Computer pattern recognition of printed music*

by DAVID S. PRERAU

Department of Transportation/Transportation Systems Center
Cambridge, Massachusetts

INTRODUCTION

A major area of concentration of pattern recognition research has been the design of computer programs to recognize two-dimensional visual patterns. These patterns may be divided into two classes:[1] patterns representing objects in the real world (e.g., landscapes, blood cells) and patterns representing conventionalized symbols (e.g., printed text, maps). The standard notation used to specify most instrumental and vocal music forms a conventionalized, two-dimensional, visual pattern class.

This paper will discuss computer recognition of the music information specified by a sample of this standard notation. An engraving process is generally used to produce printed music, so the problem can be termed one of computer pattern recognition of standard engraved music (though the recognition procedure will, of course, be effective for music printed by any method).

The overall process is illustrated in Figure 1. A sample of printed music notation is scanned optically, and a digitized version of the music sample is fed into the computer. The digitized sample may be considered the data-set sensed by the computer. The computer performs the recognition and then produces an output in the Ford-Columbia music representation. Ford-Columbia is an alphanumeric language isomorphic to standard music notation. It is therefore capable of representing the music information specified by the original sample. (An example is shown in Figure 2.) In the form of a Ford-Columbia alphanumeric string, the output of the program can be used as input to music analysis programs,[2] music-playing programs, composer aids, Braille-music printers, music displays, commercial music printers, etc.

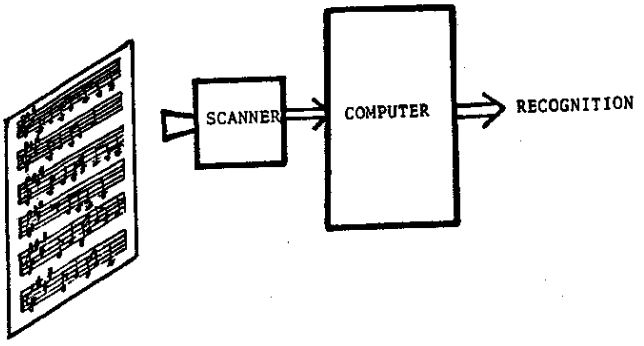


Figure 1—The overall process

THE PROGRAM

The product of this research is a computer program which recognizes standard engraved music notation. The program is called the Digital Optical-Recognizer of Engraved-Music Information, or DO-RE-MI.

DO-RE-MI is written using two computer languages, MAD and SLIP. MAD is a FORTRAN-like language, and SLIP is a list-processing language. The version of SLIP used in this study is embedded in MAD, and the combination of the two languages may be considered an extended MAD language with list-processing capability. DO-RE-MI was programmed on the MIT Compatible Time-Sharing System (CTSS), utilizing the IBM 7094 computer.

The set of symbols used in standard music notation is large. Therefore, it was decided that DO-RE-MI would be designed to recognize only a subset of these symbols; but a subset that would include all the more important symbols of music notation. This allows the program output to have practical utility, since many applications do not require full recognition. (For example, an analysis of the statistical frequency of occurrence of different pitch intervals would require only recognition of notes, clefs, accidentals, and key-signatures.) Moreover, the program is designed to be

* This paper is based on a thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Massachusetts Institute of Technology, Department of Electrical Engineering, September 1970; the work was supported in part by the National Institutes of Health.

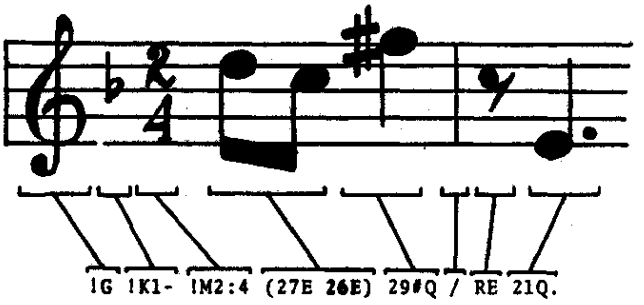


Figure 2—An example of the Ford-Columbia music representation

modular. Thus, it should not prove too difficult to add to DO-RE-MI, if desired, the capability of recognizing any of the secondary notational parameters not now recognized. This is in contrast to a previous work on computer recognition of printed music by Pruslin[3] which dealt with only a small subset of music notation and was not readily extendable to the remaining notational symbols.

DO-RE-MI is divided into three sections: Input, Isolation, and Recognition. In brief, the Input Section inputs a sample of standard engraved music notation to the computer, the Isolation Section isolates the notational symbols, and the Recognition Section performs the recognition. A flow-chart of the program is shown in Figure 3.

INPUT

The Input Section takes a sample of music notation and, using a flying-spot scanner, digitizes the sample. Several samples of source music were selected, and a positive transparency made for each. Each sample was chosen to contain two to three measures of duet music. Such a sample will be called a "picture". The scanner used is SCAD,[4] which was developed by Dr. Oleh Tretiak at M.I.T. SCAD will scan a transparency, measure the light transmission at a large number of points on the transparency, and convert these measurements to digital form.

For each picture, SCAD scans a raster of 512 rows by 512 columns, an inch in the original sample corresponding to about 225 points. SCAD finds a 3-bit number, T, corresponding to the transmittance of light through each raster point. The original picture contained, at least ideally, only black regions and white regions. It is therefore reasonable to compress the data

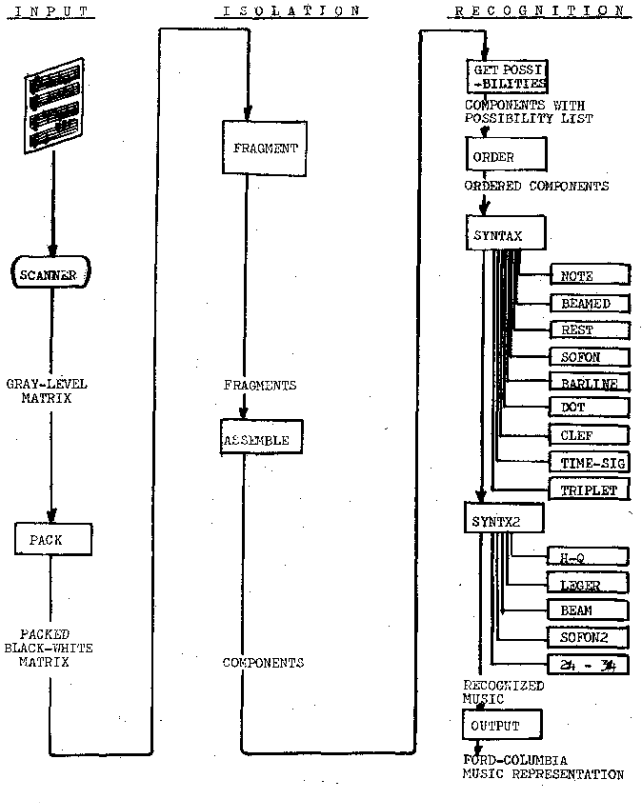


Figure 3—DO-RE-MI flowchart

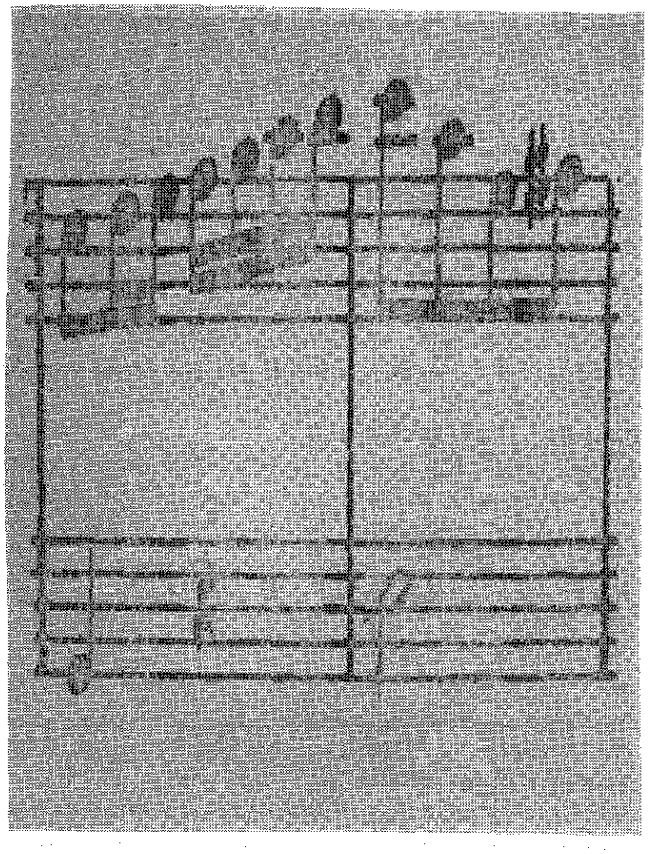


Figure 4—Printout of a picture

to one bit per point. This is done by choosing a threshold, Θ, and considering all points with transmittance $T > \Theta$ to be "White", and all points with transmittance $T < \Theta$ to be "Black". Since the transmittance of most points in the picture will not be near the threshold, the choice of Θ is not crucial.

The result is a matrix whose entries correspond to the digitized points of the music sample. (One such matrix is shown in Figure 4, representing "Black" by a point and "White" by a blank for display purposes.) Thresholding is done by a routine which is called PACK since it also packs the matrix for storage economy. This packed matrix is the data-base for the program.

ISOLATION

As shown in Figure 3, the packed Black-White matrix produced by the Input Section is passed to the Isolation Section. The function of this Section is to isolate each of the music notational symbols represented in the matrix, so that the program can then attempt to recognize individual symbols (rather than having to deal with arbitrary groups of symbols or with parts of symbols). The symbols must be isolated from the staff-lines upon which they are superimposed, and from each other. The staff-lines can be considered as graphical interference; they connect symbols that would normally be disconnected, they camouflage the contours of symbols, and they fill in symbol areas that would normally be blank. Thus, recognition is greatly simplified if the symbols are extracted from the staff-lines. This is a problem of pattern recognition in the presence of qualitatively defined interference, as the graphical positions of the staff-lines are qualitatively defined (i.e., five lines, nominally horizontal, straight, and equally spaced).

The process of symbol isolation must not destroy or significantly distort the original picture, for such destruction or distortion will make recognition difficult if not impossible. A method of isolation with minimal picture distortion has been developed. This method, called Fragmentation and Assemblage, is illustrated in Figure 5. First, the symbol fragments falling between staff-lines are picked out from the staff-line background by a relatively complex Fragmentation procedure. Each fragment is obtained by finding a list of the points that make up its contour (considering its intersections with staff-lines as part of its contour). Then, these fragments are assembled together again by associating the fragments into sets called picture components, each picture component corresponding to exactly one of the music notational symbols of the original sample. In essence, this procedure reforms the symbols, but without the staff-line interference. The reformed symbols are isolated from each other, as desired.

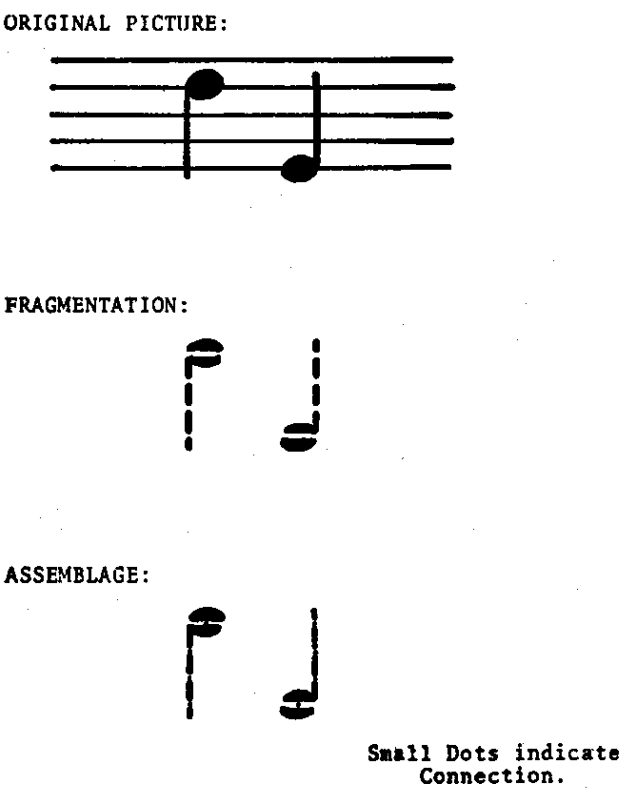


Figure 5—Fragmentation and Assemblage

For each fragment, a SLIP-list is produced containing ROWMAX, ROWMIN, COLMAX and COLMIN—the extreme rows and columns occupied by the fragment. This information is readily found from the fragment's list of contour points, and defines a rectangle bounding the fragment. The SLIP-list also contains fragment interconnection data.

In addition, a SLIP-list is produced for each component, containing a listing (by number) of the fragments that make up the component. It is then easy to find the overall ROWMAX, ROWMIN, COLMAX and COLMIN of the component from the data in the fragment SLIP-lists. (For example, the COLMIN of the component is just the minimum of the COLMINs of the fragments.) In this way, a bounding rectangle is found for each component.

The Fragmentation and Assemblage method is somewhat intricate and will be discussed in more detail in a future paper. However, it is important here to note the data reduction that it accomplishes. As illustrated in Figure 6, a picture is originally stored as a huge (250,000 point) matrix. The fragment data for the picture is stored as a fairly long table of contour points

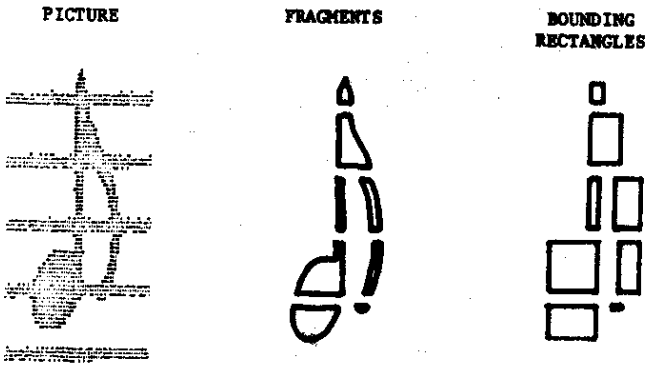


Figure 6—A picture, its fragments, and its bounding rectangles

(about 5000 to 10,000 contour points per picture). The corresponding SLIP-lists contain only the connection and bounding rectangle data (about 1000 list-entries per picture). In each step, the amount of data has been significantly reduced.

As will be seen, the small amount of easily-found information in the fragment and component SLIP-lists is enough, in most cases, to enable complete recognition of the components.

It was originally thought that each symbol would have to be fully reconstructed after Fragmentation by combining its fragments graphically, and filling in the spaces left by the staff-lines. This process was not needed. Surprisingly, the information in the Black-White picture matrix was never needed after Fragmentation. Even more surprisingly, the information in the fragment contour-point listings was needed only in five very special cases. Except for these special cases, all recognition can be done just from the ROWMAX, ROWMIN, COLMAX, COLMIN, and connection information in the SLIP-lists! This result points out another benefit of the Fragmentation-Assemblage method of symbol isolation: almost all the recognition tests can be performed on the relatively small data base of the SLIP-lists, rather than on the larger data base of the Fragment point-listings, or the even larger data base of the picture matrix.

PRELIMINARY FILTERING

The symbols of the picture having been isolated, the actual recognition procedure can begin. First, it is interesting to note that certain familiar techniques cannot be used. In standard music notation there are some symbols that are not characters, in that they have one or more graphical parameters whose value may be different for each occurrence of the symbol. Consider Figure 7. On the top staff there are two occurrences of the same symbol. However, the two are not geometrically identical, since the spacing is affected by the notes on the bottom staff. Thus, these symbols are not characters. Many of the techniques used for recognition of characters, such as template-matching, cannot be used to recognize non-character symbols. Such techniques are therefore not very useful in the recognition of music notation. Alternative techniques must be employed.

Music notation has many strong syntactic properties. In combination with small lists of possibilities for each unknown symbol, these syntactic properties can be used with good results to recognize each component. Thus, it was decided to use a preliminary filter to reduce the set of possible symbols corresponding to the unknown to a small subset, and then to use the syntactic properties for unique classification.

Upon consideration of the form of the data after Fragmentation and Assemblage (i.e., the SLIP-lists) and of the graphical properties of music notational symbols, it was decided that a simple but powerful preliminary filtering could be obtained by examination of the normalized overall size of the component. A close look at the symbols of standard music notation reveals that each symbol type is significantly different in overall height or overall width from almost all others.

In order to find the size range of each music symbol, a survey of standard music notation was made. Many samples of each type of notational symbol were measured, and the overall height and width of each symbol tabulated. These measurements must be normalized since different samples of music will in general be of different scale. The average height of a staff-space (which shall be denoted as SPHGT) appears to be a good normalization factor, so that by dividing the physical dimensions by SPHGT, the measurements can

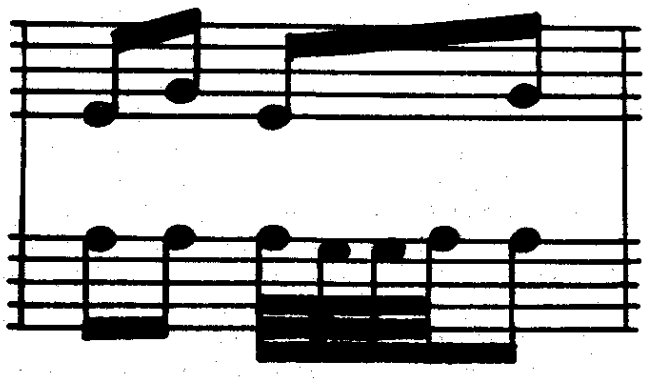


Figure 7—Non-character symbols in music notation

be converted into ratios that are independent of absolute dimensions.

Each normalized pair of measurements is plotted in a normalized-height vs. normalized-width property-space. A region is then delineated in the property-space for each type of music symbol. This region is chosen to be the smallest rectangular region enclosing all the plotted points for that symbol. The symbol regions in the property-space are shown in Figure 8. The figure shows, for example, that the measured natural signs were always approximately 0.6 to 0.8 SPHGTs in width, and 3.2 to 3.9 SPHGTs in height. Note from the figure that there is very little overlap of the regions.

In finding the list of possible assignments for an unknown component, it appears far better to include a few extra possibilities than to leave out the correct one. Therefore, it seems reasonable to enlarge the rectangular regions of the property-space to allow additional

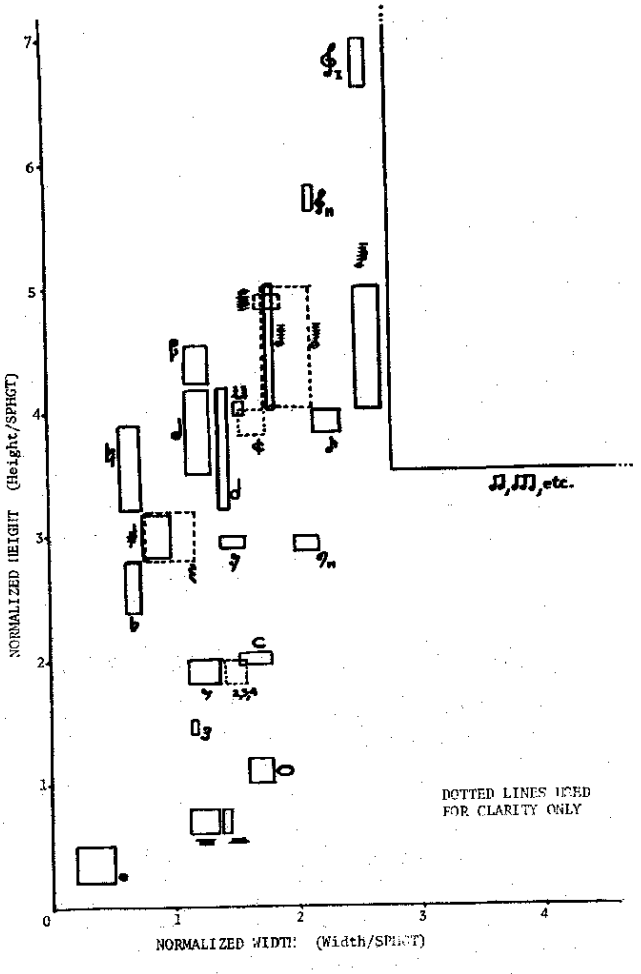


Figure 8—Normalized-height vs. normalized-width property-space

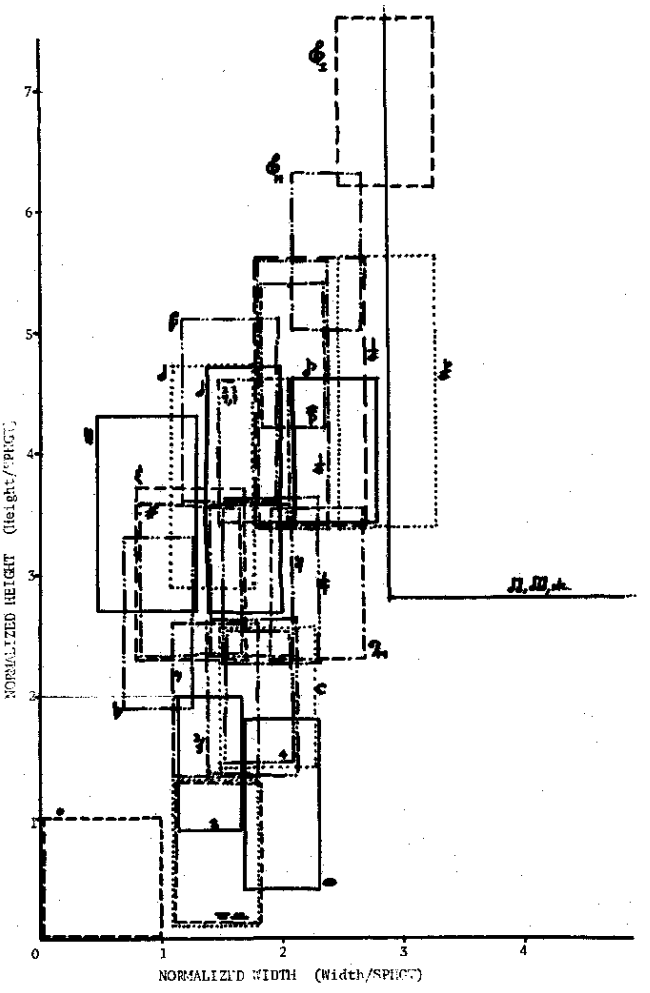


Figure 9—The H-W space (The normalized-height vs. normalized-width space with enlarged regions)

tolerance. This allows for processing effects due to the scanner and quantization, and for such characteristics of printed music as the widening of a symbol-line when it crosses a staff-line, etc. The normalized Height-Width property-space with the enlarged regions is called "the H-W space". This is shown in Figure 9. Note that the number of overlapping regions at a point is usually between three and five, with a maximum of eight. The points with the higher numbers of overlaps usually occur where the corners of many regions meet. Since most plotted points for unknown components will be found near the middle of the regions of their corresponding symbols, the areas of the property-space corresponding to the larger numbers of overlaps are rarely encountered in practice.

For every tested picture, the point in the H-W space plotted for each component fell in the region representing the symbol actually corresponding to that

component. Since the H-W space contains enlarged regions, and since it uses normalized values, it is reasonable to use the same H-W space for recognizing any sample of music notation. However, if for certain printed music it is found that the regions in the H-W space are not properly delineated, it is only necessary to find an H-W space based on this new set of symbols. Changing the H-W space in DO-RE-MI requires nothing more than changing a few parameter values. Or, one might store several H-W spaces, and then call the one corresponding to the music style in which the sample has been printed. This ability to change the H-W space so easily is an attribute of the modularity of the program.

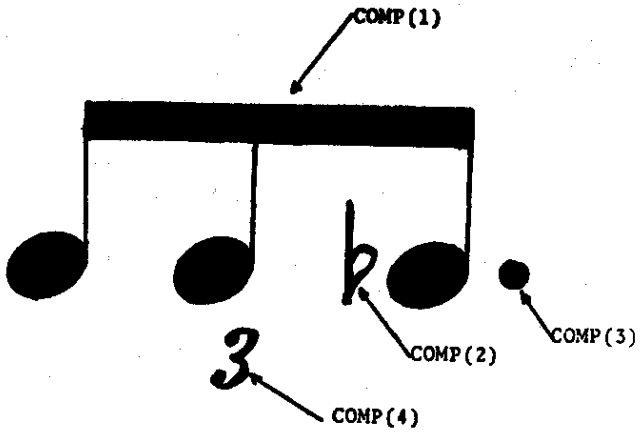
The Get-Possibilities Routine (GETPOS) performs the preliminary filtering. Given the H-W space, a short list can be generated of the symbols that can possibly correspond to a given unknown component. This is done by finding the point in the property-space representing the normalized-width and the normalized-height of the component. The regions in which this point falls specify the music symbols which can correspond to the component. When the Possibility List has been found by GETPOS, it is added to the SLIP-list of the component.

It is interesting to note the power of the preliminary filter. It is based on a simple overall property of the component, normalized size, and is completely independent of the internal features of the component. Yet, it is able to reduce the number of possible symbols corresponding to the component usually to about three to five with a maximum of eight. In addition, use of the data found in Fragmentation and Assemblage makes the determination of the normalized overall size a trivial calculation.

ORDERING

The components have been found by Fragmentation and Assemblage independent of their positions on the staves. It is therefore necessary to associate each component with the staff from which it was extracted, and to find the left-to-right order of the components within each staff. Knowledge of this ordering is needed to produce the final output sequence. In addition, it is useful because two important graphical features of symbols in music notation can be ascertained from the order information: immediate symbol context and symbol nestedness.

The Order Routine (ORDER) finds the left-to-right ordering of the components in each staff, forming an order list for each staff. Due to the two-dimensionality of the placement of music symbols, the ordering process is not simple. (This is in direct contrast to printed text



ORDERING BY LEFTMOST POINT:
COMP(1), COMP(4), COMP(2), COMP(3).

ORDERING BY CENTER OF MASS:
COMP(4), COMP(1), COMP(2), COMP(3).

ORDERING BY THE "ORDER" ROUTINE:
COMP(1)-L, COMP(4)-L, COMP(4)-R,
COMP(2)-L, COMP(2)-R, COMP(1)-R,
COMP(3)-L, COMP(3)-R.

Figure 10—An example of three different ordering methods

where the characters are in a one-dimensional string, and where the ordering process is trivial.) Consider the music sample of Figure 10. Note, for example, that Component 2 is neither to the left nor to the right of Component 1, but that a more complex two-dimensional relationship exists between them. Thus, ordering the components by one-dimensional techniques will not be satisfactory. An ordering by leftmost point would give the sequence: COMP(1), COMP(4), COMP(2), COMP(3). An ordering by center of mass would give: COMP(4), COMP(1), COMP(2), COMP(3). Neither is a good representation of the situation.

DO-RE-MI orders the components by both their leftmost and rightmost points in the same list. For Figure 10, this method gives the sequence: COMP(1)-Left, COMP(4)-Left, COMP(4)-Right, COMP(2)-Left, COMP(2)-Right, COMP(1)-Right, COMP(3)-Left, COMP(3)-Right. This ordering more accurately represents the overlapping of COMP(4) and COMP(2) by COMP(1).

SYNTACTIC TESTS

Most of the symbols of standard music notation (as opposed to the alphabetic symbols of printed text)

have a strong set of symbol-to-symbol contextual syntactic properties. For example, in the standard music considered by this study, the "flat" sign may appear in only one of two contexts (as shown in Figure 11):

(1) In a key signature to the right of a clef or bar-line.
(2) As an accidental to the left of a note.

Any other appearance of a flat is syntactically incorrect. Also illustrated in Figure 11 is the rule that the first symbol of a line of music must be a clef. This type of syntactic property can be used to great advantage in recognition.

Music notation contains many redundancies and these too can be used to aid recognition. There are two types of redundancies: syntactic redundancy and graphical redundancy. Figure 11 shows an example of syntactic redundancy. Since the line starts with a *G*-clef, the first flat of the key-signature (if any) must be on the middle line, i.e., a *B*-flat. Conversely, if the first symbol of the key-signature is on the middle line, it must be a flat. Continuing in the same manner, if the first key-signature symbol is a flat, the second key-signature symbol (if any) must be on the top space, and must be a flat. An example of graphical redundancy would be the *F*-clef, where a component cannot be recognized as an *F*-clef, even if it passes all other tests, unless two other components are recognized as dots and are in the correct position to be the *F*-clef dots. (If so, all three components are considered to form the *F*-clef symbol).

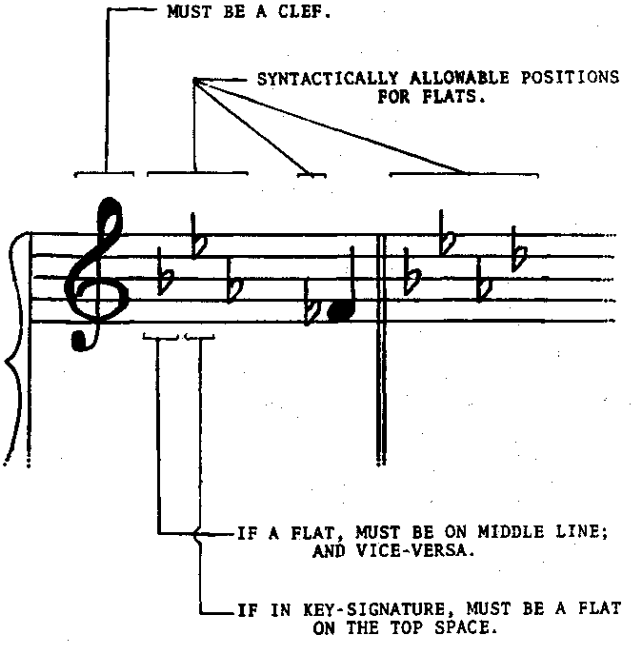


Figure 11—Syntax and redundancy in standard music notation

Another type of syntactic property exhibited by music notational symbols is positional syntax. Many symbols can occur only in a fixed position on the staff, or only in a certain region (e.g., above the staff). For example, the two dots of a repeat sign will always be found in the two middle spaces of the staff. Also, each rest has a standard vertical position and thus will always occupy the same position on the staff (except for a few special cases).

The Syntax Routine (SYNTAX) uses syntactic properties to perform the final recognition of the components. It examines all the components in sequence as they appear on each staff's order list. For each component, SYNTAX performs tests on every entry on the Possibility List, and eliminates from the list all those possibilities which fail any of these tests. Contextual syntactic properties are tested. Some feature tests must be employed, but only when the other three types of tests still yield ambiguities. This occurs, for example, when a sharp, a flat or a natural appears as an accidental (i.e., to the left of a note). In this usage, the three symbols can be syntactically equivalent. Since they are approximately the same size and would occupy approximately the same vertical position, they often cannot be differentiated by the H-W space or by position tests. In this case, feature tests must be used. As will be seen, a simple feature test is used to differentiate among the three.

As shown in Figure 3, SYNTAX calls nine subroutines, each testing a different class of symbol-possibility, e.g., notes, rests, clefs, time-signatures, etc. SYNTX2 is Part 2 of the Syntax tests, and is called when one of the five special cases requiring information from the Fragment contour-point lists is found. In all other cases, recognition is completed by SYNTAX and its nine subroutines, and the only information used is that in the SLIP-lists, i.e., the bounding rectangles on each fragment and component plus the interconnection data.

A REPRESENTATIVE 'SYNTAX' SUBROUTINE

As an illustration of the procedures used in SYNTAX and SYNTX2, a representative SYNTAX subroutine, the Sofon Test, will be discussed in some detail. ("Sofon", pronounced sō-fon, is a term that is coined here from the initials "Sharp Or Flat Or Natural" to denote a symbol which is any of these three, inde-

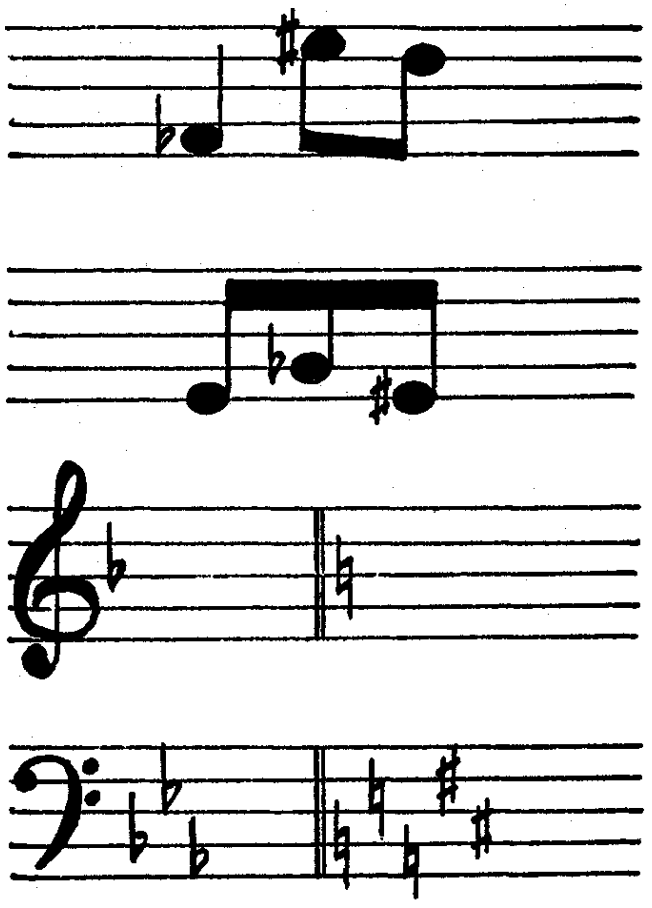


Figure 12—Situations where sofons occur

pendent of whether the symbol appears in a key-signature or as an accidental. No such general term for this set of symbols exists in the music vocabulary, and such a term is needed since these symbols can often be treated together during symbol recognition.) The Sofon Test (SOFON) tests all components that have either "sharp" or "flat" or "natural" on their Possibility List.

There are strong contextual syntactic tests, redundancy tests and positional syntastic tests that can be applied to sofons. Thus, a sofon must be either:

1. To the left of a note or a beamed-together notegroup, and close to it (i.e., within 1 SPHGT horizontally), and at least partially overlapping it vertically.
2. Overlapped horizontally by a beamed-together notegroup, and at least partially overlapping it vertically.
3. To the right of a clef or bar-line and on the correct pitch-space. (The "pitch-space" of a sofon is defined as the pitch of the line or space which the sofon is "on." This is determined by the vertical position of the sofon and the clef currently in effect.) In this case, the correct pitch-spaces are:

   For a sharp: *F*
   For a flat: *B*
   For a natural: *F* or *B*

4. To the right of another of the same sofon type, and close to it, and on the correct pitch-space. The sequence of correct pitch-spaces for key-signature sofons is as follows:

   For sharps: *F, C, G, D, A, E, B*
   For flats: *B, E, A, D, G, C, F*
   For naturals: Either of the above sequences.

   The initial sharp or flat of a key-signature can also be to the right of a natural.

In the first two cases the sofon is used as an accidental; in the latter two cases it is used in a key-signature. Typical situations are illustrated in Figure 12.

Since accidental sofons are syntactically equivalent, the three types of sofons often cannot be separated from each other by the above tests (though SYNTAX will eliminate all other possibilities from the Possibility List). In this case, a feature test must be used.

After consideration of many other separation

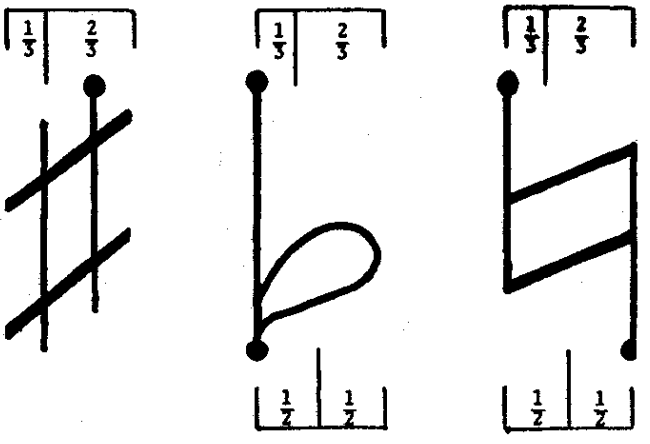


Figure 13—Sofon separation

algorithms, it was noted that the three sofons could be separated by examining only their top and bottom points, as follows (see Figure 13):

(1) Sharp vs. Flat; Sharp vs. Natural:

—For a sharp, the topmost point occurs in the *rightmost two-thirds* of the component width.
—For a flat or natural, the topmost point occurs in the *leftmost one-third* of the component width.

(2) Flat vs. Natural

—For a flat, the bottommost point occurs in the *left half* of the component width.
—For a natural, the bottommost point occurs in the *right half* of the component width.

Thus, examination of the topmost point of the component separates sharps from flats and naturals, and examination of the bottommost point separates flats from naturals.

The sofon separation test requires finding the topmost and bottommost points of the component, and these can be found easily from the contour-point lists. This is one of the cases where it may be necessary to employ SYNTX2 and the fragment contour-point lists. Point lists for only two fragments must be examined: the fragments containing the ROWMIN and the ROWMAX of the component. In fact, often these fragments fall completely on one side or the other of the two-thirds or one-half points of the component's width. When this is so, the sofon separation can be made solely from the information in the SLIP-lists. Then, there is no need to look at the contour-point lists, and SYNTX2 does not have to be called at all.

OUTPUT

The Output Routine (OUTPUT) takes the music which was recognized by the Syntax Routine (SYNTAX and SYNTX2) and produces the final DO-RE-MI output according to the Ford-Columbia Music Representation. As has been mentioned, Ford-Columbia is an alphanumeric language which is substantially isomorphic to standard music notation. It was developed by a musician, Stefan Bauer-Mengelberg.[5] OUTPUT stores the Ford-Columbia representation of the recognized music on a SLIP list. For example, a printout of the final output for the picture of Figure 4 is shown in Figure 14. (Output restrictions on SLIP-lists in CTSS do not allow accurate printing of all the symbols of the Ford-Columbia character set; therefore, a modified Ford-Columbia is printed out with commas representing blanks and apostrophes representing exclamation points.) In this printout, "I01" indicates that the first instrument's line begins, "26E" indicates an *E*ighth note on space 26 (the third space on the top staff), parentheses indicate beams, "/" indicates a bar-line, etc.

```
('I 0 1,(26 E,(27 S, 28 S)),((29 S, 30 S, 31 S, 32 S)),/,(33 E, 31 E,
28 E, 29=E), 'I 0 2, 21 Q, R Q,/, 26 H)
```

Figure 14—Output for the picture of Figure 4

RESULTS OF TEST RUNS

DO-RE-MI was tested on some representative pictures, including the one shown in Figure 4. In all cases, the program produced the desired Ford-Columbia representation of the input picture with complete accuracy. All symbols in the subset of music notation considered by DO-RE-MI were correctly recognized. In addition, all symbols not in that subset were correctly recognized as such.

Some overall statistics on the test-runs: DO-RE-MI correctly isolated all the music notational symbols into a total of 137 components. These components were formed by 527 fragments. All 137 components were correctly recognized by DO-RE-MI (including thirteen components which were each a group of three to four notes beamed together). An average test-run took about four minutes, from packed Black-White matrix to Ford-Columbia output, for a test picture of two to three measures of duet music (i.e., four to six single measures of music). This time figure could be significantly reduced by various means; however, minimization of run-time was not a major goal of this work.

CONCLUSION

This work is an investigation into computer recognition of a class of conventionalized, two-dimensional, visual patterns: standard engraved music notation. Important aspects of the problem involve pattern recognition in qualitatively-defined interference, recognition of positionally two-dimensional patterns, use of syntax and redundancy properties in recognition, and recognition of non-character symbols. A simple preliminary filter

proved very effective. Bounding rectangles on fragments and components unexpectedly provided all necessary data for recognition in almost all cases, and thus examination of detailed feature properties was rarely required. The program produced should be able to be expanded to the recognition of all printed music. In addition, the pattern recognition techniques developed can possibly be applied to computer recognition of such things as maps, graphs, organic chemistry symbols, circuit diagrams, blueprints, aerial photographs, etc.

ACKNOWLEDGMENT

I would like to gratefully acknowledge the encouragement, support, and counsel of Professor Murray Eden of M.I.T., who guided me throughout the course of this research, and of Professor Allen Forte of Yale, Dr. Oleh Tretiak of M.I.T. and Professor Francis Lee of M.I.T. The interest and assistance of the members of the Cognitive Information Processing Group of the M.I.T. Research Laboratory of Electronics is also greatly appreciated.

REFERENCES

1 M EDEN
Recognizing patterns
P Kolers and M Eden editors Chapter 8
The MIT Press Cambridge MA 1968

2 H B LINCOLN
The current state of music research and the computer
Computers and the Humanities September 1970

3 D H PRUSLIN
Automatic recognition of sheet music
Doctoral Thesis MIT Cambridge MA January 1967

4 O J TRETIAK
Scanner display (SCAD)
Quarterly Progress Report, No. 83 MIT Research Laboratory of Electronics October 1966

5 S BAUER-MENGELBERG
(of the IBM Systems Research Institute New York City)
Representing music to a computer: A primer of the Ford-Columbia music representation (DARMS)
Manuscript in preparation