, CA 94062

Example 4a. Leo Ornstein: Piano Sonata No. 8—original print.



Example 4b. Leo Ornstein: Piano Sonata No. 8—unedited *NoteScan* results.



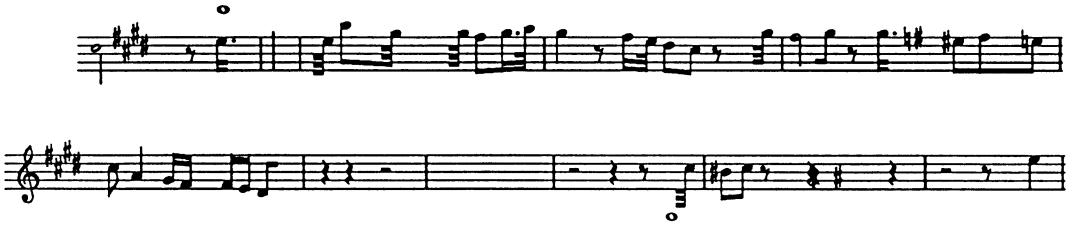
Example 4c. Leo Ornstein: Piano Sonata No. 8—*NoteScan* results edited and transposed using *Nightingale*.

OMR

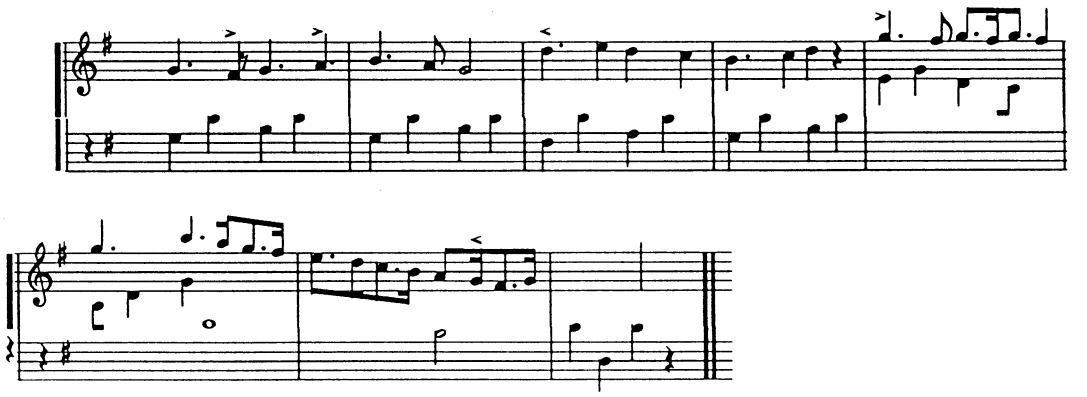
OMR reported scanning time only, with the following results:

Handel	0:22
Clementi	1:03

In lieu of other measures of time, the respondent for *OMR* provided actual output from his program (shown below). His comments point to the kinds of problems that those who experiment with optical recognition often encounter. He remarked that (1) the examples were "very light"; (2) the noteheads were smaller than those on which his program had been trained; and (3) when noteheads are not recognized, stems can be mistaken for barlines.



Example 5a. *OMR*: Unedited scan of the Handel example.



Example 5b. *OMR*: Unedited scan of the Clementi example.

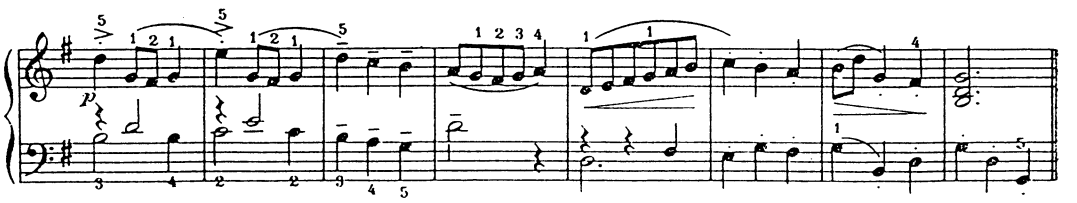
Examples 5a and b illustrate one feature that occurs commonly in optical recognition: mistaken object types. In Example 5a dynamics indications have been replaced by white noteheads. In Example 5b it appears that the fingering numeral 2 has been read as the accent > and the numeral 3 as the accent < . Although in this case the error is obvious, in other situations, errors generated by scanners can be hard to detect visually and may survive until the data is actually put to use in an application. Recognition software can also generate completely spurious objects, especially from specks of dirt on the scanned page. Thus the proofreading of scanned material is often more cumbersome than that of ordinary typesetting.

SAM

The developers of *SAM* provided scanning times for both the Clementi example and a Bach keyboard minuet. Times for the Clementi example are shown in Table 5.

Operation	Clementi
Input time	5:15
Image processing time	6:10
Screen correction time	5:55
Output time	2:20
Total elapsed time	19: 40

Table 5. *SAM*: Time (in minutes and seconds) elapsed in the scan of the 8-bar Clementi example. A 486/33 was used.



Example 6. *SAM*: Final staff of the Bach minuet scanned.

```
Unit 20
Line0 xs 0 ys 121 xe 264 ye 117
Line1 xs 0 ys 142 xe 264 ye 138
Line2 xs 0 ys 163 xe 264 ye 159
Line3 xs 0 ys 184 xe 264 ye 180
Line4 xs 0 ys 205 xe 264 ye 201
Line5 xs 0 ys 331 xe 264 ye 326
Line6 xs 0 ys 352 xe 264 ye 348
Line7 xs 0 ys 372 xe 264 ye 368
Line8 xs 0 ys 393 xe 264 ye 389
Line9 xs 0 ys 414 xe 264 ye 410
Crotchet1 xp 135 yp 183 xs 0 ys 0
Crotchet1 xp 215 yp 191 xs 0 ys 0
Crotchet2 xp 40 yp 163 xs 0 ys 0
Crotchet2 xp 86 yp 143 xs 0 ys 0
Crotchet1 xp 134 yp 392 xs 0 ys 0
Crotchet1 xp 39 yp 341 xs 0 ys 0
Crotchet1 xp 211 yp 370 xs 0 ys 0
Staccato xp 134 yp 218 xs 8 ys 7
Staccato xp 214 yp 219 xs 8 ys 6
Decrescendo xp 31 yp 240 xs 216 ys 30
Staccato xp 212 yp 336 xs 8 ys 6
Staccato xp 132 yp 424 xs 0 ys 0
```

```
Unit 21
Line0 xs 0 ys 122 xe 232 ye 119
Line1 xs 0 ys 142 xe 232 ye 140
Line2 xs 0 ys 163 xe 232 ye 161
Line3 xs 0 ys 184 xe 232 ye 182
Line4 xs 0 ys 205 xe 232 ye 203
Line5 xs 0 ys 330 xe 232 ye 327
Line6 xs 0 ys 351 xe 232 ye 348
Line7 xs 0 ys 372 xe 232 ye 369
Line8 xs 0 ys 393 xe 232 ye 390
Line9 xs 0 ys 414 xe 232 ye 410
Minim xp 43 yp 184 xs 0 ys 0
Minim xp 43 yp 215 xs 0 ys 0
Minim xp 43 yp 240 xs 0 ys 0
Crotchet1 xp 38 yp 341 xs 0 ys 0
Crotchet1 xp 177 yp 411 xs 0 ys 0
Crotchet1 xp 106 yp 372 xs 0 ys 0
Dotdurmod xp 71 yp 169 xs 16 ys 8
Dotdurmod xp 72 yp 213 xs 16 ys 9
Dotdurmod xp 71 yp 239 xs 16 ys 8
Staccato xp 41 yp 312 xs 8 ys 6
Staccato xp 109 yp 335 xs 8 ys 6
Staccato xp 173 yp 439 xs 8 ys 5
```

Example 7. *SAM*: File excerpt representing the last two bars of Example 6.

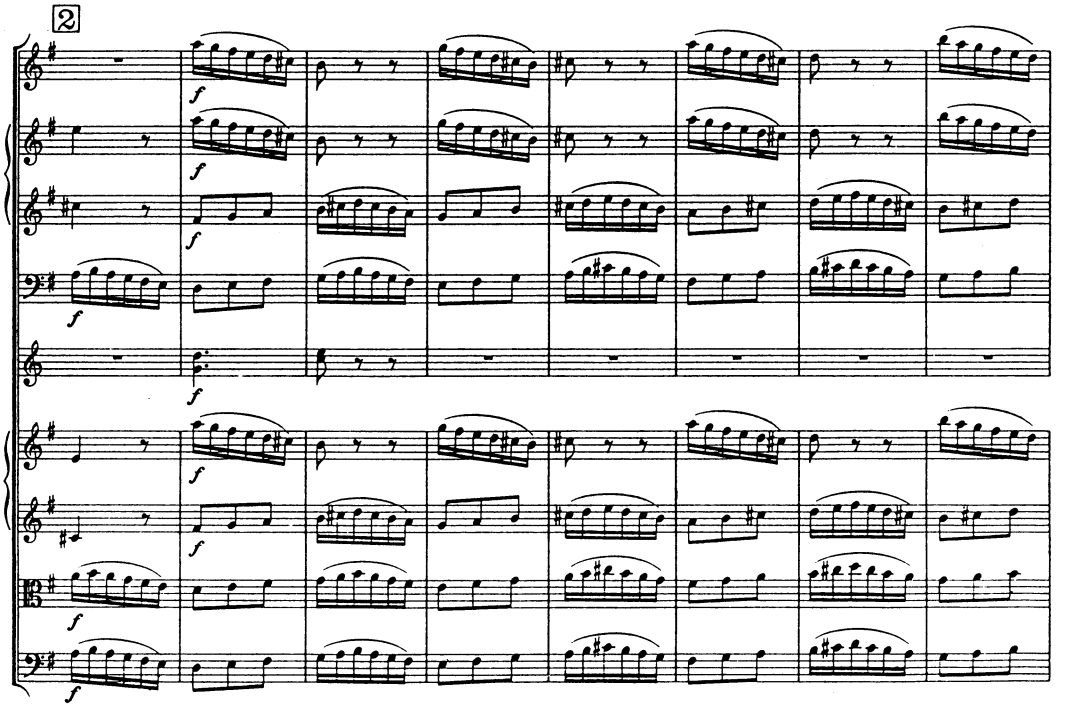
The minuet is one of 32 bars—four times the length of the Clementi example. It is also somewhat more difficult (at least in the second half) in texture as well as number and proximity of symbols than the Clementi example. These factors should be taken into account in examining the performance figures given in Table 6.

Operation	Bach minuet
Input time	4:40
Image processing time	21:47
Screen correction time	4:30
Output time	1:13
Total elapsed time	32:10

Table 6. *SAM*: Time required to recognize a 32-bar Bach minuet.

SightReader

SightReader submitted information concerning the scanning of one page of a Haydn symphony. The original is shown as Example 8. This has more staves (9) per system than the other examples submitted but is relatively free of objects other than black notes, accidentals, beams, and slurs.

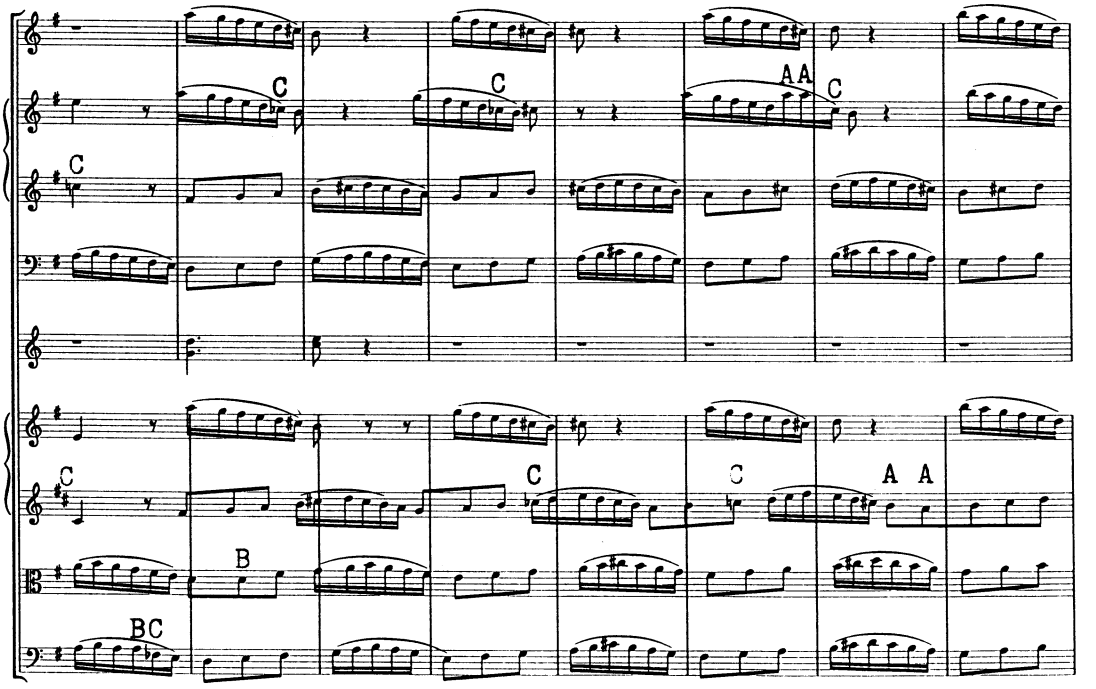


Example 8. *SightReader*: One system from Haydn's Symphony No. 8 (original image, reduced).

This example provides a useful opportunity to address the issue of counting the number of graphic objects, which may be significantly greater than the number of musical objects as we normally conceive of them. Horizontal line objects, for example, may be said to include 1 system, or 9 staves, or 45 staff lines. Vertical line objects include 8 barlines (each presented as two graphic objects, because of the section breaks) as well as the barline that forms part of the brace. A graphics recogniser must take separate account of the curved end pieces and the two curly brackets. This example contains 256 noteheads; these are attached to 254 stems. Ten stems have flags. Musicians would be inclined to consider these 520 graphics objects to make up 256 musical entities ("the notes"). We also find 24 eighth rests and 7 whole rests. There are 20 single beams (of three eighth notes) and 30 double beams (of six sixteenth notes); there are also 30 slurs. By way of miscellaneous markings, the example contains 9 clef

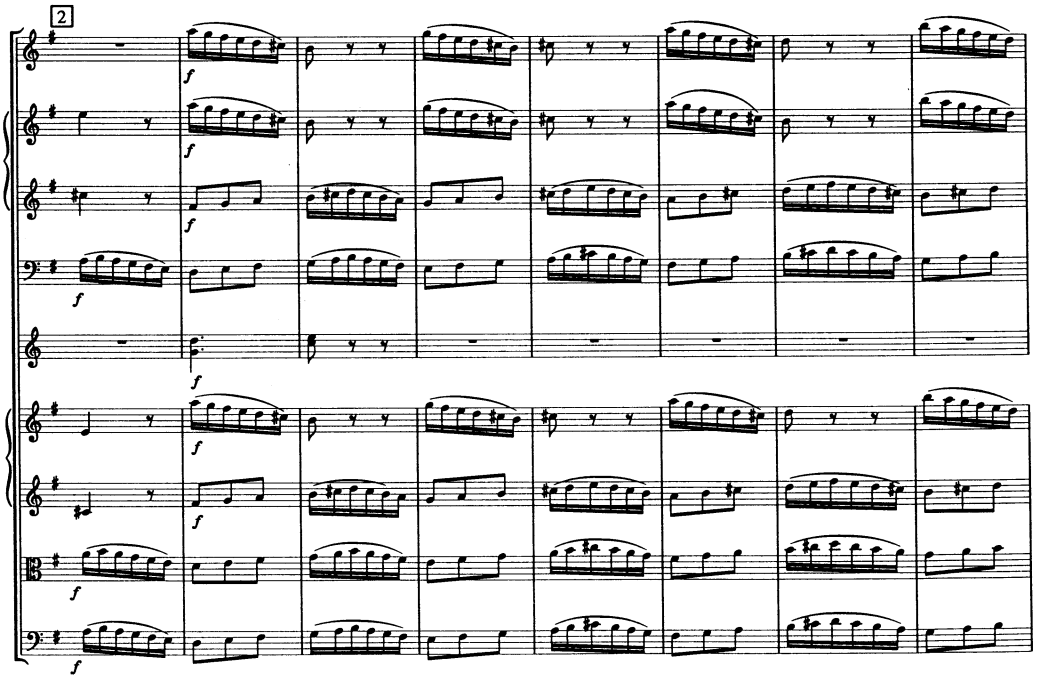
signs, 36 accidentals, 9 dots of prolongation, 9 dynamics signs, and 1 rehearsal number (containing two graphic parts—a box and a numeral).

SightReader provided samples of acquisition before and after correction. These are shown as Examples 9 (below) and 10 (on the following page).



Example 9. *SightReader*: Lower system of Haydn symphony after acquisition and translation to *SCORE* but before correction. Error categories are coded A (superfluous notes), B (erroneous pitch), and C (error in use of accidental).

Notice that in Example 9 there are superfluous notes (Error Type A; 4 instances), mistaken pitches (Type B; 2 instances), and superfluous, misconstrued, and misplaced accidentals (Type C; 8 instances). These errors were attributed to *SCORE*'s difficulty in placing barlines accurately. Slurs and beams are captured by *SightReader*, while dynamics markings are added editorially in *SCORE* files (Example 10). Durational values with the wrong visual grammar (*i.e.*, quarter rests in place of two eighths; 11 instances) are also corrected editorially.



Example 10. *SightReader*: Same system after editorial correction using *SCORE*.

Operation	Haydn symphony excerpt
Input time	00:30
Image processing time	00:48
Screen correction time	19:30
Output time	01:15
Total elapsed time	22:08

Table 7. *SightReader*: Elapsed time (in minutes and seconds) in the recognition of one page (two systems including the one shown above) of Haydn's Symphony No. 8.

Quantifying Program Performance

The easiest way to rate the efficiency of the system is to count the number of objects on the page and express the number recognized correctly as a percentage. It is not necessarily a very satisfactory method, since the misrecognition of one object can obviate the correct recognition of another that is contingent. From the user's point of view, some objects may be worth weighting more heavily than others, but such weightings would be relative to the repertory at hand.

We asked respondents to indicate the relative reliability of their programs in correctly recognizing all the object types that occur in the two examples we distributed. Their self-assessments are shown in Table 8.

Object	Midi-Scan	Music-Reader	Note-Scan	OMR	SAM
white notes	x	5	5	4	2
black notes	1	1	1	2	2
rests	2	2	3	3	2
stems	1	1	1	2	2
beams	2	1	2	3	3
clefs	1	3	5	4	2
barlines	1	1	2	2	1
braces	x	x	x	x	x
sharp signs	1	1	2	2	2
natural signs	1	1	2	2	2
fermatas	x	x	x	x	x
slurs	x	4	x	x	x

Table 8. Self-estimates of levels of accuracy in recognition of objects. An x means the object is not attempted. A score of 1 means it is easily captured. *SightReader* did not provide evaluations.

Measures of the rapidity with which a musical passage can be interpreted give only a very preliminary indication of software performance. Accuracy is essential if optical recognition is to become a useful tool.

How might accuracy be assessed? We asked our respondents to suggest ways in which this might be computed. The predominant view expressed was that one might divide the total number of objects correctly identified by the total number of objects in the example. If, for example, there are 100 objects in an example and 60 are correctly identified, a score of 60% would be achieved.

In moving from the generalization to the actuality, a number of considerations arise:

- In all cases mentioned here there is a discrepancy between the number of objects present and the number attempted. This is a result of the implementation of object classes one by one (Table 8 indicates classes unattempted with an "x"), since at the present time there is no program that attempts to identify all the object classes present in the relatively simple examples that we circulated. The program is seen to be more competent if the total number of objects correctly identified is divided by the total number attempted.

  Suppose that there are 100 objects in an example, 80 fall into classes that are attempted, and 60 are correctly identified. Whereas by Method 1 (total present/total identified) a score of 60% would be attained, by Method 2 (total attempted/total identified) a score of 75% would be attained. To the end user who is interested in a competent end result, the first measure gives a fairer picture.

- Systems do not all define objects in the same way. A "note" may be an indivisible unit in one system whereas in another a notehead and a notestem may be classed as two separate objects. Two evaluators scanning the same hypothetical example may find 100 objects if the composite note is considered to be the smallest unit of information and 125 objects if the graphical parts of the note (heads, stems, flags, *et al.*) are taken to be individual objects.

- Grande suggests also tabulating spurious objects falsely recognized and subtracting them from the total correctly identified before dividing by the total present.

The first consideration is compounded by the substitution of "friendly examples," that is, those lacking object classes that the program does not attempt to recognize, since there will be little discrepancy between the first and second methods of calculation. *SightReader*, for example, claims an accuracy of 94.2% in the Haydn example presented. This is based on 778 *SCORE* items in the file on which the reconstruction is based. Among these 45 required correction. The page in question contained the system shown in Example 8 and the preceding system, which is very comparable in musical detail. It

is clear that by defining objects in a graphical sense rather than a musical sense the numbers of items is elastic and could have been greater still [see pp. 135-6].

Another issue that complicates measures of performance is how the same items would be treated if encoded (*e.g.*, alphanumerically) as input to the program used to edit the example. While no system in common use requires separate representation of every staff line and many facilitate the creation of a template to set up systems, there is some open-endedness in most systems. The *DARMS*-based *Note Processor* would create the rehearsal number, for example, by assembling two right angles to form a box and placing a numeral within it (3 objects); *SCORE* in some instances requires insetting a second beam to create sixteenth-note groups from eighth-note groups. In notation programs there has always been a tradeoff between ease of use and degree of control over the result. In evaluating software for optical recognition it will be as important as ever to remember that intended use is the most important factor to consider in selecting a system.

Those whose sole objective is to acquire files (*e.g.*, MIDI) for sound applications can happily ignore the handling of beams and rehearsal numbers as well as system layout and the like. Reengineering of systems designed in the first instance for sound output to support complex notational output remains a difficult task.

Considering that the field of optical recognition is still in its infancy, it would be inappropriate to insist on common examples, since it is in the nature of scanning programs to perform best on material that resembles the material used for training. This will inevitably vary widely from program to program. As we demonstrated at the beginning, the graphic quality of materials is generally poorer than one would assume on a cursory glance, but there are many different kinds of typographical poverty. At a technical level, no two kinds may correspond in the problems they pose. Thus, good editing tools for passing the scanned result to an applications program become essential. The route to competent programs for sound output should be shorter than that to programs for notation, since less information needs to be conveyed.

The Future

Two Japanese researchers with substantial experience in the development of optical recognition, Hirokazu Kato and Seiji Inokuchi, assert that the goal of finding a single method for recognizing all musical symbols may be unrealistic. Their work uses a five-layer processing model that ascends from the pixel level through "primitives" (symbol elements such as flags and beams), complete symbols, and musical meaning (pitch and duration) to interpretations of the acquired information. This model implicitly suggests the degree to which the intellectual obstacles to accurate recognition arise from the complexity of the visual grammar of music.

An Annotated Bibliography
including Theses-in-Progress

Baumann, Stefan. "*DOREMIDI*: Document Recognition of Printed Scores and Transformation into MIDI." Master's thesis, German Research Center for Artificial Intelligence, 1992.

Available as a printed research report from the above Center, P.O. Box 2080, 67608 Kaiserslautern, Germany. Further information: Stefan Baumann, German Research Center for AI, Erwin-Schroedingerstr., 67663 Kaiserslautern, Germany; tel.: +49 631/205-3343; fax: 631/205-3210; e-mail: *baumann@dfki.uni-kl.de*.

Bainbridge, David. "Preliminary Experiments in Musical Score Recognition." Undergraduate project, Dept. of Computer Science, University of Edinburgh, 1991.

This author's work continues, with output to *CSound*, at the Dept. of Computing, University of Canterbury, Christchurch, New Zealand; e-mail: *dbain @cosc.canterbury.ac.nz*.

Baumann, Stefan. "Transforming Printed Piano Music into MIDI" in the *Proceedings of the Workshop on Syntactical and Structural Pattern Recognition, Sep.1992* (Bern).

Blostein, Dorothea, and Henry S. Baird. "A Critical Survey of Music Image Analysis" in *Structured Document Image Analysis*, ed. Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto (Berlin: Springer-Verlag, 1992), pp. 405-34.

Review of terminology, procedures (line adjacency graph, run-length histograms, template matching), symbol classification methods, and a useful list of references.

Blostein, Dorothea, and Nicholas P. Carter. "Recognition of Music Notation: SSPR '90 Working Group Report" in *Structured Document Image Analysis*, ed. Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto (Berlin: Springer-Verlag, 1992), pp. 373-4.

Report of a workshop on Syntactic and Structural Pattern Recognition held at Murray Hill, NJ on 13-15 June 1990.

Carter, Nicholas P., and Richard A. Bacon. "Automatic Recognition of Printed Music" in *Structured Document Image Analysis*, ed. Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto (Berlin: Springer-Verlag, 1992), pp. 456-65.

Reports progress in developing an object recognition system which is independent of font types and sizes, with scanned examples of keyboard passages from C. P. E. Bach [as shown in *CM 1991*] translated into *SCORE* code.

Carter, Nicholas P. "A New Edition of Walton's *Façade* Using Automatic Score Recognition" in *Advances in Structural and Syntactic Pattern Recognition*, ed. Horst Bunke (Singapore: World Scientific), pp. 352-62.

Describes problems encountered in replicating a printed score from the early twentieth century.

Carter, Nicholas P. "Segmentation and Preliminary Recognition of Madrigals Notated in White Mensural Notation," *Machine Vision and Applications* 5 (1992), 223-30.

Describes efforts to recognize printed notation used in the late sixteenth and early seventeenth centuries.

Choi, James. "Optical Recognition of the Printed Musical Score." M.S. thesis, Northwestern University, [1992]. Further information: *phantom@merle.acns.nwu.edu*.

Clarke, Alastair, Malcolm Brown, and Mike Thorne. "Problems to be faced by the Developers of Computer Based Automatic Music Recognisers," *Proceedings of the International Computer Music Conference 1990* (San Francisco: Computer Music Association, 1990), pp. 345-347.

Couasnon, Bertrand. "Segmentation et reconnaissance de partitions musicales guidées par une grammaire."

Couasnon's grammatical approach to recognition of printed scores involves a dual neural net/rule-base system approach to recognition of printed scores. Further information: IRISA, INSA - Dept. Informatique, 20, av. des Buttes de Coesmes, 35043 Rennes Cedex, France; tel.: +33 99/28-64-91; fax: 99/63-67-05; e-mail: *couasnon@irisa.fr*.

Fujinaga, Ichiro. "Optical Music Recognition using Projections." M.A. thesis, McGill University, 1988.

This project has been continued in doctoral research about OMR software scheduled for completion in 1994. Further information: Ichiro Fujinaga, Peabody Conservatory of Music, 1 E. Mt. Vernon Place, Baltimore MD, 21202; e-mail: *ich@music.mcgill.ca*.

Fujinaga, Ichiro, Bo Alphonce, and Bruce Pennycook. "Issues in the Design of an Optical Music Recognition System," *Proceedings of the International Computer Music Conference 1989* (San Francisco: CMA, 1989), pp. 113-6.

Fujinaga, Ichiro, Bo Alphonce, Bruce Pennycook and Natalie Boisvert. "Optical Recognition of Musical Notation by Computer," *Computers in Music Research* 1 (1989), 161-4.

Fujinaga, Ichiro, Bo Alphonce, Bruce Pennycook, and Glendon Diener. "Interactive Optical Music Recognition," *Proceedings of the International Computer Music Conference 1992* (San Jose: 1992), pp. 117-21.

This system developed at McGill University in Montreal runs on the NeXT workstation and outputs files to the public-domain *Nutation* editor developed by Diener. The program builds a database of previously identified symbols. Success is reported with both printed and manuscript sources.

Itagaki, Takabumi, Masayuki Isogai, Shuji Hashimoto, and Sadamu Ohteru. "Automatic Recognition of Several Types of Musical Notation" in *Structured Document Image Analysis*, ed. Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto (Berlin: Springer-Verlag, 1992), pp. 466-76.

Describes the use of *SMX [Standard Musical Expression]* as an intermediate output format for scanned notation to be routed to diverse end-uses including printing scores, Braille scores, and electronic performance. The test repertory consisted of piano works including a Chopin étude and the Fantaisie-impromptu. Experiments with Braille input and recognition of Labanotation were also conducted.

Kato, Hirokazu, and Seiji Inokuchi. "A Recognition System for Printed Piano Music using Musical Knowledge and Constraints" in *Structured Document Image Analysis*, ed. Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto (Berlin: Springer-Verlag, 1992), pp. 435-55.

The authors describe their five-tiered approach to recognition and offer accuracy figures for four piano works scanned ranging from 83% (the second movement of Beethoven's Sonata "Pathétique") to 96% (Beethoven's "Für Elise").

Leplumey, Ivan, and Jean Camillerapp. "Comparison of Region Labelling [in the] Musical Score," *Proceedings of the First International Conference on Document Analysis* (St. Malo, 1991), pp. 674-82. Also in French in *AFRCET: 8e Congrès Reconnaissance des Formes et Intelligence Artificielle* (Lyon: Villeurbanne, 1991), 3, 1045-1052.

Concentrates on segmentation algorithms to facilitate score recognition.

Leroy, Annick, and Bertrand Couasnon. "Editeur plume multifonction: application a un editeur plume musical" ("Multipurpose Pen Editor: Application to a Pen-based Music Editor").

A pen-based music editor is under development in this thesis-in-progress. Further information: Poste 4204 IRISA, INSA - Dept. Informatique. 35043 Rennes Cedex, France; tel.: +33 99/28-64-00; e-mail: *couasnon@irisa.fr.*

Ng, Kia-Chuan, and Roger D. Boyle. "Segmentation of Music Primitives," *Proceedings of the British Machine Vision Conference 1992*, pp. 472-80.

A fuller explanation of the approach described herein.

Pennycook, Bruce, "Towards Advanced Optical Music Recognition," *Advanced Imaging* (April, 1990).

Perrotti, F. A., and R. A. Lotufo. "Pré-processamento, Extração de Atributos e Primeiro Nível de Classificação para un Sistema de Reconhecimento Ótico de Símbolos Musicais" ["Preprocessing, Feature Extraction, and First Classification Level for an Optical Recognition System"] in *VI Brazilian Symposium in Computer Graphics and Image Processing, SIBGRAPI. 19-22 October 1993.* Recife (Brazil), 1993.

A solution for the preprocessing, feature extraction, and first level of classification in optical music recognition is presented. The attributes are extracted from a thinned and vectorized image of the score. Critical points and their supporting environs are matched and clustered to identify candidate symbols. Experimental results are supported.

Further information: *perrotti@dca.fee.unicamp.br* and *lotufo@dca.fee.unicamp.br.*

Prerau, David. "Computer Pattern Recognition of Standard Engraved Music Notation." Ph.D. dissertation, Massachusetts Institute of Technology, 1970.

Although the amount of music the author was able to scan was extremely limited, the thought he applied to the task has remained fundamental in most projects originating in recent years.

Roth, Martin. "Optical Music Recognition Bibliography." Electronic text. Zurich: ETH, December 1992.

A recent compilation incorporating references from earlier bibliographies by Alastair Clarke, Henry Baird, Dorothea Blostein, and Karl Tombre. Readers are welcome to supply additions by sending them to *roth@ips.id.ethz.ch.* The material is available by FTP to *maggia.ethz.ch* (129.132.17.1), login *ftp*, directory */pub/roth/omrbib.*

Sonka, Milan, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision.* London: Chapman and Hall, 1993.

See the contribution on *AMSR*.

Wolman, Amnon, James Choi, Shahab Asgharzadeh, and Jason Kahana. "Recognition of Handwritten Music Notation," *Proceedings of the International Computer Music Conference 1992* (San Jose: 1992), pp. 125-7.

Describes research based on the premise that while a program which reads handwritten music could probably read printed music, the reverse is not true. Users are required to write music on commercially available printed score paper.

Yadid, Orly, Eliyahu Brutman, Lior Dvir, Moti Gerner, and Uri Shimony. "*RAMIT*: [A] Neural Network for Recognition of Musical Notes," *Proceedings of the International Computer Music Conference 1992* (San Jose: 1992), pp. 128-31.

Describes the Neocognitron, a neural network model for visual recognition of black and white patterns.

Note: An extensive bibliography of writings on optical recognition is maintained by Martin Roth. It is accessible by FTP to *maggia.ethz.ch* [129.132.17.1] in the directory */pub/roth/omrbib*. Both compressed and uncompressed files are available.

MusicReader:
An Interactive Optical Music Recognition System

The *MusicReader* is an interactive one-pass optical music recognition system. It can capture information from a printed score and translate it to both a *DARMS* file (for printing) and a MIDI file (for playback) in a reasonable time. The system, which is implemented on a 640k 386/486 personal computer, supports interactive software requiring the use of a mouse, computer keyboard, and color display. We have adopted the basic approach developed by David Prerau for his dissertation at the Massachusetts Institute of Technology in 1970. This approach follows scanning with the removal of staff lines, the identification of connected pixel components, the classification of resulting musical entities, and the production of *DARMS* code as output.

The heart of our system is the one-pass classifier, which includes a robust staffline remover, and an integrated beam and chord analyzer. In its current state of development, our technology can facilitate computer-aided analysis, reorchestration, and (re)printing. *MusicReader*'s performance is best on monophonic sources in standard music notation (such as fake books and part scores), moderate on simpler piano scores, and fair on more complex music.

The review of the field made recently by Dorothea Blostein and Henry Baird makes it unnecessary to discuss previous work. In our view an understanding of the work of Prerau and Nicholas Carter, whose work began in the late Eighties at the University of Surrey, England, is essential to progress in this field.

The Music Reader System

The following five sets of computer data, discussed below, are used:

1. A pixel representation of the musical score
2. A file containing the classified connected components with horizontal and vertical placement
3. A *DARMS* representation
4. A Brinkman score representation
5. A Standard MIDI file

Interactive routines are used in the scanning process (to generate the pixel representation), in the classification process, and, normally, in the editing of the *DARMS* text file. Each routine is described in detail below.

We have found that with present computer technology, the best results are obtained through the analysis of connected staves representing several measures. Backtracking is not used in the classifier; this results in advantages and some minor disadvantages.

1. Scanner Program

A considerable portion of time is spent in the output of the scanning and input of the classification process: the files are large. A 2-inch segment of an 8-1/2 by 11-inch page takes 2 x 300 x 8.5 x 300 pixels, *i.e.*, 1.6 megapixels. Data compression techniques such as those used for facsimile transmission can reduce the size considerably, but these require coding and decoding. Since the pixels are thrown away anyway, we have opted for an intermediate position and use run encoding of the black pixels. The use of 16-bit integers is all that is required. Typical file lengths are 100 to 200 kilobytes. Therefore we have a file consisting, for each scanned line, of the beginning and end of each run of black pixels. An end-of-line marker (-1) is used to indicate the end of a line.

For staff-line identification purposes, the image must be a rotated one, as if the scanning had gone from left to right, rather than from top to bottom. This can be done in two ways: turning the page around, or doing a software rotation. The difficulty with a software rotation is that it requires a large memory for the required pointer arithmetic, limiting the number of staffs that may be readily scanned to five or so.

In the course of the project, we have developed utilities that will translate TIFF files and unrotated run-encoded files. A utility to remove short white runs is sometimes of use. For the scanning process itself, a series of tests indicates that on our scanner, a Hewlett Packard Scanjet, dark images without automatic threshold control enhance the classification process.

A preliminary scan at 6:1 compression (*i.e.*, 50 dots per inch) is used to reproduce a crude image on the monitor screen. The operator then uses a mouse to click the upper left and lower right of a rectangle enclosing the staffs to be analyzed at once. When all the rectangles have been identified, the scanning itself can proceed. The final scan is at 300 dots per inch and results in a rotated run-length encoded file.

2. Classifier Program

Most of the time is spent on classification. Whereas the input is a run-encoded rotated image of a few staffs, the output is a text file with the following format items:

- the staff number
- left side of component (in pixels)
- right side of component (in pixels)
- vertical position of component using *DARMS* representation
- a component identifier, as close to *DARMS* code as feasible

The classifier program performs many functions, but the main three, discussed below, are these:

A. Staffline identification and removal
B. Component identification and classification
C. Beam/chord classification

A. STAFFLINE REMOVAL

While in many kinds of image analysis the isolation of components can proceed in a straightforward way, in musical scores the stafflines connect many objects and render them interpretable. For this reason, we concentrate on components with some black pixels between the stafflines. The identification of the stafflines is also essential for the *DARMS* encoding of the components [*DARMS* specifies vertical heights rather than pitches]. We find this to be the most frustrating part of optical music analysis. We err on the side of excess by dropping parts that are common both to stafflines and other objects. This is a tradeoff which is forced by our design decision to use a one-pass classifier, a restriction placed upon us by our meager hardware memory resources. The classifier sweeps along the staff, removing stafflines. As connected components are identified, they are classified, written to the output file, and released from memory.

Our staffline removal algorithm basically performs a correlation of the current line and the next line in the left-to-right scan, and if a long thin element is near a current staffline, it is associated with the staffline, and the position of the staffline is adjusted. We assume only that stafflines belonging to a common staff are parallel and that the total number of stafflines is a multiple of five. These staffline-like elements are shown in light grey on the screen, and the other pixels are shown in white. When the staffline elements are terminated (usually because they bump into components such as barlines, etc.) they are removed from memory.

Although the staffline removal algorithm is rugged, the starting position is, of course, important. The program initially knows neither the number nor the position of the stafflines. However, when it is determined that the number of short runs is a multiple of five, the program stops, beeps, and displays what it thinks are the stafflines. The operator may agree and save the information or disagree and press a key that gives the number of staffs (currently an integer from 1 to 9). In this second case, the program is told the position of the first and last of the five stafflines of each staff. Occasionally initial clefs may be accidentally omitted. The effect is small relative to the difficulty encountered in accurately identifying C clefs, which have a large number of different presentations.

B. COMPONENT ANALYSIS

A connected component analysis is carried out simultaneously with staffline removal. If a connected component—a set of pixels, all of which are connected—exceeding a size and position threshold is found, it is displayed in green on the screen. The tentative *DARMS* classification, including vertical position, is presented, again in green. The operator indicates whether to save, change, analyze (for beams and/or chords), or delete the component. In the case of a change, the prompt turns from green to red, and the operator simply types in the *DARMS* representation.

In the beam/chord analyzer, the component is rotated in memory, and the stemlines are identified much as the stafflines are identified. These stemlines are displayed in yellow, and the *DARMS* code associated with each stemline, which is based on a connected component analysis made when the stemlines are removed, is presented to the operator, who may save, change, or discard them. The remaining connected components are classified as beams or notes, and this determines the *DARMS* representation. A *DARMS* comment is applied to each of these components, indicating that they are all part of the same beamed or chorded component. In addition, the stem direction, the note duration, and the beaming are indicated. Since most notation programs will perform beaming automatically, and beaming is unnecessary for MIDI files, there is some redundancy in this information. We find that the beaming information is useful, however, in the editing of the *DARMS* code.

Since several measures cannot fit in one display, whenever a barline is identified, the screen is refreshed, moving that barline to the left side of the screen.

The classification of the components uses features which include and extend those presented by Prerau: the size of the bounding rectangle, the relative position of the top and bottom pixels (used to discriminate sharps, flats and naturals), the width at the middle in a vertical and horizontal position, the density of the component compared to the bounding rectangle, and the number of holes. The features are normalized to the staffline spacing.

C. FEATURE DATA

The features are stored by giving upper and lower bounds for the position features. A distance measure based on deviance from these bounds, the number of holes, and the density is used to classify the component. This information may be used to fine-tune the feature table. It may be customized for a particular musical source or class of printing.

Special utilities operate on a subsidiary output file which contains the features for all the components. The utility analyzes all of these files appearing in a subdirectory, prints out the maximum and minimum values for the features as well as the mean and standard deviations, and displays them on the screen. This information may be used to update the feature data.

3. Sort/Syntax Program

The output from the classifier is not in order. What is desired is a sort by staff, and then by position in the staff. The comments related to beam/chord number are stripped at this stage. Certain syntax simplifications are made. Notes appearing at the same time are (optionally) clustered to form a chord. The notes appearing at the same time are ordered from long duration to short. Dots appearing after an F-clef are dropped, double dots before a barline are merged into a repeat sign, dots above notes form staccato marks, and dots after notes are used to lengthen durations. Sharps and flats appearing at the beginning, if they are in the proper sequence, form a key signature.

Logical problems with the handling of beamed notes within *DARMS* require the use of linear decomposition, the creation of separate tracks of information to accurately represent nonhomophonic textures (for example the presentation of soprano and alto parts on a single staff). We have found it difficult to automate linear decomposition, even though we do retain stem directions. It seems preferable to drop the beaming information completely and restore it with the notation program.

4. *DARMS* Interpreter

The *DARMS* interpreter provided by Brinkman (1990) is used to assign absolute time values to all *DARMS* events. The *MusicReader* interpreter also assigns note volumes intelligently. The output may be displayed and analyzed. The interpreter is perhaps most useful in checking the syntax of the *DARMS* code.

5. MIDI Filemaker

A translation to Standard MIDI files enables the data to be used with most sequencers and notation programs. The MIDI files are also valuable for data verification.

The *DARMS* interpretation and MIDI file conversion utilities are supported by an assembly language editor (available as shareware) in which error messages are displayed with the lines of text in the editor, facilitating correction.

Program Performance

Program performance varies with the source material. Generally speaking, the scanning of single-line music with quarter notes or eighth notes, beamed sixteenth and thirty-second notes, and unbeamed chords is flawless. Individual sixteenth- and thirty-second notes are harder to identify because of the tails. Rests that occur in staves containing only one part are usually recognized; they can be troublesome in multipart music on one staff. Because of the ambiguous nature of white space, half notes and whole notes are difficult; they are currently recognized about 70 percent of the time. Either the staffline goes though them, and they look like quarter notes or quarter noteheads, or they are between stafflines and they become split into two components.

Clef recognition is difficult. The C-clef is most problematical, followed by the F-clef; the G-clef is usually recognized. Finally, beam/chord components with a multiplicity of stemlines going to one component are often misclassified, usually by presenting too many elements on each stem.

Despite these limitations, many of which are due to the poor quality of the starting images, we feel that such an analysis is always more efficient than encoding a work in *DARMS* alphanumerically, especially since the manually encoded *DARMS* file will need to be checked anyway.

Some deficiencies in the current program are inherent in the approach, while others are readily remedied. The problem of fragmentation of components owes to the use of the one-pass staffline removal algorithm. Its importance varies with the repertory. It would be a significant obstacle to a repertory containing a preponderance of half and whole notes, such as hymns. The problem of multiple stems attached to a single notehead is one of music representation. It should be capable of solution, perhaps using graph-based methods. The current program does not recognize text, but we are implementing a template-matching algorithm that may help. The text algorithm may be useful for the classifier as well.

Hardware enhancements would include porting to other popular workstations, such as the Macintosh, NeXT, and Sun SparcStation, and testing with hand-held scanners.

The system is now available for use, and user feedback would be much appreciated.

References

Brinkman, Alexander. *Pascal Programming for Music Research.* Chicago: University of Chicago Press, 1990.

Dydo, Stephen. "Data Structures in 'The NoteProcessor'," *Proceedings of the International Computer Music Conference, 1977* (Urbana: University of Illinois, 1977), pp. 311-16.

Merkley, Paul, and William F. McGee. "Optical Scanning of Music," *Computers and the Humanities*, May 1991.

Prerau, David. "*DO-RE-MI*: A Program that Recognizes Music Notation," *Computers and the Humanities*, 9/1 (1975), 25-29.

Further Information: William F. McGee, Nepean DSP Services, 73 Crystal Beach Drive, Nepean, Ontario, Canada K2H 5N3; tel.: (613) 828-9130; e-mail: *mcgee@citr.ee.mcgill.ca*. Paul Merkley, Department of Music, University of Ottawa, Ottawa, Canada K1N 6N5; tel.: (613) 564-9239; fax: (613) 564-5643; e-mail: *merkley@acadvm1.uottawa.ca*.

Music Score Recognition: Problems and Prospects

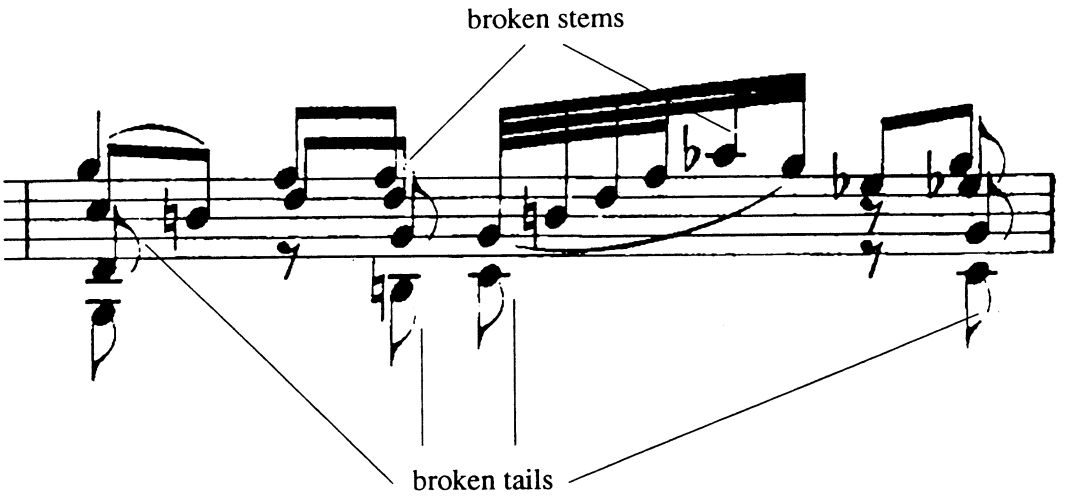
The problem of recognition of music scores has much in common with the processing of other types of documents (mechanical engineering drawings, circuit diagrams, maps, and texts containing conventional English characters, Chinese and Japanese characters, and mixed alphanumerics and graphics [Yamamoto 1993]). Not only are they all binary (black and white) images but they also contain a mixture of line segments, characters, and domain-specific symbols. Current thinking in the document recognition community seems to agree on several key points, namely that no one recognition technique is going to solve the recognition problem for cases such as those listed above, that context will play a significant role in any successful recognition system, and that it would be useful to have not only databases of typical documents but also appropriate models of image degradation for use in system development and testing. It is interesting to note that some work on seemingly black and white originals makes use of the grey-scale image in order to direct the thresholding process and thereby create a better approximation to the symbol shapes in the image (Roach 1988, Yamamoto 1993).

While line-finding is a fundamental process in the graphic analysis of many categories of binary image, music is unique in making use of sets of five roughly horizontal, parallel, and equidistant lines as its reference grid. Another shared problem is that of touching symbols. Text annotations of circuit diagrams and engineering drawings sometimes touch the objects to which they refer; maps frequently contain touching or intersecting symbols. Music could be said to have this characteristic but to a greater degree because most symbols will touch or be superimposed upon the stafflines and, even after symbol isolation (staffline recognition), symbols which touch or are superimposed will probably remain.

It should be remembered that the first work in the field, undertaken in the late 1960s and early 1970s, was severely limited by hardware (Pruslin 1967, Prerau 1970). Now that computers commonly have sufficient storage capacity to handle large quantities of data (an 8.5" x 11" page digitized at 300 dots per inch [d.p.i.] requires about 1 MByte of storage) and enough processing power to undertake recognition tasks involving such quantities of information, music score recognition has become purely a software problem. Scanners capable of producing binary images at 300 or more d.p.i. and sophisticated music notation programs are now widely available and these provide the other ingredients necessary for the development of an integrated music score recognition system.

General Problems in Music Recognition

The music fragment shown below will be used to illustrate some of the significant problems involved in attempting to develop algorithms which can read music notation. It is important to emphasize that such algorithms need to be generally applicable in order to be truly useful. It is easy to make assumptions about scores early on in the development phase which turn out to be untrue upon examination of a wider range of repertories. This is not purely a criticism of music score recognition research but seems applicable to work on the other forms of binary image (circuit diagrams, etc.) recognition, where assumptions are sometimes made in modelling the structure of symbols which turn out to be inadequate when faced with the degraded images which exist in the real world. This is particularly the case in industry, where the automatic conversion into electronic form of old and deteriorating documents is often of particular interest precisely because the paper-based originals are of poor quality and are therefore reaching the end of their useful life.



Example 1. J. S. Bach: an extract from the *Sonatas and Partitas for Unaccompanied Violin* (Dover Edition; after original engraving of 1894 by Breitkopf und Härtel).

Some of the basic requirements of a score recognition system include coping with variations in the following image parameters:

1. The **size** of original. Pocket scores may not have to be read by an automatic system but a reasonable requirement would encompass sparse instrumental parts with 6 or 8 staves per page, up to full orchestral scores. Staff height may vary between, say, 6 and 12mm.

2. The **rotation** of the original (the stafflines may be straight but not horizontal).

3. The **bowing** of stafflines. This is sometimes present in the original due to the printing process but may be introduced by the use of a hand scanner to acquire the image or by an intermediate photocopying stage.

4. The presence of **noise** in the image (either black specks, often referred to as "salt and pepper" noise, or dropouts, *i.e.*, white patches in symbols which should be solid black as well as slight variations along supposedly straight edges due to quantization).

5. **Discontinuities** such as breaks in stafflines or notestems (see Example 1).

6. Digitizing **resolution** (commonly 300 d.p.i.).

7. The music symbol **font**.

To put these requirements in context, it should be noted that omnifont text recognition in a range of sizes, in particular with touching characters, is still a current research topic (Yamamoto 1993). In comparison, score recognition requires coping not only with a range of music sizes, fonts, and distortions but also with the superimposition of symbols, not to mention the additional need for a built-in omnifont, size-independent character recognition facility which understands text underlay. Also, the above list does not include what is probably the most difficult problem in score recognition, *i.e.*, dealing with superimposed symbols, although it is also a complex process to extract implied information after symbol recognition has been achieved. This is amply illustrated by Example 1 with its varying number of voices, superimposed beamed groups, and staggered simultaneities. A more detailed discussion of the above parameters follows.

Specific Problems

1. Size

In order to process images of varying size, all measurements which are made on the image need to be relative rather than absolute. This is commonly achieved in current systems by normalizing all dimensions with respect to the vertical distance between adjacent stafflines. The other significant issue relating to size is the loss of symbol detail which occurs in some small scores. This leads to problems for the recognition engine which are similar to those produced by some instances of noise, poor-quality original pages where symbols are fragmented, or low resolution digitizing.

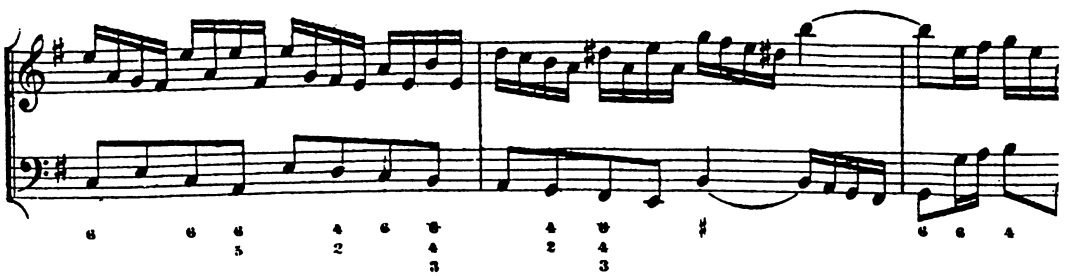
The use of multiple recognition modules, at least one of which would use context, is a possible solution in order to produce a "best fit" recognition result for an input image. It would also seem promising to undertake iterative recognition processing, perhaps in conjunction with a cost-function to direct results towards such a "best fit," albeit somewhat daunting to implement. This bears similarity to the annealing process for automatic layout of printed music described by Byrd (1980).

2. Image Rotation

It is reasonable to assume that the original will only be rotated slightly, at most, especially if a sheet-fed scanner has been used, but it is important to realize that only slight rotation is needed in order to render useless those staffline-finding algorithms which rely on finding horizontal lines. Also, some rotation may be present in the original image, so regardless of positioning relative to the scanning element, rotation is still a factor. The technique for staffline-finding must therefore include pre-processing to rotate the page, where necessary, or otherwise take account of rotation in dealing with line fragments. This issue is linked to the following parameter because stafflines cannot be assumed to be straight either (whether horizontal or not), so the staffline-finding process is forced to use a piecewise technique which permits bowed line fragments.

3. Staffline Distortion (Bowing)

Curvature in music score images, particularly of stafflines, is a source of difficulty for algorithms which assume that lines are straight. The human vision system has little difficulty in accommodating such distortions, so the widespread nature of the problem is perhaps not widely realized. Bowing can also be introduced by the scanning process, for instance if a hand scanner is used (see Example 2), or by photocopying the original prior to scanning.



Example 2. J. S. Bach: extract from the Flute Sonata in E Minor (Bach Gesellschaft; acquisition by hand scanner at 300 d.p.i.).

4. Noise (Dirt)

Noise in the input image takes various forms. Common examples are the black specks which appear in white background areas of a score due to the printing process or variations in the reflectivity of the paper. These need to be ignored by the recognition system but care must be exercised because similar-sized black regions may, depending on the position on the page, constitute a faint duration dot or be the sole difference between an 'l' and a 't' in a string of text. Similarly, white specks may appear within a region intended to be solid black. These alter the structure of the region and also reduce the effectiveness of a template-matching approach. Another effect which can be viewed as a form of noise is illustrated in Example 2, where lines vary in thickness. This is common, sometimes due to the quality of the original but often simply to the quantization process.

5. Discontinuities

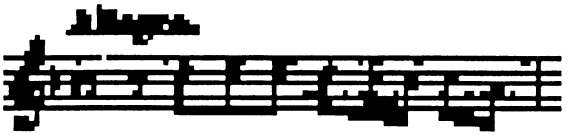
This is a specific form of noise which manifests itself in the form of linebreaks (stafflines, stems, barlines, tails, etc. can all be affected) as illustrated in Example 1. This makes line-tracking, or other similar techniques which rely on line continuity, difficult and necessitates provision for traversing such gaps. It can also make the detection of line fragments more difficult if these are required to be unbroken.

6. Resolution

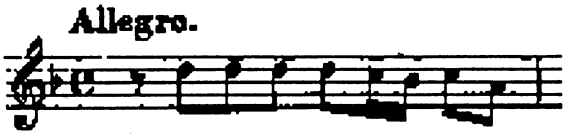
It is interesting to examine the amount of information which the human vision system is able to extract even from a low resolution image. The following illustrations show resolutions of 37.5, 75, and 150 d.p.i. with a progressive factor of four increase in the amount of data required to store each image.

It can be seen that the stafflines are readily identifiable even at the lowest resolution. A treble clef and a one-flat key signature followed by some beamed groups are also apparent, although a degree of uncertainty exists over most of the pitches. By the use of context, an eighth-note rest can be identified preceding the first beamed group.

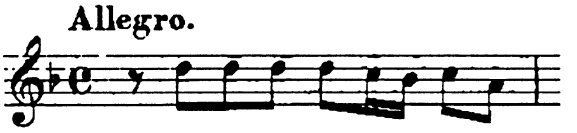
As soon as the resolution is increased to 75 d.p.i., however, the extract can be read accurately, and 150 d.p.i. seems perfectly adequate for recognition purposes (thus making 300 d.p.i. appear excessive, particularly in view of the fact that it requires 16 times the quantity of data used by the 75 d.p.i. image).



37.5 dots per inch resolution



75 dots per inch resolution



150 dots per inch resolution

7. Font Variation

The wide variation in music symbols due to different fonts should be apparent from a typical selection of material from assorted publishers (Fujinaga 1988, p. 78). A score recognition system has to make its decision as to which symbol is present based on appropriate font-independent parameters. For instance a full-size treble clef will normally have a loop extending above the staff and a tail below the staff regardless of its slope, stroke width or height. Similarly, a bass clef will have two dots on either side of the F-line regardless of the ornateness of its main body, although detecting these dots may not be a trivial task.

Conclusion

In summary, some of the parameters which make music score recognition such a difficult problem have been presented. Progress has been made in coping with some of these variables, but current systems are still limited in scope. Also, it is still an open question how much information should be included in the output data file. This depends on whether it is just the musical content of the original which is being extracted or if it is also important to retain appearance, *i.e.*, the precise placement and style of symbols.

References

Blostein, Dorothea, and Henry S. Baird. "A Critical Survey of Music Image Analysis" in *Structured Document Image Analysis*, eds. Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto (Berlin: Springer-Verlag, 1992), pp. 405-34.

Byrd, Donald. "Music Notation by Computer." Ph.D. dissertation, Indiana University, 1980.

Carter, Nicholas P. "Automatic Recognition of Printed Music in the Context of Electronic Publishing." Ph.D. dissertation, University of Surrey, 1989.

Carter, Nicholas P. "A New Edition of Walton's *Facade* Using Automatic Score Recognition" in *Advances in Structural and Syntactic Pattern Recognition*, ed. Horst Bunke (Singapore: World Scientific, 1992), pp. 352-362.

Carter, Nicholas P., and Roger A. Bacon. "Automatic Recognition of Printed Music" in *Structured Document Image Analysis*, eds. Henry S. Baird, Horst Bunke and Kazuhiko Yamamoto (Berlin: Springer-Verlag, 1992), pp. 456-65.

Fujinaga, Ichiro. "Optical Music Recognition Using Projections." M.A. thesis, McGill University, Montréal, 1988.

Prerau, David S. "Computer Pattern Recognition of Standard Engraved Music Notation." Ph.D. dissertation, Massachusetts Institute of Technology, 1970.

Pruslin, D. H. "Automatic Recognition of Sheet Music." D.Sc. dissertation, Massachusetts Institute of Technology, 1967.

Roach, J. W., and J. E. Tatum. "Using Domain Knowledge in Low-level Visual Processing to Interpret Handwritten Music: An Experiment," *Pattern Recognition*, 21/1 (1988), 33-44.

Yamamoto, K., ed. *Proceedings of the Second International Conference on Document Analysis and Recognition (ICDAR '93), Tsukuba Science City, Japan. October 20th-22nd.* IEEE Computer Society Press, 1993.

Further information: Nicholas P. Carter, Department of Physics, University of Surrey, Guildford, Surrey GU2 5XH, England, UK; tel.: +44 483/300800; fax: 483/300803; e-mail: *N.Carter@ph.surrey.ac.uk*.

How Practical is Optical Music Recognition as an Input Method?

Over the academic year 1992-3, Nicholas Carter, a leading researcher in the field of optical recognition of printed music, was in residence at the Center for Computer Assisted Research in the Humanities. Since the Center has been at work for almost ten years in the development of electronic representations of musical works, the circumstances were conducive to running a fairly well controlled test of the efficiencies of optical input vis-à-vis the CCARH standard input. This test was run in December 1993.

The Two Approaches

Carter's recognition program is designed to work with *SCORE* notation software. That is, after the score is converted to a bitmapped image, the recognition software attempts to identify objects and arrange these into a *SCORE* input file. The file is then converted to a *SCORE* parameter file, from which a musical score is printed. Errors are then corrected and the score is reprinted.

The CCARH system (*MuseData*; see pp. 11-28) involves realtime entry of pitch and duration through an electronic keyboard, syntax checking of this input, alphanumeric entry of information not enterable from a musical keyboard (articulation, dynamics, text underlay, grace notes and ornaments, basso continuo figuration, etc.), and proofreading, editing, and reprinting of printed scores. Proof-hearing also plays a major role in the Center's data verification process, but since no such operation was included in the optical recognition test, we discounted it in quantifying the performance of each approach.

Although in the test that was run a common goal—the replication of a single Haydn symphony—was pursued, there is one overall difference between the systems that could have a dramatic effect on more extensive comparison. *SCORE* is designed solely to capture and replicate an existing score. As an input process in its own right, it is not intended to facilitate assembly of a score from parts or other more varied approaches to acquisition. *MuseData* software can work from a score or from single parts or even from quite disheveled manuscript sources to a representation that supports not only notation but also sound output and certain kinds of analytical pursuits. The combination of *SightReader* plus *SCORE* has been aimed from the beginning at republication of preprinted material, whereas *MuseData* has from the beginning been aimed at the storage of musical information in a format hospitable to many and varied uses. Since files are organized by system and page in *SCORE*, the original format is replicated exactly in the recreation. This is not generally true for *MuseData*, which is organized by part and movement. Layout can, however, be altered in both systems.

The Music

The music on which this test was run was the Breitkopf und Härtel edition (1907) of Haydn's Symphony No. 1 (1759). The work was selected and photocopies for both contestants were made by Dr. Carter.

The work is generally laid out on three systems of six staves each to a page. There are six parts:

(1) Oboes 1 and 2
(2) Horns in D 1 and 2
(3) First violin
(4) Second violin
(5) Viola
(6) Violoncello and bass

There are three movements:

1. A *Presto* in cut time	5 pages
2. An *Andante* in 2/4	2.5 pages
3. A *Finale* (also marked *Presto*) in 3/8	2.5 pages

The oboes and horns are omitted in the second movement, which is laid out with five four-stave systems to a page. Page 8 contains two four-stave systems (end of the *Andante*) and two six-stave systems (start of the *Finale*). The second and third movements are in binary form. The outer movements are in D Major, while the *Andante* is in G Major.

Although the work is one of the musically simplest orchestral pieces by Haydn, it is not free of notational complexities. These include triple stops, abbreviation signs, grace notes, triplets, and trill signs in the string parts; polyphonically differentiated passages on the oboe staff; single notes with opposing stems on the horn staff; and so forth. Dynamics indicators, staccatos, ties, slurs, rehearsal numbers, and repeat signs are also present. In the main, however, the work consists principally of eighth and sixteenth notes, with some quarter, half, and whole notes and their corresponding rests.

The graphic images, while exhibiting many of the phenomena cited in preceding articles, were of generally good quality in that they are clear, with good contrast of black and white.

Quantitative Results

Because the unit of measure in *SCORE* is the page while in *MuseData* it is the movement, direct comparison of elapsed time was not immediately possible. We converted the *SightReader* page times to movement times by making the following

assignments: times for pp. 1-5 = Movement 1 (*Presto*); times for pp. 6, 7, and half of p. 8 = Movement 2 (*Andante*); times for half of p. 8 plus pp. 9 and 10 = Movement 3 (*Finale*).

As an input method, *SightReader* has only two stages—(1) image acquisition (TIFF) and conversion to a *SCORE* input file and (2) screen editing of the provisional score. The *MuseData* acquisition process is considerably more involved and requires a total of 10 steps—three (Stage 1) devoted to pitch-and-duration acquisition and syntax checking and seven (Stage 2) devoted to merging parts, printing a draft score, correcting it, and reprinting it. Some of these steps are very short and their times are reported only in summary fashion.

The following respective timings (reported in hours, minutes, and seconds) were reported:

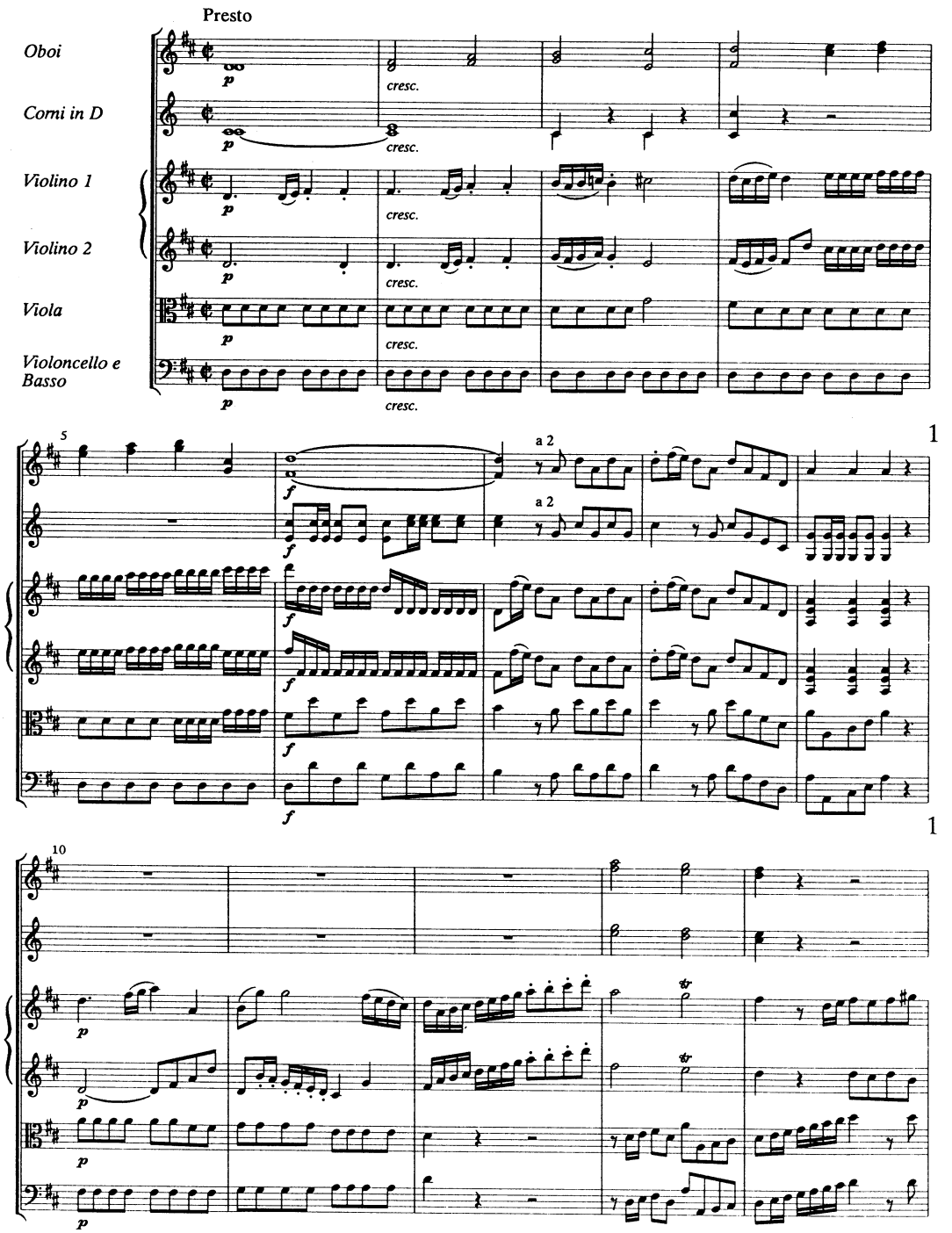
Movement	TIFF --> SCORE	Hand-editing	Total
Presto	0:05:49	5:46:00	5:51:49
Andante	0:04:55	2:24:05	2:29:00
Finale	0:02:24	1:34:05	1:36:29
Total	0:13:08	9:44:10	10:07:18

Table 1. Input (*SightReader*) and editing (*SCORE*) for the Haydn Symphony No. 1.

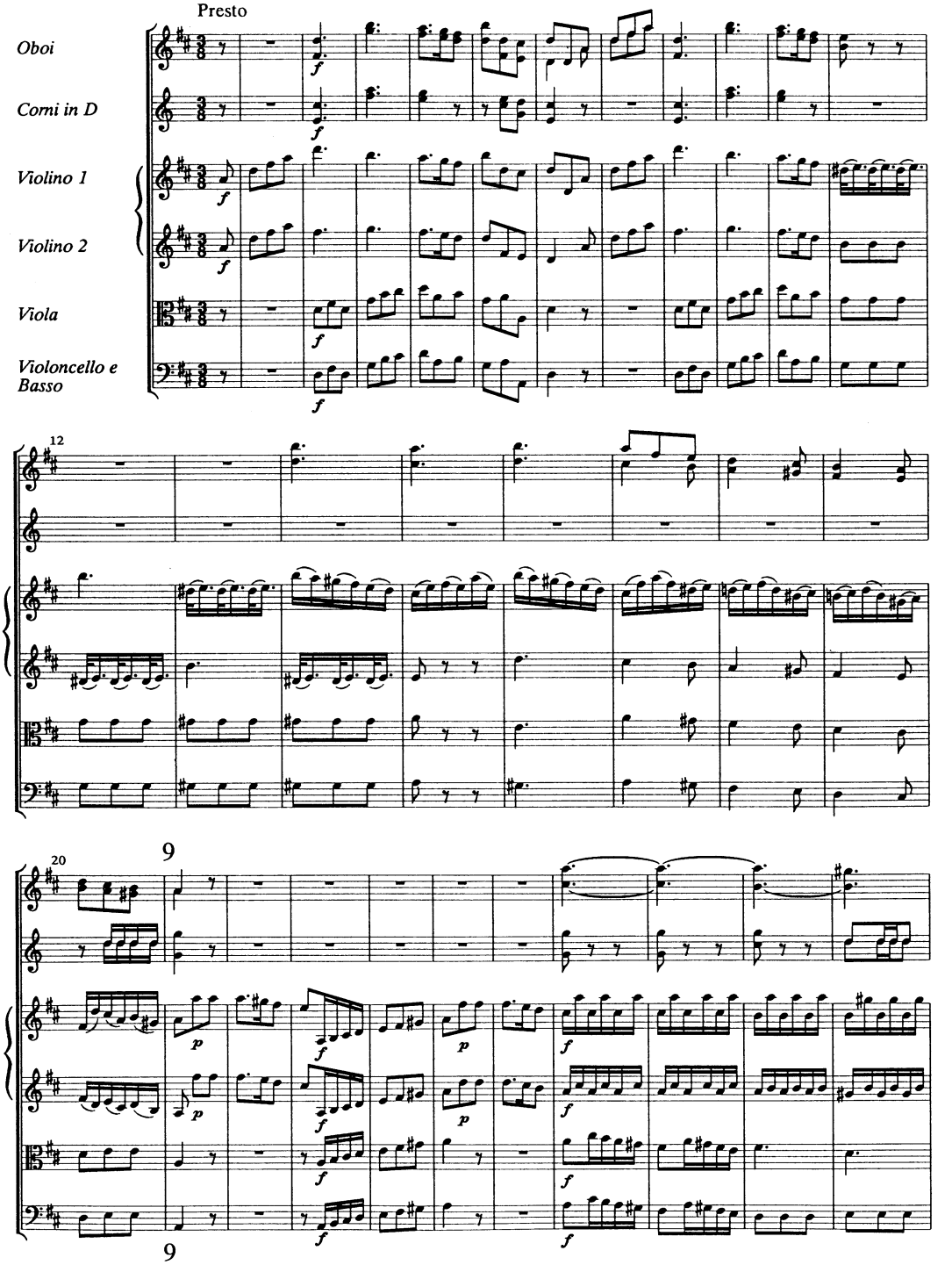
Movement	Stage 1	Stage 2	Total
Presto	1:55:00	4:24:00	06:19:00
Andante	1:20:00	2:52:00	04:12:00
Finale	1:05:00	2:04:00	03:09:00
Total	4:20:00	9:20:00	13:40:00

Table 2. *MuseData* input (Stage 1) and editing (Stage 2) times for the Haydn Symphony No. 1.

When the *MuseData* figures are consolidated to resemble the two steps in the *SightReader/SCORE* process (Table 3), then it becomes apparent that the *MuseData* editing time is marginally faster, but the original acquisition time is inevitably much slower, since it represents realtime performance, part by part, movement by movement.



Example 1. Beginning of the first movement of Haydn's Symphony No. 1 (*MuseData* reconstruction).



Example 2. Beginning of the third movement of Haydn's Symphony No. 1 (*MuseData* reconstruction).

System	Input	Editing	Total
Sight/SCORE	0:13:08	9:44:10	10:07:18
MuseData	4:20:00	9:20:00	13:40:00
Difference	4:06:52	0:24:10	03:32:42

Table 3. Comparison of *SightReader/SCORE* and *MuseData* systems of input.

It appears that efficient recognition is powerless to speed editing. One question that prospective users of recognition software must assess for themselves is whether they are prepared to make the same commitment of time to the editing process after optical music recognition as they do without it.

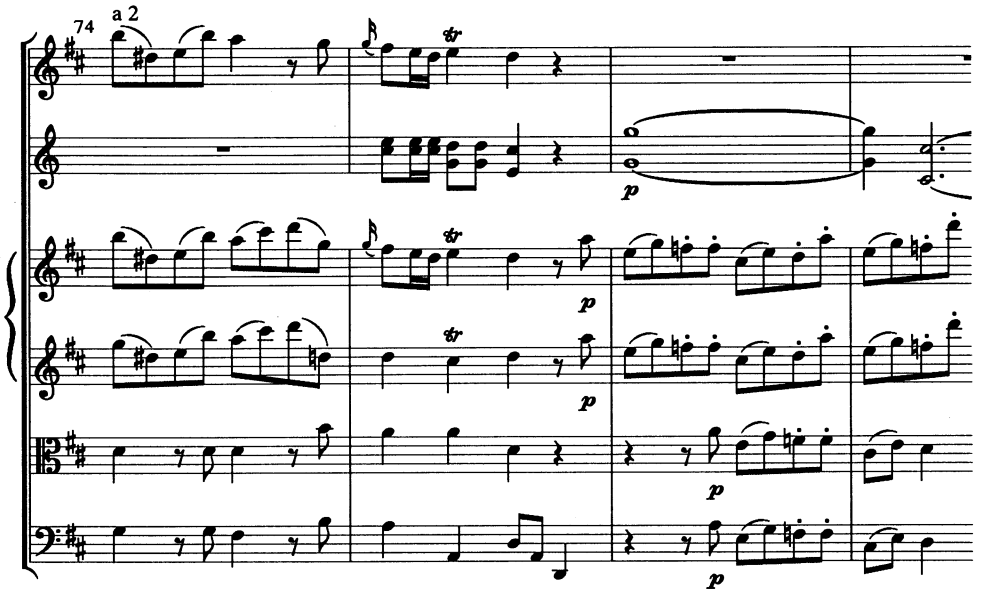
Qualitative Results

Judging the aesthetic quality of the output is entirely subjective. *SightReader* output resembles the materials shown on pp. 136-7. We reproduce the first pages of the first and third movements of the Haydn symphony from the *MuseData* printing system as Example 1 and 2.

As for accuracy, we originally took both "editions" to have been brought to the level of 100% accuracy. For the *SightReader* approach, accuracy consists of complete fidelity to the original material. This goal was achieved. For *MuseData*, however, accuracy consists of a faithful reproduction modified, as conditions warrant, to produce a fully sensible and workable set of information to be used for diverse applications. Frances Bennion and Ed Correia from CCARH found two errors in the Breitkopf edition related to the non-cancellation of accidentals. One case is given in Example 3. Numerous questions of visual grammar—ambiguous slurs, and so forth—also were encountered.

In the end, users will need to decide for themselves what level of data verification is required for their applications. A publisher wishing only to reprint a score may consider it unwarranted to proofread the original material before scanning. A performer working from an erroneous part or score or a MIDI file listener would undoubtedly prefer corrected material.

An overriding concern is that since *SCORE* is a model of completeness in the information that it represents, users of a scanning program that provides output to a simpler scheme, such as MIDI, or of limited sets of features (by excluding white notes, for example) must expect to spend a significantly greater amount of time in postprocessing than was the case in either instance here.



Example 3. Bars 74-77 of Movement 1 (*Presto*) of Haydn's Symphony No. 1. In Bar 74 the final note for second violins (Staff 4) was not marked with a natural in the original edition.

Our test narrowly avoided the inclusion of spurious material (an error that is unlikely to occur except through the use of scanners). When the symphony was photocopied for the participants, two pages of another Haydn symphony in the same meter strayed into the stack of pages representing the Finale. The scanning program was completely indifferent to this material, but the live data entry specialists found the abrupt change of key and simultaneous reduction in the number of parts, from 6 to 5, disconcerting. They investigated and discovered the error. This necessitated revising the results for *SightReader*, which originally reported 12 pages of material and an addtional 66 minutes of processing and editing time, and rerunning the *MuseData* test for the *Finale*; the material was of course more familiar the second time.

The moral is obvious: technology, however sophisticated, is no substitute for common sense. For the moment, scanning programs don't investigate; they simply scan. Even within the domain of manual input, the eye can be deceived. This is why we regard proofhearing as an essential step in the process of data verification. Old editions have their share of errors and notational anomalies. Since so much effort is required to encode musical data in large quantities, the additional human intelligence required to rectify these seems well worth providing.

We expect all of the systems represented here to improve dramatically over time. We also expect optical music recognition to become practical for printing and analysis applications in due course. In these, as well as in sound applications via MIDI, the user's error tolerance and time available for the necessary postprocessing will determine the point at which use of the technology becomes practical.

We wish to thank Nicholas Carter, Edmund Correia, Jr., and Frances Bennion for participating in this experiment.

CCARH is willing to make available to other recognition software developers copies of the materials used in this comparison for the purpose of evaluating the performance of their programs.