Adaptive Optical Music Recognition

Ichiro Fujinaga

Faculty of Music
McGill University
Montréal, Canada

June 1996

A Thesis submitted to the Faculty of
Graduate Studies and Research
in partial fulfillment of the requirements of the degree of
Doctor of Philosophy

0-612-29937-6

To My Parents and Maïko

Abstract

The basic goal of the Adaptive Optical Music Recognition system presented herein is to create an adaptive software for the recognition of musical notation. The focus of this research has been to create a robust framework upon which a practical optical music recognizer can be built.

The strength of this system is its ability to learn new music symbols and handwritten notations. It also continually improves its accuracy in recognizing these objects by adjusting internal parameters. Given the wide range of music notation styles, these are essential characteristics of a music recognizer.

The implementation of the adaptive system is based on exemplar-based incremental learning, analogous to the idea of "learning by examples," that identifies unknown objects by their similarity to one or more of the known stored examples. The entire process is based on two simple, yet powerful algorithms: k-nearest neighbour classifier and genetic algorithm. Using these algorithms, the system is designed to increase its accuracy over time as more data are processed.

Résumé

Le système de reconnaissance optique de la musique proposé ici a pour but de créer un logiciel adaptif qui permet de reconnaître la notation musicale. L’objectif principal de cette recherche a été de concevoir une structure solide sur laquelle on peut construire un système pratique de reconnaissance de la musique.

La force de ce système réside dans sa capacité d'apprendre de nouveaux symboles musicaux et des notations manuscrites. En adjustant ses paramètres internes, le système accroit sa précision dans la reconnaissance des divers éléments. Étant donné le vaste éventail de styles de notation musicales, ces caractéristiques constituent l’essentiel d’un système de reconnaissance de la musique.

La mise en oeuvre d’un tel système est basée sur le concept de « l’apprentissage par l’exemple » : le système identifie des éléments inconnus en les comparant avec un ou plusieurs éléments connus déjà emmagasinés. Le processus tout entier s’appuie sur deux algorithmes simples mais puissants : l’algorithme du plus proche voisin et l’algorithme génétic. Ces algorithmes permettent au système d'augmenter sa précision d’opération en fonction de la quantité de données qu’il a traitées.

Acknowledgements

I would first like to acknowledge Prof. Bo Alphonce and Prof. Bruce Pennycook for their wonderful support and supervision during this project which lasted over a decade.

I would also like to thank the following people who had helped and supported me over the years: Elizabeth Azcona-Hartmark, Natalie Boisvert, Prof. Mary Cyr, Brenda Dalen, Francois Couture, Prof. Malcolm Forsyth, Tom Hall, Jim and Maria Harley, Garth Hobden, Arnee Holloway, Prof. Susumu Horiguchi, Kharim Hogan, Shelly Irvine, Cynthia Leive, Roger Lord, Prof. Bruce Minorgan, Marie Moscato, Prof. David Ostry, the Paquette family, especially Maïko and Sylvie Paquette, Prof. Paul Pedersen and Jean Pedersen, Prof. Fordyce "Duke" Pier, René Quesenel, Meijane Quong, Charlene Redekopp, David Rosenthal, Martin Roth, Rosemary Speakman, Dale Stammen, Jason Vantomme, Lise Viens, Dr. Susan Weiss, Van Wilmott, and Dr. Geoffery Wright.

The research was, in part, supported by a research grant from the Social Sciences and Humanities Research Council of Canada and an equipment grant from the Apple Canada Educational Foundation.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 The goal

The basic goal of the Adaptive Optical Music Recognition (AOMR) project is to design an adaptive system for computer recognition of musical notation that works with a certain degree of user interaction. The focus of this research has been to create a robust framework within which a practical optical music recognition (OMR) system can be built.

1.2 Overall design

The AOMR system described here is composed of a database and three interdependent processes: recognizer, editor, and learner. Operating on the scanned image of a musical score, the recognizer locates, separates, and classifies music symbols into musically meaningful categories. The classification is based on the k-nearest neighbour (k-NN) rule aided by a database of symbols and their features collected from previous sessions.

The output of the recognizer is corrected by a musically trained human operator using a music notation editor. The editor can provide both visual and audio feedback of the output. Glen Diener's *Nutation,* a public-domain music editor, which displays and playbacks the result of the recognition process, was experimentally used for this purpose. Commercially available music editors may be used. The result is stored in the symbol database used by the classifier and the learner. This database can also be used as a basis for constructing a representation of the score suitable for other applications. The learner improves the speed and accuracy of future recognition sessions by continuously rearranging the database and optimizing classification strategies.

1.3 Adaptive systems

The most interesting feature of this system is its ability to learn and adapt incrementally to its environment. Rather than using statistical or deterministic methods of pattern recognition, commonly used by engineers and other OMR systems, an adaptive exemplar-based system is used here to recognize music scores.

1.3.1 What is an adaptive system?

An adaptive system is characterized by the ability to undergo modification of its behaviour in response to new conditions, demands, and circumstances of the surrounding environment. For a recognition system, it means that the system will be able to learn novel objects and that it will continually improve its accuracy in recognizing those objects. Given the wide range of music notation typefaces, this is an essential component for a music recognizer.

1.3.2 Implementation of the adaptive system

The present implementation of the adaptive system is based on an exemplar-based incremental learning system. An exemplar-based pattern recognition scheme classifies an unknown object by comparing it to the known examples already stored in its database. "Incremental" here means that the system learns gradually as new samples are added to the database.

Typically, a learning system is nurtured with training data. Once the designer is satisfied with the performance of the system, the various parameters of that system are fixed. In other words, no modification takes place when the system is actually used in the field. Here, no distinction is made between training data and real data: all incoming data are treated as training data, and the system parameters are continually changing.

The reorganization of the recognition tactics, such as the parameter tuning, is managed by the system itself rather than the human expert. This process seems to correspond to human incremental development of expertise. The adaptiveness of the system is founded on two very simple yet powerful concepts: k-NN rule and genetic algorithm.

Using these algorithms, the system is designed to increase its accuracy over time as more data are processed. The accuracy of the recognizer can be increased by having many examples and by selecting the appropriate importance attached to each feature used to recognize the symbols. If required, the system can decrease the recognition time on its own. In the k-NN classification system the recognition time is proportional to the size of the database. By reducing the size of the database, therefore, the recognition time can be reduced.

Exemplar-based systems have often been criticized for their relatively large storage requirement and for inefficiency. The recent dramatic increase in economically available memory space along with similar increase in the speed of desktop computers have made the use of exemplar-based systems quite feasible. It is not unreasonable to demand megabytes of RAM, gigabytes of hard disk space, and a fast microprocessor.

Furthermore, the efficiency of this particular application is not crucial as manual preparation of a score by a human copyist could take over an hour per page. Also, as most desktop computers are personal computers (in other words, they are not used constantly) there are many free cycles that can be exploited by the learning system.

1.3.3 The advantages of an adaptive music recognition system

There are three main reasons why an adaptive music recognition system is desirable. It should be able to recognize a large number of symbols and the arrangements of these symbols that make up the score; it should be able to learn new music symbols; and it should be able to recognize handwritten scores.

Similarities between the recognition of printed text and of music are often cited, yet there are important differences. In music there is a basic set of symbols, such as rests, clefs, and accidentals, that have fixed size and orientation, corresponding to the letters, digits, and punctuation symbols in printed text. But unlike text, music scores contain many symbols that vary in size and orientation, such as arpeggio marks, slurs, ties, barlines, pedal markings, and voice-leading lines. Also, noteheads are often grouped together with other such components as stems, flags, and beams. Thus, the recognition system for music must be able to recognize a very large number of configurations of symbols.

Another very important difference is that in the case of alphabets, although there are new font designs, it is unlikely that a new alphabet symbol will be added within the next few years. Music notation on the other hand, is a more evolving system with new symbols continually being added. Consequently the set of music symbols is much larger than that of alphabet symbols. Read's book of notation lists about four-hundred different symbols that are currently in use (Read 1979). The learner section makes the system adaptive both to the evolving nature of music notation in general, where new symbols are created as performance or compositional requirements dictate, and to specific notational "dialects," including handwritten scores and different historical notations.

Until very recently, most scores of new compositions were prepared by hand owing to the expensive process of engraving music. These scores are generally of very high quality; because music must be sight-read in real time, there is an enormous pressure to have the music easily legible. Not only do performers tend to be discouraged by music that is difficult to read, but the processing resources and time devoted to decoding the music notation will presumably reduce the resources and time needed to perform it. For this reason, many high-quality handwritten scores should be recognized by the system. And there is another reason why machine recognition of handwritten notation would be valuable. Because of the availability of music editing software on microcomputers today, music that would once have been copied by hand is now often done on the computer. Yet because of the awkward user-interface (screen, keyboard, and mouse), many musicians prefer using the pen-and-paper method of setting music down, although they do appreciate the output of high-resolution laser printers. Note that the user interface to computers grew out of and remains a tool primarily for alpha-numeric input. Similarly, many graphic artists and draftsmen still prefer the traditional working tools, not surprisingly, since the tools these artists and craftsmen use have been tailored over the years to their needs. Thus, an ideal scenario is to draft the music by hand, scan it into the computer, edit, if necessary, and then print it out.

There are other benefits to adaptive systems. Different copies of the system may evolve along different lines, much in the same way as natural selection, each system developing its own expertise according to the needs of the users. Consequently, a copy of the system can be made to specifications, either with a *tabula rasa* database or primed for one particular notational repertoire, publisher, or composer. Another important advantage from the designer's point of view is that various adjustable parameters in the recognition process need not be predetermined. The wider implication of similar adaptive systems both in music and other domains will be discussed in the conclusions.

1.4 Applications

There are many areas of possible application of the machine-readable representation of musical scores. For music publishers, it can be used to produce new editions based on old editions and manuscripts. It can be used to preserve out-of-print editions for which the master plates are either lost or no longer usable. It can be used to create automatically engraved-quality scores based on manuscripts.

Musicologists can use it for various purposes including the preparation of scholarly editions that compare concordances between manuscripts and printed scores. Performers and composers can use it for part extraction and transpositions, Braille translation, automatic MIDI file creation, and thus automatic playback which in turn would allow score-assisted recognition of musical performance via audio, and "what-if" demos for music theory and orchestration studies. Such a playback system would also allow for computer-aided music practice in the form of intelligent music-minus-one for chamber music, concertos, and conducting practices. It would also simplify the preparation of music psychology experiments such as the study of music expression.

Although some of these applications can be performed now with commercially available music editing software, the tedious task of entering music manually has hindered development of most of these applications. For reviews of other methods of input see Carter et al. (1988) and Fujinaga (1988).

Once a sufficiently large amount of music is scanned and stored in a database, there are further applications. Music scholars can use the database to study musical structures and style, either manually or automatically. In the latter case, the computer can be used to verify algorithmic analytical tools and theories. Music publishers may establish an on-demand music-score printing, where music can be printed on a customer's local printer. In a multimedia environment, a database may be used for a low-bandwidth, high-quality audio distribution system. Rather than sending high-bandwidth CD-quality audio on the network, which may be difficult because of the amount of data involved, scores can be sent to the local workstation, where audio is recreated locally through the use of synthesizers. Also, music scores can be searched and viewed on screens on the network for browsing or sight-reading purposes where printed music is not necessarily required.

1.5 Design of the dissertation

In the next chapter, some recent papers on other OMR research will be reviewed. Many of the image processing and pattern classification techniques used in the program are explained in Chapter 3. Chapter 4 describes the program, and concluding remarks are presented in Chapter 5.

2. REVIEW OF OMR RESEARCH

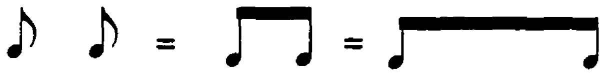
Until recently, research into OMR has been restricted to two MIT doctoral dissertations (Prusslin 1966, Prerau 1970). With the availability of inexpensive optical scanners, much research began in the 1980's. More recent research projects have been reported in issues of *Computing in Musicology* (Hewlett and Selfridge-Field, 1987–94). An excellent historical review of OMR systems is given in Blostein and Baird (1992). Here, some of the Japanese-only papers and other research not covered in that review will be summarized. Commercial software is now available from Musitek (MIDISCAN), Grande Software (Note Scan), and Yamaha.

2.1 Aoyama and Tojo (1982)

This relatively early paper, published only in Japanese, contains many techniques that are used by more recent research in optical music recognition systems. The system is divided into three stages: input, segmentation, and recognition and syntax check. In the input stage, the image is binarized, staffline height and staffspace height are obtained, and stafflines are located. In the segmentation stage, the stafflines are removed and symbols are segmented using connected component analysis. Finally the segmented symbols are classified and verified.

The following observations about the music score are made:

1) It is two-dimensional.
2) Spatial information is important.
3) Line drawing, image, and characters are mixed, and their position is not specified.
4) Because of fine lines, high resolution scanning is necessary.
5) Symbols having the same meaning may have different graphic representations,

e.g.,  .

6) Symbols are placed according to spatial syntactic rules.

From the recognition viewpoint, scores contain symbols that are

1) suitable for template matching and
2) suitable for a structural analysis method.

The input score is assumed to be printed and free of broken symbols, but can be of any size (within limits) and staves may be bent or slightly broken. The system uses a 254-dpi (dots per inch) drum scanner with 8-bit gray level.

The image is scanned twice. In the first scan, groups of vertical scan lines are obtained (a figure shows nine groups across the page, each group containing a few lines separated by 1 mm). The stafflines are located as follows:

1) Binarization of the scan lines are achieved through the use of a histogram.
2) Y-projection of each group is taken, and if each group contains n lines, projections with n or n–1 pixels are considered to be staffline candidates.
3) By using the result of 2) and creating a histogram of black runs and white runs from the staffline candidates, staffspace height and staffline height are obtained.
4) The candidates for stafflines are finalized using the information obtained in 3).

In the second scan, because of the large amount of information involved, each staff is considered separately. In each staff window, the picture is vertically run-length coded (this is the direction in which the page is physically scanned in their drum scanner).

The system removes most of the stafflines, but to avoid excessive segmentation of symbols such as half-notes and flats when the stafflines are removed, the regions of the staffline left and right of the runs adjacent to the symbol are marked so as not to be deleted (see Figure 2.1). At the end, runs that straddle the staffline position and that have the staffline width are removed.

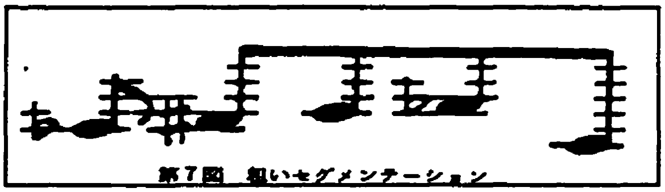


Figure 2.1 Image after coarse segmentation (Aoyama and Tojo 1982).

Next, black noteheads are searched with a template on stafflines or between stafflines, and temporarily removed if found. The black noteheads are only temporarily removed because the real goal of this section is to find holes (in flats, half noteheads, and whole noteheads). Once found these symbols can be marked so that when the rest of the stafflines are removed the symbols will not be fragmented. The holes are detected by the system looking for short horizontal white runs between stafflines. Once the holes are marked the black noteheads are restored, and stafflines are finally removed.

The resulting image is segmented through connected component analysis. The height and the width of the bounding box of each segment are used to coarsely separate the connected components into ten groups (see Figure 2.2). The height and width are normalized using the staffspace height.

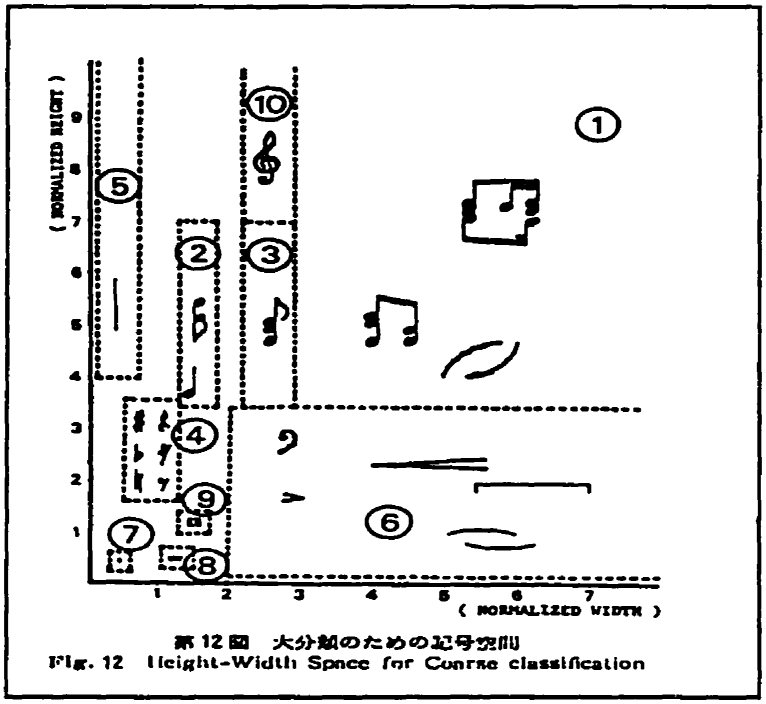


Figure 2.2 Coarse classification (Aoyama and Tojo 1982).

In the group with flagged notes and beamed notes, flags and beams are separated from noteheads by removing thin regions (stems). Analysis of the note configuration is performed by way of features such as width, height, center of gravity, ratio of area / area of the bounding box, head count, flag count, and H-type (any of 11 head-stem configurations).

In another group of accidentals and rests, a tree classifier based on horizontal and vertical run-lengths is used to separate the members of this class. A table containing information about relative position of components is employed to recognize composite symbols (e.g. 𝄎 𝄋 𝄐 𝄢:).

Finally, syntax rules concerning the position of symbols and the constant number of beats in a measure are used to double-check the recognition result. The spatial rules are:

1) key signatures appear after the clef symbol;
2) if there is a treble clef and key signature starts with a sharp, the sharp must be on the top staffline;
3) accidentals appear to the left of the notehead.

Although not implemented, the possibility of recognizing expressive markings (*pp, andante, a tempo,* etc.) by their character count is suggested.

2.2 Maenaka and Tadokoro (1983)

Maenaka and Tadokoro aimed at building a system that would be portable, compact, easy-to-use, and inexpensive. To meet these design goals, they used an 8-bit microprocessor (MC6809) and a TV camera as input device. They mention the possibility of using a facsimile machine as an alternate inexpensive input device. The overall system architecture is shown below (see Figure 2.3).

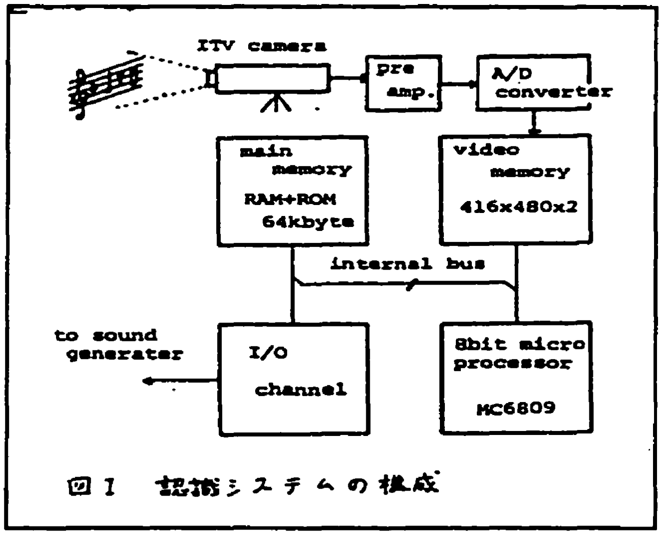


Figure 2.3 Overall architecture (Maenaka and Tadokoro 1983).

Since the maximum address space on an 8-bit processor is 64K bytes, which is not large enough to address the entire image information, a separate independent memory is used for the image. Although the memory had the capacity to store 1024(H) x 512(V) x 4-bit of video information, the camera's hardware limitations resulted in only 416(H) x 480(v) x 2-bit subset of usable memory.

A simple memory access method is devised to access a pixel and its square neighbouring pixels so that filtering, projection calculations, and other basic pattern recognition algorithms can be performed efficiently. The TV camera is equipped with zoom lens and close-up lens is fixed on a camera stand. Three standard 100-watt lamps are used for lighting. Due to camera limitations, sheet music size of A4 format had to be divided into four sections. Adjusting the gain and the bias of the analog-to-digital converter and the lighting eliminated the need to use the histogram method or notchless binary transformation method for preprocessing. A simple fixed binary threshold method was sufficient for successful pattern recognition. Yet, because of the optical characteristics of the close-up lens, the four corners of the images were badly distorted. The paper also discusses the problem of the change in the aspect ratio during the acquisition.

The processing time of the system will be of an order of magnitude slower than if it uses a minicomputer system; hence, an effort was made to keep the processing algorithms simple and to avoid excess access to large image areas. It was decided not to implement expensive algorithms such as high-order pattern matching and spectral analysis.

The following symbols are considered a bare minimum set of music fonts and are used as recognizable objects: treble clef, bar line, double barline, repeat barline, final barline, whole note, half note, quarter note, eighth note, sixteenth note, beamed eighth and sixteenth notes, whole rest, half rest, quarter rest, flat, sharp, natural sign, and dot of prolongation.

In order to find a fragment of the target object, the pattern in the *i*th space of a staff, $S_i(x)$ is defined as:

$$S_i(x) = f(x, y_5(x)) + (2.5 - i)\alpha),$$

$$\text{where } f(x,y) = \begin{cases} 0, \text{ if pixel is white;} \\ 1, \text{ if pixel if black.} \end{cases}$$

$y_5(x)$, is the position of the middle line in the y-direction (the vertical axis), and α is the space between stafflines.

$$P(x) \equiv \sum_{i=0}^{4} S_i(x)$$

counts the number of spaces, at x, contained in the object fragment. $P(x)$ can be used to locate a symbol but it can also be used for classification.

To track the position of the five stafflines the following algorithm is used. $B(i)$ shows the correlation against the position of the current five lines and is defined as

$$B(i) = \sum_{j=1}^{5} \sum_{i=0}^{N} f(x+j, y_5(x) + (2-k)\alpha + i)$$
$$i = \{1, 0, -1\}.$$
$$\text{If } B(1) > B(0) > B(-1) \rightarrow y_5(x+1) = y_5(x) - 1.$$
$$\text{If } B(-1) > B(0) > B(1) \rightarrow y_5(x+1) = y_5(x) + 1.$$

Thus the position of the middle staffline at the next position, $y_5(x+1)$, is incremented or decremented by 1 relative to $y_5(x)$, the current position of the middle staffline.

Because a simple method usually means shorter processing time, the fixed-point sampling method and the Sonde method (counting of black-to-white transitions) are used for recognition of the objects.

The objects are first coarsely classified into three groups. At any point x if $P(x) > 0$ and $\sum_{i=0}^{4} [S_i(x) * S_i(x+1)] > 0$ (to allow for noise), then the object is classified as follows:

Class A if $P(x) = 1$,
Class B if $P(x) = 2$, and
Class C if $P(x) \geq 3$.

To further classify the object, certain number of fixed regions are sampled to find any black pixels. For example, to find eighth rests, six regions are sampled. The six-bit long vector is compared with the standard pattern. If a series of tests fails, the object is considered to be a musical note and proceeds to the next stage. The size of the region for sampling is adjusted according to the size of a staffline height.

2.2.1 Classification of notes

If $P(x)$, which is a note candidate, has the value 1 or 2, it is either stem-less or has stem up (remember that $P(x)$ basically counts spaces that have black pixels in them), so that the smallest i with $S_i = 1$ is chosen as the possible position. If $P(x) \geq 3$, it is considered to be a note with stem down, and thus the largest i with $S_i = 1$ is chosen as the possible position of the notehead.

Given i, there are still three possibilities for the position of the notehead: the notehead can be in the space, on the line above, or on the line below (see Figure 2.4). To precisely determine the position of the notehead, the area below and above the enclosing stafflines is traced.

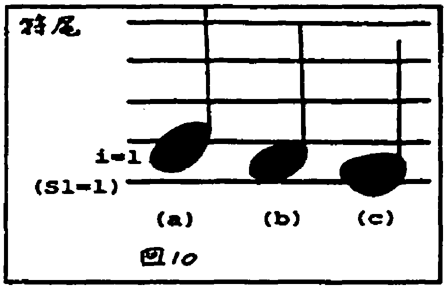


Figure 2.4 Possible position of the notehead (Maenaka and Tadokoro 1983).

The existence of stems and flags can be determined by sampling fixed neighbouring regions. To distinguish between a black notehead and a white notehead, two different algorithms are used depending on whether the note is placed on the staffline or between the stafflines.

For the notehead between two stafflines, the lines equidistant from the two stafflines are scanned from left to right. If the black pixel changes to white before the notehead ends the note is considered white, otherwise it is considered black. For the notehead that is on a staffline, the area around the notehead is scanned vertically to look for black-to-white transition. This scan is performed several times at different positions along the horizontal axis. If only a very small number of vertical scans have the transition, then it is considered black; otherwise it is considered white (see Figure 2.5).

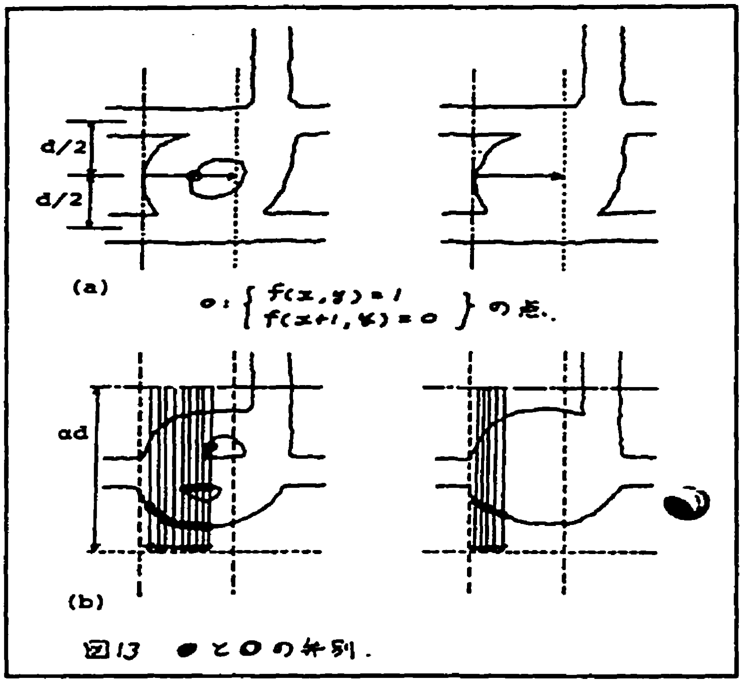


Figure 2.5 Finding white noteheads (Maenaka and Tadokoro 1983).

2.2.2 Classification of beams

When there is a beam, $P(x) \geq 1$, so that the existence of beams must be checked before proceeding with classifications for notes and rests. The vertical sums of black pixels are calculated for regions wider than the width of a notehead. If there is a sudden change in the sum, the position is noted, and $P(x)$ is reduced by one and then passed onto one of the three classes (see Figure 2.6).

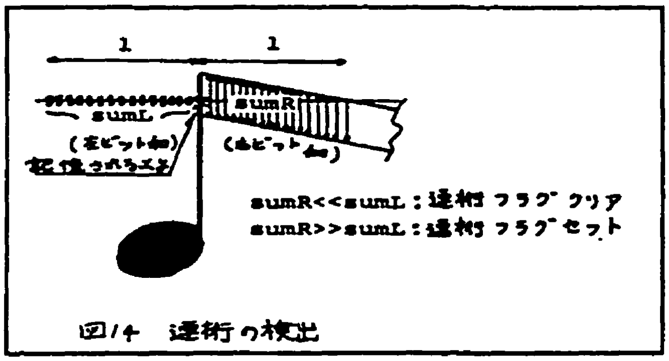


Figure 2.6 Finding beams (Maenaka and Tadokoro 1983).

2.2.3 The output format

As real-time process was not possible and as there was no need to share the data, the output was coded in a way convenient to the sound generating device (MIDI was not yet available).

2.2.4 The experimental results and observations

The various algorithms are coded in Pascal and simulated on a computer system with the same microprocessor; thus it is estimated that it ran probably ten times slower than if everything had been coded in an assembler language and if a specialized memory access method had been used.

2.2.5 Recognition results

Because of the poor quality of the image and the noise, some of the algorithms are not as robust as expected. Also, owing to the large number of parameters involved, such as weights for the fixed sampling and beam windowing width, the correct choices were difficult to find. Further, the values had to be changed depending on the contrast level of the input image. The error rate is reported to be less than 1 error per image (1/4 of page); the accuracy can be increased by increasing the sampling points, but that also results in increase in process time. The process time for 3 measures of music containing 1 quarter note and 23 beamed eighth notes was 4 minutes and 11 seconds. In general, depending on the score, it took 4 to 10 minutes to process one line of monophonic music.

2.3 Kim, Chung, and Bien (1987)

This paper presents a complete OMR system using a TV camera as input and mechanical robot for playback. Unlike the WABOT-2 system (Matsushima 1985), this one is designed to recognize music scores with different font size under poor illumination and without special hardware. The five major processing steps are: preprocessing, coarse classification, fine classification, music syntax check, and interface to music performing device.

The music symbols recognized include: flagged and beamed notes and rests up to 16th note value, treble and bass clef, single and double bar lines, sharp, flat, natural, five simple time signatures, and key signatures up to three accidentals. The system also makes the following assumptions:

1) music symbols are darker than background;
2) music symbols are randomly distributed on the staves; and
3) the distance between two symbols is larger than a quarter of the staffspace.

In preprocessing, an input gray-image is enhanced by the 3x3 Laplacian convolution operator:

$$H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 12 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

to remove blurring between adjacent symbols.

The staff detection algorithm is as follows:

1) Create histogram of average gray-level of horizontal lines.
2) Assign threshold that maximizes the expected value of the between class variance.
3) Label horizontal lines as staffline candidates depending on the threshold.

A gray-level input image is converted to a binary image by adaptive thresholding. At the same time each staff nucleus (staff and symbols belonging to that staff) is separated from the others.

To remove the stafflines, each point x on a staffline, is kept if the vertical neighbourhood satisfies one of three conditions: If only one pixel above is black, or if both of two pixels below are black, or if the four pixels above and four pixels below contain at least five black pixels. Otherwise, the point x is removed.

X-projection is used for symbol segmentation. Coarse classification is performed on each segmented symbol using the height and the width of the minimum bounding box after normalization by staffspace height. The symbols are classified into one of the nine groups. Four of the nine groups or regions in the height/width space (Prerau 1970) need no further processing since there is only one type of symbol within these classes. For the rest of the classes, fixed partial template matches and the Sonde method are used to finalize the classifications of the unknown symbols. Simple music syntax is invoked to check and correct relative duration and pitches of notes.

2.4 Martin and Bellissant (1991)

In the project by Martin and Bellissant (1991) a neural network is used both for staffline removal and connected component object classification.

2.4.1 Skew correction

For the skew correction of stafflines, the concept of chord is introduced. The chord of orientation θ in P is the discrete line segment of slope θ inscribed in connected component C, where P belongs to C (see Figure 2.7).

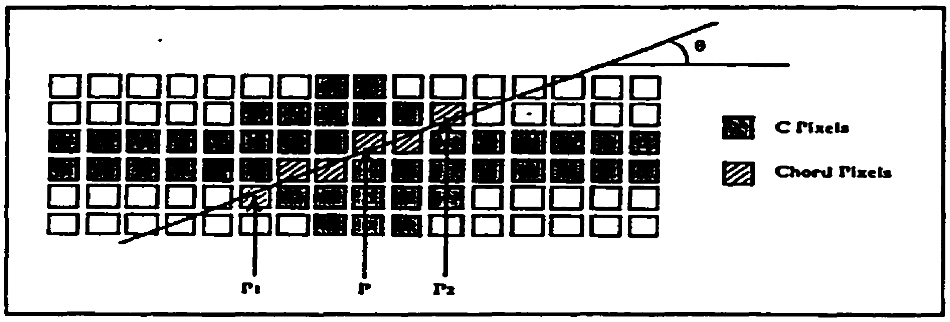


Figure 2.7 Chord of orientation θ in P (Martin and Bellissant 1991).

The chord length $L(P, \theta)$ is defined as the distance between the two boundary points of C that intersect with the chord. In the continuous case, there would be an infinite number of chords of θ at P, but the number is finite in a discrete case, and if one limits θ to be ± a few degrees, the number is greatly reduced.

Assuming that the whole page is skewed at some number of degrees ("less than one degree practically" [Martin & Bellissant 1991b, 418]), all points in the center column of the entire image are considered P and a few values of θ are examined to find P_0 and θ_0 so that $L(P_0, \theta_0)$ is maximized. Then rotation with $-\theta_0$ center at P_0 is applied to the entire image for deskewing. The chord length is calculated using an efficient line-tracing algorithm.

2.4.2 Finding and tracking the staves

Coarse approximation of the position of the staffline is derived by taking the y-projection of the entire unskewed image. This information is used to erase stafflines not overlaid by music symbols. Also, the upper and lower bounds of each staffline are computed, enabling greater accuracy in evaluating the position of the noteheads.

To erase the stafflines, each column is scanned; if a black run-length is found near the position of the y-projection histogram, has similar width and does not belong to a symbol, then it is erased. The problem is how to determine if the black runs belong to a symbol or not. In other words, the black run has the width of the staffline but it may be part of a symbol, e.g., slurs, bass clef, etc. To solve this problem, a larger context is considered. Ideally, if the point does not belong to a symbol, there will only be one "long" chord at the horizontal, i.e., at $\theta = 0$. Yet in practice, due to noise and distortion, the longest chord may not actually occur at $\theta = 0$, so a multi-layered neural network with 228 inputs using gradient back propagation is used to recognize whether a point belongs to a symbol or not. The window used for chord calculation is 50x30 pixels centered at the center of the possible staffline (the black-run). This prevents most of the points belonging to a symbol, but also part of staffline, to be erased. The procedure also leaves some points not belonging to symbols intact. That artifact will be removed at a later stage.

Apparently, the notes are classified by some ad-hoc rule-based system using elliptical shaped template matching. The vertical and horizontal Sonde method is used to count the number of flags and beams attached to noteheads and stems. The other symbols are classified by thinning the symbols which are then processed by another neural net. After a classical thinning operation is performed, some points are marked as endpoints, junction points, and "bending" points. The minimum enclosing rectangle, which has been size-normalized, is arbitrarily partitioned into windows. A set of binary valued variables is used as input to the net. There are two classes of variables. One is of the type (t, w), where t is one of end point, junction point, and bending point, and w is a window. The other type is of the form $(w_i, w_j), i \neq j$, for all i and j, where $(w_i, w_j) = 1$, if at least one segment of the skeleton has one of its extremities in w_i and the other in w_j, otherwise $(w_i, w_j) = 0$.

The neural net used here seems to include a decision-tree building algorithm to include specialized hidden cells that are connected only to certain input cells (features), as well as totally connected hidden cells, those that are connected to all input cells.

The authors conclude, despite the reported 96.5% recognition rate of the net, that "performance in the classification area is less impressive when compared to statistical methods; we noticed, as others before, that a nearest neighbour classifier is usually enough to reach the same recognition rate [as] best multi-layer perceptron.... But it should be noted that nearest neighbour can also be implemented as multi-layer automata networks" (Martin and Bellissant 1991b, 1109).

2.5 McGee and Merkley (1991)

The subject of recognition is lined notation of chant with square neumes (see Figure 2.8). The elimination of four stafflines is performed by finding "sufficiently long" thin horizontal lines. At the same time they are straightened. Classification is performed using a set of bounding rectangles for each neume. The authors have also experimented with a "thin-line coding" method originally developed for fingerprint identification for neume classification. The input resolution is 300 dpi.

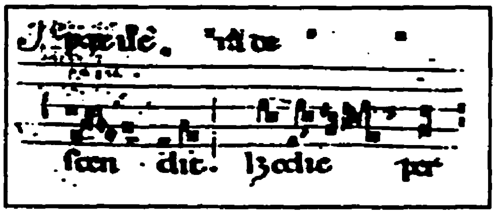


Figure 2.8 Sample notation (McGee and Merkley 1991).

2.6 Sicard (1992)

Sicard uses a rather low-resolution 100 dpi input. The staffline detection uses a y-projection and fails if the skew is more than ±10°. The entire page seems to be rotated and stafflines are removed "using an algorithm similar to [Roach (1988)]." Different algorithms are specialized for different classes of symbols: vertical run-lengths are calculated for finding thick lines (beams); vertical lines (stems and barlines) are located by using the x-projections; accidental identification involves a thinning algorithm; noteheads are localized using "edge detection, break-point extraction, and diameter evaluation methods" (Sicard 1992, 575); and other symbols are identified using templates. Sicard reports an average 97% accuracy, where the 3% error is attributed to notehead location errors, with a process time of about three minutes per page on a Sun SPARC workstation.

2.7 RAMIT (1992)

Yadid-Pecht et al. use a neural network, named RAMIT, to recognize music symbols. The net used is a one-dimensional version of the two-dimensional Neocognitron (Fukushima and Miyake 1982). The Neocognitron is a multi-layered net that has variable connection between the cells in adjoining layers. It is shift-invariant, and selectivity to deformed pattern is adjustable. The net can learn supervised or non-supervised. RAMIT has two hidden layers in addition to the input layer, which presumably responds to each pixel. Layer 1 responds to "horizontal lines of 11x1 pixels and Layer 2 responds to three elements of Layer 1" (Yadid-Pecht et al. 1992, 128). During the preprocessing, the skew of staffline is determined, and coarse rotation of the whole page is performed. For finer adjustment, the stafflines are sheared.

2.8 Miyao et al. (1992)

The two interesting features of this system are that it incorporates a music notation grammar to aid in recognition, and that, unlike most systems, the stafflines are removed after the notes (including noteheads, stems, flag, and beams) are extracted. (The description of the research is available only in Japanese).

Three observations are made about music notation characteristics:

1) The position of the clef, key signature, and time signatures can be predicted from the position of the staff and bar lines.
2) Other symbols, including dots, ties, slurs, tenuto, accent, staccato, fermata, etc., are positioned relative to stems, barlines, and notes.
3) The size of symbols are relative to staffspace height.

The system finds the position of the staves, then notes are searched and removed. After the stafflines are removed, the remaining symbols are coarsely grouped according to their size and position, and symbols are classified by using structural features or template-matching.

A piece-wise linear Hough Transform is used to find the staffline based on the staffline and staffspace height calculated from vertical black and white run-lengths. Bar lines that span two staves are located using x-projections. The black noteheads are extracted using a rectangular mask (staffspace height x width of notehead, which is 2 x staffspace height). The position on the stafflines and another between the stafflines are scanned with the mask. White noteheads are distinguished from black noteheads by the number of white pixels in the mask area. The half note and whole note are distinguished using template matching.

Note candidates found outside of the staff are verified by searching for ledger lines. If no ledger line is found, the candidacy is revoked. Given a notehead, stems are searched by looking at the left and the right edges. If a stem is not found, the note candidacy is rescinded as well. Notational rules such as "no three stems to a notehead" are applied to make sure that recognized symbols are grammatically correct. The number of flags and beams are determined by counting the number of black runs near the stems. After removal of the stafflines, connected components are grouped, by the height, width, and relative position from the middle staffline. All measurements are normalized with staffspace height.

Coarsely grouped fixed-size symbols are further classified using 6x6 meshed templates. The symbol is divided into a 6x6 mesh and each mesh is represented by the ratio of the number of black pixels to white pixels. The thirty-six numbers are represented as a vector and compared with the vectors of prototypes using Euclidean distance measures. The unknown symbol is classified to be the same as the closest prototype above a certain threshold. Unclassified symbols are reconnected by inserting the stafflines that are removed, and then distance calculation is repeated. For size-varying symbols, such as slurs and dynamic hairpins, vertical and horizontal run-lengths are used for classification. Finally, spatial rules are used to finalize the classification decisions.

An accuracy of 93% to 98% with a processing time of 3 to 20 minutes per page using a Sony (NWS-821) workstation is reported. The input scanner has a resolution of 240 dpi.

2.9 Modayur et al. (1992)

The bi-level system described here uses morphological algorithms for symbol detection at a low-level and a high-level module that verifies the output of the low-level module and then incorporates notational syntax to aid in the spatial positioning of the symbols. The authors claim that the recognition task can be performed in near real time and achieves accuracy in excess of 95% on the sample they processed, with a peak accuracy of 99.7% for the quarter and eighth notes.

Some of the assumptions made include:

- The stafflines are equally spaced and there are five lines to a staff.
- The size of the different symbols is relative.
- The image does not have a large skew.
- The notes are proportionally spaced relative to note duration.
- Accidentals are placed directly in front of the note they alter.
- Stems, in general, go down when attached to the left of the note. They go up when attached to the right of the note.
- The stem length is normally the length of one octave.
- A quarter rest is at the center of the staff.
- A half rest touches the third line above, while a whole rest touches the fourth staffline below.

To locate stafflines, the image is opened with a 35-pixel wide horizontal line, but the stafflines are not removed. The structuring elements employed throughout this symbol detection phase would “loosely” follow the shape of the medial axis (the skeleton) of the feature shape being sought. This is done to incorporate a certain degree of tolerance in the detection process. Thus, a few missing foreground pixels, broken edges, blurred corners, etc., do not affect the output of the symbol detection process.

The system is able to recognize twelve symbols: treble and bass clefs, sharp, flat, whole notehead, half notehead, quarter notehead, eighth rest, quarter rest, stem, beam, and half-beam. The system runs on an MVI- Genesis 2000 image processing workstation and takes 2 minutes to process a 512x480 image.

2.10 Kobayakawa (1993)

A very efficient recognition system (10 seconds per page) is described. This is achieved by actively searching for common music symbols. The system consists of Sun SPARC 2 and Omron Luna workstations, the latter being connected to a 200 dpi scanner and a Yamaha DX7 MIDI synthesizer.

To locate the stafflines, thirty-two vertical lines spread across the page are scanned for black runs. Any runs whose length is less than the median of the black run lengths are considered as a candidate for a staffline. For each of these candidates, the image is scanned horizontally and if a horizontal line is found to cover 70% of the score width then that line is considered a staffline. These stafflines are removed if there is a white pixel a certain distance above and below the center of the stafflines.

To locate the black noteheads, the image is scanned horizontally for black runs at staffline positions and center point between the stafflines. Two maxima are found from the histogram of these run lengths. The maximum with few pixels ("about 2 pixels") are considered to come from vertical line segments (stems and barlines) and the second peak ("about 15–18 dots") is assumed to be contributed by black noteheads. In the rhombic (diamond-shape) region around the center of the longer runs, the number of black pixels is counted. If the count is greater than 95% of the region then it is considered to be a black notehead.

The sharp and the natural signs are distinguished from the noteheads by determining that the distance between two nearby vertical line segments are close together. The barlines are separated from other vertical line segments because of their height being the same as the height of the staff or longer. If these barlines are close together they are considered double barlines, in which case, two small dots indicating repeat signs are sought. The remaining vertical lines are considered stems if they are close to a notehead or if there are noteheads between the two endpoints of the line segment.

After the stems are removed, the side opposite to the noteheads is scanned in the vertical direction to look for flags or beams. If any black pixels are found, a connected component is assembled. If the width of the component is less than twice the width of the notehead and the slant (presumably the angle of the line connecting midpoints of the left and the right edges of the component) is steep, then it is considered a flag.

The remaining symbols are recognized using template-matching. These templates are prepared from various example scores, edited with a bit-map editor, then encoded in run-length format. The reported recognition rates are:

Scenes from Childhood, op. 15/6 (Schumann):	99.6%
Fantasie-Impromptu, op. 66 (Chopin):	98.3%
Turkish March (Mozart):	94.8%

2.11 Roth (1994)

The system consists of the following seven steps.

2.11.1 Rotation

To correct skews, the image is rotated by shearing horizontally and vertically. The actual amount of shearing is determined manually.

2.11.2 Vertical run-length statistics

The median lengths of vertical runs of black and white pixels are used to estimate the staffline height (from black runs) and the staffspace height (from white runs). The size of all the staves on a page is assumed to be the same.

2.11.3 Locate and delete stafflines

The stafflines are located by searching for groups of five peaks in the y-projection, then they are tracked from the middle outwards to get accurate y-position in each image column. This operation corrects slightly skewed or bent stafflines. Once located, the stafflines are deleted from the image. In order not to affect symbols too much, lines are deleted only when their width is close to the overall staffline height.

2.11.4 Locate and delete vertical lines

By examining the x-projections of each staff, vertical lines are located. This task is refined later through application of the technique of mathematical morphology. Note that any vertical line segments (thin objects) are removed, which include stems, bar lines, and lines within sharps, flats, and naturals.

2.11.5 Connected component labeling

The remaining components are identified. A list of components and references from each pixel to the component it belongs to is created. "A fixed space above and below the staff is included in the region of interest, the total height of the region is three times the staff height. This allows for recognition of up to four ledger lines. For this region connected components are derived" (Roth 1994, 18).

2.11.6 Symbol recognition

Before symbols are classified, "separated white notehead (due to staffline removal) are merged and connected black noteheads (due to chords) are separated using heuristics" (Roth 1994, 19). In addition, Roth employs a fairly complex decision tree to classify various music symbols using the following features: height, width, area, and center of gravity. The location with respect to other components, vertical lines, and stafflines is also taken into consideration.

2.11.7 Lipsia document generation

Finally the recognized element is reproduced using the Lipsia music notation editor. Preliminary but successful use of mathematical morphology operators is also reported.

2.12 Summary

Although many innovative OMR systems have been developed over the last decade, there are major limitations to their use as practical OMR. As mentioned, the number of different music symbols commonly used exceeds four hundred, yet, most of the available programs can recognize no more than a few dozen symbols. This is a serious limitation because these programs are not designed to learn new symbols. The lack of learning capability limits the recognition of handwritten music as well. The automatic recognition of well-formed handwritten music will be extremely useful for musicians. The AOMR described here overcomes these limitations by incorporating a flexible learning mechanism thus enabling it to recognize virtually unlimited numbers of music symbols, including handwritten manuscripts.

3. TECHNICAL BACKGROUND

3.1 Pattern recognition system

In general, a pattern recognition process consists of three major phases (see Figure 3.1).

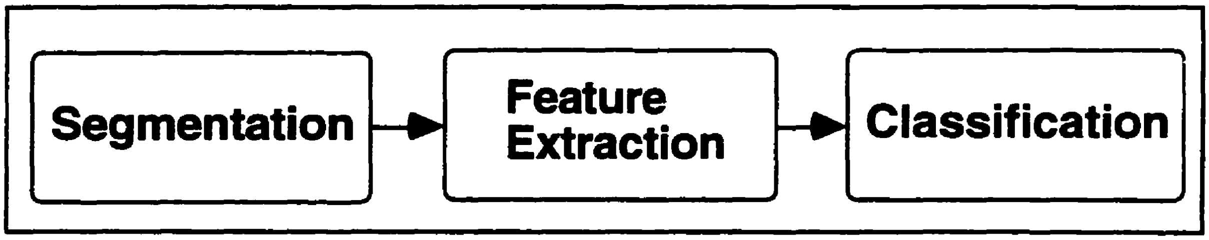


Figure 3.1 Pattern recognition system.

In the segmentation phase, objects to be classified must be found and isolated from the rest of the scene. This is accomplished by partitioning a digital image into disjoint (non-overlapping) regions. Features are sets of the measurable properties of a given symbol, such as size and shape. The feature extraction phase measures these properties, producing a set of measurements called feature vector. A decision regarding the classes to which the object belongs is made during the classification phase. Classification is based on the features vector.

3.2 Pattern recognition system design

During the designing stage of a pattern recognition system, strategies and algorithms to be used for each of the three phases in pattern recognition must be determined.

3.2.1 Object locator

An object locator is a set of algorithms that isolates the images of the individual objects in the complex scene. In music recognition this is not a trivial problem. Stafflines connect most of the symbols. Also, there are some music symbols that are made up of disconnected components: for example, bass clef, fermata, and octavo lines. Furthermore, many symbols such as beamed notes are made up of more elementary objects: noteheads, stems, and beams. In AOMR, run-length coding (Section 3.5), projections (Section 3.6), and connected component analysis (Section 3.7) are used along with other specialized algorithms to remove the stafflines then segment the symbols.

3.2.2 Feature selection

Feature selection involves deciding which features best distinguish among the various object types and should thus be measured. (For features considered in AOMR see Section 3.9 below.) The procedure of selecting "good" features is not formalized; as Castleman states: "frequently intuition guides the listing of potentially useful features" (Castleman 1979, 321). Cover and Van Campenhout (1977) rigorously showed that in determining the best feature subset of size *m* out of *n* features, one needs to examine all possible subsets of size *m*. For practical consideration, some non-exhaustive feature selection methods must be employed. Many methods exist for finding near-optimal solutions to this problem in a finite time, such as sequential forward selection, sequential backward elimination (Kittler 1978), and branch and bound algorithm (Narendra and Fukunaga 1977, Hamamoto et al 1990). The latter method guarantees the optimal features subset without explicitly evaluating all possible feature subsets under the assumption that the criterion function used satisfies the "monotonicity" property. Unfortunately, in AOMR there is no guarantee that this constraint, or even the more relaxed "approximate monotonicity" (Foroutan and Sklansky 1987) can be met. Furthermore, although branch and bound can reduce the search space drastically, the calculation may become impractical in cases where there are many features (more than 10–20). It should also be noted, however, that Hamamoto et al (1990) have shown that the "monotonicity" constraint need not be satisfied in order to obtain successful results in practice.

The problem becomes more complex as Cash and Hatamian (1987) have shown. The weighting of each feature used in a similarity measure can markedly improve the recognition rate. In other words, the optimal use of features involves not only choosing the correct subset of the features but how much of each feature should contribute to the final decision. In the branch and bound method, the goal was to find a set of binary weights for the features (0 or 1), but the problem now is to determine the weights which can be any real number. In AOMR, the genetic algorithm (3.11) is used to find the near-optimal set of weights from this infinite possibility.

3.2.3 Classifier

Designing a classifier consists of establishing a mathematical basis for the classification procedure and selecting the type of classifier structure.

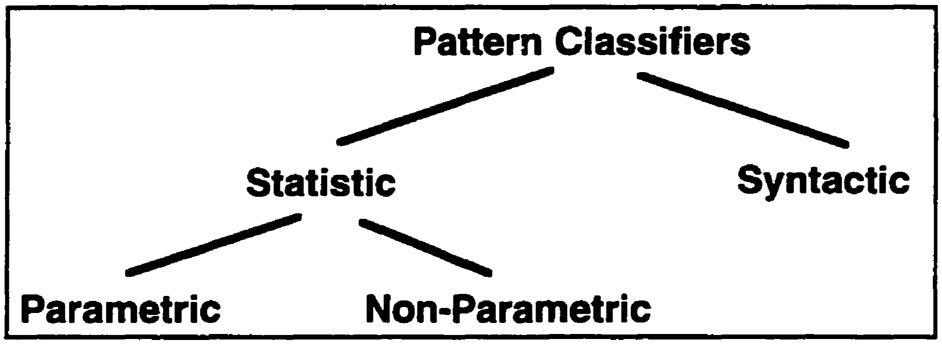


Figure 3.2 Different types of pattern classifiers.

There are two major types of pattern classifiers: syntactic and statistic. The latter can be further divided into parametric and non-parametric classifiers and any of them can be trained with or without supervision (Figure 3.2).

Syntactic pattern classification explicitly exploits the composite nature of a shape in the classification process. Syntactic pattern classification is based on obtaining a grammar relating certain strings of patterns to each other. For example, a grammar can be constructed for describing an eighth note consisting of a notehead, a stem, and a flag.

Statistical classification is based on a statistical measure of shapes. A classifier that assumes a probability distribution function of a given sample is called a parametric classifier. The Bayes classifier is an example of parametric classifiers. The non-parametric classifiers do not assume any probability distribution functions of the given sample. The k-NN classifier described below falls into this category.

3.2.3.1 Classifier training

Once the basic decision rules of the classifier have been established, the particular threshold values that separate the classes must be determined. This is generally done by training the classifier on a group of known objects called the training set. A number of objects from each class, previously correctly identified, constitutes the set. The measurement space is partitioned by decision lines that minimize the error of the classifier when tested with the training set. The idea is that if the training set is representative of the objects to be encountered in the field, then the classifier should perform about as well on the real objects as it did in the training set.

3.2.3.2 Performance evaluation / Error-rate estimators

The process of learning requires a method of evaluation or self-monitoring. A learning system must be able to evaluate its own performance so that it can be improved. Here the leave-one-out error rate estimator is used to evaluate the expected error rate of the classifier. This estimator is a special case of the general class of cross-validation error estimates. In k-fold cross-validation, the known objects are randomly divided into k-mutually exclusive partitions of approximately equal size. The objects not in the test partition are independently used for training and the resulting classification is tested on the corresponding test partition. The average error-rates over all k partitions is the cross-validation error-rate. Thus, when k is one, every sample in the training set is classified using all the other samples in the set.

3.3 Nearest neighbour classifier

Loftsgaarden and Quesenbery (1965) proposed a very useful and simple method for non-parametric estimation of the probability density function p(X) of a random variable X from N observations of X. This method is known as the k-NN method. The application of this method to the classification problem is the k-NN rule that classifies an observation with unknown classification by assigning it to the class most heavily represented among its k-nearest neighbours.

3.3.1 Bayes probability of error

Let each of the objects to be classified belong to one of M classes denoted by $C_i, i = 1,2,\ldots,M$. Let $P(C_i)$denote the *a priori* probability of occurrence of objects belonging to class C_i. Let $\mathbf{x} = (x_1,x_2,\ldots,x_d),\mathbf{x} \in E^d$denote the set of *d* measurements (features) made on an object and let $p(X|C_i)$ denote the probability density function of $\mathbf{x}$ given that the pattern on which $\mathbf{x}$ was observed belongs to class C_i. Then it is well known that the decision rule that minimizes the expected probability of error (misclassification) in making a decision on $\mathbf{x}$is to choose class C_i if:

$$p(\mathbf{x}|C_i)P(C_i) > p(\mathbf{x}|C_j)P(C_j) \text{ for all } j \neq i.$$

It is also known that the resulting Bayes probability of error, which is optimal, meaning that the error is the smallest possible, is given as:

$$P_e = 1 - \int_{-\infty}^{\infty} \max_i [p(\mathbf{x}|C_i)P(C_i)] d\mathbf{x}.$$

To be able to use the above Bayes decision rule it is required to know the *a priori* probabilities $P(C_i)$ and the class conditional probability density functions $p(\mathbf{x}|C_i)$ for all *i*.

3.3.2 Non-parametric classification

Non-parametric decision rules, such as the k-NN rule, are attractive because no *a priori* knowledge is required concerning the underlying distributions of data. In the non-parametric classification problem, we have available a set of *n* feature vectors taken from a collected data set of *n* objects (the set of pre-classified samples) denoted by $\{\mathbf{x},\Theta\} = \{(\mathbf{x}_1,\theta_1),(\mathbf{x}_2,\theta_2),\ldots,(\mathbf{x}_n,\theta_n)\}$, where $\mathbf{x}_i$ and θ_i denote, respectively, the feature vector on the *i*th object and the class label of the *i*th object. The labels θ_i are assumed to be correct and are taken from integers $\{1,2,\ldots,M\}$, i.e., the patterns may belong to one of the *M* classes.

3.3.3 Nearest neighbour rule

The nearest neighbour search consists in finding the closest point to a query point among N points in a *d*-dimensional space. The NN rule assigns an unclassified sample to the same class as the nearest *n* stored, correctly classified sample. The only means by which the NN method can improve its performances, given a similarity measure, is by increasing the number of training set patterns: these then have to be stored and compared individually with any test patterns presented to the system. The most interesting theoretical property of the NN rule is that, for any metric, and for a variety of loss functions, large-sample risk incurred is less than twice the Bayes error.

Let $\mathbf{x}$ be a new object (feature vector) to be classified and let $\mathbf{x}_k^* \in \{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\}$ be the feature vector closest to $\mathbf{x}$, where closeness is measured by some similarity measure such as Euclidean distance between $\mathbf{x}$ and $\mathbf{x}_k^*$ in E^d. The nearest neighbour rule classifies the unknown object to class θ_k^*. Let $P_e^n(NN) = \Pr\{\theta \neq \theta_k^*\}$ denote the resulting probability of misclassification (error), where θ is the true class of X, and let $P_e(NN)$ denote the limit of $P_e^n(NN)$ as $n \to \infty$. I has been shown by Cover and Hart (1967) that as $n \to \infty$, the nearest neighbour error is bounded in terms of the Bayes error by:

$$P_e \leq P_e(NN) \leq P_e\left[2 - M\left(\frac{P_e}{M-1}\right)\right].$$

Thus, the probability error of the NN-rule is bounded above by twice the Bayes error. Therefore the asymptotic probability of error of the NN rule is close to optimal. (Asymptotic here refers to a very large number of samples). Furthermore, using a suitable modification such as the k-NN rule, one can decrease the probability of error to closer to the optimal.

The main criticism directed at the NN method is the large amount of storage and the resulting computation involved because it stores all the sample data. Thus there has been considerable effort in "editing" or "thinning" the data in an attempt to store only a subset of it. Some of these techniques are described below.

3.4 Modified k-NN classifiers

The apparent necessity to store all the data and the resulting excessive computational requirements have discouraged many researchers from using the rule in practice. In order to combat the storage and computation problems, many researchers, starting with Hart (1968), propose schemes to "edit" the original data so that fewer feature vectors need be stored. These schemes are based on the idea of selecting a small representative subset of the training set so that NN classification with the reduced subset achieves a performance that is close to or better than the performance of NN classification with the complete set.

3.4.1 Condensed k-NN

The editing procedure creates a decision boundary defined by a small number of samples belonging to the outer envelopes of the clusters. Clearly, samples that do not contribute to defining the boundary—e.g., those deeply imbedded within the clusters—may as well be discarded with no effect on subsequent performance. This is the idea behind the condensing technique first suggested by Hart (1968).

The goal of condensing is to construct a consistent subset which, when used as a stored reference set for the k-NN rule, correctly classifies all remaining points in the sample set. The following algorithm creates a consistent subset:

1) Setup two bins STORE and GRABBAG.
2) The first sample is placed in STORE.
3) The second sample is classified by the NN rule, using as a reference set the contents of STORE. If the second sample is classified correctly it is placed in GRABBAG; otherwise it is placed in STORE.
4) Proceeding inductively, the *i*th sample is classified by the current contents of STORE. If classified correctly it is placed in GRABBAG, otherwise it is placed in STORE.
5) After one pass through the original set, the procedure continues to loop through GRABBAG until termination, which can occur in one of two ways:
   a) GRABBAG is exhausted.
   b) One complete pass through GRABBAG with no transfer to STORE.

3.4.2 Edited k-NN

Edited k-NN was introduced by Wilson (Wilson 1972, Wagner 1973), criticized by Penrod and Wagner (1977) and modified by Devijver and Kittler (1980). An editing algorithm is used to reduce the number of pre-classified samples and to improve the performance of the rule:

For each *i:*

1) Classify sample S_i, using k-NN rule as though it has not been classified.
2) If S_i is mis-classified then discard it.

Thus the edited k-NN edits out "poor" samples and not only reduces storage requirements of the k-NN for the future classification of unlabeled samples but also claims to have a better asymptotic performance. After the criticism of Penrod and Wagner (1977), mostly on Wilson's leave-one-out procedure, Devijver and Kittler (1980) modified it based on "holdout" or partitioning technique:

1) Make a random partition of the sample set into N subsets $S_1, S_2, \ldots, S_N$.
2) Classify the patterns in S_i using $S_{(i+1) \bmod N}, i = 1, 2, \ldots, N$.
3) Discard all the patterns from the sample that were mis-classified at step 2.

Furthermore, they suggested the multi-editing method where the algorithm above is repeated until the last iteration produces no editing.

3.4.3 Other improvements

Dudani (1976) introduced a k-NN rule called the distance-weighted k-NN rule. This is a k-NN classification rule with the facility to weigh more heavily the evidence of samples nearer to the unknown observation. This is intuitively appealing and promised more accurate results, albeit at the expense of more computation overhead.

In a recent paper, Parthasarathy and Chatterji (1990) showed that for large sample-size problems, the best performance of the traditional k-NN rule with a mechanism to resolve ties (either by randomly choosing the winner or by finding one more neighbour to break the tie) is comparable to the performances of Dudani's classifier and is preferred because of the improved computational efficiency.

The use of the k-NN rule in practical applications has been frequently ruled out because of the storage and computational complexity. The difficulty can be partly remedied by fast algorithms for searching nearest neighbours.

In the effort to make the computation more efficient, Ramasubramanian and Paliwal (1992) have proposed an algorithm, based on work by Vidal (1986), to reduce the amount of distance calculations when searching for nearest neighbours. By pre-calculating the distance between all the library points and some arbitrary-fixed anchor points in the space, then using triangle-inequality, much of the distance calculations between the unknown sample and the stored samples can be eliminated. Experimental results show a savings of over 90% in calculation time. The penalty for this method is increase in storage of $O(n(m+1))$, where m is the number of anchor points used. Because of the rather large n used in AOMR, probably a small m would be preferable. If $m = 1$, it will be possible to order the vectors for an even faster search.

3.4.4 Voronoi diagram and Gabriel graph

Optimal selection of those samples that define the boundary with the complete set can be obtained using Voronoi diagrams. Unfortunately construction of a Voronoi diagram is quite demanding in terms of storage and computational complexities. A similar Gabriel graph can be used which seems to exhibit performance similar to the Voronoi diagram, yet is much less demanding with respect to storage and computation. The worst case for Voronoi diagram calculation for *n* elements in *d* dimensions will take at least $O\left(n^{[d/2]}\right)$ time, while computation time for the Gabriel editing algorithm is between $O\left(dn^2\right)$ and $O\left(dn^3\right)$ (Bhattacharya et al. 1992).

3.5 Run-length coding

Run-length coding is a simple data compression method where a sequence of identical numbers is represented by the number and the length of the run. For example, the sequence {3 3 3 3 5 5 9 9 9 9 9 9 9 9 9 9 9 9 6 6 6 6 6} can be coded as {(3, 4) (5, 2) (9, 12) (6, 5)}. In a binary image, used as input for the recognition process here, there are only two values: one and zero. In such a case, the run-length coding is even more compact, because only the lengths of the runs are needed. For example, the sequence { 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1} can be coded as {7, 4, 13, 8, 2}, assuming 1 starts a sequence (if a sequence starts with a 0, the length of zero would be used). By encoding each row or column of a digitized score the image can be compressed to about one tenth of the original size. Furthermore, by writing programs that are based on run-length coding, a dramatic reduction in processing time can be achieved.

3.6 Projections

Projections are the count of black pixels along parallel lines. Here, only the count along the vertical lines (x-projections) and horizontal lines (y-projections) are used (see Figure 3.3).

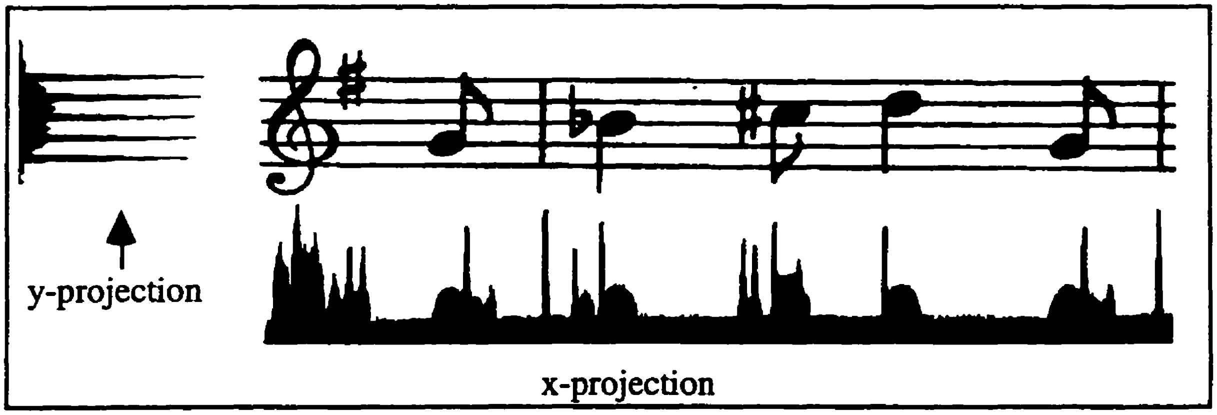


Figure 3.3 X- and y- projections.

The generalized projection transform, called Radon transform of $g(x, y)$ at (s, θ), for the two-dimensional case is:

$$[Rg](s,\theta) = \int g(s\cos\theta - u\sin\theta), s\sin\theta + u\cos\theta)du.$$

This is the integral of g along a line that passes through the point $(s\cos\theta, s\sin\theta)$ with slope $-\mathrm{ctn}\theta$ (Herman 1979, 81–104). When θ is $\pi/2$ and 0, the transforms result in x- and y-projections, respectively:

$$[R,g](s,\pi/2) = \int g(u,s)du \quad \text{and} \quad [R,g](s,0) = \int g(s,u)du$$

In the discrete case, given $P(i, j)$ of an m x n digital image, the equations above become:

$$X(j) = \sum_{i=0}^{m} P(i,j), \quad 0 \le j \le n \quad \text{and} \quad Y(i) = \sum_{j=0}^{n} P(i,j), \quad 0 \le i \le m$$

In the early part of this research, the projections were used extensively for the music recognition process. Currently, the projections are used only during the process of staffline detection.

3.7 Connected component

The connected component is an important concept in image segmentation when determining if a group of pixels is considered to be an object. A connected set is one in which all the pixels are adjacent or touching. The formal definition of connectedness is as follows:

> Between any two pixels in a connected set, there exists a connected path wholly within a set.

Thus, in a connected set, one can trace a connected path between any two pixels without ever leaving the set.

Point P of value 1 (in a binary image) is said to be 4-connected if at least one of the immediate vertical or horizontal neighbours also has the value of 1. Similarly, point P is said to be 8-connected if at least one of the immediate vertical, horizontal, or diagonal neighbours has the value of 1 (see Figure 3.4).

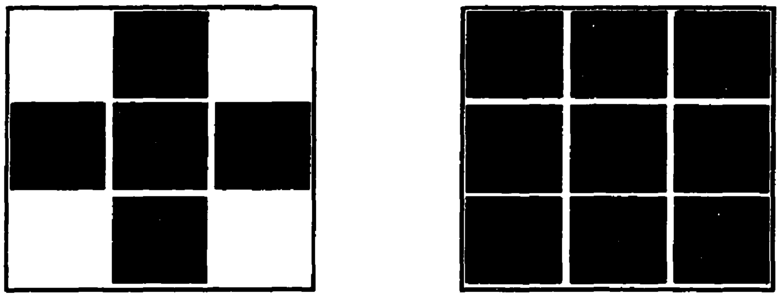


Figure 3.4 Possible neighbours of 4- and 8-connected components.

Two algorithms to find connected components in a binary image are explained below. The first method requires two scans but is simple. The second method, the one that is currently implemented in AOMR requires only one scan, but recursion is involved.

3.7.1 Method 1: Two-pass connected component labeling

The main task is to label each point in each component with a unique value. In the first scan, for each black pixel P, the three neighbouring pixels above and the left-hand pixel of P are examined (see Figure 3.5).

1) If all four are not labeled; P gets a new label;
2) if only one of them is labeled, then P gets that label; or
3) if two or more are labeled, then P gets one of the labels and the fact that the labels are equivalent is recorded (i.e., they belong to the same component).

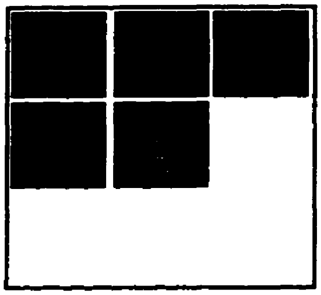


Figure 3.5 Pixels examined on the first scan.

At the end of the first scan, every black pixel has a label, and labels in different 8-connected components are guaranteed to be different. Within a component, however, there may be several different labels. The equivalent pairs that were recorded are sorted into equivalent classes and one label is chosen, arbitrarily, to represent that class, and therefore the component. In the second scan each point in a component will receive the same unique number (see Figure 3.6).

```
Input
    A A   A
A A A       A   A
  A     A     A

After scanning the first row:
    B B   C
A A A       A   A
  A     A     A

After scanning the second row:
    B B   C
D D D       C   E  (D = B)
  A     A     A

After scanning the third row:
    B B   C
D D D       C   E
  D     F     C    (E = C)

After the second scan:
    B B   C
B B B       C   C
  B     F     C
```

Figure 3.6 Two-pass connected component labeling.

3.7.2 Method 2: Depth-first tree traversal

Since the entire page is converted to vertical run-length representation in AOMR, an algorithm to find connected components using this representation was developed.
The goal of this analysis is to label each pixel of a connected component with a unique number. This is usually a time-consuming task involving visiting each pixel twice, labeling and re-labeling (see above). By using graph theory (depth-first tree traversal) and the vertical black run-length representation of the image, the processing time for finding connected components can be greatly reduced.

Here is the overall algorithm:

1. All vertical runs are first labeled, UNLABELED.
2. Start at the leftmost column.
3. Start at the first run in this column.
4. If the run is UNLABELED, do a depth-first search.
5. If not last run, go to the next run and repeat Step 4.
6. If not last column, go to next column and repeat Step 3.

The basic idea, of traversing the tree structure, is to find all runs that are connected and label them with a same number. A run X on column n is a father to another run Y, if Y is on the next column (n + 1) and X and Y are connected. Y is called a child of X. In a depth-first search, all children of a given father are searched first recursively, before finding other relatives, such as grandfathers. Note that, a father can have any number of sons and each son may have any number of fathers. Also, by definition of run-length coding, no two runs in the same column can be connected directly. The result is a representation of the image that is run-length coded and connected-component labeled, providing an extremely compact, convenient, and efficient structure for subsequent processing.

3.8 Features

Features are sets of the measurable properties of a given symbol. The feature extraction phase measures these properties, producing a set of measurements called a feature vector. There are many special characteristics of music scores that can be exploited to select appropriate features that may aid in the classification. Scores are often shared in the orchestra and in the church, and therefore tend to be rather large and have gross and global graphical features so that they can be read from a distance. The scores are also meant to be read in real time; thus, they are designed to be read quickly which also led the designers of music symbols to concentrate on global features rather than on details.

The following features are currently used in the AOMR system: width; height; area of the object (Ao); and area of the bounding box (Ab = width x height); rectangularity: Ao / Ab, which represents how well an object fills its bounding box; aspect ratio: width / height, which can distinguish slender objects from roughly square or circular objects; number of holes, and normalized central moments which provide a more detailed numerical description of the shape.

Other potential features are listed below but are not currently implemented (Figure 3.7). One of the reasons they are not currently implemented is that most of these require boundary points. Because boundaries in many music symbols can be noisy and broken, features involving boundary points were thought to be unreliable. But if these boundaries can be smoothed (by filters), or if the broken parts of symbols can be restored before features are extracted, then features below, involving boundary points, should become useful.

Perimeter: length of boundary

Radii: R_{min}, R_{max} are the minimum and the maximum distances, respectively, to boundary from the center of mass

Eccentricity or *elongation:* R_{max} / R_{min}

Euler number: number of connected region - number of holes

Roundness or *compactness:* $\gamma = \frac{(perimeter)^2}{4\pi(area)}$,
for a disc, γ is minimum and equals 1

Fourier descriptors

Chain coding

Figure 3.7 Features not used in AOMR.

3.8.1 Moment

Moment is one of the main features used in AOMR and it has many attractive attributes. The moment techniques have an appealing mathematical simplicity and are very versatile. The method of moments provides a robust technique for decomposing an arbitrary shape into a finite set of characteristic features. In general, moments describe numeric quantities at some distance from a reference point or axis. Moments are commonly used in statistics to characterize the random variable distribution and in mechanics to characterize bodies by spatial distribution of mass. Here, the image is considered to be a two-dimensional density distribution function. Moments have a very interesting property that can be stated in the following theorem.

Moment Representation Theorem:

> *The infinite sets of moment* $\{m_{pq}, p, q = 0, 1, \ldots\}$ *uniquely determine* $f(x, y)$ *and vice versa.*

What this means is that any image can be completely described by an infinite series of numbers. In practice this is not feasible, yet being able to obtain a series of numbers, especially the low-order moments that describe a shape, is nonetheless very useful. In fact, the low-order moments tend to describe more global shape characteristics than higher-order moments which tend to be noisy and unreliable shape descriptors in digital images.

Prokop and Reeves state that "a major strength of this approach is that it is based on a direct linear transformation with no application-specific 'heuristic' parameters to determine." On the other hand, "a major limitation of the moment approach is that it can only be directly applied to global shape identification tasks" (Prokop and Reeves 1992, 458). This fits precisely with the objectives of music symbol recognition where global shape is the most distinguishing feature, as opposed to, for example, alphabets or Chinese ideograms where the details are more important. The objects of recognition using moments in other machine classification systems include aircraft (Dudani et al 1977), ships (Smith and Wright 1971), buildings, and bridges (Gilmore and Boyd 1981). Note that these objects are classifiable by global shapes.

3.8.1.1 Cartesian moment definition

The two-dimensional Cartesian moment, m_{pq}, of *order* $p+q$, of a density distribution, $f(x,y)$, is defined as

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y) dxdy.$$

The two-dimensional moment for a $(M \times N)$digitized image with discrete density distribution $g(x,y)$, is

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

A moment set of order n consists of all moments, m_{pq}, so that $p+q \leq n$ and contains $\frac{1}{2}(n+1)(n+2)$ elements.

Various types of moments are available (orthogonal, rotational, and complex moments, as well as moment invariance). Here, relatively simple normalized central moments are used as only the size and location invariance is needed for music symbols; orientation invariance is not required.

3.8.1.2 Properties of moments

The low-order moment values represent well-known fundamental geometric properties of a distribution or a body.

3.8.1.2.1 Zeroth-order moments: Area

The definition of the zeroth-order moment m_{00}, of the image $g(x,y)$,

$$m_{00} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)$$

represents the total mass or the area, if $g(x,y)$ is binary, of the given image.

3.8.1.2.2 First-order moments: Centre of mass

The first order moments, $\{m_{10}, m_{01}\}$, are used to locate the center of mass of the object. The coordinates of the center of mass $(\bar{x},\bar{y})$ are given by

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

If the object is moved so that the centre of mass is at (0, 0), then the moments computed for that object are referred to as central moments and are designated by μ_{pq}. The central moment of order $(p+q)$becomes

$$\mu_{pq} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} (x-\bar{x})^p (y-\bar{y})^q g(x,y).$$

(Note that $\mu_{10} = \mu_{01} = 0$.)

The normalized central moments denoted by η_{pq} are invariant to size:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}},$$

where

$$\gamma = \frac{p+q}{2} + 1, \quad p+q = 2,3,\ldots$$

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

3.8.1.2.3 Second-order moments

The second-order moments, $\{m_{02}, m_{11}, m_{20}\}$, known as the moments of inertia, can be used to determine the principal axes of the object, where the principal axes may be described as the pair of axes about which there are the minimum and the maximum second moment. Other useful object features involving the second-order moments include:

Orientation:

$$\theta = \frac{1}{2}\tan^{-1}\left[\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right]$$

Oriented bounding rectangle: the smallest rectangle enclosing the object that is also aligned with its orientation.

Best-fit ellipse: the best-fit ellipse is the ellipse whose second moment equals that of the height.

Eccentricity : indicates the distribution of the mass.

$$\varepsilon = \frac{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}}{area}$$

Radii of Gyration : "the radii of gyration about the origin is the radius of a circle centered at the origin where all the mass may be concentrated" (Prokop and Reeves 1992, 440):

$$ROG_{com} = \sqrt{\frac{\mu_{20} + \mu_{02}}{\mu_{00}}}$$

3.8.1.2.4 Higher-order moments

The two third-order central moments, $\{\mu_{30}, \mu_{03}\}$, describe the skewness of the image projection. Skewness is a classical statistical measure of a distribution's degree of deviation from symmetry about the mean. Two of the fourth-order central moments, $\{\mu_{40}, \mu_{04}\}$, describe the kurtosis of the image projection. Kurtosis is a classical statistical measurement of the "peakedness" of a distribution.

3.8.1.3 Moment computation

In the actual software implementation of moment calculation the following equalities are used to drastically decrease computation time:

$$\begin{aligned}
\mu_{00} &= m_{00} \\
\mu_{20} &= m_{20} - \bar{x}m_{10} \\
\mu_{02} &= m_{02} - \bar{y}m_{01} \\
\mu_{11} &= m_{11} - \bar{y}m_{10} \\
\mu_{30} &= m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2 m_{10} \\
\mu_{12} &= m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2 m_{10} \\
\mu_{21} &= m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01} \\
\mu_{03} &= m_{03} - 3\bar{y}m_{02} + 2\bar{x}^2 m_{10}
\end{aligned}$$

3.9 Similarity measure

Once the features of the objects are measured and assembled into a vector, a method to compare these vectors for "similarity" is needed. There are many ways to define "similarity" or "closeness" of two vectors. Since these are subjective terms, the similarity measure that results in accuracy and efficiency is chosen. Unlike other classifiers, where one measure is decided in advance, for adaptability purposes many different measures can be implemented in AOMR. Hence different measures can complement each other in classification design (in terms of confidence levels). In different environments some measures may be more useful than others.

3.9.1 Common metrics

Three common metrics used are called City-block, Euclidean, and Chessboard, these being special cases of the Minkowsky metric which is defined as:

$$d_p(x,y) = \left\{ \sum_{i=1}^{n} |x_i - y_i|^p \right\}^{1/p}.$$

Note: The variable x represents the known vectors in the stored library and y represents the unknown vector to be classified.

City-block (p = 1)

$$d_1 = \sum_{i=1}^{n} |x_i - y_i|$$

Euclidean (p = 2)

$$d_2 = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2}$$

Chessboard (p = ∞)

$$d_\infty = \underset{i \in N}{Max} |x_i - y_i|$$

Another metric proposed by Chaudhur et al. (1992) is defined as:

$$d_N = |x_{ixy} - y_{ixy}| + \frac{1}{n - \left\lfloor \frac{n-2}{2} \right\rfloor} \sum_{\substack{i=1 \\ i \neq ixy}}^{n} |x_i - y_i|,$$

where $|x_i - y_i|$ is maximum for $i = ixy$, and

$\lfloor a \rfloor$ indicates the integral part of a, i.e., the largest integer $\leq a$.

The following similarity measures require statistics about the existing feature vectors already in the library.

3.9.2 Mahanalobis distance

$$d_h = \sum_{i=1}^{n} \frac{(\bar{x}_i - y_i)^2}{\sigma_i^2}$$

This measure (Cash & Hatamian 1987, 303) is attractive because the number of comparisons required is constant regardless of the size of the library.

3.9.3 Weighted normalized cross correlation

(Cash & Hatamian 1987, 303)

$$R = \frac{\sum_{i=1}^{n} w_i x_i y_i}{\sqrt{\sum_{i=1}^{n} w_i x_i^2 \sum_{i=1}^{n} w_i y_i^2}},$$

where w_i are the weights.

Some of the possible definitions for the weight are:

$$w_i = \frac{1}{\bar{\sigma}_k}, w_i = \bar{\sigma}_k, w_i = \frac{\sigma_i}{|x_i|}, \textit{ and } w_i = \frac{\sigma_i}{\bar{\sigma}_i}.$$

3.9.4 The problem of evaluating weights

The weights can be used in measures other than the weighted normalized cross correlation (3.9.3). For example, weighted Euclidean distance can be defined as:

$$d_2 = \left(\sum_{i=1}^{n} w_i (x_i - y_i)^2 \right)^{1/2},$$

where w_i are the weights.

Those features that are found to be more reliable than others should be given more importance when making classifications. The idea behind this is to try to make the intra-class distance as small as possible. For the Euclidean distance measure, weights can be adjusted so that the more reliable features make larger contributions to the distance between two feature vectors. The problem now is how to select the appropriate weighting factors.

Determining which weights will result in the most accurate classification is an extremely compute-intensive task, for the optimal set can only be obtain by examining all possible combinations (Foroutan and Sklansky 1987). Fortunately, the task can be performed both through background processing and by using idle resources of workstations on a network. The exhaustive search for optimal set of weights, however, remains intractable (testing with five different values for weights for all features would take several thousand years on the fastest workstations available today). Some improvements can be made to speed up this calculation as described below, yet, the vast improvement for this problem came from applying the genetic algorithm in the selection process as explained in Section 3.10.

3.9.5 Reducing similarity measure computation time

As Bryant (1989) notes, it is almost never necessary to finish the distance calculations, since the current minimum distance is known. In summation-type similarity measures, one can exit the loop when the running total exceeds the minimum distance already calculated.

By reordering the feature vector in descending values of the weights, further increase in the efficiency of the calculations can be obtained, since the features with higher weights will contribute more to the final distance than those with smaller weights.

3.10 Genetic algorithms

Genetic algorithms (Holland 1975, Davis 1987, Goldberg 1989) are used here to find the optimal set of weights for the feature vectors during distance calculations. With the benefit of this algorithm, the entire AOMR system has a greater chance of survival. It allows the system to find, within a reasonable amount of time, the near-optimal set of weights, whereas under normal circumstances, the exhaustive search would take too long to find such a set.

Genetic algorithms are currently used in problem-solving systems based on computational models of the evolution of individual structures via processes of selection and reproduction. More precisely, genetic algorithms maintain a population of individuals that evolve according to specific rules of selection and other operators such as crossover and mutation. Each individual in the population receives a measure of its fitness in the environment. Selection focuses attention on high-fitness individuals, thus exploiting the available fitness information. Since the individual's genetic information (chromosomes) is represented as arrays of binary data, simple bit manipulations allow the implementation of mutation and crossover operations.

The entire process may be described as follows (see Figure 3.8):

1) Evaluate the fitness of all the individuals in the population.
2) Select parents, recombine the "genes" of the selected parents to produce offspring.
3) Perturb the mated population stochastically (mutation).
4) Discard the old population and iterate using the new population.

Each individual in the population is evaluated for its fitness using a fitness function. Given a particular individual, the fitness function returns a single number; this is the primary place in which the traditional genetic algorithm is tailored to a specific problem.

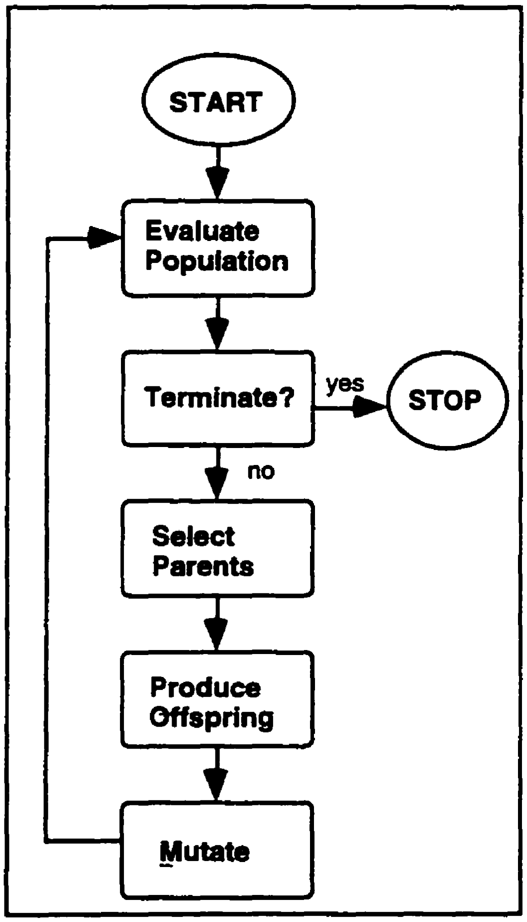


Figure 3.8 Overall process of genetic algorithm.

During the reproductive phase of genetic algorithms, parents are selected and mated, producing offspring that will comprise the next generation. A selection operator is used to favor the fittest parents for reproduction. High-fitness individuals may be used several times for reproduction and low-fitness individuals may not be used at all. When two parents are selected, their chromosomes are recombined to produce new offspring using crossover and mutation operators.

Crossover operators exchange substrings of two parents to obtain two offspring. The purpose of the crossover operator is to combine useful parental information to form new and, one hopes, better performing offspring. Such an operator can be implemented by choosing a point at random, called the crossover point, and exchanging the segments to the right of this point. For example, let

```
Parent 1 = a1 a2 a3 a4 : a5 a6 a7
Parent 2 = b1 b2 b3 b4 : b5 b6 b7
```

and suppose that the crossover point has been chosen randomly as indicated by the colon. The resulting offspring would be:

```
Child 1 = b1 b2 b3 b4 : a5 a6 a7
Child 2 = a1 a2 a3 a4 : b5 b6 b7
```

Crossover rate is the probability per individual of undergoing recombination.

Mutation randomly alters each gene with a small probability, typically less than 1%. This operator introduces innovation into the population and helps prevent premature convergence on a local maximum. The evolution is terminated when the population attains certain criteria such as simulation time, number of generations, or when certain percentages of the population share the same function value.

Genetic algorithms have been successfully applied to solve many optimization and other computationally intensive problems (Davis 1991). In music, genetic algorithms have been used for timbral design (Horner et al. 1992, Horner et al. 1993, Takala et al. 1993, Vuori and Välimäki 1993) and as a compositional aid to generate pitch patterns (Horner and Goldberg 1991).

4. DESCRIPTION OF THE PROGRAM

In this chapter, general workings of the AOMR software is described. The program is divided into seven sections:

1. Staff removal
2. Text removal
3. Segmentation
4. Feature extraction
5. Classification
6. Score reconstruction
7. Learning phase

Given an optically scanned page of a music score, the system first locates and removes the staves. The textual materials, such as lyrics and expression markings are also removed. The remaining symbols on the page are then located and separated from one another for classification. The classification is dependent on the shape of each symbol. The numerical descriptions of the shape are called features, the calculation of which is called the feature extraction. Once the features of the symbol are determined, they are used for classification, which means assigning symbol names to unknown objects. The score is then reconstructed to visually verify the accuracy of the classifier. Finally, the system attempts to improve its performance in the learning phase.

4.1 Staff detection and removal

One of the initial challenges in any OMR systems is the treatment of the staves. For musicians, stafflines are required to facilitate reading the notes. For the machine, however, they become an obstacle by making the segmentation of the symbols very difficult. The task of separating background from foreground figures is a unsolved problem in many machine pattern recognition systems in general.

There are two approaches to this problem in OMR systems. One way is to try to remove the stafflines without removing the parts of the music symbols that are superimposed. The other method is to leave the stafflines untouched and devise a method to segment the symbols (Carter 1989, Fujinaga 1988).

In the AOMR system described here, the former approach is taken, that is, the stafflines are carefully removed, without removing too much from the music symbols. This decision was taken basically for three reasons: 1. Symbols such as ties are very difficult to locate when they are placed right over the stafflines. (See Figure 4.1). 2. One of the hazards of removing stafflines is that parts of music symbols may be removed in the process. But due to printing imperfection or due to damage to the punches that were used for printing (Fujinaga 1988), the music symbols are often already fragmented, without removing the stafflines. In other words, there should be a mechanism to deal with broken symbols whether one removes the stafflines or not. 3. Removing the stafflines simplifies many of the subsequent steps in the recognition process.

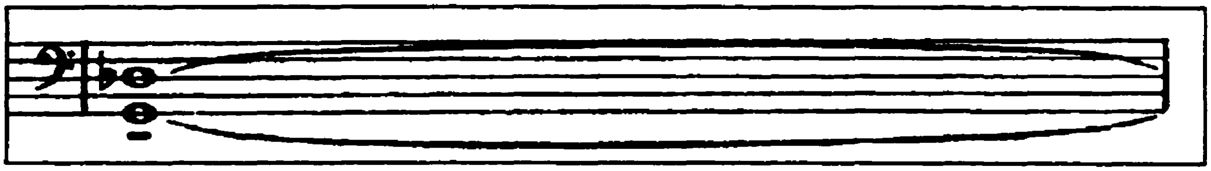


Figure 4.1 Tie superimposed over staff.

4.1.1 The complexity of the process

The following procedure for detecting and removing staves may seem overly complex, but it was found necessary in order to deal with the variety of staff configurations and distortions such as skewing.

The detection of staves is complicated by the variety of staves that are used. The five-line staff is most common today, yet the "four-line staff was widely used from the eleventh to the thirteenth century and the five-line staff did not become standard until the mid-seventeenth century, (some keyboard music of the sixteenth and seventeenth centuries employed staves of as many as fifteen lines)" (Gardner 1979, 28). Today, percussion parts may have one to several lines. The placement and the size of staves may vary on a given page because of an auxiliary staff, which is an alternate or correction in modern editions (Figure 4.2); ornaments staff (Figure 4.3); ossia passages (Figure 4.4), which are technically simplified versions of difficult sections; or more innovative placements of staves (Figure 4.5). In addition, due to various reasons, the stafflines are rarely straight and horizontal, nor parallel to each other. For example, some staves may be tilted one way and another on the same page or they may be curved.

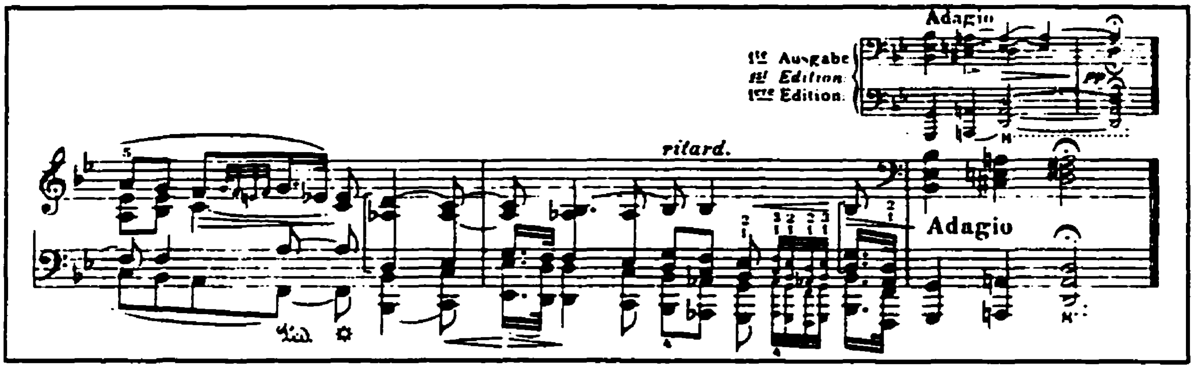


Figure 4.2. An example of an auxiliary staff.

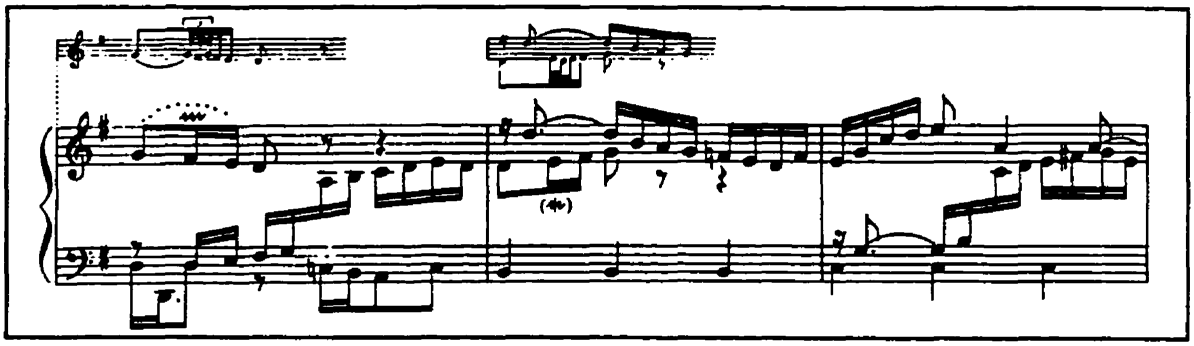


Figure 4.3. An example of ornament staves.

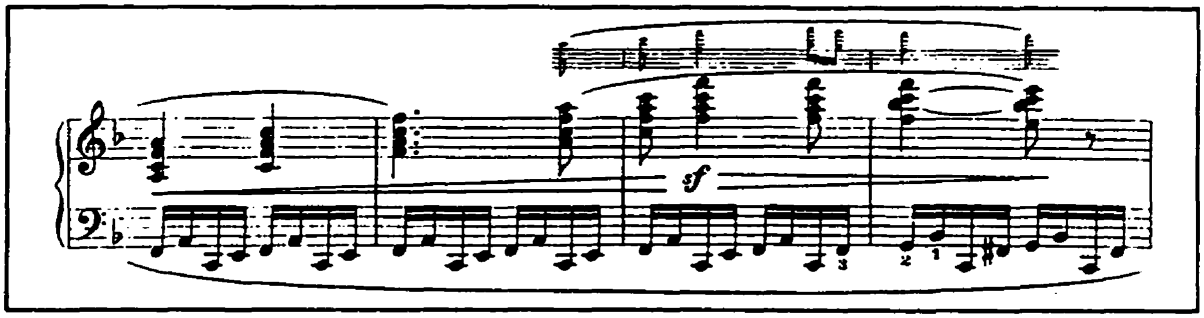


Figure 4.4. An example of ossia staff.

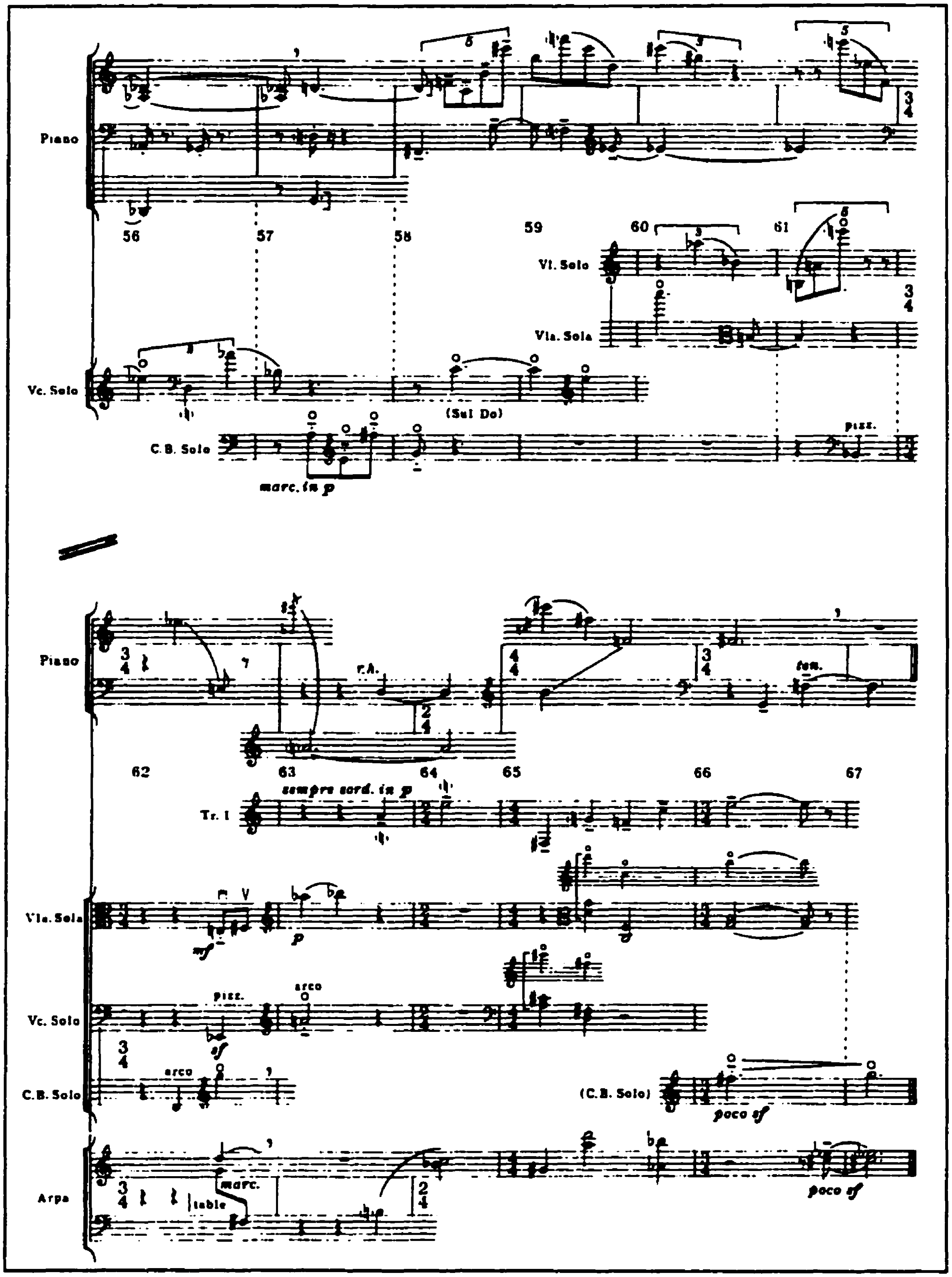


Figure 4.5. An example of innovative staff layout.

4.1.2 The reliability of staffline_height and staffspace_height

In order to design a robust staff detector that can process a variety of input, one must proceed carefully, not making too many assumptions. There are, fortunately, some reliable factors that can aid in the detection process.

The thickness of stafflines, the **staffline_height**, on a page is more or less consistent. The space between the stafflines, the **staffspace_height**, also has small variance within a staff. This is important, for this information can greatly facilitate the detection and removal of stafflines. Furthermore, there is an image processing technique to reliably estimate these values. The technique is the vertical run-lengths representation of the image.

If a bit-mapped page of music is converted to vertical run-lengths coding, the most common black-runs represent the staffline_height (Figure 4.6) and the most common white-runs represents the staffspace_height (Figure 4.7). Even in music with different staff sizes, there will be prominent peaks at the most frequent staffspaces (Figure 4.8). These estimates are also immune to severe rotation of the image. Figure 4.9 shows the results of white vertical run-lengths of the music used in Figure 4.8 rotated intentionally 15 degrees. It is very useful and crucial, at this very early stage, to have a good approximation of what is on the page. Further processing can be performed based on these values and not be dependent on some predetermined magic numbers. The use of fixed threshold numbers, as found in other OMR systems, makes systems inflexible and difficult to adapt to new and unexpected situations.

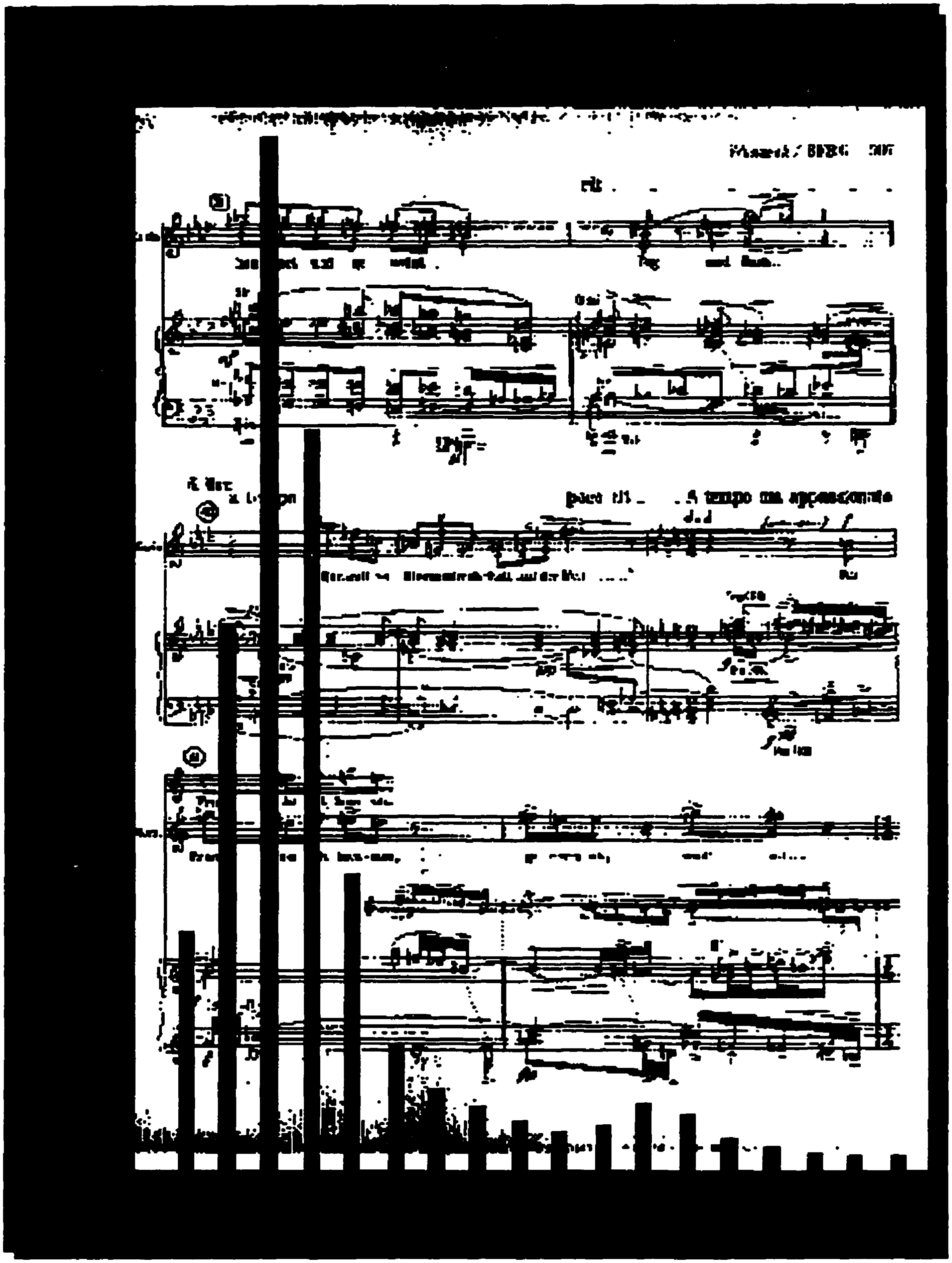


Figure 4.6 Estimating staffline_height by vertical black runs. The graph shows that the staffline_height of 4 pixels is most prominent.

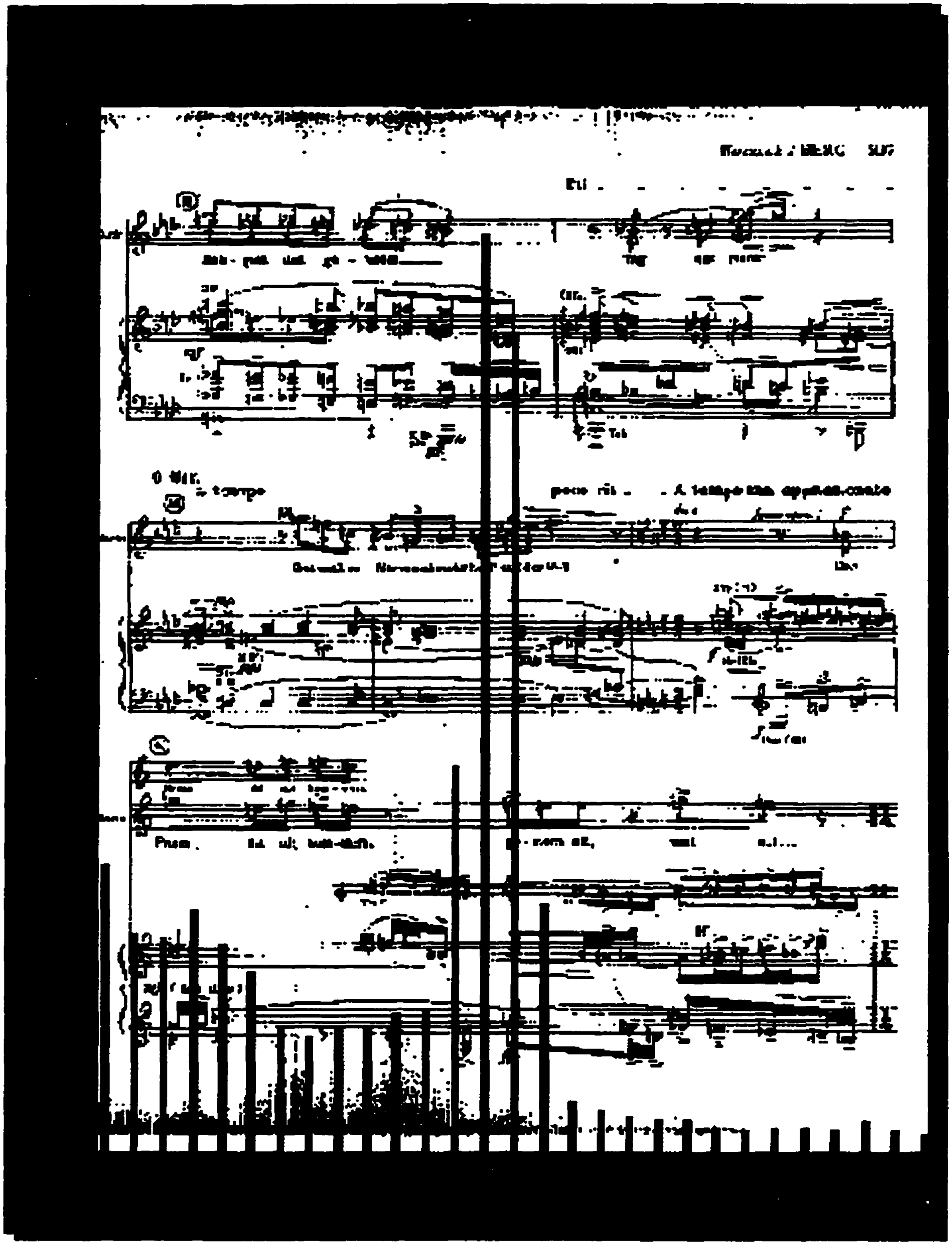


Figure 4.7 Estimating staffspace_height by vertical white runs. The graph shows that the staffspace_height of 14 pixels is most prominent.

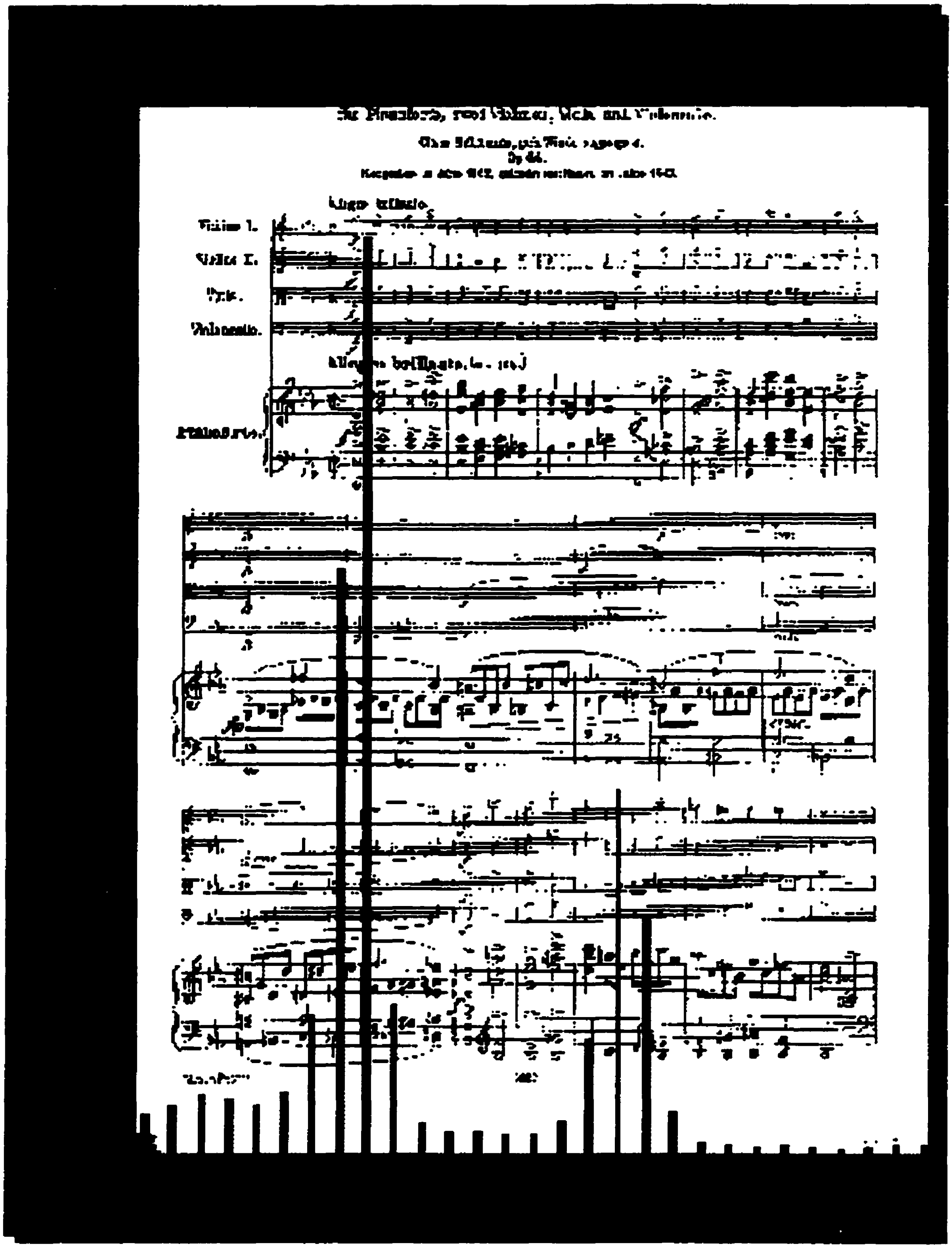


Figure 4.8 Estimating staffspace_height by vertical white runs with multiple-size staves.

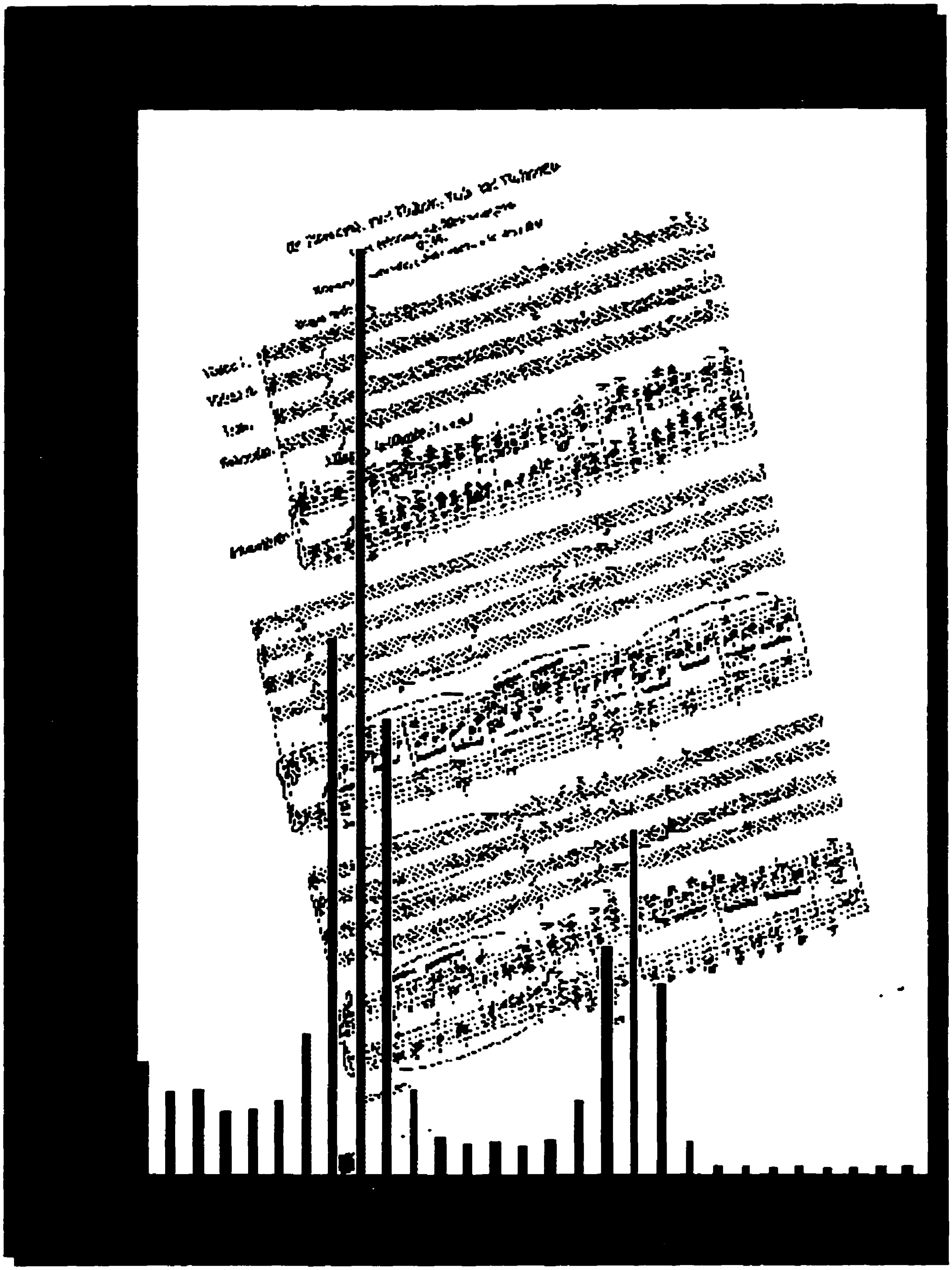


Figure 4.9 Estimating staffspace_height by vertical white runs of a skewed image. The music used in Figure 4.8 is rotated 15 degrees.

4.1.3 The process

The locations of the staves must be determined before they can be removed. The first task is to isolate stafflines from other symbols to find the location of the staves. Any vertical black runs that are more than twice the staffline_height are removed from the original. (See Figure 4.11, Figure 4.10 is the original). A connected component analysis is then performed on the filtered image and any component whose width is less than staffspace_height is removed (Figure 4.12). These steps remove most objects from the page except for slurs, ties, dynamics wedges, stafflines, and other thin and long objects.

The difference between stafflines and other thin objects is the height of the connected component; in other words, the minimal bounding boxes that contain slurs and dynamics wedges are typically much taller than the minimal bounding box that contains a staffline segment. Removing components that are taller than staffline_height, at this stage, will potentially remove stafflines because if the page is skewed, the bounding boxes of stafflines will also have a height taller than the staffline_height. Therefore, an initial de-skewing of the entire page is attempted. It is hoped that this would correct any gross skewing of the image. Finer local de-skewing will be performed on each staff later. The de-skewing, here, is a shearing action; that is, a part of the image is shifted up or down by some amount. This is much simpler and a lot less time-consuming than true rotation of the image, but the results seem satisfactory. Here is the algorithm:

1. Take the narrow strip (currently set at 32 pixels-wide) at the center of the page and take a y-projection. Make this the reference y-projection.
2. Take a y-projection of the adjacent vertical strip to the right of the center strip. Shift this strip up and down to find out the offset that results in the best match to the reference y-projection. The best match is defined as the largest correlation coefficient, which is calculated by multiplying the two y-projections.
3. Given the best correlated offset, add the two projections together and make this the new reference y-projection. The offset is stored in an array to be used later.
4. If not at the end of the page, go back to Step 2.
5. If the right side of the page is reached, go back to Step 2, but this time move from the center to the left side of the page.
6. Once the offsets for the strips of the entire page are calculated, these offsets are used to shear the entire image. (See Figures 4.13 and 4.14).

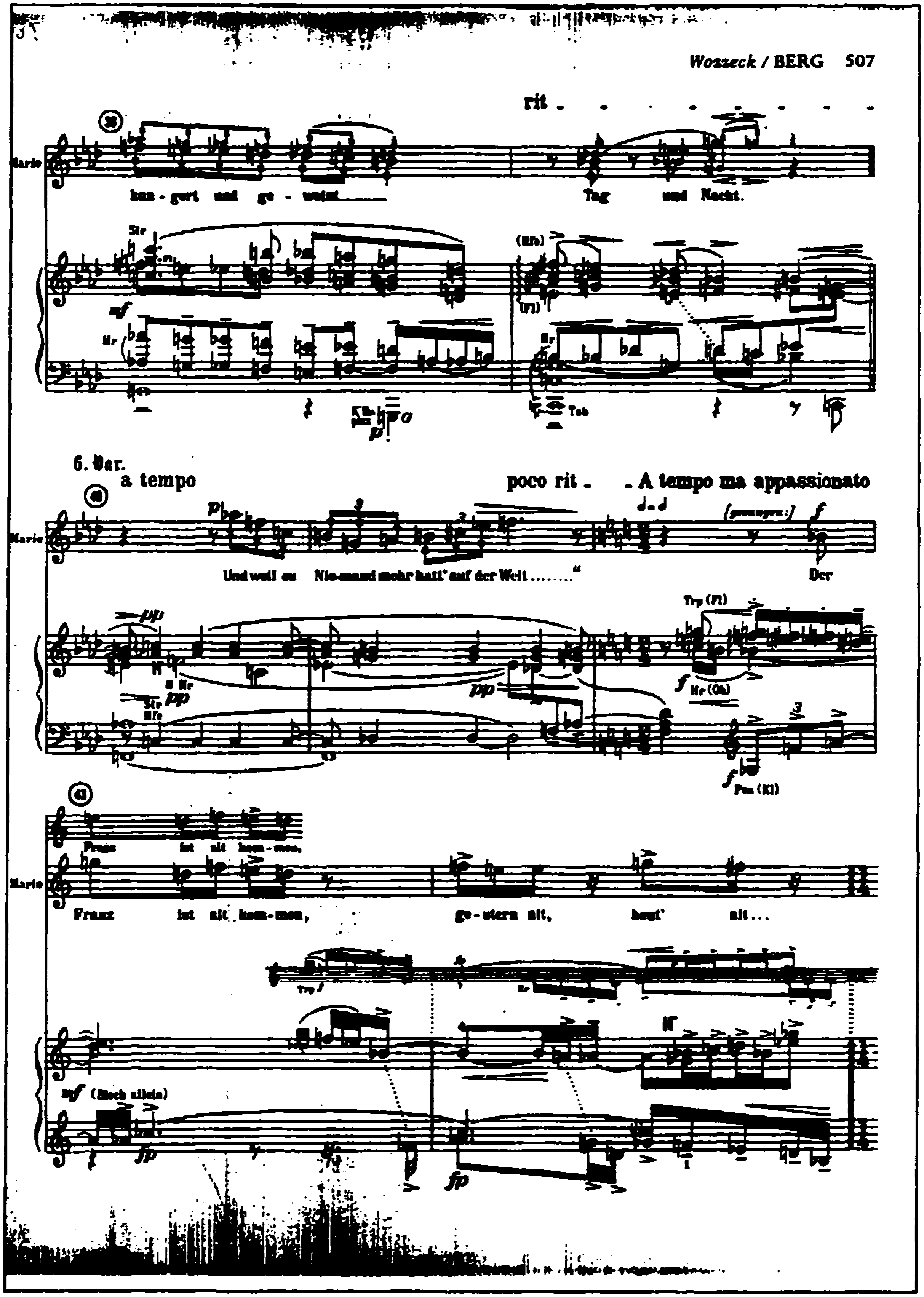


Figure 4.10 The original.

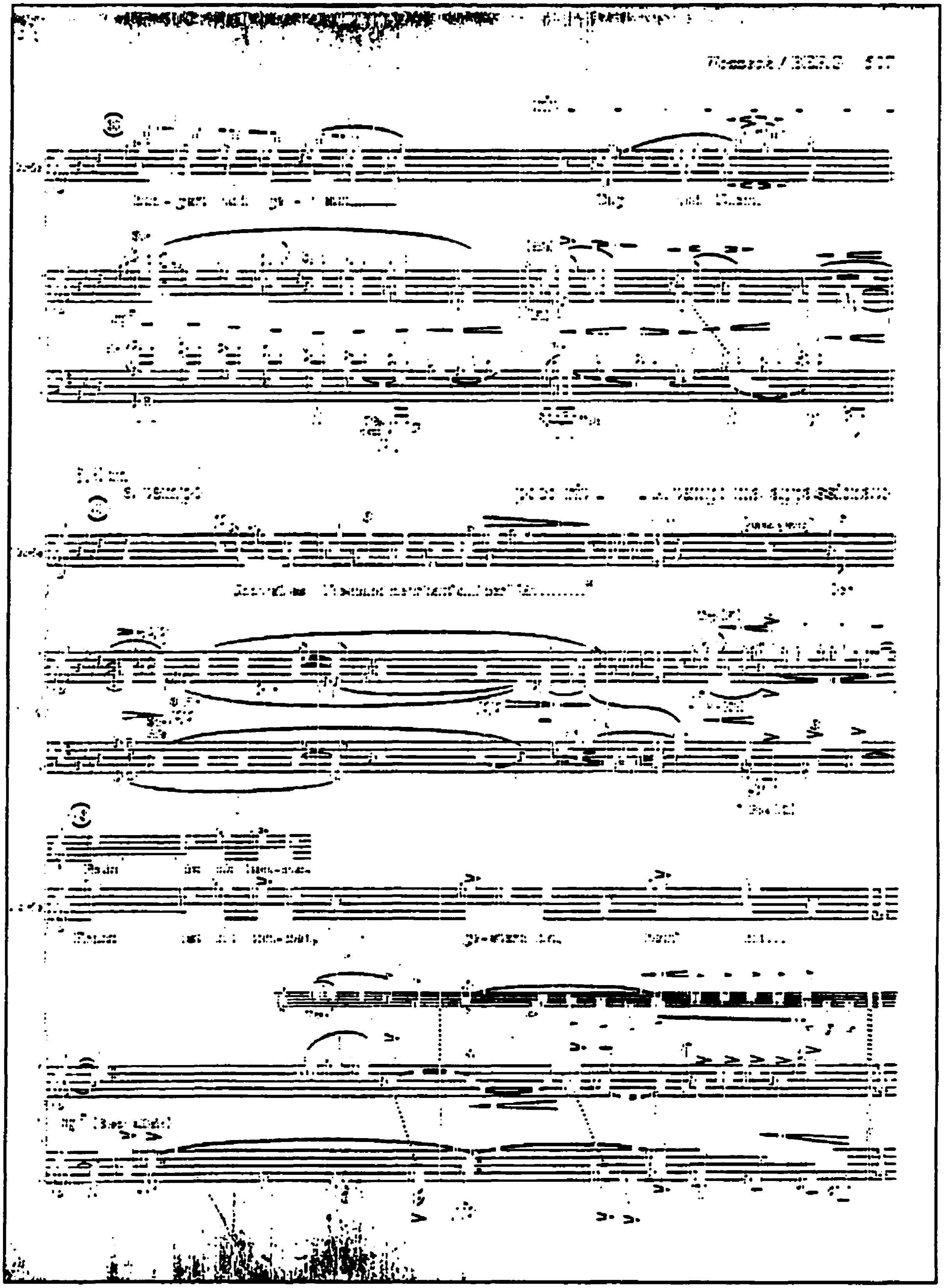


Figure 4.11 Vertical black runs more than 2 x staffline_height removed.

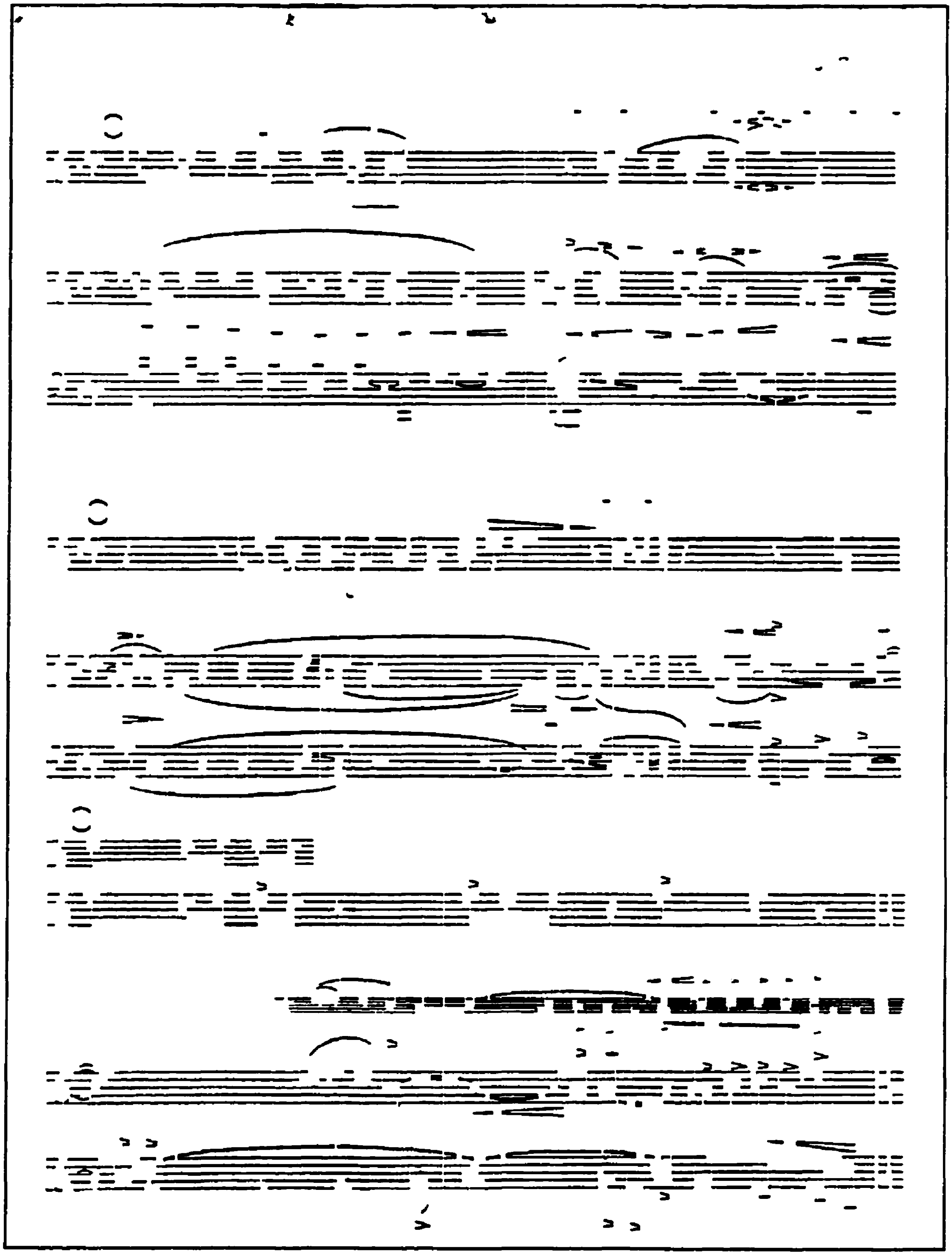


Figure 4.12 Connected-components narrower than staffspace_height removed.

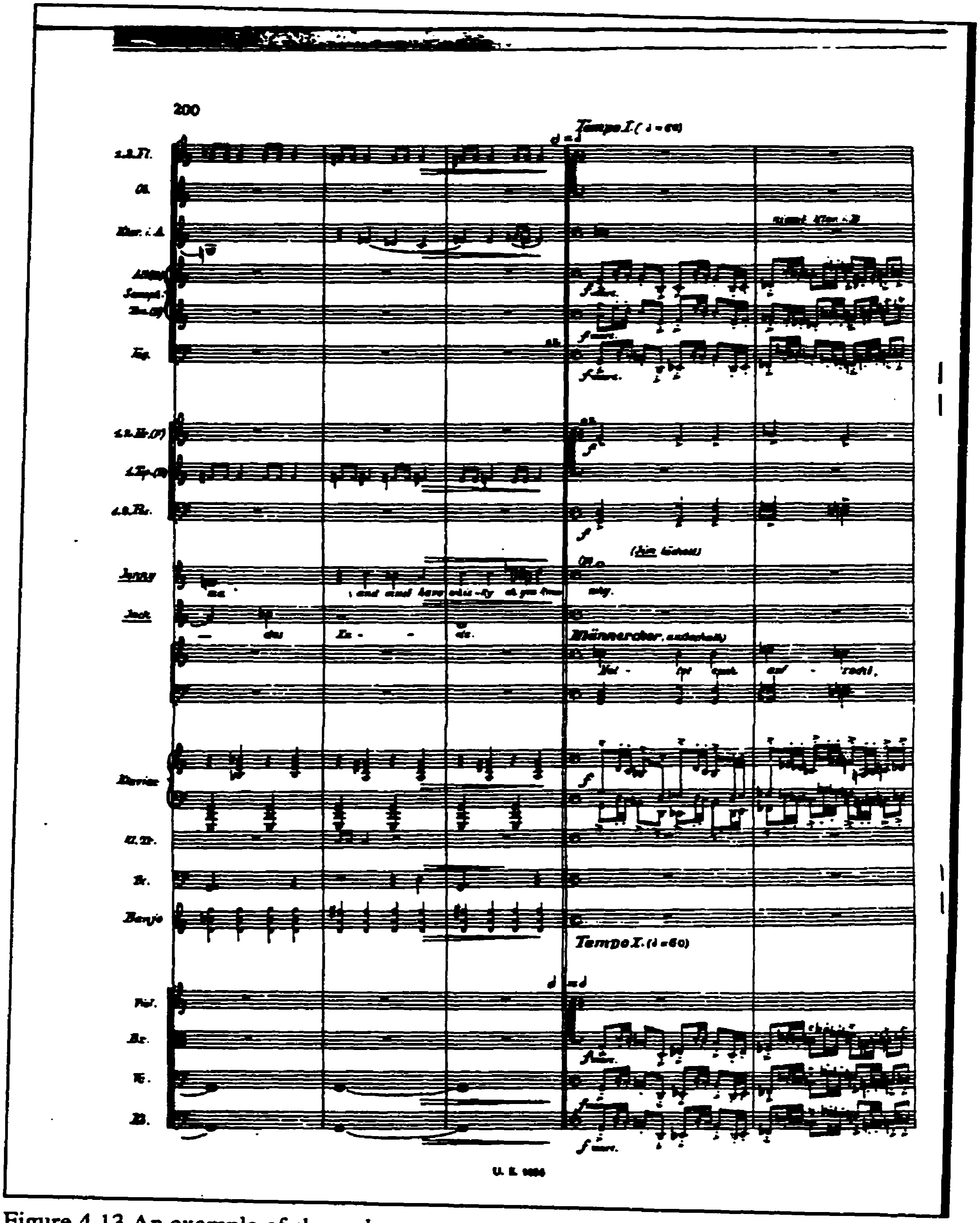


Figure 4.13 An example of skewed page.

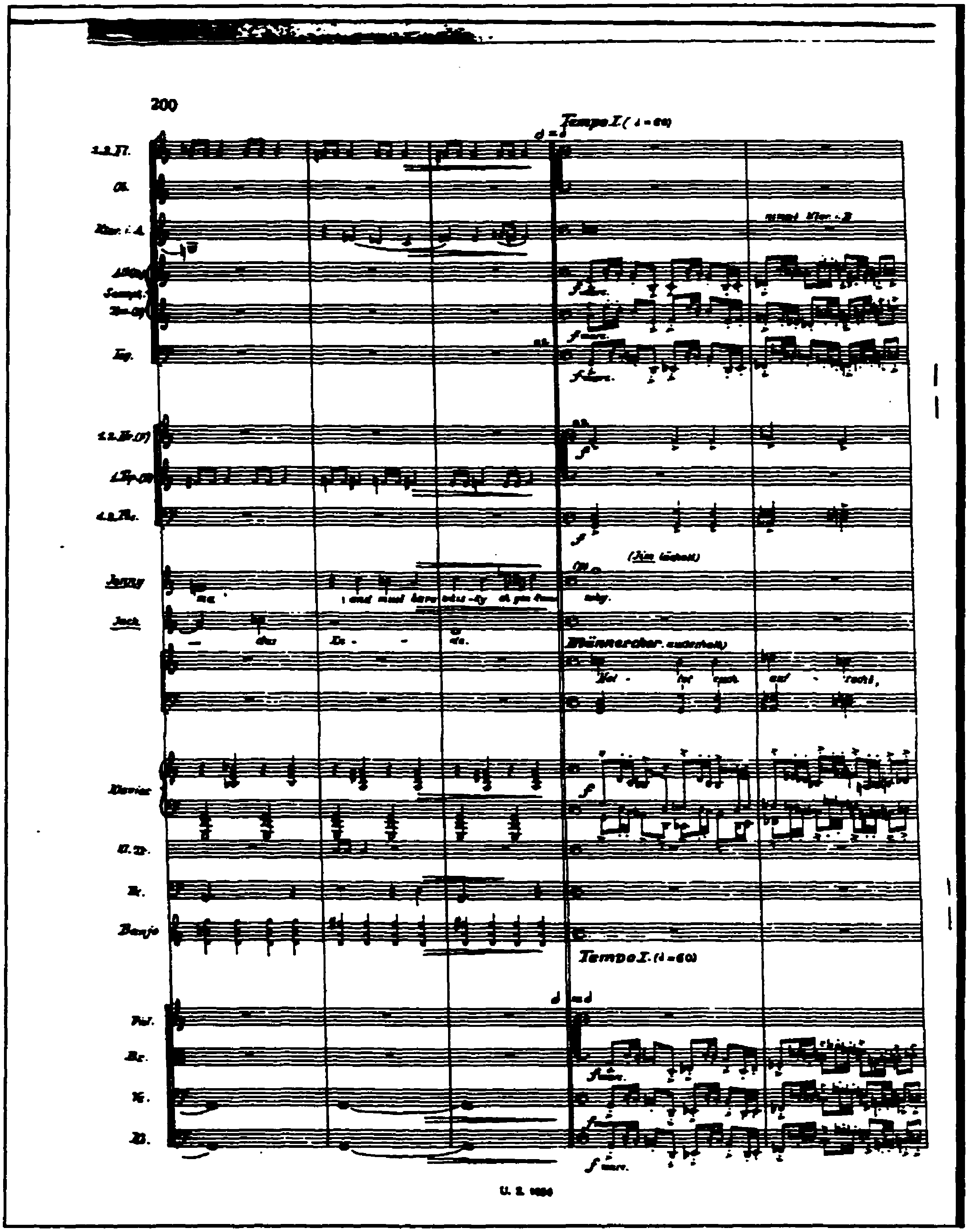


Figure 4.14 De-skewed image of Figure 4.13 by shearing.

Note that because the run-length coded version of the image is used for shearing, only one operation per column is needed, making the operation extremely efficient.

Assuming now that the image is relatively level, i.e. stafflines are horizontal, taller components, such as slurs and dynamic wedges, are removed. The filter here is still rather conservative, since if a long staff line is still skewed, as a component, it may have a considerable height (Figure 4.15). This precaution is needed because staves on a page are often distorted in different ways.

The result now consists of mostly staffline segments, some flat slurs, and flat beams. At this point, y-projection of the entire image is taken again (Figure 4.16). The derivative of the y-projection is used to locate the maxima in the projection (Figure 4.17). Using this information along with the known staffspace_height, the possible candidates for the staves are selected. For each of these candidates, x-projection is taken to determine if there is more than one staff, by searching for any blank area in the projection. Also a rough idea of the left and the right edges of the staff can be determined from the x-projection (See Figures 4.18 and 4.19).

At this point, the run lengths of the region bounding a staff, are calculated in order to obtain a more precise estimate of the staffline_height and staffspace_height of this particular staff. Also, a shearing operation is performed again to make the staff as horizontal as possible.

Using the y-projections employed during the shearing process, the vertical positions of the stafflines can be ascertained. By taking an x-projection of the region defined by the stafflines, the horizontal extents of the staff are determined.

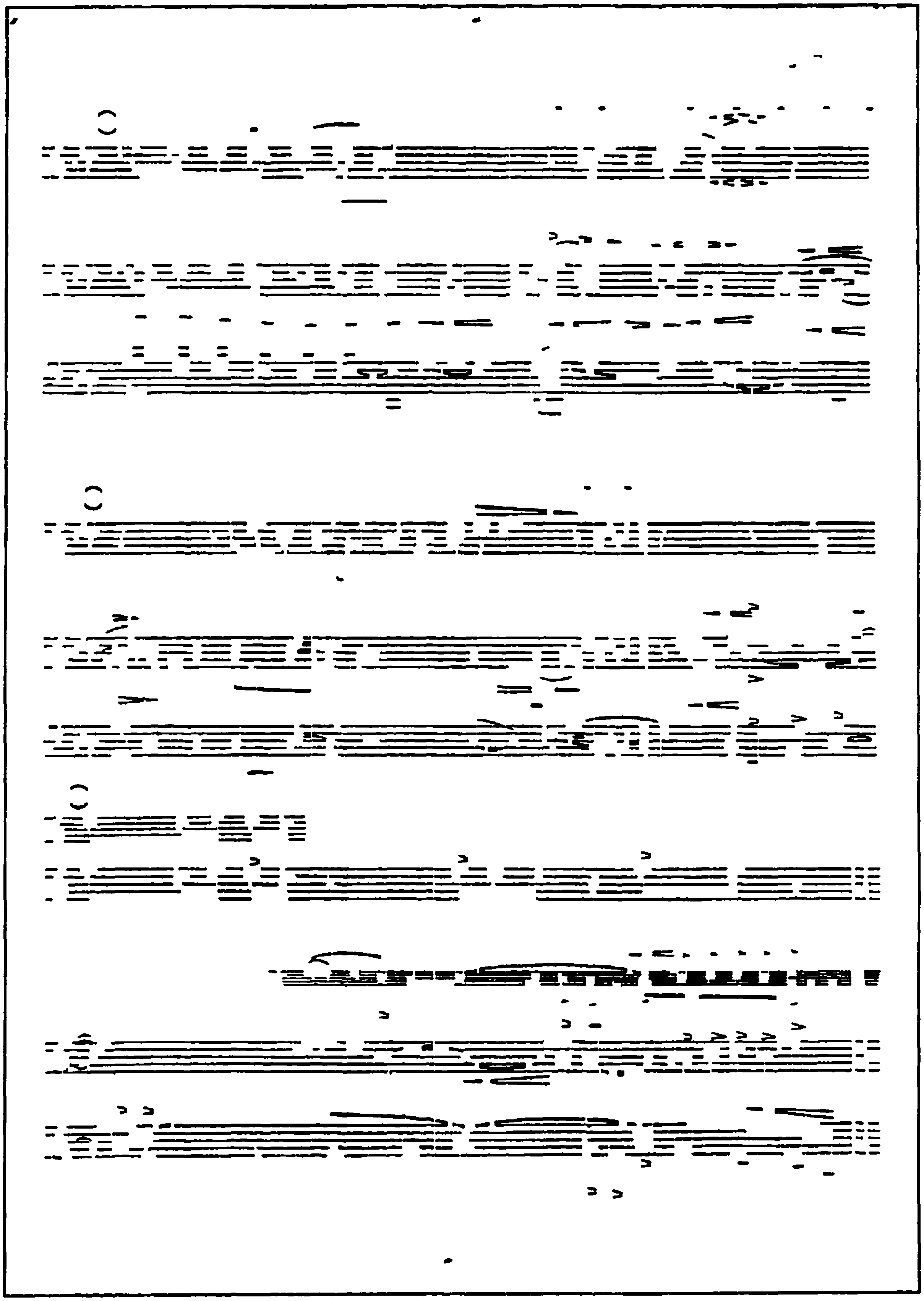


Figure 4.15 Tall connected components removed from Figure 4.12.

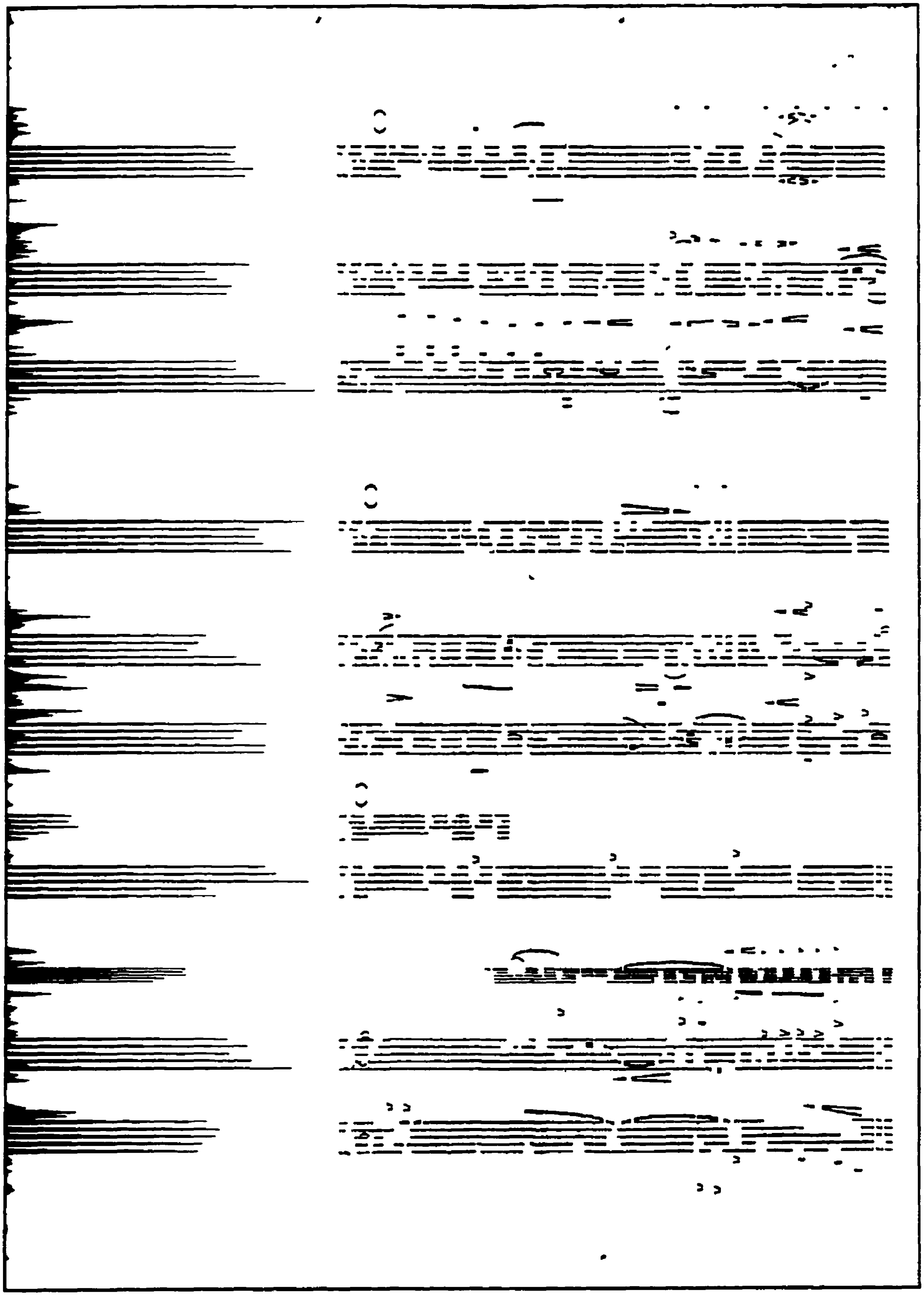


Figure 4.16 Y-projection of Figure 4.15.

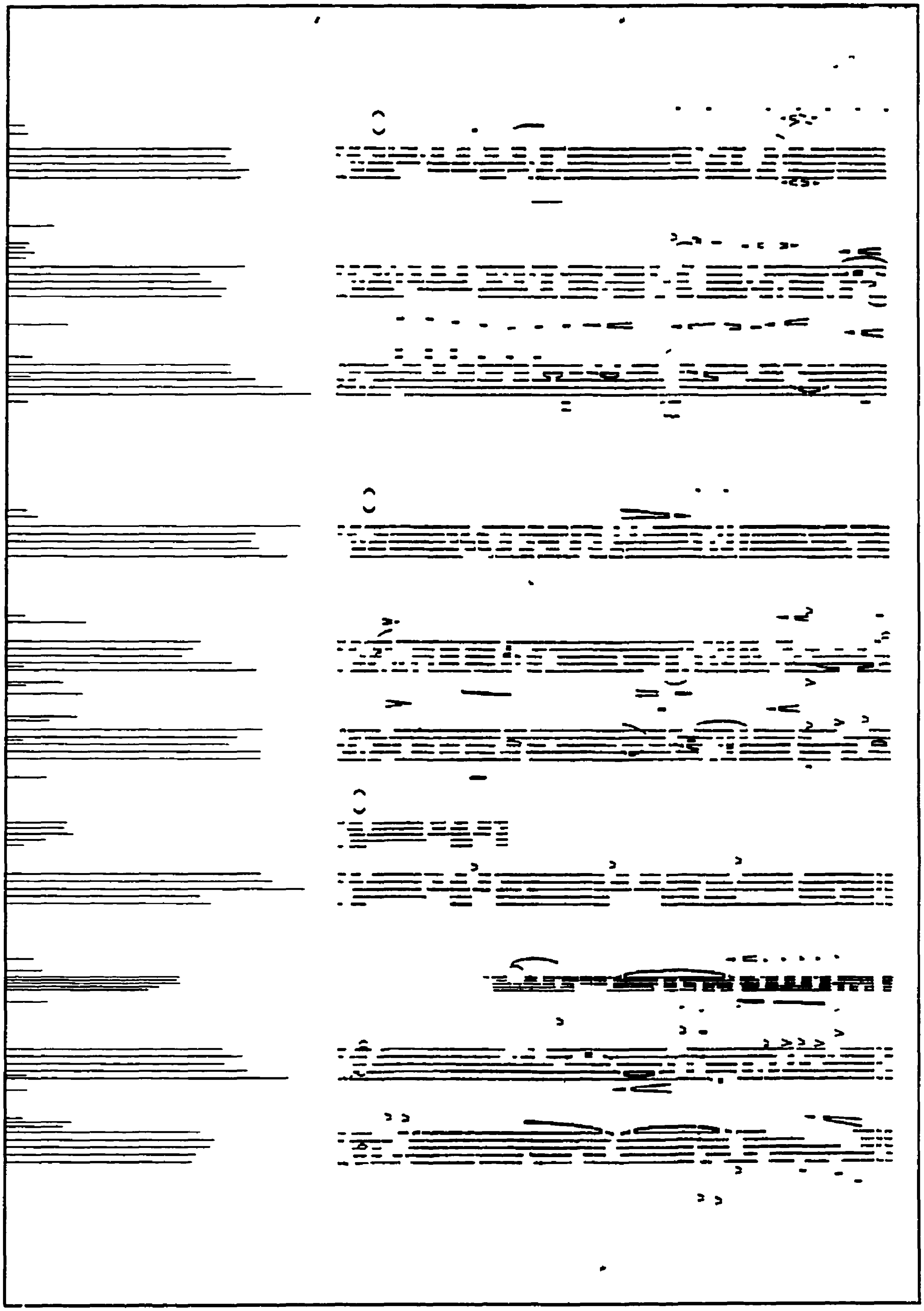
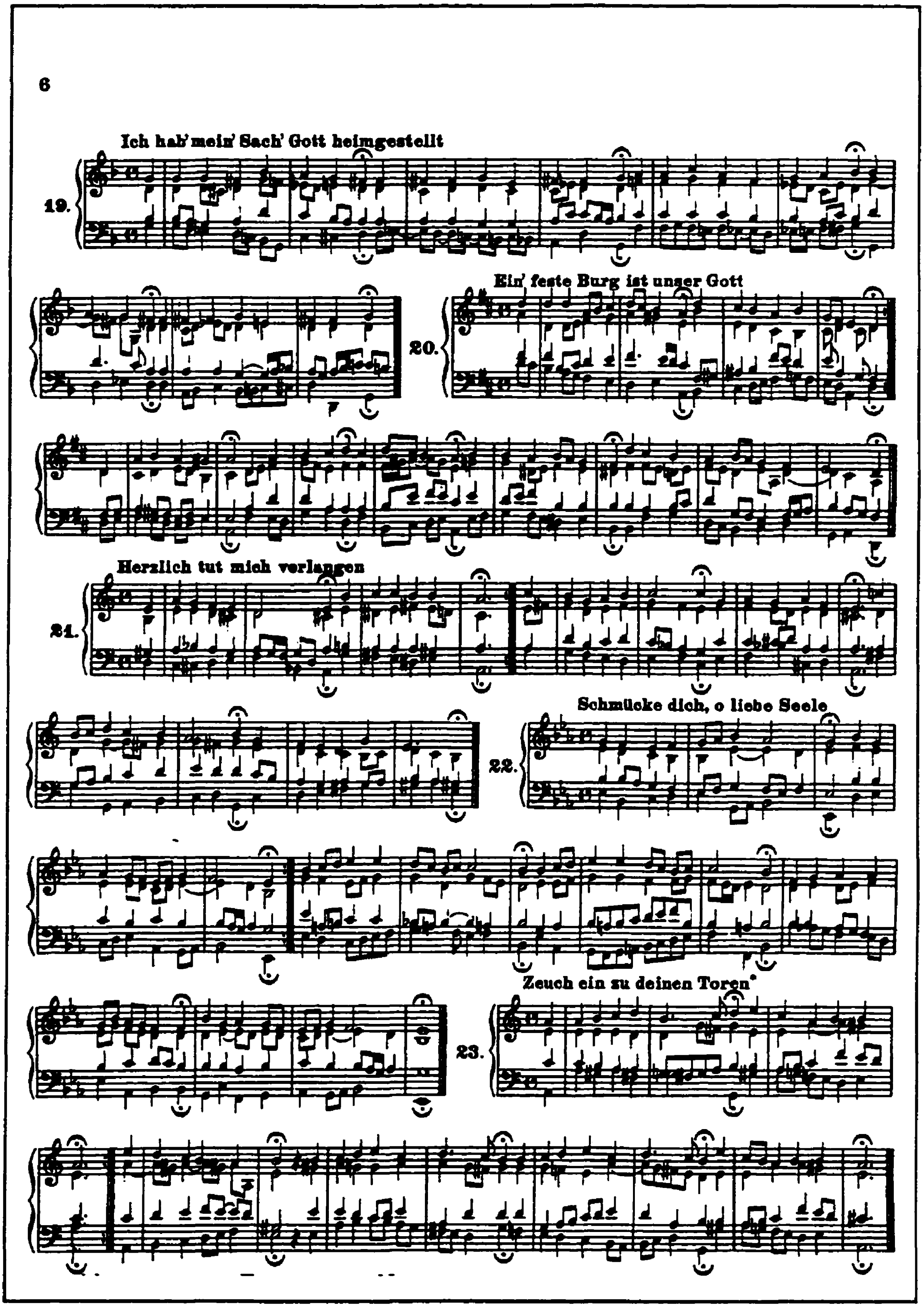


Figure 4.17 Y-projection (maxima only) of Figure 4.15.



6
Ich hab' mein' Sach' Gott heimgestellt
19.
Ein' feste Burg ist unser Gott
20.
Herzlich tut mich verlangen
21.
Schmücke dich, o liebe Seele
22.
Zeuch ein zu deinen Toren*
23.

Figure 4.18 An example of staves placed side-by-side.

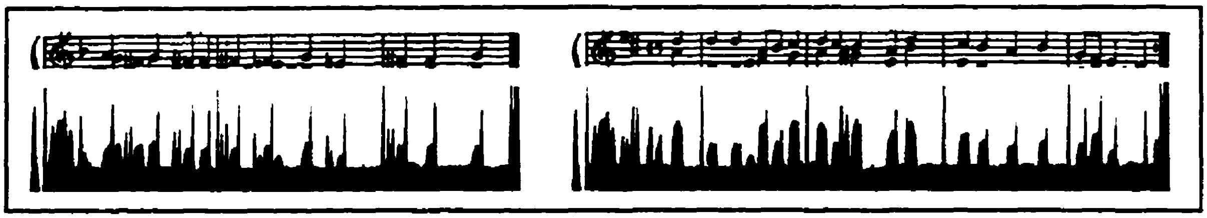


Figure 4.19 X-projection of the top staves of the second system in Figure 4.18.

The next step, knowing the positions of the stafflines, is to remove them. Since the image now consists mainly of staffline segments (Figure 4.20), the strategy is to delete everything but the stafflines; then the image can be XORed with the original image so that, in effect, the stafflines are removed.

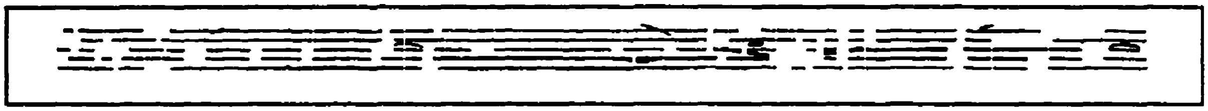


Figure 4.20 Isolated staff, from sixth staff of Figure 4.15.

At this point, the stafflines are assumed to be flat, so any components taller than the stafflines can be removed (Figure 4.21). This operation differs from the similar operation performed on the entire image, since the more accurate staffline_height that applies to this particular staff is now available.

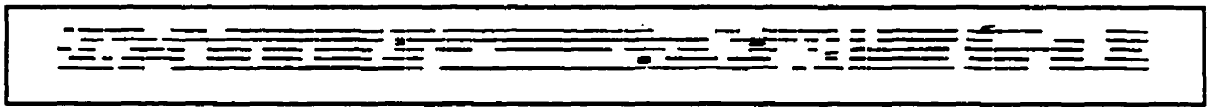


Figure 4.21 Tall connected components removed.

Also, given the exact positions of the stafflines, components that are between the stafflines are removed (Figure 4.22).

The result is XORed with the original image. Given two bit-mapped images A and A', where A' is a subset of A (A' is derived from A), an XOR operation has the following important property: All black pixels in A' are removed from A. For example, Figure 4.22 and Figure 4.23 are XORed resulting in Figure 4.24.

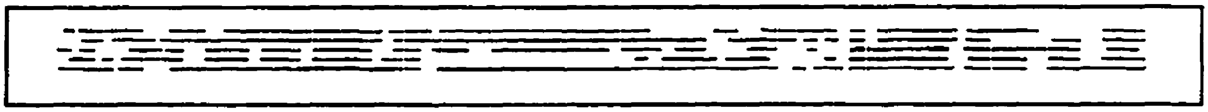


Figure 4.22 Objects between the stafflines removed.

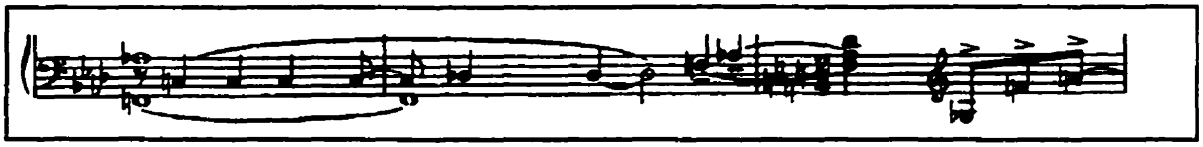


Figure 4.23 The original sixth staff of Figure 4.10.

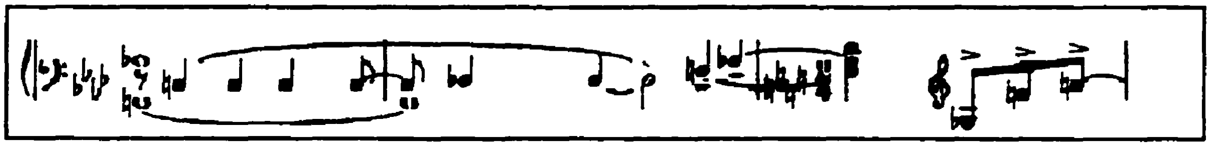


Figure 4.24 The result of XORing Figures 4.22 and 4.23.

Several examples of the staffline removal are shown in Figures 4.25 to 4.36. The time the program takes to remove the stafflines, including reading the input image and writing the resultant image, of 32 pages of different types of music, was approximately 20 minutes, or less than 40 seconds per page on a Sun SPARC 2. All of these image processings, such as filtering and XORing, are performed either on the run-length codes or connected components and not directly on the bit-map, thus making computations extremely efficient.

4.1.4 A note on scanning resolution

The resolution of scanning is 300 dpi (dots-per-inch) which seems to be satisfactory for standard piano music or instrumental parts that may have eight to ten staves per page. The 300 dpi resolution, however, is not fine enough for orchestral scores or miniature scores. For these types of scores, scanning resolution of 600–1000 dpi is needed. Ideally, the thinnest object (usually the stems) should have the thickness of three to five pixels.

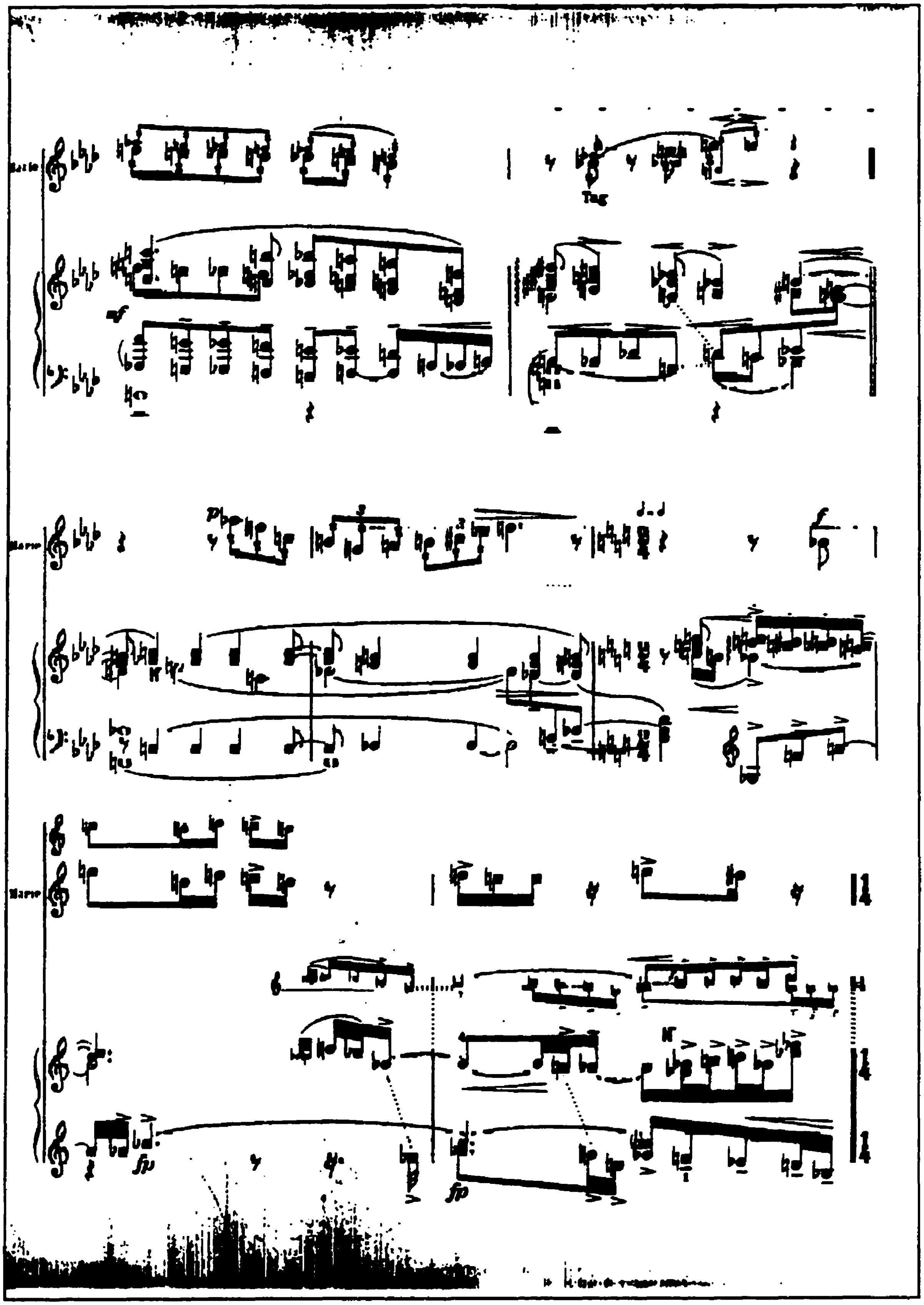


Figure 4.25 Stafflines removed from Figure 4.10.

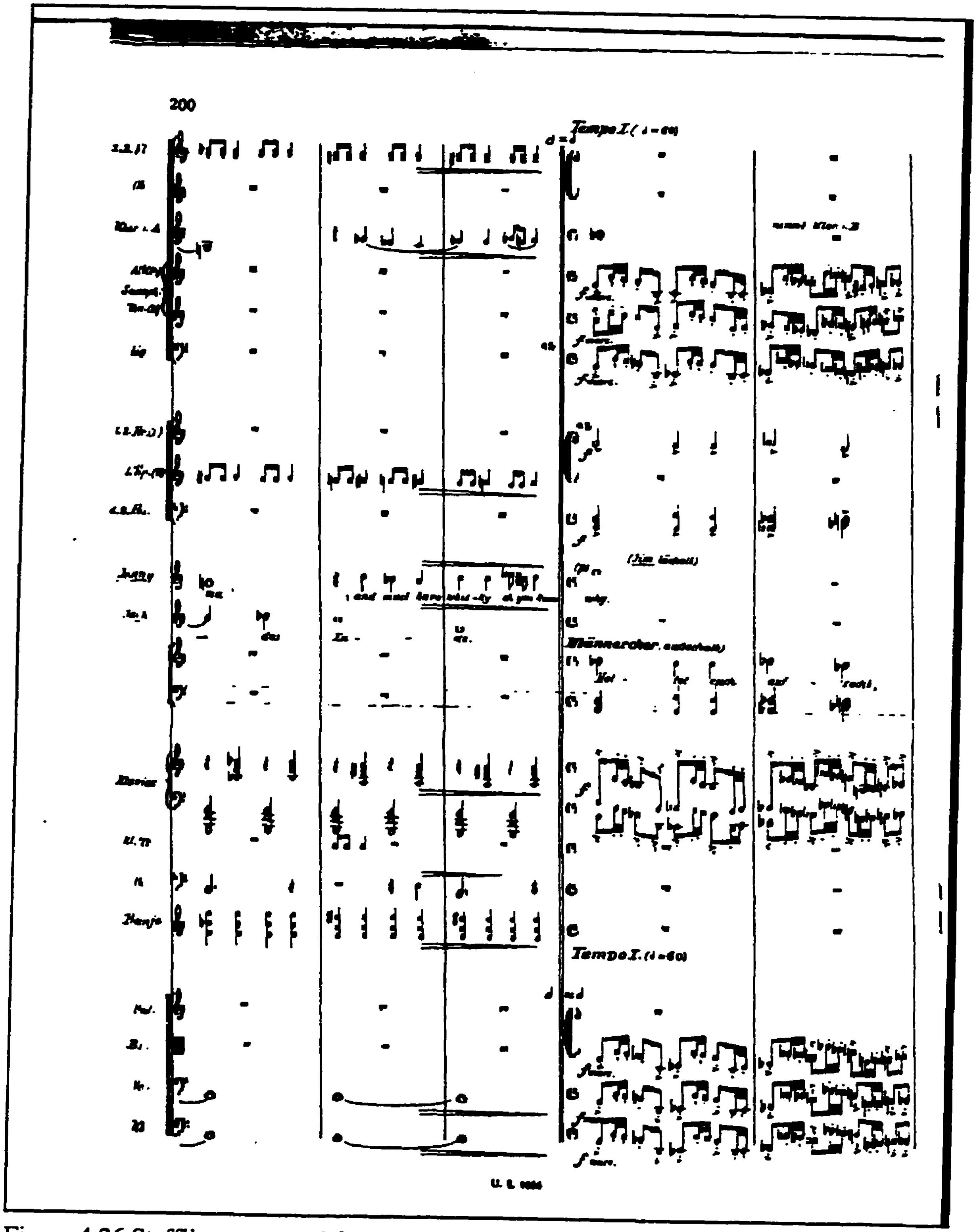


Figure 4.26 Stafflines removed from Figure 4.14.

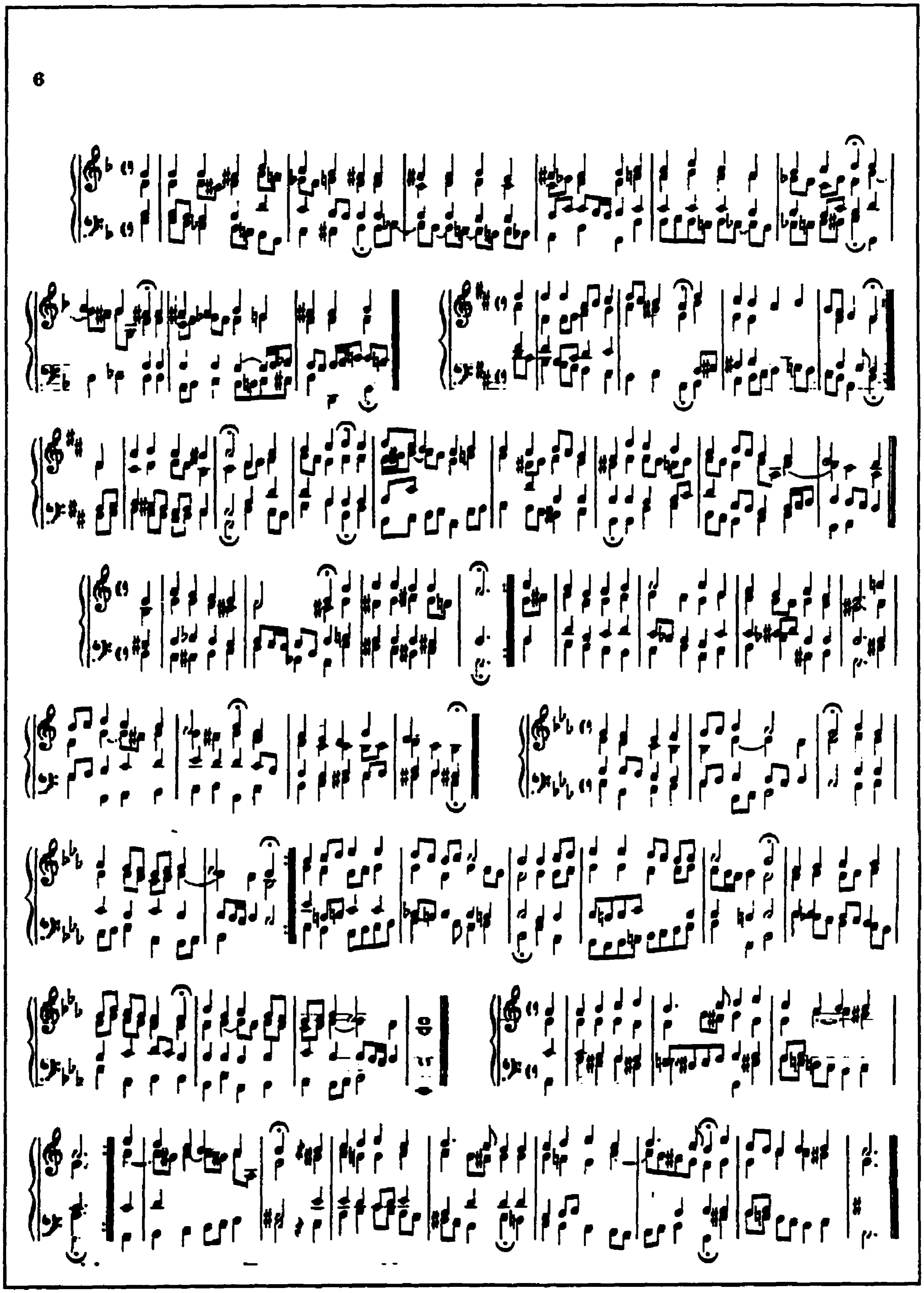


Figure 4.27 Stafflines removed from Figure 4.18.

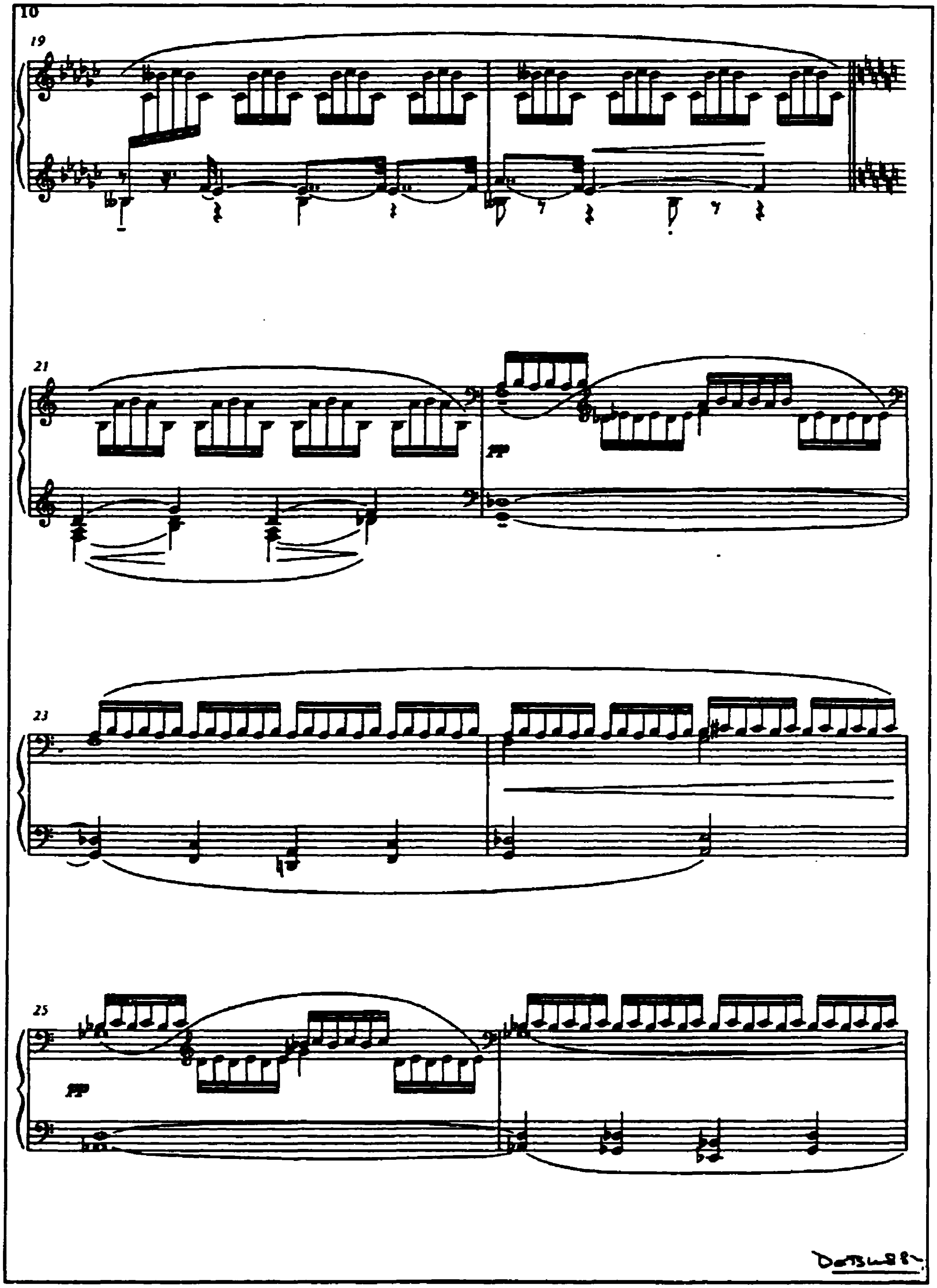


Figure 4.29 The original.

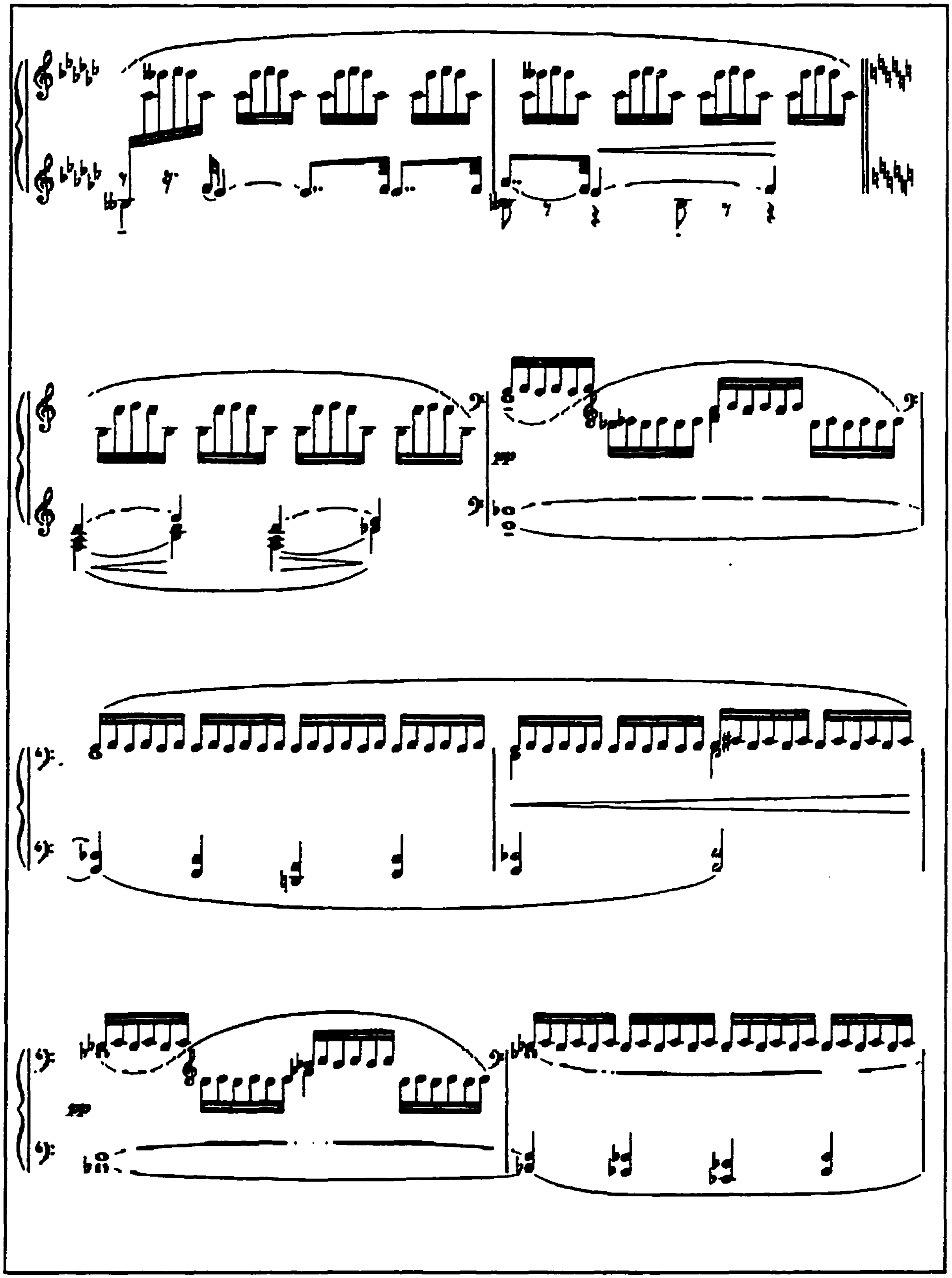


Figure 4.30 Stafflines removed from Figure 4.29.

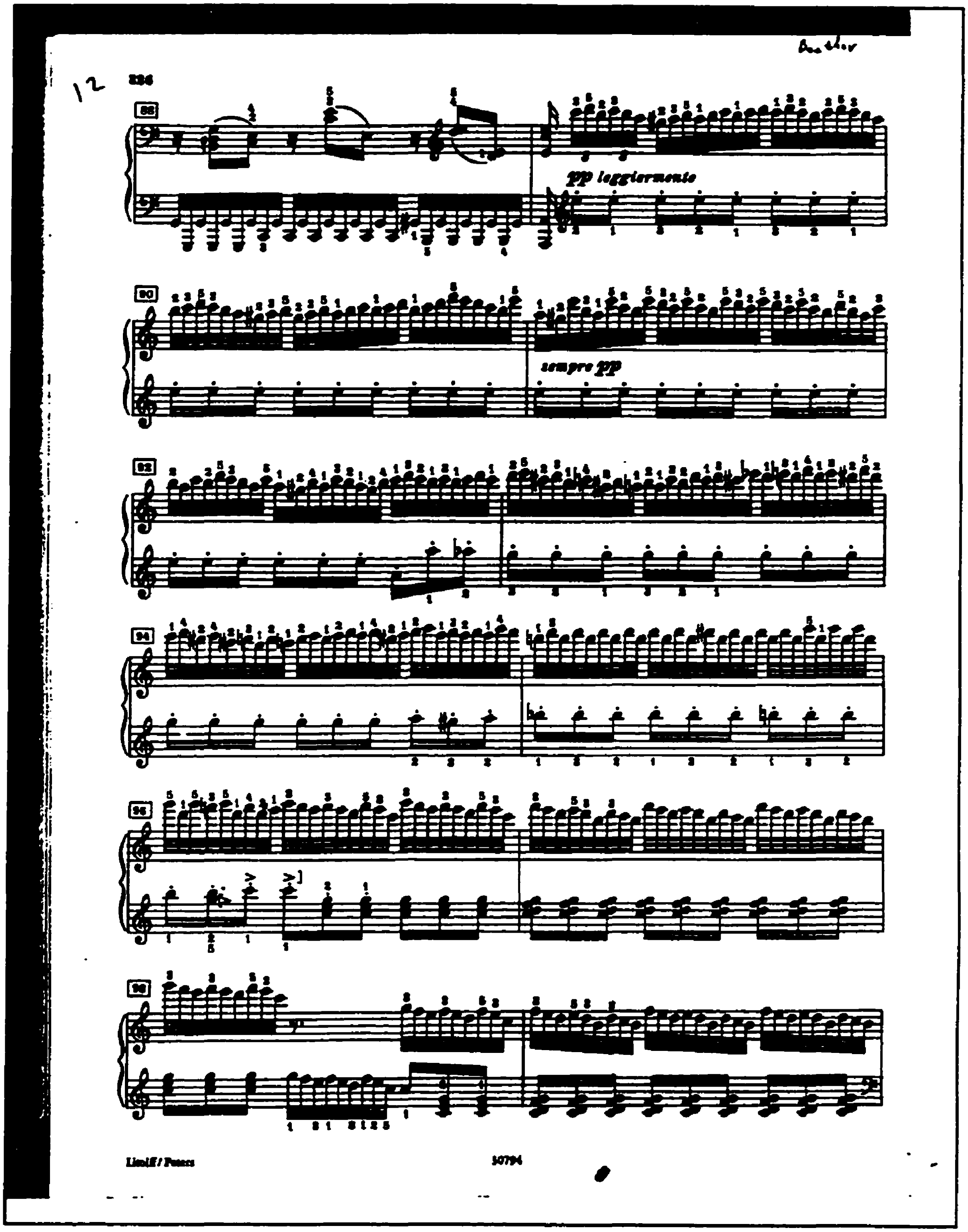


Figure 4.31 The original.

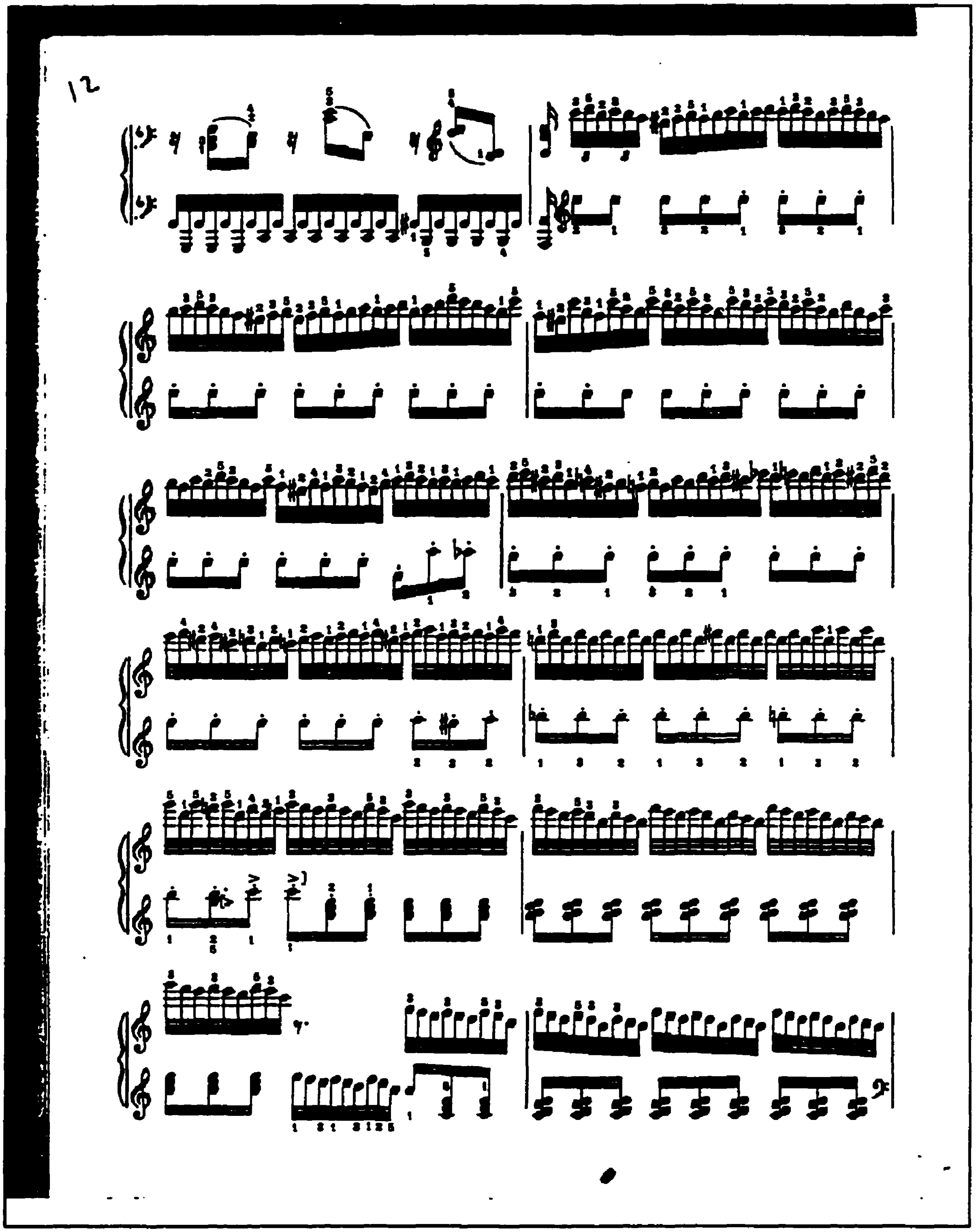


Figure 4.32 Stafflines removed from Figure 4.31.

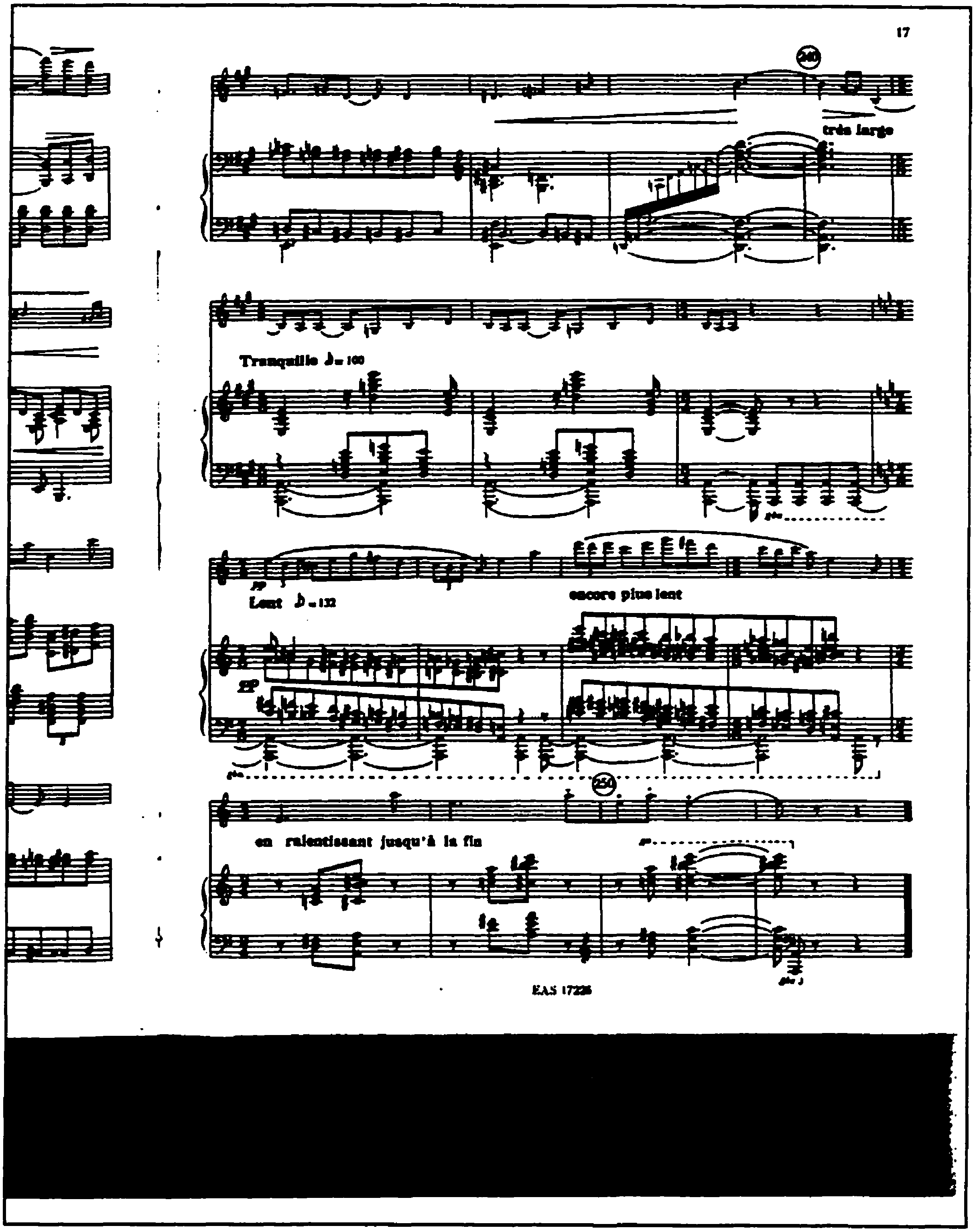


Figure 4.33 The original.

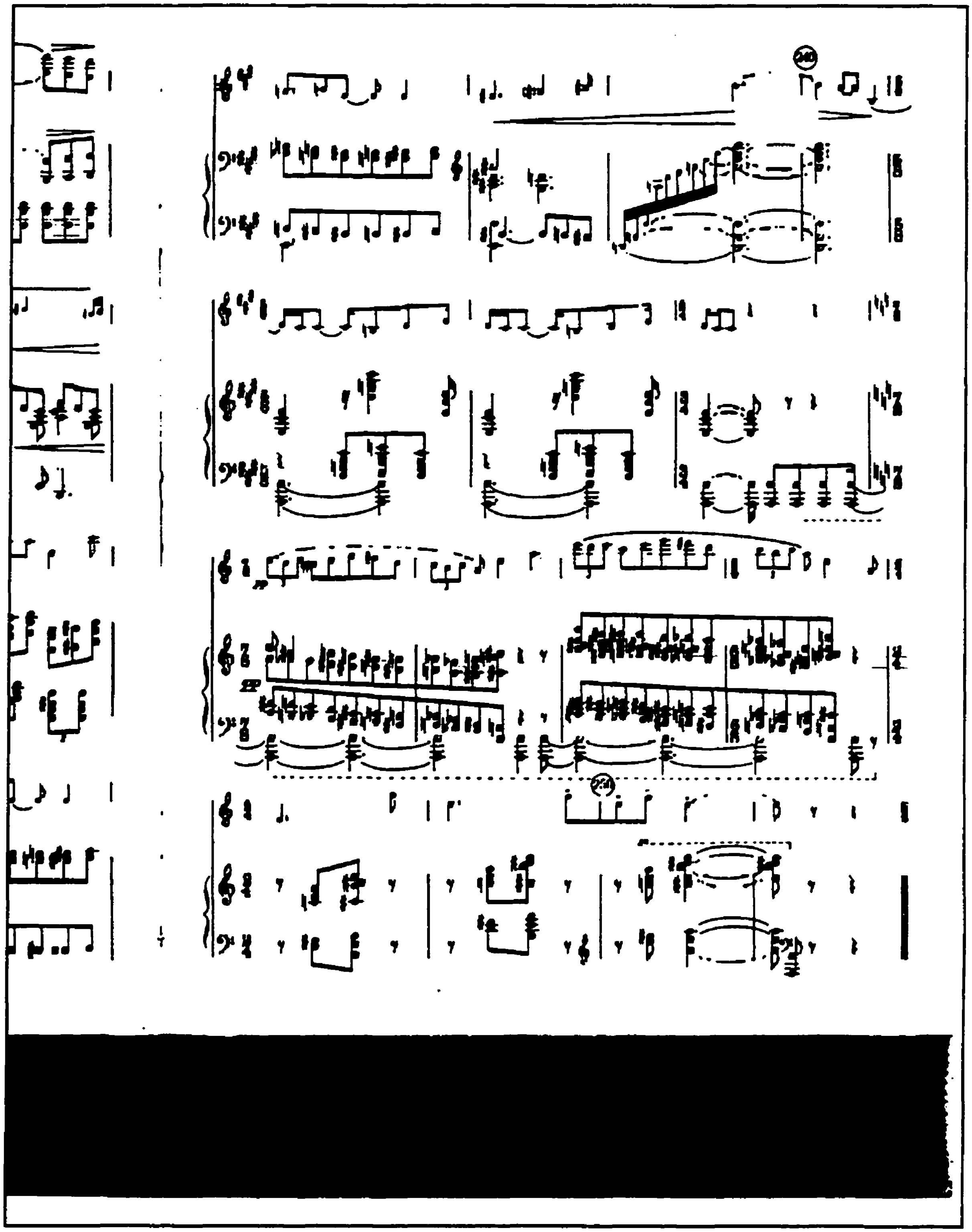


Figure 4.34 Stafflines removed from Figure 4.33.

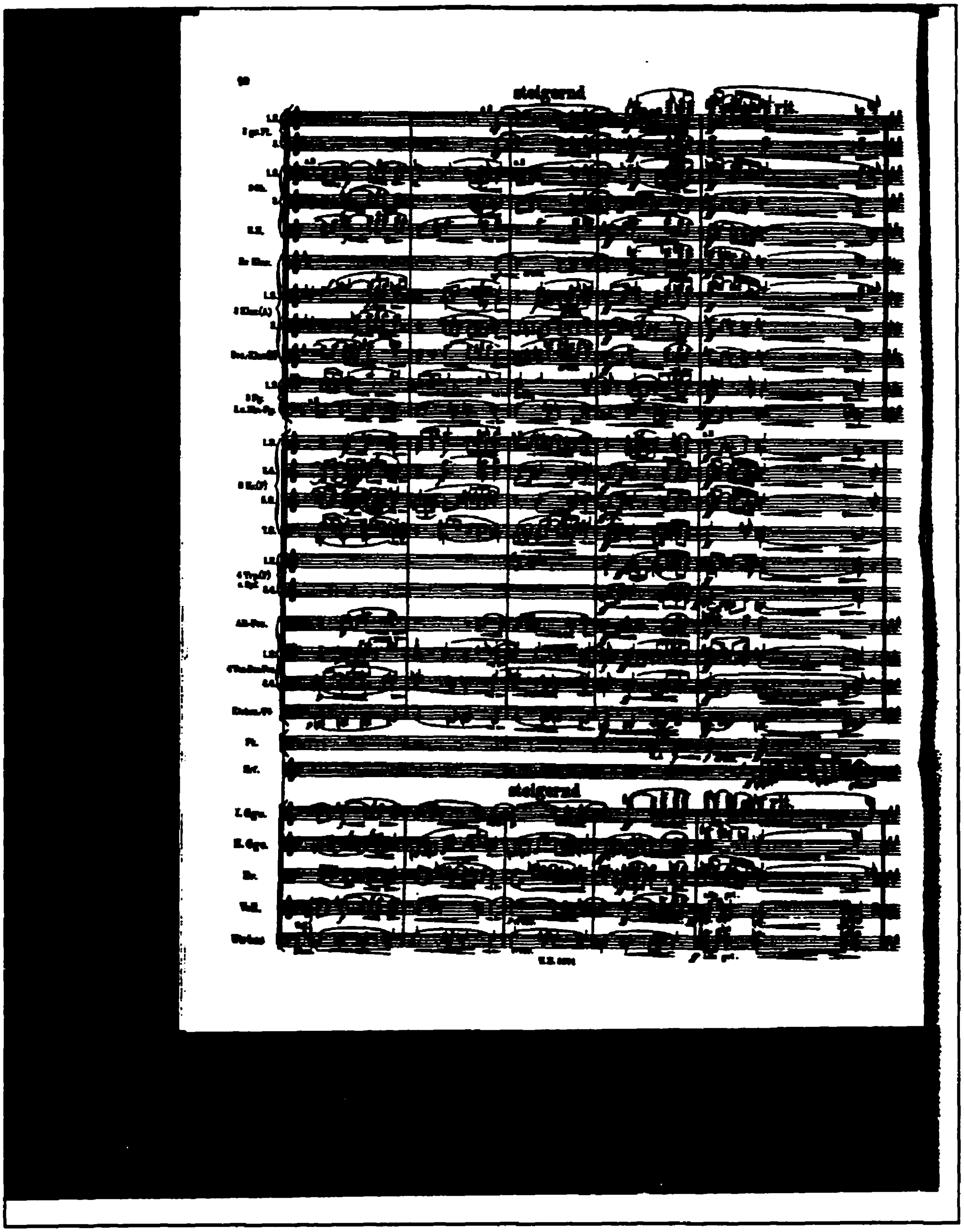


Figure 4.35 The original.

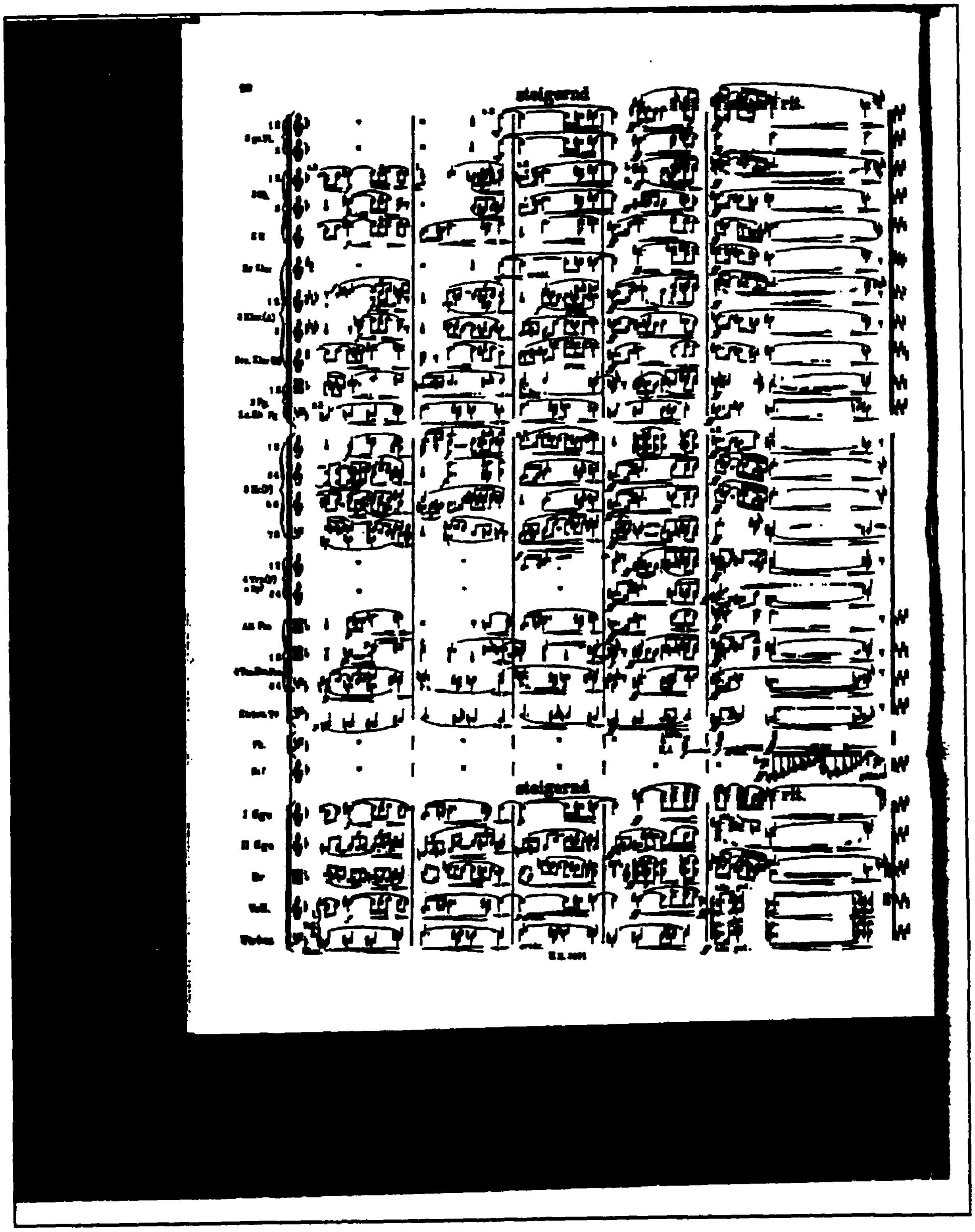


Figure 4.36 Stafflines removed from Figure 4.35.

4.2 Text removal

In order to lessen the burden on the classifier, text, such as lyrics and performance indications, is removed as much as possible. The intention is to use a separate program, specialized for optical character recognition, to process the texts on the page.

Text is distinguished from musical symbols by using the characteristics that text symbols have basically the same height and are placed side by side. The problem here is similar to finding texts in document image analysis (Nagy 1989), where texts need to be separated from graphics in maps (Taxt, Flynn, and Jain 1989), newspapers (Akiyama and Hagita 1990), and drawings (Fletcher and Kasturi 1988), or when locating destination address on envelopes (Jain and Bhattacharjee 1992; particularly difficult here are finding address labels on the newspapers and magazines delivered by mailing).

Simple yet effective heuristics are used to locate texts, which can appear almost anywhere on the page. First, perform a connected component analysis on the entire page. Second, determine if each connected component may qualify as a letter; if so it must further qualify to be a letter within a word, i. e., a single letter is not removed as dynamic markings such as *p*, *f*, or numerals for tuplet notation or fingerings can be processed by the AOMR program.

Here are the criteria for a letter:

1. That its "average height" and "average width" are larger than some predetermined minimum value. (this lower limit will skip punctuation markings, which are considered separately.)
2. That its aspect ratio (height / width) is within a certain range. This step is needed to remove slurs and pedal markings; it also removes some connected letters.

Note that staves and everything attached to them become very large connected components and are discarded by the second criterion.

Here is the criterion for a letter within a word:

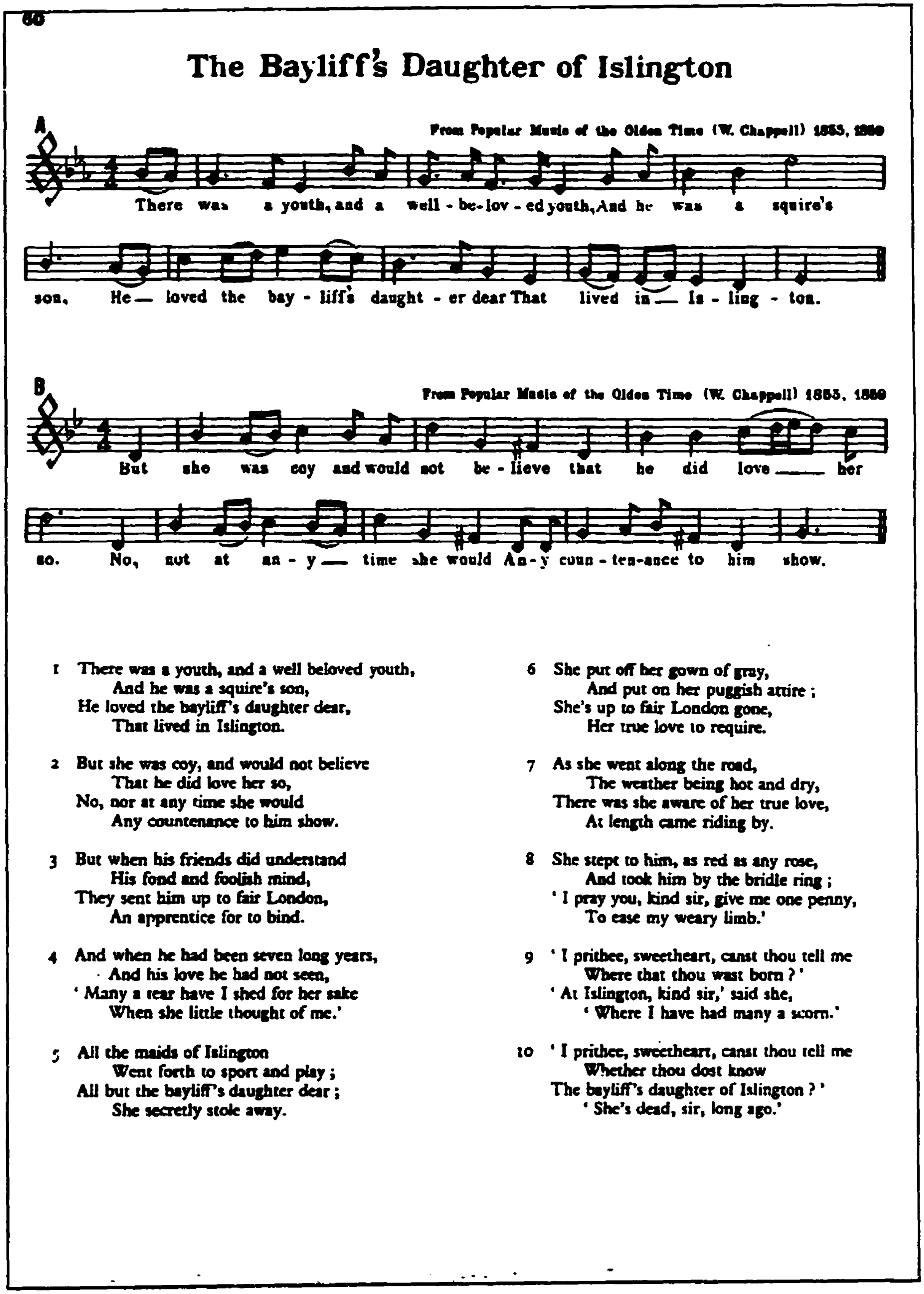
> If another letter can be found that is horizontally close to it, it is considered a letter within a word. The closeness depends on the size of the letters.

The result of the above processes are three classes of connected components:

I. Those considered as letters belonging to a word.
II. Those that were too small to be considered as letters.
III. Those that were possible letters, but rejected because no other letters were found that were close to it.

The connected components in Class II are revisited to see if they may be punctuations (period, comma, quotation mark, etc.) belonging to one of the letters in Class I by the fact they are close to them.

Although these simple rules help to eliminate most words on a page, as shown in Figures 4.37 to 4.47, there are two kinds of cases where this algorithm fails. One is when letters are connected to each other. These result in the low aspect ratios, because they have relatively wider width than a letter. The other is when the letters are touching the staffline, in which case the elimination is difficult because notes that are attached to staves may easily be mistaken for letters.


60
A
From Popular Music of the Olden Time (W. Chappell) 1855, 1859
There was a youth, and a well - be - lov - ed youth, And he was a squire's
son, He — loved the bay - liff's daught - er dear That lived in — Is - ling - ton.
B
From Popular Music of the Olden Time (W. Chappell) 1855, 1859
But she was coy and would not be - lieve that he did love — her
so. No, nor at an - y — time she would An - y coun - ten - ance to him show.

The Bayliff's Daughter of Islington

1 There was a youth, and a well beloved youth,
And he was a squire's son,
He loved the bayliff's daughter dear,
That lived in Islington.

2 But she was coy, and would not believe
That he did love her so,
No, nor at any time she would
Any countenance to him show.

3 But when his friends did understand
His fond and foolish mind,
They sent him up to fair London,
An apprentice for to bind.

4 And when he had been seven long years,
And his love he had not seen,
'Many a tear have I shed for her sake
When she little thought of me.'

5 All the maids of Islington
Went forth to sport and play;
All but the bayliff's daughter dear;
She secretly stole away.

6 She put off her gown of gray,
And put on her puggish attire;
She's up to fair London gone,
Her true love to require.

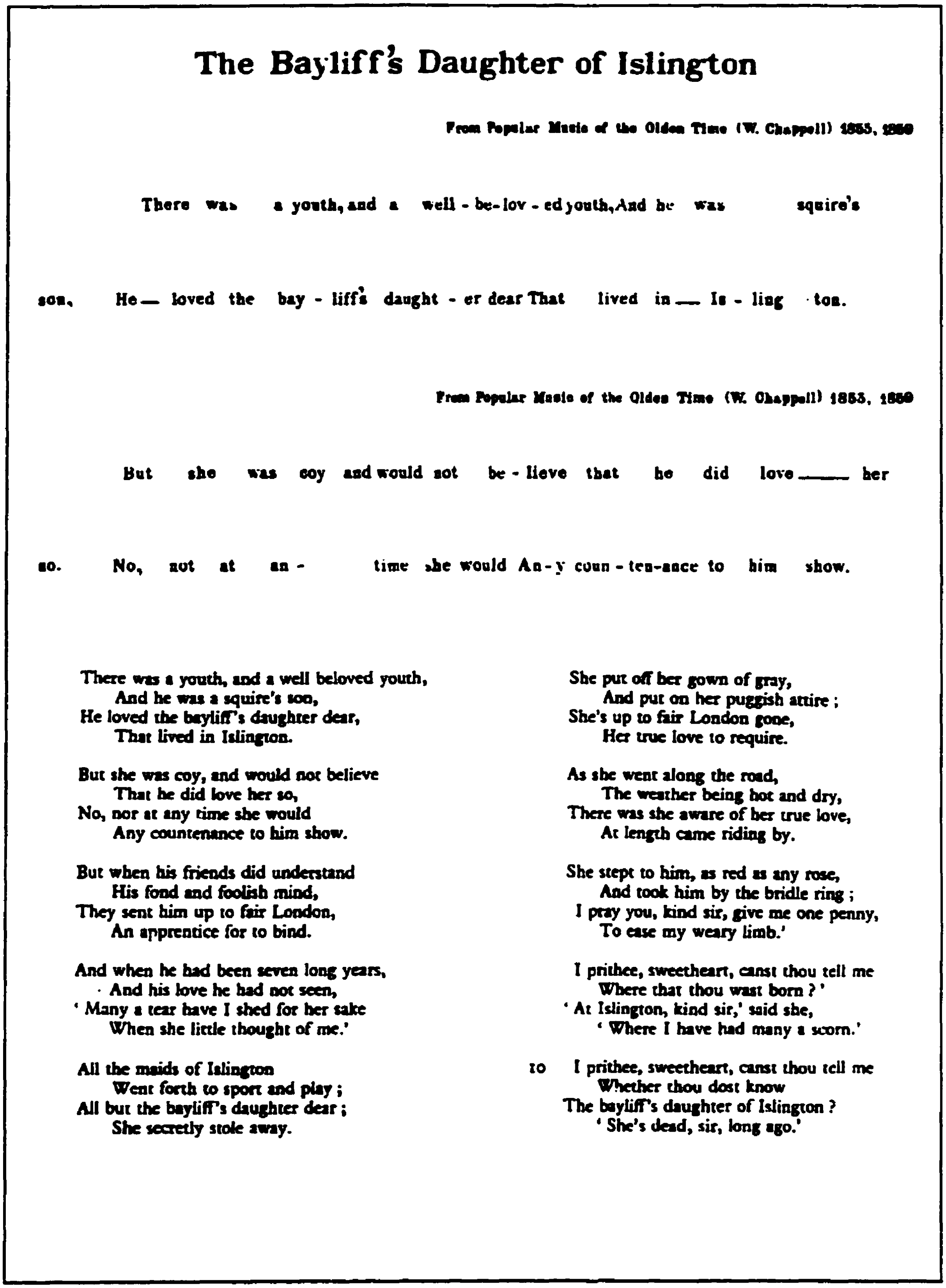
7 As she went along the road,
The weather being hot and dry,
There was she aware of her true love,
At length came riding by.

8 She stept to him, as red as any rose,
And took him by the bridle ring;
'I pray you, kind sir, give me one penny,
To ease my weary limb.'

9 'I prithee, sweetheart, canst thou tell me
Where that thou wast born?'
'At Islington, kind sir,' said she,
'Where I have had many a scorn.'

10 'I prithee, sweetheart, canst thou tell me
Whether thou dost know
The bayliff's daughter of Islington?'
'She's dead, sir, long ago.'

Figure 4.37 The original with text.



The Baylif's Daughter of Islington

From Popular Music of the Olden Time (W. Chappell) 1855, 1859

There was a youth, and a well - be- lov - ed youth,And he was squire's

son, He — loved the bay - liff's daught - er dear That lived in — Is - ling - ton.

From Popular Music of the Olden Time (W. Chappell) 1853, 1859

But she was coy and would not be - lieve that he did love — her

so. No, not at an - time she would An - y coun - ten - ance to him show.

There was a youth, and a well beloved youth,
And he was a squire's son,
He loved the bayliff's daughter dear,
That lived in Islington.

But she was coy, and would not believe
That he did love her so,
No, nor at any time she would
Any countenance to him show.

But when his friends did understand
His fond and foolish mind,
They sent him up to fair London,
An apprentice for to bind.

And when he had been seven long years,
And his love he had not seen,
'Many a tear have I shed for her sake
When she little thought of me.'

All the maids of Islington
Went forth to sport and play;
All but the bayliff's daughter dear;
She secretly stole away.

She put off her gown of gray,
And put on her puggish attire;
She's up to fair London gone,
Her true love to require.

As she went along the road,
The weather being hot and dry,
There was she aware of her true love,
At length came riding by.

She stept to him, as red as any rose,
And took him by the bridle ring;
I pray you, kind sir, give me one penny,
To ease my weary limb.'

I prithee, sweetheart, canst thou tell me
Where that thou wast born?'
'At Islington, kind sir,' said she,
'Where I have had many a scorn.'

I prithee, sweetheart, canst thou tell me
Whether thou dost know
The bayliff's daughter of Islington?
'She's dead, sir, long ago.'

Figure 4.38 Texts extracted from Figure 4.37.

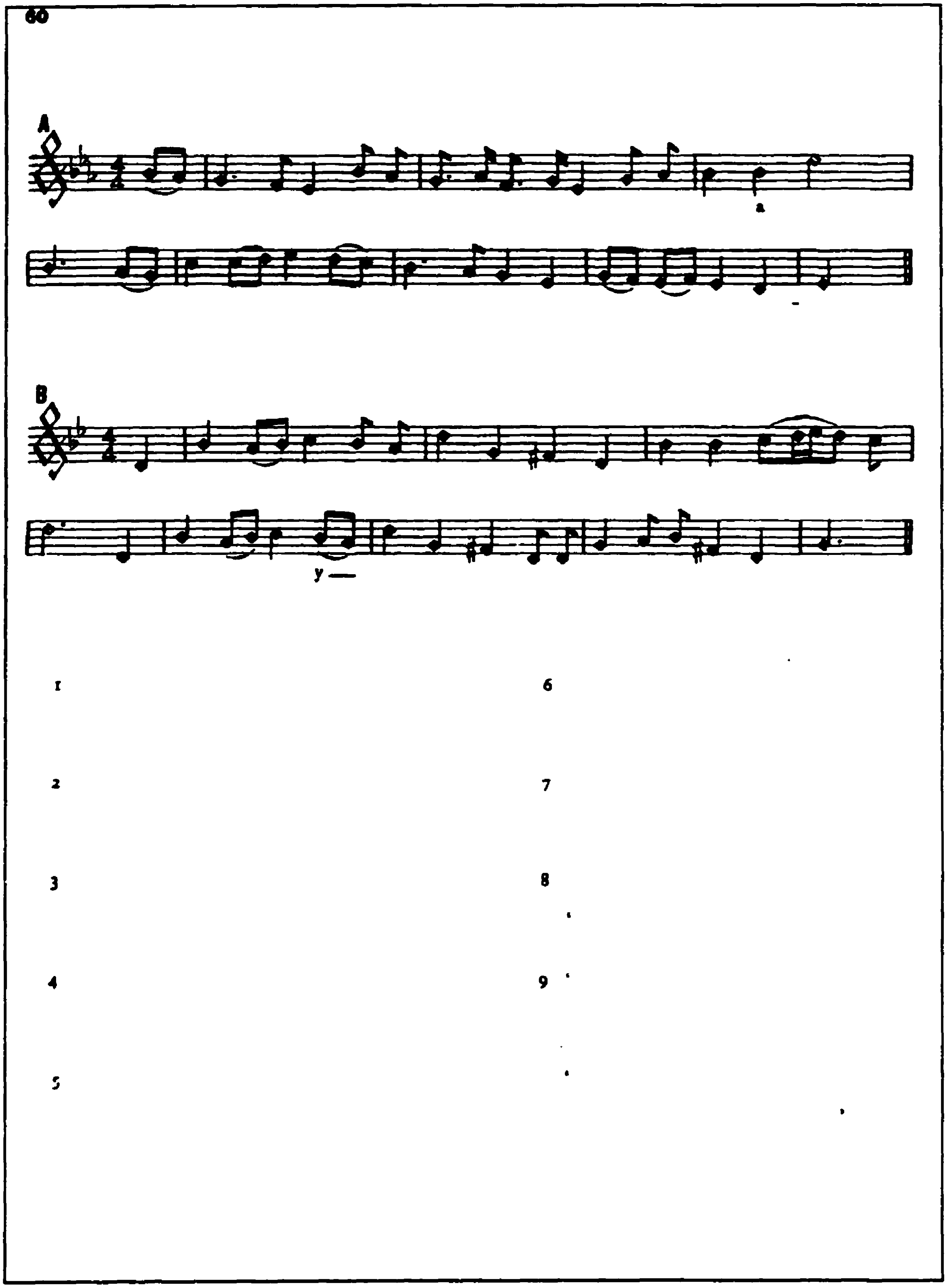


Figure 4.39 Text removed from Figure 4.37.

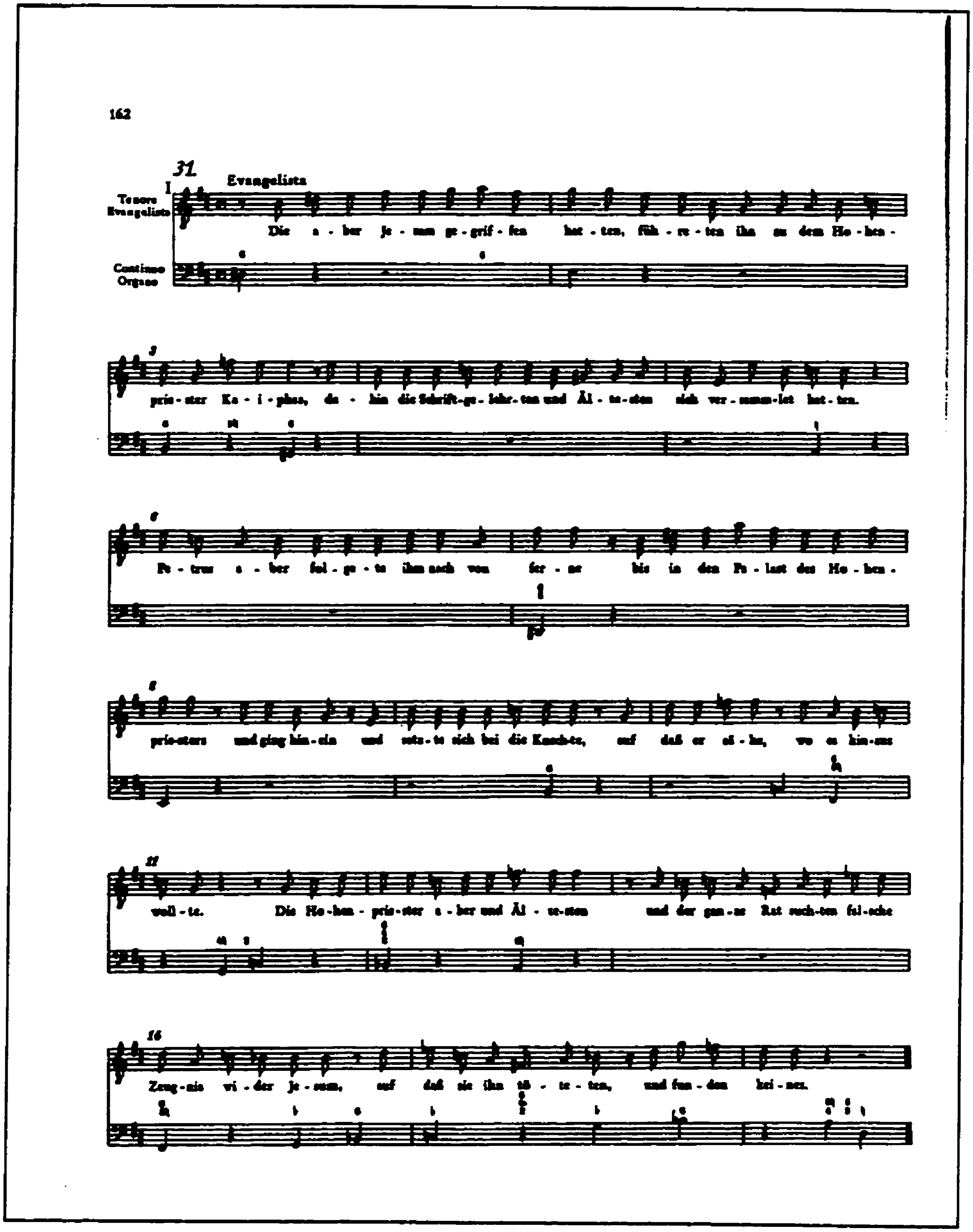
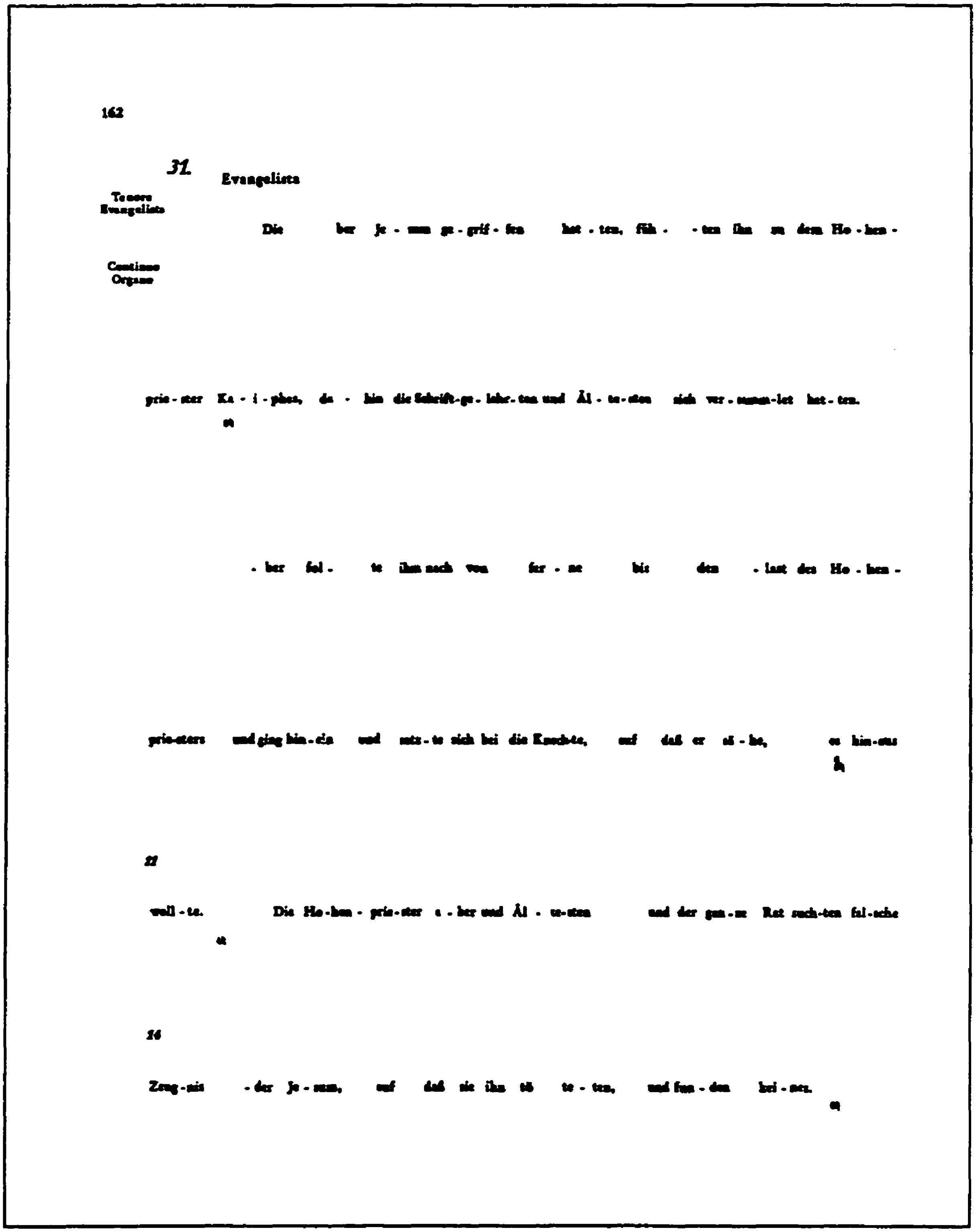

162
31
Evangelista
I
Tenore Evangelista
Continuo Organo
Die a - ber Je - sum ge - grif - fen hat - ten, füh - re - ten ihn zu dem Ho - hen -
3
prie - ster Ka - i - phas, da - hin die Schrift - ge - lehr - ten und Äl - te - sten sich ver - samm - let hat - ten.
6
Pe - trus a - ber fol - ge - te ihm nach von fer - ne bis in den Pa - last des Ho - hen -
8
prie - sters und ging hin - ein und setz - te sich bei die Knech - te, auf daß er sä - he, wo es hin - aus
11
woll - te. Die Ho - hen - prie - ster a - ber und Äl - te - sten und der gan - ze Rat such - ten fal - sche
16
Zeug - nis wi - der Je - sum, auf daß sie ihn tö - te - ten, und fun - den kei - nes.

Figure 4.40 The original with text.



162

31. Evangelista

Tenore Evangelista

Die ber Je - sum ge - grif - fen hat - ten, füh - - ten ihn zu dem Ho - hen -

Continuo Organo

prie - ster Ka - i - phas, da - hin die Schrift-ge - lehr - ten und Äl - te-sten sich ver - samm-let hat - ten.

- ber fol - te ihm nach von fer - ne bis den - last des Ho - hen -

prie-sters und ging hin - ein und setz - te sich bei die Knech-te, auf daß er sä - he, es hin-aus

22

woll - te. Die Ho-hen - prie-ster a - ber und Äl - te-sten und der gan - ze Rat such-ten fal-sche

26

Zeug - nis - der Je - sum, auf daß sie ihn tö te - ten, und fun - den kei - nes.

Figure 4.41 Text extracted from Figure 4.40.

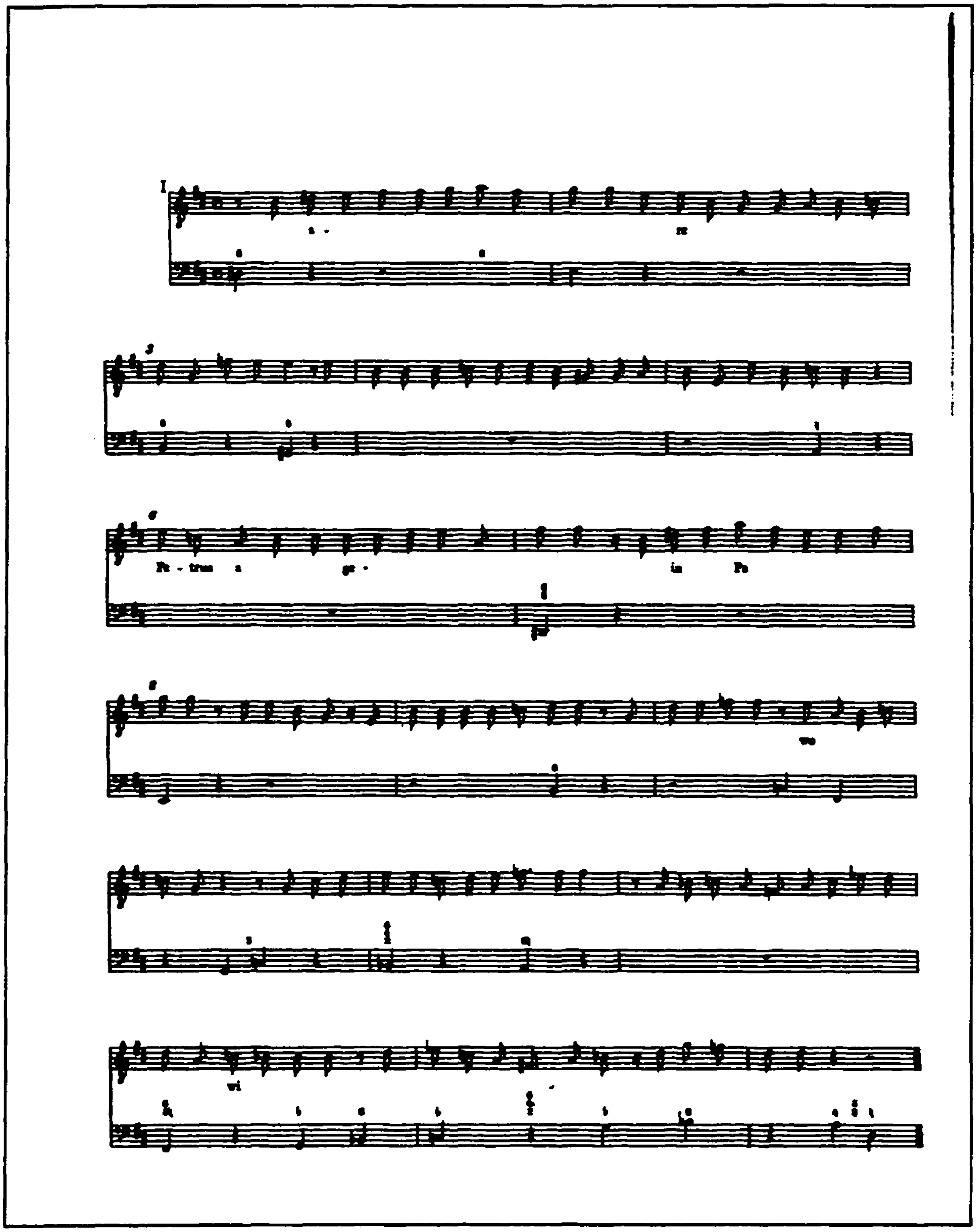
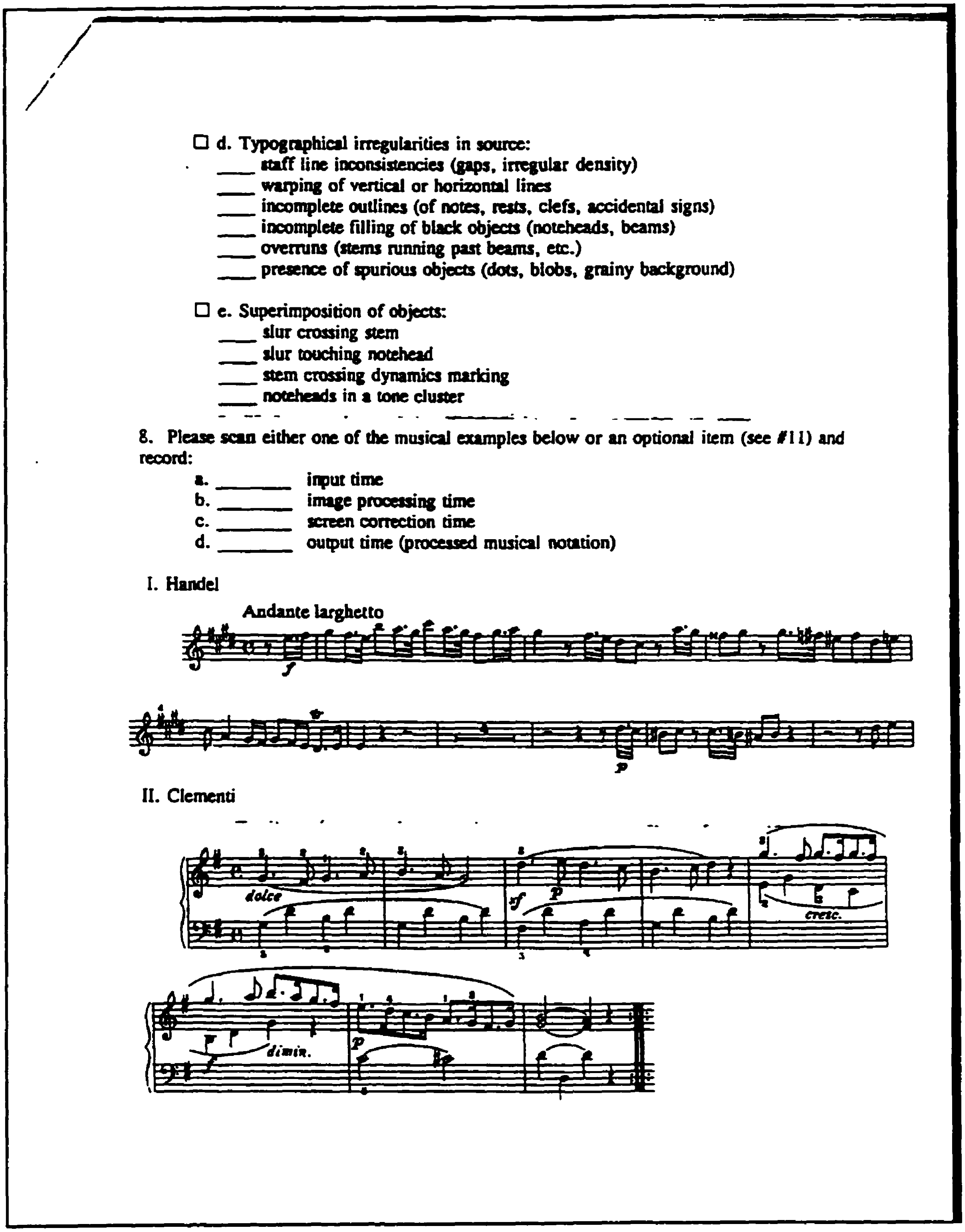


Figure 4.42 Text removed from Figure 4.40.



□ d. Typographical irregularities in source:
____ staff line inconsistencies (gaps, irregular density)
____ warping of vertical or horizontal lines
____ incomplete outlines (of notes, rests, clefs, accidental signs)
____ incomplete filling of black objects (noteheads, beams)
____ overruns (stems running past beams, etc.)
____ presence of spurious objects (dots, blobs, grainy background)

□ e. Superimposition of objects:
____ slur crossing stem
____ slur touching notehead
____ stem crossing dynamics marking
____ noteheads in a tone cluster

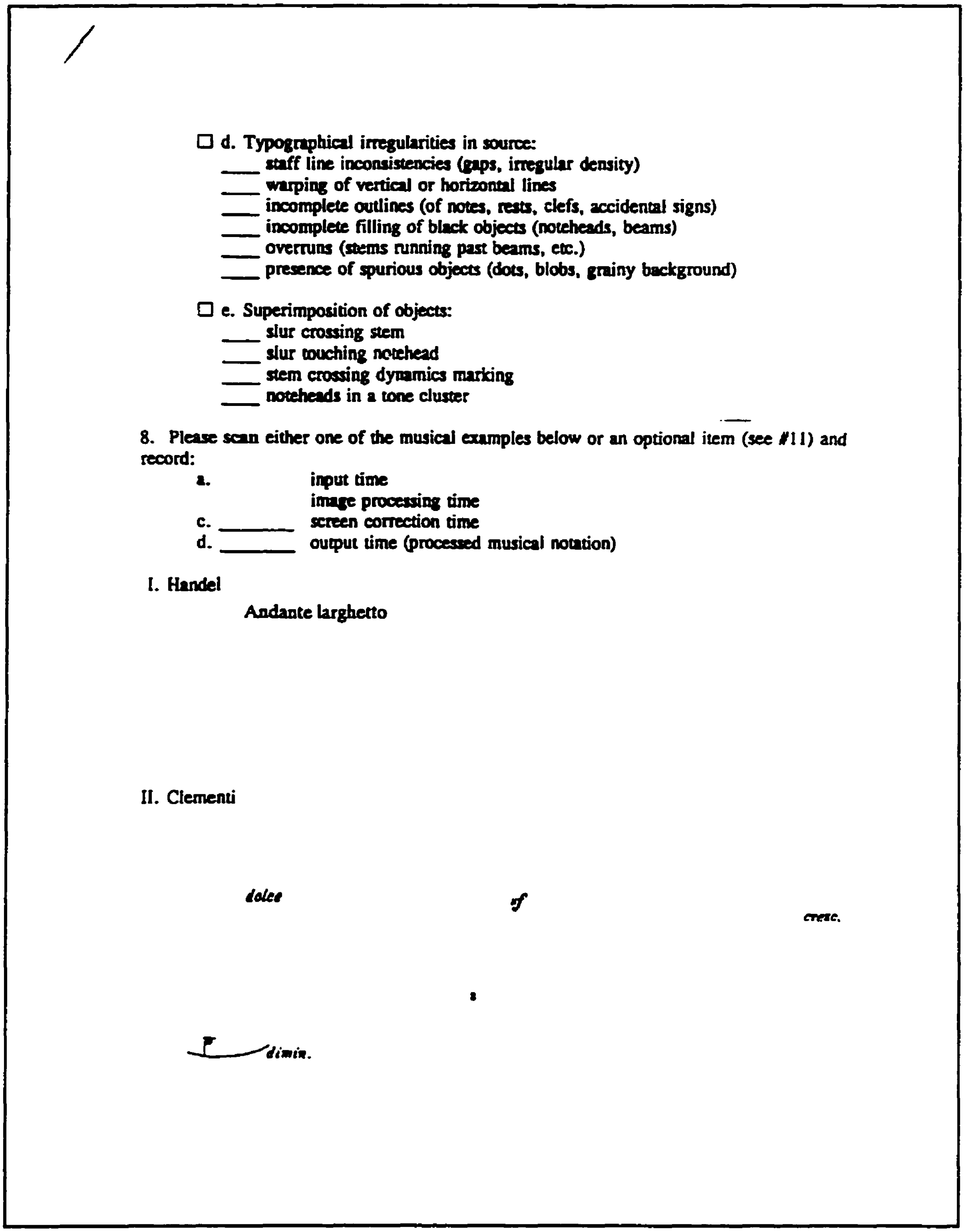
8. Please scan either one of the musical examples below or an optional item (see #11) and record:

a. ________ input time
b. ________ image processing time
c. ________ screen correction time
d. ________ output time (processed musical notation)

I. Handel

II. Clementi

Figure 4.43 The original with text.



/

□ d. Typographical irregularities in source:
____ staff line inconsistencies (gaps, irregular density)
____ warping of vertical or horizontal lines
____ incomplete outlines (of notes, rests, clefs, accidental signs)
____ incomplete filling of black objects (noteheads, beams)
____ overruns (stems running past beams, etc.)
____ presence of spurious objects (dots, blobs, grainy background)

□ e. Superimposition of objects:
____ slur crossing stem
____ slur touching notehead
____ stem crossing dynamics marking
____ noteheads in a tone cluster

8. Please scan either one of the musical examples below or an optional item (see #11) and record:
a. input time
image processing time
c. ________ screen correction time
d. ________ output time (processed musical notation)

I. Handel

Andante larghetto

II. Clementi

dolce sf cresc.

dimin.

Figure 4.44 Text extracted from Figure 4.43.

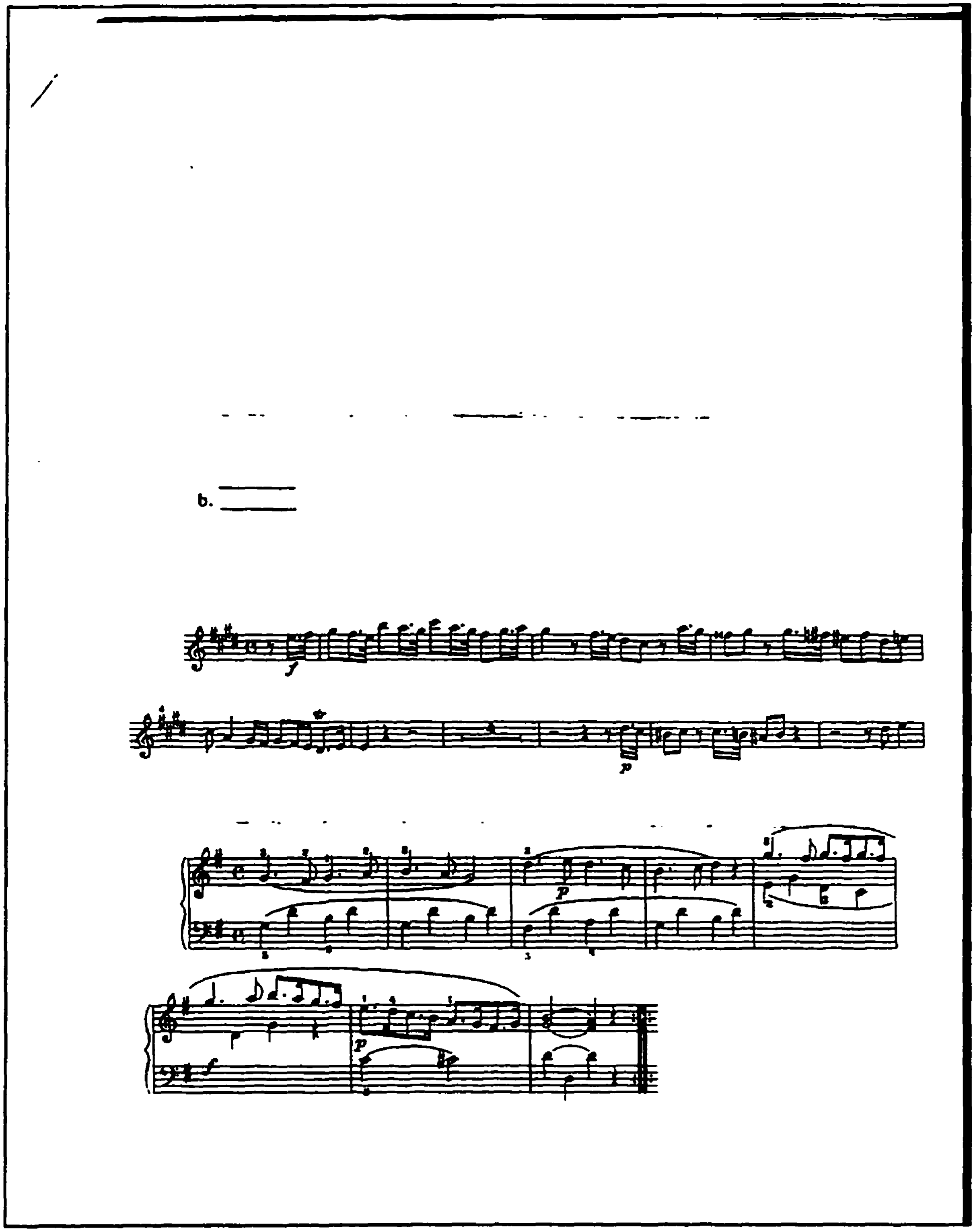


Figure 4.45 Text removed from Figure 4.43.

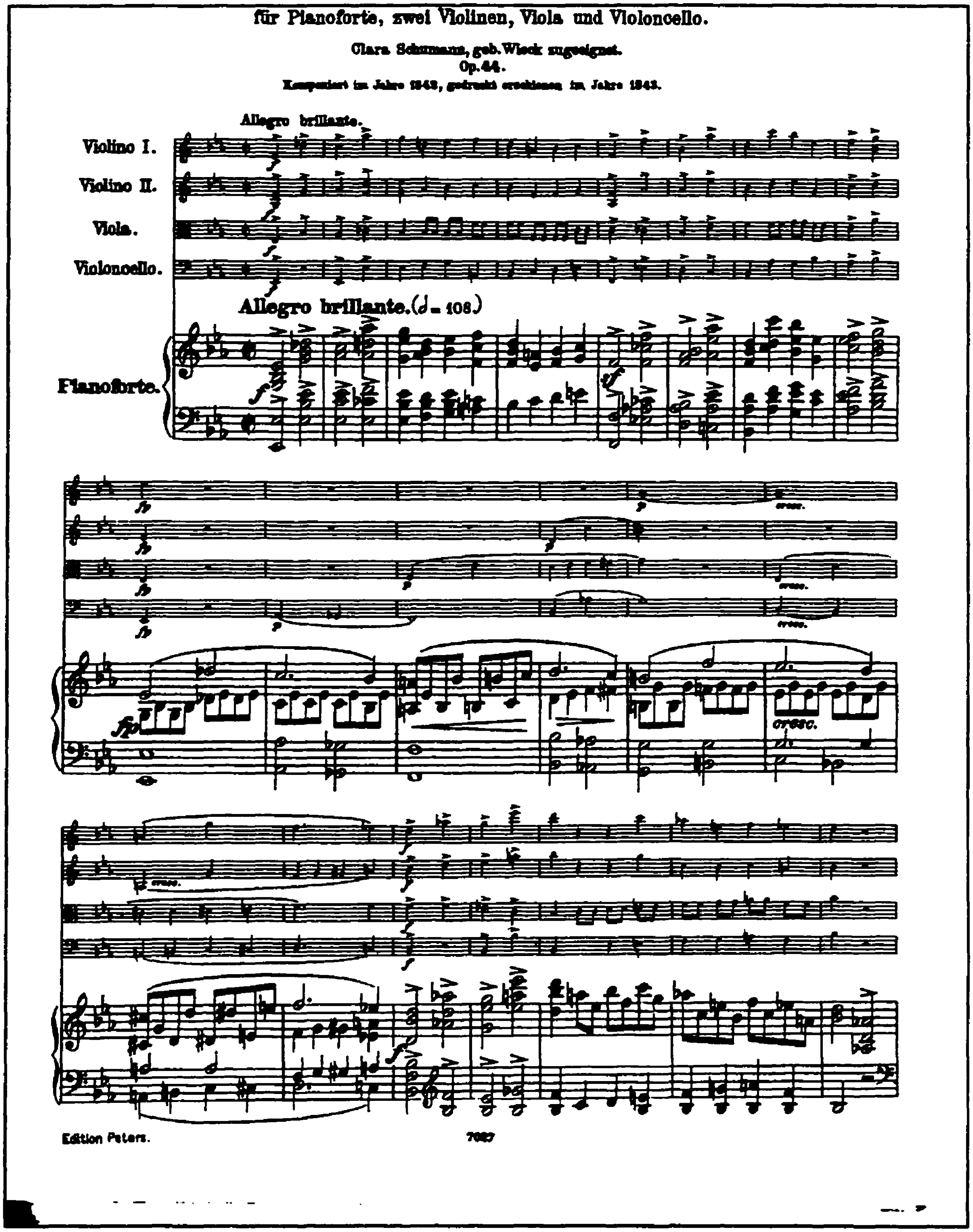

für Pianoforte, zwei Violinen, Viola und Violoncello.
Clara Schumann, geb. Wieck zugeeignet.
Op. 44.
Komponiert im Jahre 1842, gedruckt erschienen im Jahre 1843.
Allegro brillante.
Violino I.
Violino II.
Viola.
Violoncello.
Allegro brillante. (𝅗𝅥 = 108.)
Pianoforte.
f
fp
p
cresc.
Edition Peters.
7087

Figure 4.46 The original.

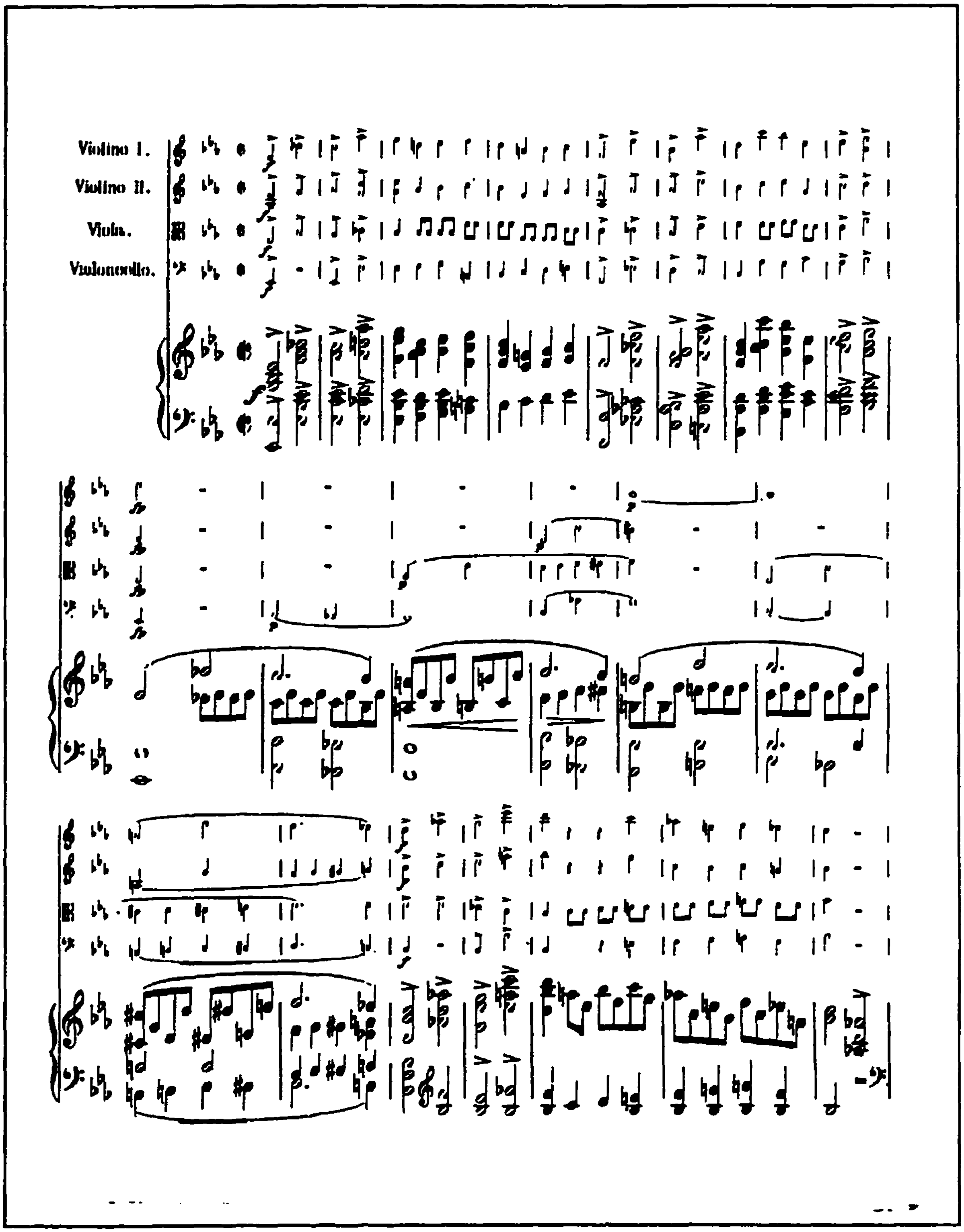


Figure 4.47 Stafflines and texts removed from Figure 4.46.

4.3 Segmentation

Segmentation is the process where symbols are separated from each other. This task is accomplished by the connected component analysis of the page after the stafflines and texts are removed. The analysis naturally separates the symbols because, by convention, most music symbols are not connected. In practice, however, symbols do touch and, of course, notes in a chord touch each other (see Figure 4.48).

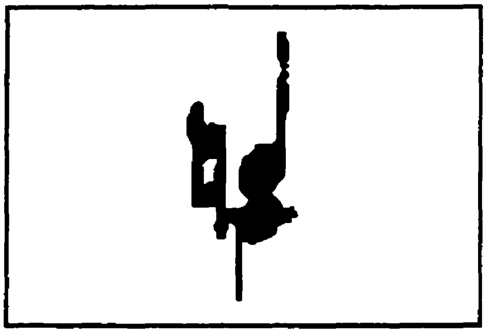


Figure 4.48 An example of attached music symbols.

In most pattern recognition systems, the segmentation stage precedes the classification stage, i. e., all the symbols are separated before being classified. In order to successfully segment symbols, it is necessary to know in advance, the characteristics of all the symbols. Since this is not possible in an environment where symbols may be connected in various ways, such as chords and beamed notes, and new symbols may be introduced, a more flexible method, which allows further segmentation during the classification stage, is implemented. The tactic deployed is explained in the Classification (4.5) section. Prior to the classification, each connected component is analyzed to extract its features.

4.4 Feature extraction

Features are the quantifiable aspects of a given symbol and are sets of the measurable properties of the symbol. The feature extraction phase calculates these descriptions, producing a set of measurements called feature vector for each connected component.

The following features are currently used in the AOMR system: width; height; area; area of the bounding box (width * height); rectangularity: Ao / Ab, which represents how well an object fills its bounding box; aspect ratio: width / height, which can distinguish slender objects from roughly square or circular objects; average number holes per horizontal and vertical scan lines; and normalized central moments, which provide a more detailed numerical description of the shape.

4.5 Classification

This phase uses the k-nearest neighbour (k-NN) classification technique to determine the class of a given unknown symbol on the basis of its feature vector. There are many reasons why the k-NN classification scheme is well-suited to this application. Aside from its simplicity and intuitive appeal, the classification requires no *a priori* knowledge about the underlying distribution of symbols in the feature space. This enables the system to learn new classes of symbols. Furthermore, a symbol class may occupy two or more disjunct regions. This is important because some musical symbols such as beams and slurs vary greatly in their shape and size; and other symbols such as the quarter rest and the tenor clef have completely different shapes depending on the music publishers (see Figure 4.49). Finally, the most significant reason for using this classifier is its ability to learn; that is, its accuracy improves as more data is collected.

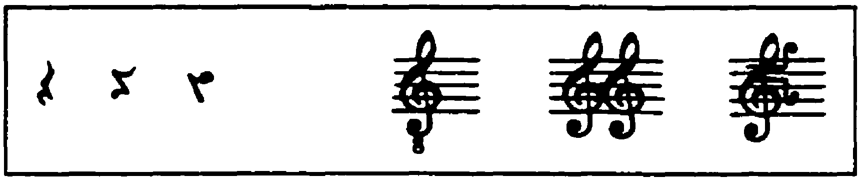


Figure 4.49 Examples of quarter rests and tenor clefs by different publishers.

As described in 3.3, a measure of the distance between an unclassified symbol and previously classified symbols is calculated between their feature vectors. The class represented by the majority of k-closest neighbour is then assigned to the unclassified symbol. Typically, such classes are actual music symbols, such as treble clef, notehead, and eighth rest, in which case, the program moves on to the next object. There are, however, four special classes of symbols that require further processing. These are:

1. STEM_COMPLEX (notes, chords, beamed notes)
2. CURVES (ties, slurs)
3. SPLIT_X
4. SPLIT_Y

4.5.1 Stem_complex

When a connected component is identified as a **stem_complex**, stems are automatically removed. The connected component is scanned horizontally and any wide black runs are removed. Then a connected component analysis is performed on the resulting image, and components that are narrow and tall are then marked for deletion in the original image (see Figure 4.50). Simply removing short horizontal black runs will not work because many things including flags will be removed (see Figure 4.51).

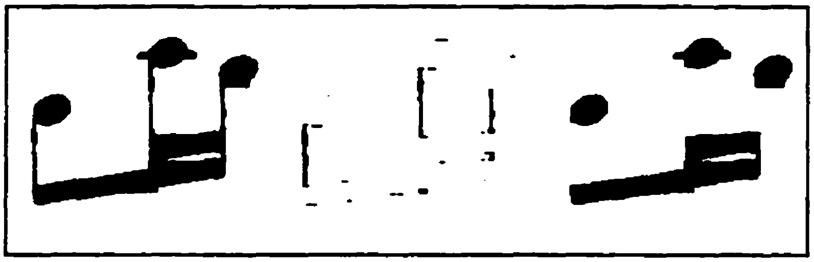


Figure 4.50 Removing stems from beamed notes.

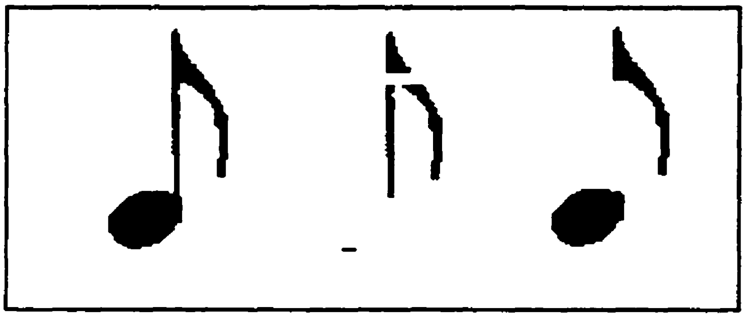


Figure 4.51 Removing stem from an eighth note.

4.5.2 Curves

In order to numerically define the shape of ties and phrase marks, the Bezier curve, originally developed for automobile designs (Hearn and Baker 1986, 195), is used. Bezier curves can define many types of curves with only four points (two endpoints plus two intermediate points) and are used widely in the computer-graphics field. Furthermore, the Bezier curve is implemented in the PostScript language, used for score reconstruction below.

In engraved-quality music, the phrase, slurs, and ties are not simple curves. They are thin at the ends and thicker in the middle. The algorithm to find the Bezier points of a curve works best if the curve has single pixel thickness. Thus, the phrase marks and ties are first "thinned" using a thinning algorithm. Thinning algorithms are used in many pattern recognition problems such as fingerprint identification (Kamesawara and Rao 1978), logic and electrical schematic interpretation (Jarris 1977), and character recognition (Kumar et al. 1991). Thinning is a method of reducing the width of a digitized pattern to a single pixel. The classic algorithm by Zhang and Suen (1984) is implemented here.

Given the notation of 3x3 window around point P1:

P9	P2	P3
P8	P1	P4
P7	P6	P5

,

the algorithm uses two passes as follows:

1. Pixel P1 is deleted from the digital image if it satisfies the following:
   a) P2 * P4 * P6 = 0 (i.e., if any one of the pixel is 0)
   b) P4 * P6 * P8 = 0
   c) A(P1) = 1
   d) 2 <= B(P1) <= 6
2. In the second iteration, pixel P1 is deleted if it satisfies the following:
   a) P2 * P4 * P8 = 0
   b) P1 * P6 * P8 = 0
   c) A(P1) = 1
   d) 2 <= B(P1) <= 6

Where, A(P1) is the number of 01 patterns in the ordered set P2, P3, . . . , P9, and $B(P1) = \sum_{i=2}^{9} Pi$.

In order to use the curve-fitting algorithm, one of the end points must be found. This is accomplished by searching, from top to bottom, and left to right, a point that only has one neighbour. Once the endpoints are located, the least-squares method is used to find the two Bezier control points (Glassner 1990).

4.5.3 SPLIT_X and SPLIT_Y

Predefined symbols called SPLIT_X and SPLIT_Y which, when identified, direct the recognizer to further segment a given symbol either horizontally or vertically. The separation of the SPLIT_X and SPLIT_Y symbols uses the minimum values of x-projection and y-projection, respectively. (See Figures 4.52 and 4.53). This method results in an efficient and robust recognition of the near infinite configuration of chords and attached symbols.

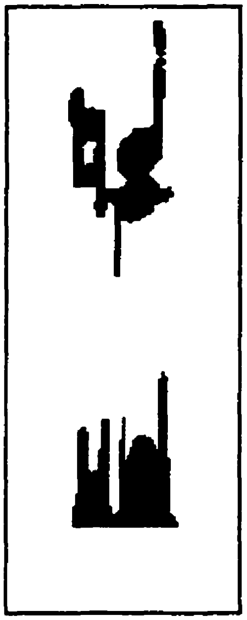


Figure 4.52 X-projection for SPLIT_X

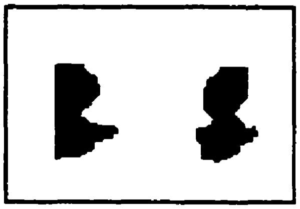
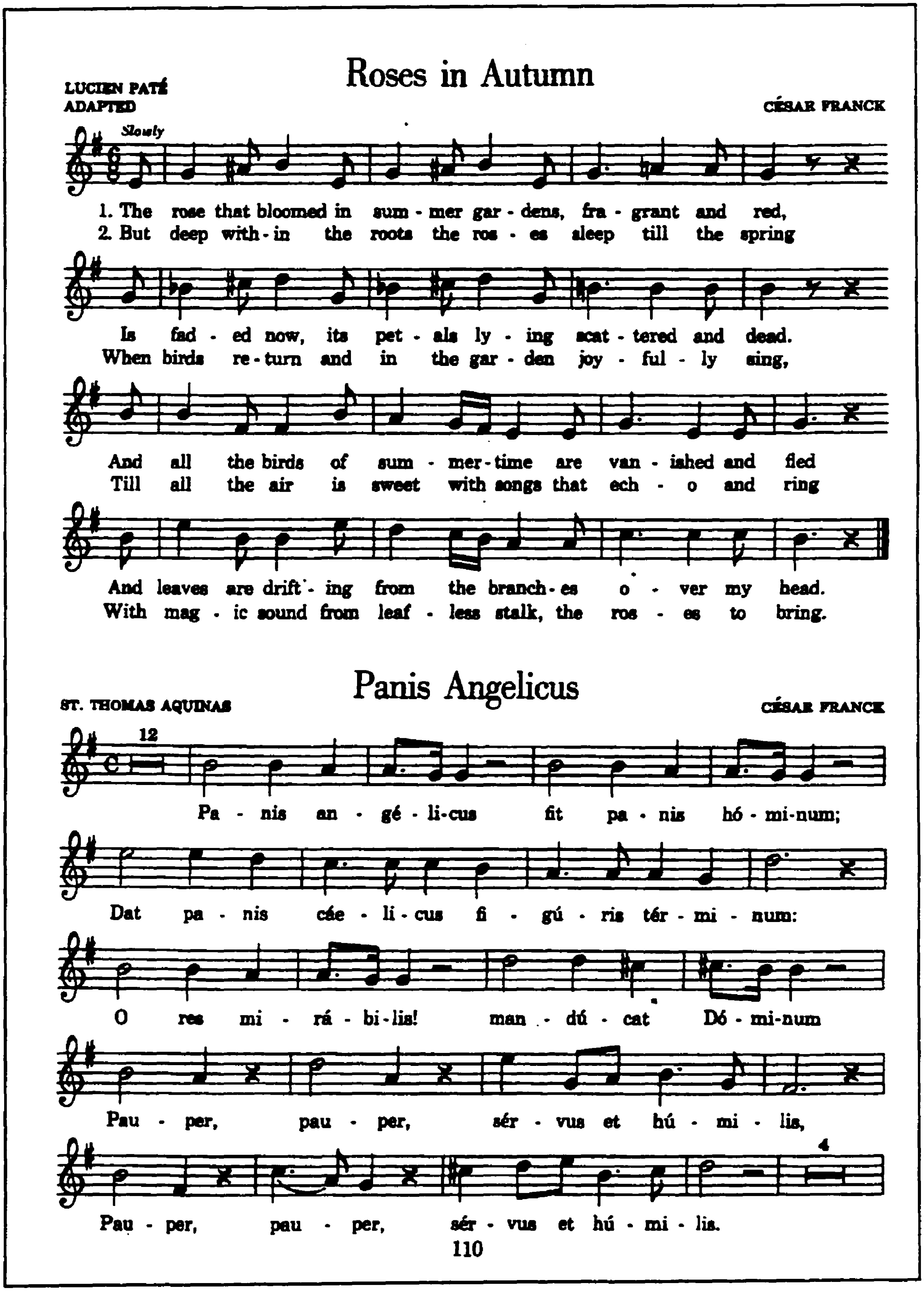


Figure 4.53 Y-projection for SPLIT_Y

4.6 Score reconstruction

Elementary score reconstruction is attempted to visually verify the accuracy of the classifier. The output is a PostScript file with x- and y-coordinates of the symbols. For stafflines, beams, stems, and barlines, the two endpoints and the thickness of the line are provided. For slurs and ties, two endpoints along with two Bezier points are indicated so that the PostScript interpreter can draw the curves (see Figures 4.54 and 4.55). The output of the recognition process can be used in various applications, see for example, Wilk (1995), which generates MIDI data.



Roses in Autumn

LUCIEN PATÉ
ADAPTED

CÉSAR FRANCK

Slowly

1. The rose that bloomed in sum - mer gar - dens, fra - grant and red,
2. But deep with - in the roots the ros - es sleep till the spring

Is fad - ed now, its pet - als ly - ing scat - tered and dead.
When birds re - turn and in the gar - den joy - ful - ly sing,

And all the birds of sum - mer - time are van - ished and fled
Till all the air is sweet with songs that ech - o and ring

And leaves are drift - ing from the branch - es o - ver my head.
With mag - ic sound from leaf - less stalk, the ros - es to bring.

Panis Angelicus

ST. THOMAS AQUINAS

CÉSAR FRANCK

Pa - nis an - gé - li - cus fit pa - nis hó - mi - num;

Dat pa - nis cáe - li - cus fi - gú - ris tér - mi - num:

O res mi - rá - bi - lis! man - dú - cat Dó - mi - num

Pau - per, pau - per, sér - vus et hú - mi - lis,

Pau - per, pau - per, sér - vus et hú - mi - lis.

110

Figure 4.54 The original.

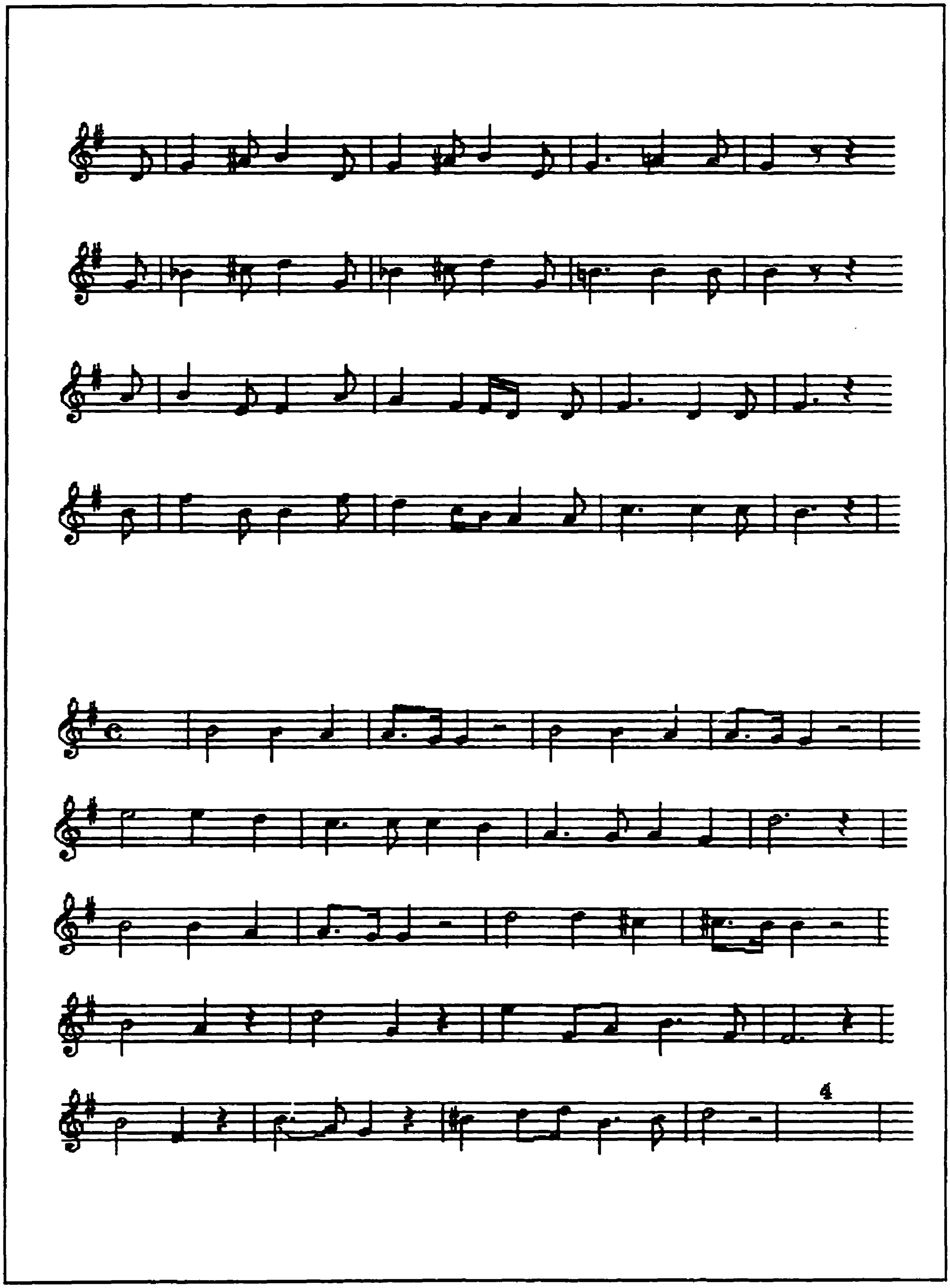


Figure 4.55 The reconstructed PostScript output of Figure 4.54.

4.7 Learning

The primary goal of the learning phase is to improve the accuracy of recognition. Enhancing the efficiency of the recognition is a secondary goal for the following reasons:

1. After an initial training period, the recognition task can be performed without human intervention through background processing and, if necessary, on multiple computers.

2. The speed of processing is directly related to the number of features used and the number of symbols stored in the database. The size of random-access memory (RAM) commonly found on today's computer limits the practical size of the database. For example, if 20 features are used for each symbol, and if each feature requires 4 bytes of storage (80 bytes per symbol), then 100,000 symbols would occupy 8 megabytes of RAM. Using Sun SPARC 2, the processing time is estimated to be about 500 ms / symbol, so that for a page containing 1000 music symbols it would take 500 seconds, or about 8 minutes.

3. It is estimated that the proofing of a page of music by a trained editor would take, depending on the complexity of the music, anywhere from a few minutes to an hour (Carter 1994b). Since most OMR systems do not claim, including AOMR, 100% accuracy, the result must be checked by human editors. Therefore, the processing time for an OMR system need only be comparable to that of a human editor.

4.7.1 Limiting the size of the database

Since there is a physical upper limit to the size of the database that can be stored in RAM, there must be a mechanism to reduce the size of the database while maintaining the accuracy. Thus, the Editing (3.4.1) and the Condensing (3.4.2) methods to reduce the size were implemented. Although both of these procedures were successful in reducing the size of the database, the accuracy suffered as the result of the reduction in the size of the stored library.

4.7.2 Accuracy

The main characteristic of the k-NN classifier is that, in theory, its accuracy increases as more data is accumulated. Simply storing classified symbols in the database increases its accuracy. Another way to improve the classification is by using different distance measures. At any time during the development process, different distance measures can be tested to see which one of the available methods achieves the best result. This approach makes the system flexible, using the best type of distance measure for the particular environment.

Although it is not complicated or time consuming to try a handful of different measures, selecting the optimal weights used in some of the measures is very difficult. This is the problem of assigning relative importance of the features when calculating the distances within the feature-space.

In many classification applications, the features are "selected," hence the term feature selection. In this process, whether or not a feature is used in the distance calculations is equivalent to deciding whether to assign 0 or 1 as the weight of each feature. This selection process requires a total of 2^f number of combinations of weights, where f is the total number of features.

The performance or the rate of accuracy of a set of weights is determined by the "leave-one-out" method, which means that, for each symbol *S* in the database, *S* is assumed to be unknown and the remaining symbols in the database are used to identify *S*. If the result corresponds to the true identity of *S*, then the system is said to identify the symbol correctly. All members of the database go through this procedure to calculate the global accuracy of the system.

Determining the set of weights that will result in the most accurate classification is an extremely computing-intensive task, for the best set can only be obtained by examining all possible combinations (Cover and Van Campenhout 1977). Furthermore, the weights of the features can be varied (using reai numbers) so that the complete set of possible combinations are virtually infinite. With the currently available computing power, exhaustive search using a relatively small database would take years to calculate. An extremely elegant and practical solution to this problem of selecting near-optimal weights is provided here by using the genetic algorithms.

4.7.3 Application of a genetic algorithm

To illustrate the use of a genetic algorithm (see 3.10 above) for finding a good set of weights, five randomly chosen pages of music are used. DATA A is created from Figure 4.54 and Figure 4.56; in DATA B, DATA C, and DATA D, the symbols from Figure 4.57, 4.58, and 4.59 are added respectively. In other words, the symbols from each page are combined sequentially to create the four datasets, thus, DATA D, for example, contains all the symbols from the four pages. Figure 4.60 shows, for each dataset, the number of different classes, total number of symbols, and the time required to find three best sets of binary weights exhaustively, i. e. by testing all possible combinations.

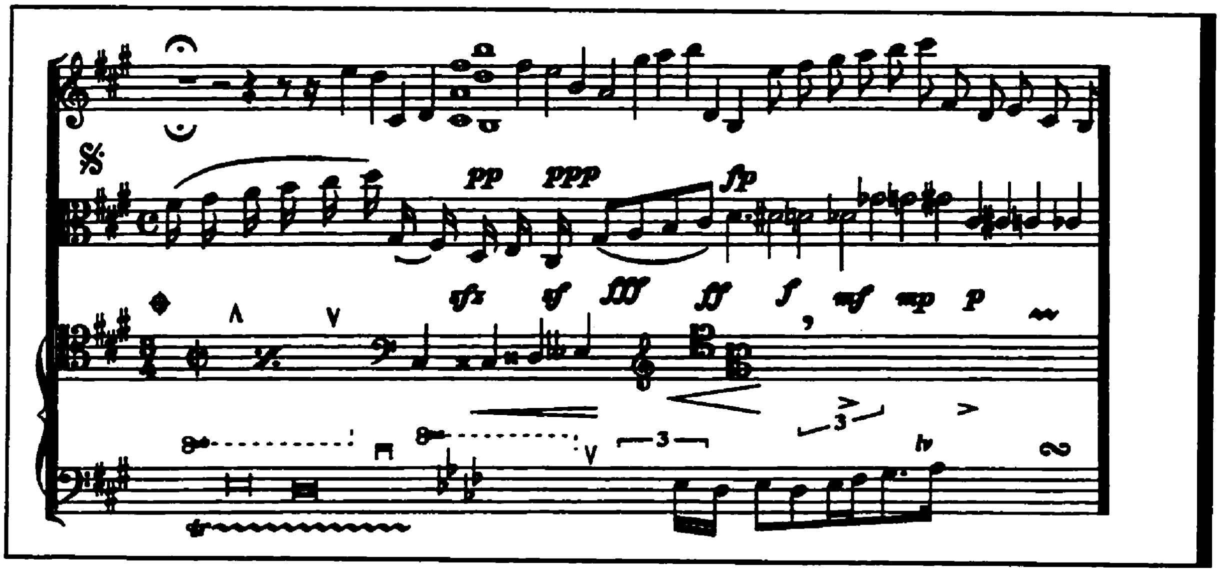
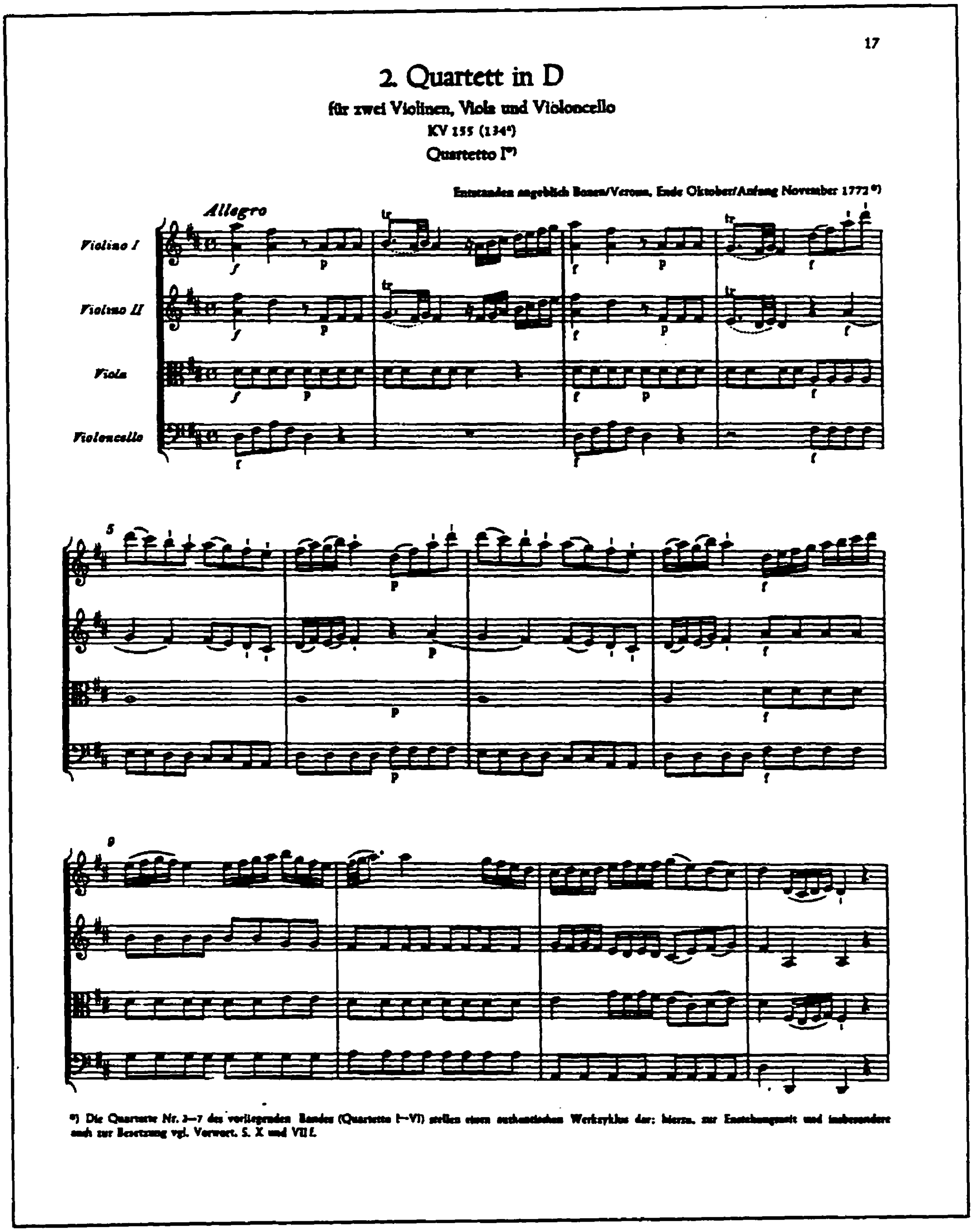


Figure 4.56 The page is used to create DATA A along with Figure 4.54.



17

2. Quartett in D

für zwei Violinen, Viola und Violoncello

KV 155 (134ª)

Quartetto I*)

Entstanden angeblich Bozen/Verona, Ende Oktober/Anfang November 1772*)

*) Die Quartette Nr. 2–7 des vorliegenden Bandes (Quartetto I–VI) stellen einen authentischen Werkzyklus dar; hierzu, zur Entstehungszeit und insbesondere auch zur Besetzung vgl. Vorwort, S. X und VIII f.

Figure 4.57 Sample page used to create DATA B.

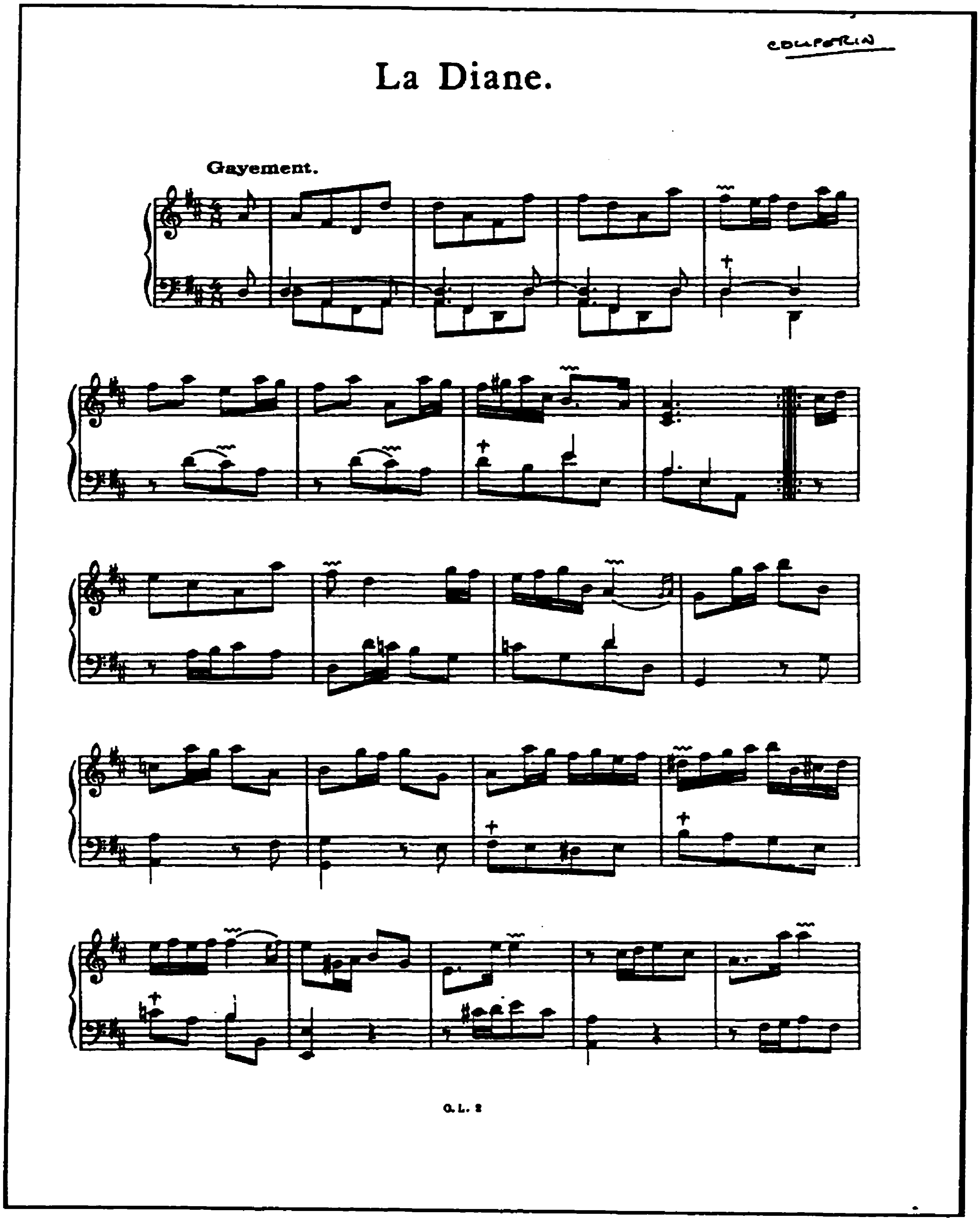


Figure 4.58 Sample page used to create DATA C.

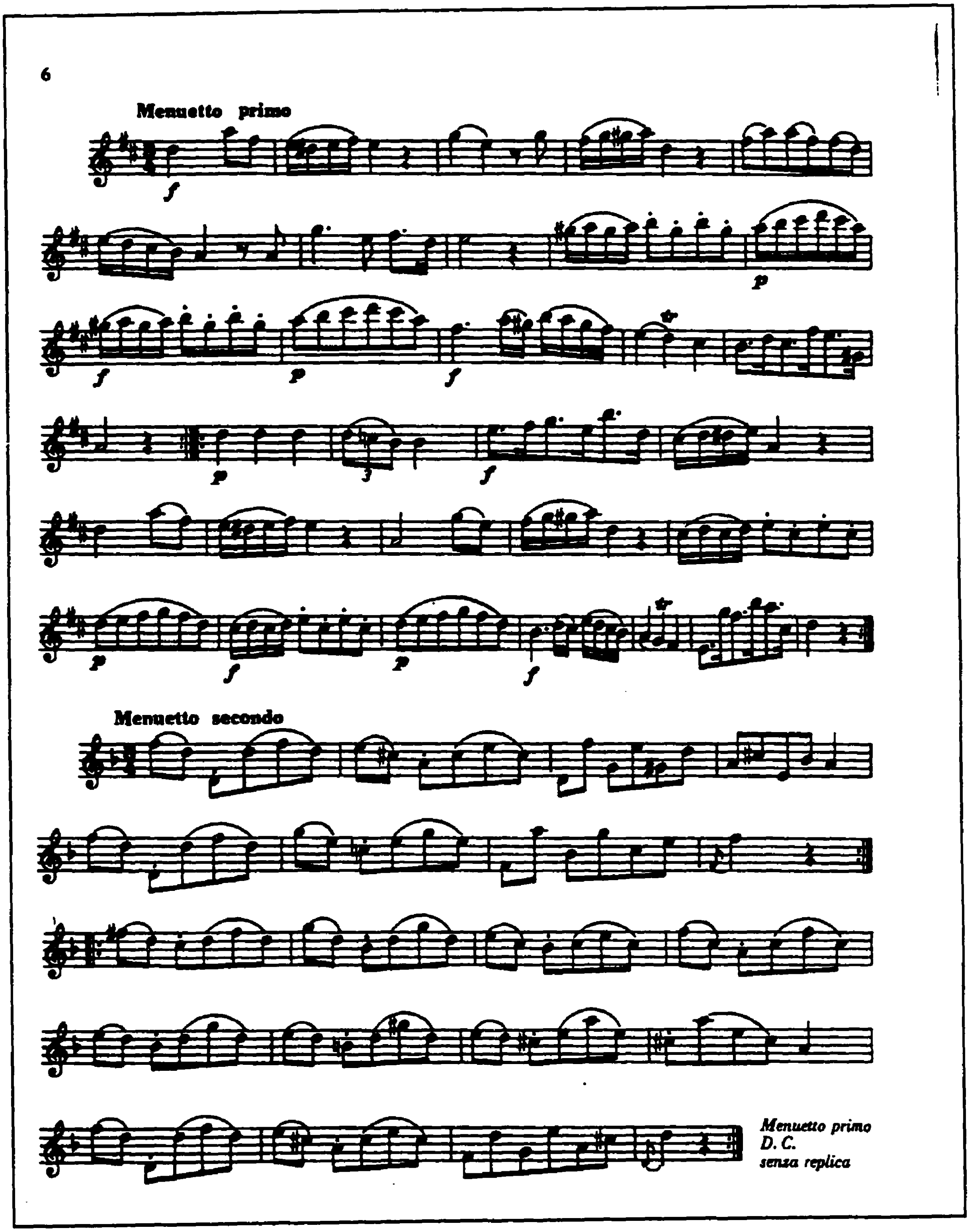


Figure 4.59 Sample page used to create DATA D.

The symbol distribution of DATA D, assembled from the four pages, is shown in Figure 4.61. The features currently implemented are listed in Figure 4.62. For each dataset, the recognition rates (the rate of 1.0 would mean 100% accuracy) using only one feature are shown in Figure 4.63 using 1-NN classification scheme (k = 1). Unfortunately, these results are not particularly useful because the combinations of the best individual features do not guarantee best results. For example, the feature 4 (x-centre of gravity) is used in all three of the top three sets for DATA A (Figure 4.64), yet by itself it is the second worst feature (0.366412). Conversely, the feature 2 (area) is the top performer by itself in DATA B (0.7000405), yet it is not used in two of the top three sets of weights (see Figure 4.64). The only way to find the set of features that result in the best recognition rate is by trying out all possible combinations. Since there are 15 features used here, there are 32767 (2^{15}) possibilities for the binary weights, and the calculation of the recognition rates takes inordinate amount of time as shown in last column of Figure 4.60.

	number of classes	number of symbols	processing time
DATA A:	19	524	24 hrs
DATA B:	29	1235	6 days (estimated)
DATA C:	32	1745	12 days (estimated)
DATA D:	32	2538	25 days (estimated)

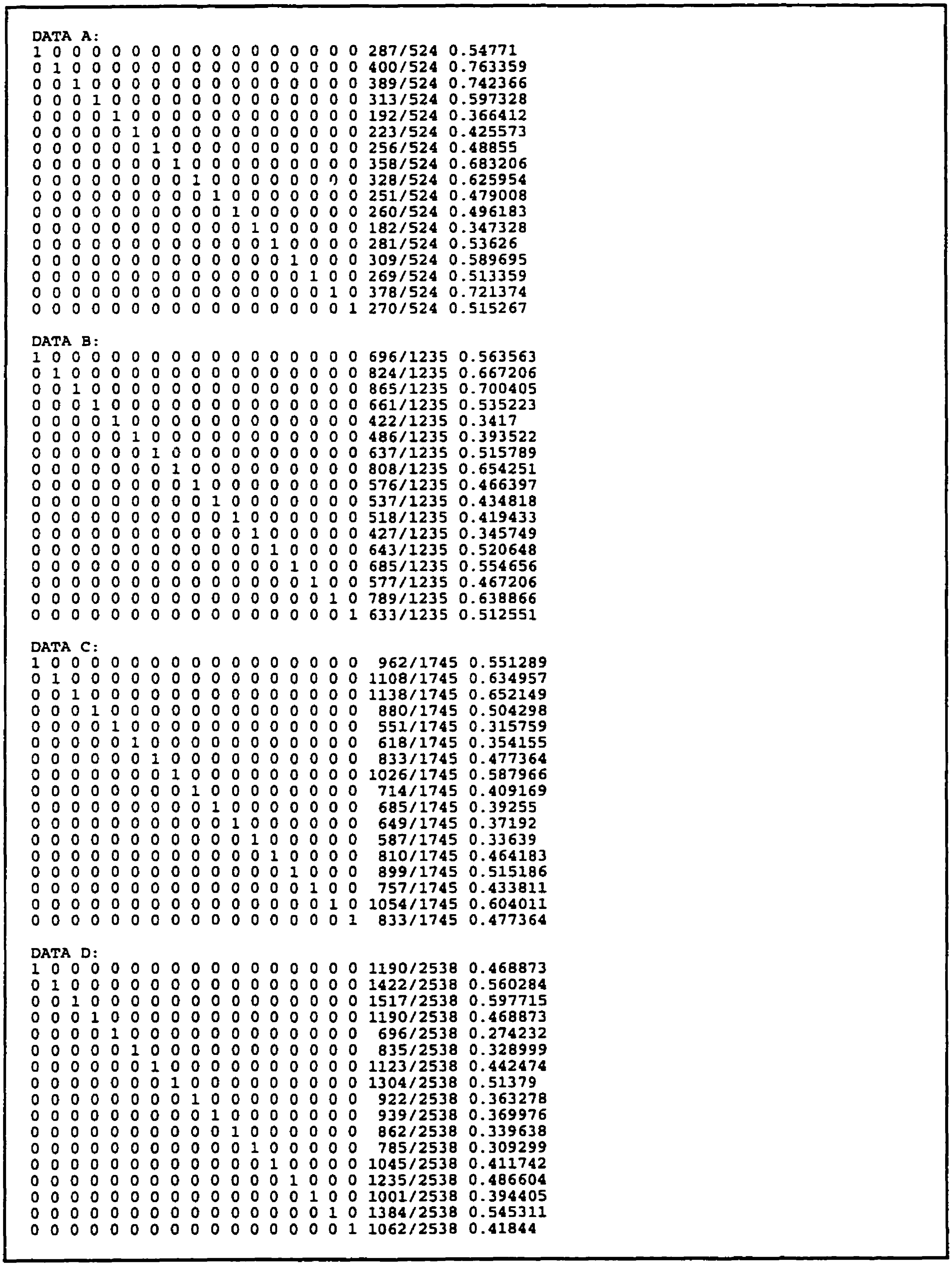
Figure 4.60 The size of each dataset and the processing time to find the optimal set of binary weights.

	count	name
1	94	sharp
2	14	flat
3	12	natural
4	33	trebleclef
5	96	dot
6	20	eighthflagdown
7	26	eighthflagup
8	5	sxflagdown
9	5	commontime
10	16	piano
11	22	forte
12	135	barline
13	6	heavybarline
14	6	wholenote
15	32	quarterrest
16	19	eighthrest
17	9	halfrest
18	168	beam1
19	68	beam2
20	5	beam3
21	24	stemsegment
22	5	brace
23	119	slur
24	21	halfnotehead
25	794	quarternotehead
26	33	quarternotehead-ledger-below
27	83	quarternotehead-ledger-middle
28	7	quarternotehead-ledger-above
29	99	splitx
30	96	splity
31	447	beam_complex
32	19	ledger_complex

Figure 4.61 Symbol distribution of DATA D.

0	width
1	height
2	area (width * height)
3	volume (pixel count)
4	x-centre of gravity
5	y-centre of gravity
6	u20 normalized central moments
7	u02 "
8	u11 "
9	u30 "
10	u12 "
11	u21 "
12	u03 "
13	n_holes_vertical
14	n_holes_horizontal

Figure 4.62 Currently implemented list of features.



```
DATA A:
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 287/524 0.54771
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 400/524 0.763359
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 389/524 0.742366
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 313/524 0.597328
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 192/524 0.366412
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 223/524 0.425573
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 256/524 0.48855
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 358/524 0.683206
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 328/524 0.625954
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 251/524 0.479008
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 260/524 0.496183
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 182/524 0.347328
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 281/524 0.53626
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 309/524 0.589695
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 269/524 0.513359
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 378/524 0.721374
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 270/524 0.515267

DATA B:
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 696/1235 0.563563
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 824/1235 0.667206
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 865/1235 0.700405
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 661/1235 0.535223
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 422/1235 0.3417
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 486/1235 0.393522
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 637/1235 0.515789
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 808/1235 0.654251
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 576/1235 0.466397
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 537/1235 0.434818
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 518/1235 0.419433
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 427/1235 0.345749
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 643/1235 0.520648
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 685/1235 0.554656
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 577/1235 0.467206
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 789/1235 0.638866
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 633/1235 0.512551

DATA C:
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  962/1745 0.551289
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1108/1745 0.634957
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1138/1745 0.652149
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  880/1745 0.504298
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0  551/1745 0.315759
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0  618/1745 0.354155
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0  833/1745 0.477364
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1026/1745 0.587966
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0  714/1745 0.409169
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0  685/1745 0.39255
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0  649/1745 0.37192
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0  587/1745 0.33639
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0  810/1745 0.464183
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0  899/1745 0.515186
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0  757/1745 0.433811
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1054/1745 0.604011
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1  833/1745 0.477364

DATA D:
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1190/2538 0.468873
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1422/2538 0.560284
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1517/2538 0.597715
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1190/2538 0.468873
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0  696/2538 0.274232
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0  835/2538 0.328999
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1123/2538 0.442474
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1304/2538 0.51379
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0  922/2538 0.363278
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0  939/2538 0.369976
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0  862/2538 0.339638
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0  785/2538 0.309299
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1045/2538 0.411742
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1235/2538 0.486604
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1001/2538 0.394405
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1384/2538 0.545311
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1062/2538 0.41844
```

Figure 4.63 Recognition rates for individual features.

```
DATA A:
1 1 1 0 1 1 0 0 1 0 0 0 0 1 1 0 1 511/524 0.975191
1 0 1 0 1 1 1 1 1 0 0 0 0 1 1 0 1 511/524 0.975191
0 1 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 511/524 0.975191

DATA B:
1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1213/1235 0.982186
1 1 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1213/1235 0.982186
1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1213/1235 0.982186

DATA C:
1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 1 1 1714/1745 0.982235
1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1 1714/1745 0.982235
1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1714/1745 0.982235

DATA D:
1 1 0 1 0 1 1 0 1 1 1 0 0 1 1 1 1 2456/2538 0.967691
1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 2456/2538 0.967691
1 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 2456/2538 0.967691
```

Figure 4.64 The best three set of weights for accuracy found by genetic algorithm for each dataset.

Furthermore, ideally the number of stored symbols in the database should be much greater. For example, using 25000 stored symbols is not unreasonable, since it would take about 2 Mbytes of storage (80 bytes per symbol) and if there are 1000 symbols on the page of music, processing time would be about 4 minutes. Finding the optimal set of features for this database, however, would take over six years!

This is why the application of a genetic algorithm (GA), which finds the near-optimal set of features in much less time, is essential. The results of the four datasets using GA are shown in Figure 4.64. The search for each dataset was stopped after 12 hours. Although these may not be the best sets (for DATA A, an exhaustive search confirmed that these are indeed the best sets), the obtained accuracy in the range of 96% to 98% seems more than acceptable.

The necessity of using a GA becomes more evident as there are two further refinements that can be made to the classification process: using a different k in the k-NN classification and using non-binary weights. In the results above the k was set to 1, but other numbers can be used. Figure 4.65 shows the best sets for k = 3 and k = 5 for DATA A, where there are slight improvements (compare with Figure 4.64). Also, any real numbers can be used as the weights for each feature. Implementing this would increase the calculation time astronomically, yet, as shown in Figure 4.66, the accuracy is improved over the binary weights. Figure 4.65 used four possible weights (0, 0.25, 0.5, and 0.75), thus the total number of combination is increased to 4^{15} or over one billion. The calculation for DATA D in this case, would take over 2000 years! Nevertheless, the power of GA methods are such that very good sets (exceeding the accuracy of the binary weights) were found within 24 hours.

```
DATA A (k=3):
1 0 0 1 1 1 1 1 1 0 0 0 0 1 1 0 1  508/518: 0.980695
1 0 0 1 1 1 1 1 1 0 1 0 0 1 1 0 1  508/518: 0.980695
1 1 1 0 1 1 0 1 1 0 1 0 1 1 1 0 1  509/520: 0.978846

DATA A (k=5):
0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1  501/512: 0.978516
0 0 0 1 1 0 0 1 1 0 1 1 1 1 1 0 1  501/512: 0.978516
0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 1  501/512: 0.978516
```

Figure 4.65 Recognition rates for DATA A using k=3 and k=5. Note that some samples are rejected because a majority of neighbours could not be established. This occurs, for example, in the 3-NN case, all three nearest neighbours are from different classes.

```
DATA C:
.75 .50 .00 .50 .00 .50 .50 .50 .50 .75 .50 .75 .50 .25 .00 1618/1634: .990208
.75 .25 .00 .75 .25 .50 .50 .25 .25 .75 .50 .75 .50 .50 .75 1618/1634: .990208
.75 .75 .50 .75 .25 .25 .25 .75 .50 .00 .25 .25 .50 .75 .75 1616/1634: .988984

DATA D:
.75 .75 .75 .75 .00 .25 .25 .75 .50 .00 .75 .50 .00 .75 .75 2465/2538: .971237
.75 .75 .50 .75 .25 .25 .75 .75 .50 .25 .00 .00 .25 .75 .75 2464/2538: .970843
.50 .75 .00 .75 .00 .25 .25 .50 .25 .50 .75 .00 .25 .75 .50 2464/2538: .970843
```

Figure 4.65 Recognition rates for DATA C and DATA D using four possible weights (0, 0.25, 0.5, and 0.75).

In general, using the binary weights and the k set to 1, the accuracy of the AOMR system is between 95% to 100% depending on the complexity of the music, the quality of typesetting or handwriting and the size of the database. The processing time is 5 to 15 minutes per page, proportional to the number of symbols on the page and the size of the database.

5. CONCLUSIONS

5.1 Future work

5.1.1 Problems

The ultimate test for an adaptive system is to observe passively its performance in various environments. From the designer's point of view, this was difficult to achieve because of the designer's desire to make the best possible system before it is completely released into the field. The tendency has been to watch the system evolve for a while, and then as soon as a problem develops, the system is modified and the process begins again.

The next step in the development is to make the system run on its own. Some of the operations—the genetic algorithms, for example—are manually initiated. Also, the evaluation of different similarity measures is not automatic. These different components must be completely integrated and made autonomous.

5.1.2 Extensions

In this research, the accuracy and the efficiency of the recognition were monitored through the learning system. This can be easily extended so that the accuracy and efficiency of various learning strategies are monitored and optimized. There are certain parameters in the genetic algorithm such as the mutation and crossover rate that can be adjusted. For error estimation, only the leave-one-out method was used here. There are other methods that can be implemented and assessed. In other words, the system explores other learning methods and evaluates their performance. This is the concept of learning to learn.

5.2 Final thoughts

In order to understand music or other manifestations of human nature, one must be aware of the bias and limitations of the investigators themselves and the tools used for the inquiry. The common serial type of computer and the associated programming language are based on procedural and formalized models of thought. In our education, formalization, reductions, and generalizations are extremely valued. In fact, these are the summit of characteristics of intelligence, at least in the modern Western world. Perhaps influenced by this, in the history of artificial intelligence, major efforts have gone into establishing formalization of human thought and perceptual processes, searching for sets of rules. Yet, in many disciplines, building rule-based models of human understanding of our world have not been successful. For example, formalizing music has been very difficult, despite many attempts made by music theorists over the years. There is an alternative approach, however. Numerous philosophers and psychologists believe that many concepts are learned directly by examples and not by rules. The proposed system here is based on that idea and the feasibility of such a system for music notation recognition has been demonstrated.

Exemplar-based adaptive systems can potentially be applied in many fields where solving problems by formalized rule-based system has failed. In the field of music alone there are various possible applications. Music structure recognition (phrase, modulation, themes, motives), timbre identification, pitch detection, and tempo tracking are some of the areas where the adaptive system can be used for enriching our understanding of music.

6. BIBLIOGRAPHY

Akiyama, T. and N. Hagita. 1990. Automated entry system for printed documents. *Pattern Recognition.* 23(11): 1141–54.

Alphonce, B., B. Pennycook, I. Fujinaga, and N. Boisvert. 1988. Optical music recognition: A progress report. *Proceedings of the Small Computers in the Arts.* 8–12

d'Andecy, V. P., J. Camillerapp, and I. Leplumey. 1994. Détecteur robuste de segments-Application à l'analyse de partitions musicales. *Actes 9 ème Congrés AFCET Reconnaissance des Formes et Intelligence Artificielle.*

Andronico, A. and A. Ciampa. 1982. On automatic pattern recognition and acquisition of printed music. *Proceedings of the International Computer Music Conference.* 245–78.

Aoyama, H. and A. Tojo. 1982a. Automatic recognition of music score (in Japanese). *Electronic Image Conference Journal.* 11(5): 427–35.

Aoyama, H. and A. Tojo. 1982b. Automatic recognition of music score. *Proceedings of the 6th International Conference on Pattern Recognition.* 1223.

Aoyama, H. and A. Tojo. 1982c. Automatic recognition of printed music (in Japanese). *Institute of Electronics and Communications Engineers of Japan (IECE).* TG PREL82–5: 33–40.

Armand, J.-P. 1993. Musical score recognition: A hierarchical and recursive approach. *Proceedings of the International Conference on Document Analysis and Recognition.* 906–9.

Bainbridge, D. 1991. *Preliminary experiments in musical score recognition.* B.Sc. Thesis. University of Edinburgh.

Baumann, S. and A. Dengel. 1992. Transforming printed piano music into MIDI. *Proceedings of International Workshop on Structural and Syntactic Pattern Recognition.* 363–72.

Bhattacharya, B., R. Poulsen, and G. Toussaint. 1992. Application of proximity graphs to editing nearest neighbour decision rules. *Technical Report No. SOCS 92.19.* McGill University.

Blostein, D. and H. S. Baird. 1992. A critical survey of music image analysis. In *Structured Document Image Analysis,* ed. H. S. Baird, H. Burke, and K. Yamamoto. Berlin: Springer. 405–34.

Blostein, D. and L. Haken. 1990. Template matching for rhythmic analysis of music keyboard input. *Proceedings of 10th International Conference on Pattern Recognition.* 767–70.

Blostein, D. and L. Haken. 1991. Justification of printed music. *Communications of the ACM.* 88–91.

Bryant, J. 1989. A fast classifier for image data. *Pattern Recognition.* 22(1): 45–48.

Bulis, A., R. Almog, M. Gerner, and U. Shimony. 1992. Computerized recognition of hand-written musical notes. *Proceedings of the International Computer Music Conference.* 110–12.

Carter, N. P. 1989. *Automatic recognition of printed music in the context of electronic publishing.* Ph.D. Thesis. University of Surrey.

Carter, N. P. 1992a. A new edition of Walton's Façade using automatic score recognition. *Proceedings of International Workshop on Structural and Syntactic Pattern Recognition.* 352–62.

Carter, N. P. 1992b. Segmentation and preliminary recognition of madrigals notated in white mensural notation. *Machine Vision and Applications.* 5(3): 223–30.

Carter, N. P. 1994a. Conversion of the Haydn symphonies into electronic form using automatic score recognition: a pilot study. *Proceedings of SPIE.* 2181: 279–90.

Carter, N. P. 1994b. Music score recognition: Problems and prospects. *Computing in Musicology.* 9: 152–8.

Carter, N. P. and R. A. Bacon. 1992. Automatic recognition of printed music. In *Structured Document Image Analysis,* ed. H. S. Baird, H. Bunke, and K. Yamamoto. Berlin: Springer. 456–65.

Carter, N. P. and R. A. Bacon. 1990. Automatic recognition of music notation. *Proceedings of the International Association for Pattern Recognition Workshop on Syntactic and Structural Pattern Recognition.* 482.

Carter, N. P., R. A. Bacon, and T. Messenger. 1988. The acquisition, representation and reconstruction of printed music by computer: A survey. *Computers and the Humanities.* 22: 117–36.

Cash, G. and M. Hatamian. 1987. Optical character recognition by the method of moments. *Computer Vision, Graphics, and Image Processing.* 39: 291–310.

Castleman, K. R. 1979. *Digital image processing.* Englewood Cliffs: Prentice-Hall.

Choi, J. 1991. *Optical recognition of the printed musical score.* M.S. Thesis (Electrical Engineering and Computer Science). University of Illinois at Chicago.

Chaudhuri, D., C. A. Murthy, and B. Chaudhuri, 1992. A modified metric to compute distance. *Pattern Recognition* 25(7): 667–77.

Clarke, A. T., B. M. Brown, and M. P. Thorne. 1988. Inexpensive optical character recognition of music notation: A new alternative for publishers. *Proceedings of the Computers in Music Research Conference.* 84–7.

Clarke, A. T., B. M. Brown, and M. P. Thorne. 1988. Using a micro to automate data acquisition in music publishing. *Microprocessing and Microprogramming.* 24: 549–54.

Clarke, A. T., B. M. Brown, and M. P. Thorne. 1989. Coping with some really rotten problems in automatic music recognition. *Microprocessing and Microprogramming.* 27: 547–50.

Clarke, A. T., B. M. Brown, and M. P. Thorne. 1990. Problems to be faced by developers of computer based automatic music recognisers. *Proceedings of the International Computer Music Conference.*

Clarke, A. T., B. M. Brown, and M. P. Thorne. 1993. Recognising musical text. *Proceedings of the SPIE.* 2064: 222–33.

Cover, T. M. and P. E. Hart. 1967. Nearest neighbor classification. *IEEE Transaction of Information Theory.* 13:21–7.

Cover, T. M. and J. M. Van Campenhout. 1977. On the possible orderings in the measurement selection problem. *Transactions on Systems, Man, and Cybernetics.* 7(9): 657–61.

Davis, L. 1987. *Genetic algorithms and simulated annealing.* London: Pitman.

Davis, L. 1991. *Handbook of genetic algorithms.* New York: Van Norstrand Reinhold.

Devijver, P. A. and J. Kittler. 1980. On the edited nearest neighbor rule. *Proceedings of the Fifth International Conference on Pattern Recognition.* 72–80.

Di Riso, D. 1992. *Lettura automatica di partiture musicali.* Masters thesis. Università di Salerno, Italy.

Diener, G. R. 1990. *Modeling music notation: A three-dimensional approach.* Ph.D. Thesis, Stanford University.

Distasi, R., et al. 1993. Automatic system for reading scores. *Proceedings of the 8th Scandinavian Conference on Image Analysis.* 1307–10.

Dudani, S. A. 1976. The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics..* 6: 325–7.

Dudani, S. A., K. J. Breeding, and R. B. McGhee. 1977. Aircraft identification by moment invariants. *IEEE Transactions on Computers.* 26: 39–46.

Fahmy, H. *A graph-grammar approach to high-level music recognition.* Technical report. Queen's University. Dept. of Computing & Information Science: 91–318.

Fahmy, H. and D. Blostein. 1991. A graph grammar for high-level recognition of music notation. *Proceedings of First International Conference on Document Analysis.* 1: 70–8.

Fahmy, H. and D. Blostein. 1992. Graph grammar processing of uncertain data. *Proceedings of International Workshop on Structural and Syntactic Pattern Recognition.* 373–82.

Fahmy, H. and D. Blostein. 1993. A graph grammar programming style for recognition of music notation. *Machine Vision and Applications.* 6: 83–99.

Fahmy, H. and D. Blostein. 1994. A graph-rewriting approach to discrete relaxation: Application to music recognition. *Proceedings of the SPIE.* 2181: 291–302.

Feng Yin; Gao Qingshi; Zhang Xiang. 1989. Principle on designing the music reading system. (in Chinese). *Mini-Micro Systems.* 10(12): 1–10.

Fischer, F. P. and E. A. Patrick. 1970. A preprocessing algorithm for nearest neighbor decision rule. *Proceedings of National Electronics Conference.* 481–5.

Fischer, K. N. 1978. *Computer recognition of engraved music.* M.S. Thesis. University of Tennessee, Knoxville.

Fletcher, L. A. and R. Kasturi. 1988. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 10(6): 910–8.

Foroutan, I. and J. Sklansky. 1987. Feature selection for automatic classification of non-Gaussian data. *IEEE Transactions on Systems, Man, and Cybernetics.* 17(2): 187–98.

Fujinaga, I. 1988. *Optical music recognition using projections.* M.A. Thesis. McGill University.

Fujinaga, I. 1992. An optical music recognition system that learns. *Enabling Technologies for High-Bandwidth Applications.* Jacek Maitan, ed. *Proceedings of SPIE.* 1785. 210–7.

Fujinaga, I., B. Alphonce, and B. Pennycook. 1992. Interactive optical music recognition. *Proceedings of the International Computer Music Conference.* 117–20.

Fujinaga, I., B. Alphonce, G. Diener, and B. Pennycook. 1992. Optical music recognition on NeXT workstation. Paper presented at the *Second International Conference on Music Perception and Cognition.*

Fujinaga, I., B. Alphonce, B. Pennycook, and K. Hogan. 1991. Optical music recognition: Progress report. *Proceedings of the International Computer Music Conference.* 66–73.

Fujinaga, I., B. Pennycook, and B. Alphonce. 1991. The optical music recognition project. *Computers in Music Research.* 3: 139–42.

Fujinaga, I., B. Alphonce, and B. Pennycook. 1989. Issues in the design of an optical music recognition system. *Proceedings of the International Computer Music Conference.* 113–6.

Fujinaga, I., B. Alphonce, B. Pennycook, and N. Boisvert. 1989. Optical recognition of music notation by computer. *Computers in Music Research.* 1: 161–4.

Fujinaga, I., B. Pennycook, and B. Alphonce. 1989. Computer recognition of musical notation. *Proceedings of the First International Conference on Music Perception and Cognition.* 87–90.

Fukushima, K. and S. Miyake. 1982. Neocognitron: A new algorithm for pattern recognition tolerant of deformation and shifts in position. *Pattern Recognition* 15(6): 455–69.

Gates, G. W. 1972. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory.* 18: 431–3.

Geggus, K. M. and E. C. Botha. 1993. A model-based approach to sheet music recognition. *Elektron. (*10)1: 25–9.

Gilmore, J. F. and W. W. Boyd. 1981. Building and bridge classification by invariant moments. *Proceedings of SPIE.* 292: 256–63.

Glass, S. 1989. *Optical music recognition.* Undergraduate project report. University of Canterbury, New Zealand.

Glassner, A. 1990. *Graphics Gems.* Boston: Academic Press.

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization, and machine leaning.* Reading: Addison-Wesley.

Goolsby, T. W. 1994. Eye movement in music reading: Effects of reading ability, notational complexity, and encounters. *Music Perception.* 12(1): 77–96.

Goolsby, T. W. 1994. Profiles of processing: Eye movements during sightreading. *Music Perception.* 12(1): 97–123.

Gowda, K. C. and G. Krishna. 1979. The condensed nearest neighbor rule using the concept of mutual nearest neighborhood. *IEEE Transactions on Information Theory.* 25: 480–90.

Hamamoto, Y., S. Uchimura, Y. Matsuura, T. Kanaoka, and S. Tomita. 1990. Evaluation of the branch and bound algorithm for feature selection. *Pattern Recognition Letters.* 11: 453–6.

Hart, P. E. 1968. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory.* 14: 515–6.

Hearn, D. and M. P. Baker. *Computer Graphics.* Englewood Cliffs: Prentice-Hall. 1986.

Hermann, G. T. ed. 1979. *Image reconstruction from projections.* Berlin: Springer-Verlag.

Hewlett, W. B. and E. Selfridge-Field. 1990. *Computing in Musicology:* A Directory of Research. Menlo Park, CA: Center for Computer Assisted Research in Humanities.

Holland, J. H. 1975. *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press. (Reprinted by Cambridge: MIT Press. 1992.)

Horner, A., J. Beauchamp, and L. Haken. 1992. Wavetable and FM matching synthesis of musical instrument tones. *Proceedings of the 1992 International Computer Music Conference.* 18–21.

Horner, A., J. Beauchamp, and N. Packard. 1993. Timbre breeding. *Proceedings of the 1993 International Computer Music Conference.* 396–8.

Horner, A. and D. E. Glodberg. 1991. Genetic algorithms and computer-assisted music composition. *Proceedings of the 1991 International Computer Music Conference.* 479–82.

Jain, A. K. and S. K. Bhattacharjee. 1992. Address block location on envelopes using Gabor filters. *Pattern Recognition.* 25(12): 1459–77.

Jarris, J. F. 1977. The line drawing editor: Schematic diagram editing using pattern recognition techniques. *Computer Graphics and Image Processing.* 6: 452–84.

Itagaki, T., S. Hashimoto, M. Isogai, and S. Ohteru, 1990. Automatic recognition on some different types of musical notation. *Proceedings of the International Association for Pattern Recognition Workshop on Syntactic and Structural Pattern Recognition.* 488.

Itagaki, T., S. Hashimoto, M. Isogai, and S. Ohteru. 1992. Automatic recognition of several types of musical notation. In *Structured Document Image Analysis,* ed. H. S. Baird, H. Bunke, and K. Yamamoto. Berlin: Springer. 467–76.

Kamesawara, C. W. and P. S. U. Rao. 1978. On the fingerprint pattern recognition. *Pattern Recognition.* 10: 15–8.

Kassler, M. 1970. An essay toward specification of a music-reading machine. In *Musicology and the computer,* ed. B. S. Brook. New York: The City University of New York Press. 151–75.

Kassler, M. 1972. Optical character recognition of printed music: A review of two dissertations. *Perspectives of New Music.* 11: 250–4.

Katayose, H., T. Fukuoka, K. Takami, and S. Inokuchi. 1990. Expression extraction in virtuoso music performances. *Proceedings of the Tenth International Conference on Pattern Recognition.* 780–4.

Katayose, H., H. Kato, M. Imai, and S. Inokuchi. 1989. An approach to an artificial music expert. *Proceedings of the International Computer Music Conference.* 139–46.

Katayose, H. and S. Inokuchi. 1989. The kansei music system. *Computer Music Journal.* 13(4): 72–7.

Kato, H. and S. Inokuchi. 1988. Automatic recognition of printed piano music based on bar unit processing (in Japanese). *Transactions of I. E. C. E.* J71–D. 5: 894–901.

Kato, H. and S. Inokuchi. 1990. The recognition system for printed piano music using musical knowledge and constraints. *Proceedings of the International Association for Pattern Recognition Workshop on Syntactic and Structural Pattern Recognition.* 231–48.

Kato, H. and S. Inokuchi. 1991. A recognition system for printed piano music using musical knowledge and constraints. In *Structured Document Image Analysis,* ed. H. S. Baird, H. Bunke, and K. Yamamoto. Berlin: Springer. 434–55.

Kim, W. J., M. J. Chung, and Z. Bien. 1987. Recognition system for a printed music score. *Proceedings of TENCON 87: 1987 IEEE Region 10 Conference 'Computers and Communications Technology Toward 2000.'* 1380: 573–7.

Kittler, J. 1978. Feature set search algorithms. In *Pattern Recognition and Signal Processing,* ed. C. H. Chen. The Netherlands: Sijthoff and Norddhoff.

Kobayakawa, T. 1993. Auto music score recognition system. *Proceedings SPIE: Character Recognition Technologies.* D. P. D'Amato, ed. 1906.

Kumar, P., D. Bhatnagar, and P. S. U. Rao. 1991. An improved parallel algorithm for thinning digital patterns. *Pattern Recognition Letters.* 12: 543–55.

Lee, M. Woo and J. Soo Choi. 1985. The recognition of printed music score and performance using computer vision system (in Korean and English translation). *Journal of the Korean Institute of Electronic Engineers.* 22(5): 429–35.

Lee, S. and J. Shin. 1994. Recognition of music scores using neural networks. *Journal of the Korea Information Science Society.* 21(7): 1358–66.

Leplumey, I., J. Camillerapp, and G. Lorette. 1993. A robust detector for music staves. *Proceedings o the International Conference on Document Analysis and Recognition.* 902–5.

Leplumey, I. and J. Camillerapp. 1991a. Comparison of region labelling for musical scores. *Proceedings of First International Conference on Document Analysis.* 674–82.

Leplumey, I. and J. Camillerapp. 1991b. Coopération entre la segmentation des régions blanches et des régions noires pour l'analyse de partitions musicales. *AFCET, 8e Congress Reconnaissance des Formes et Intelligence Artificielle.* 3: 1045–52.

Loftsgaarden, D. O. and C. P. Quesenbery. 1965. A nonparametric estimate of multivariate density function. The Annals of Mathematical Statistics. 36: 1049–51.

Maenaka, K. and Y. Tadokoro. 1983. Recognition of music using the special image-input-device enabling to scan the staff of music as the supporting system for the blind (in Japanese). *PRL83–60.* 37–45.

Mahoney, J. V. 1982. *Automatic analysis of musical score images.* B.S. Thesis. Department of Computer Science and Engineering, MIT.

Markuzon, N. 1994. Handwritten digit recognition using fuzzy ARTMAP network. *World Congress on Neural Networks-San Diego*: 4.

Martin, N. G. 1987. *Towards computer recognition of the printed musical score.* B.Sc. project report. Thames Polytechnic, London.

Martin, P. and C. Bellissant. 1991a. Neural networks at different levels of musical score image analysis system. *Proceedings of 7th Scandinavian Conference on Image Analysis.* 1102–9.

Martin, P. and C. Bellissant. 1991b. Low-level analysis of music drawing images. *Proceedings off the International Conference on Document Analysis and Recognition.* 417–25.

Martin, P. 1989. Reconnaissance de partitions musicales et réseaux de neurones: une étude. Actes 7 iéme Congrés AFCET de Reconnaissance des Formes et Intelligence Artificielle: 217–26.

Matsushima, T. 1985. Automated high speed recognition of printed music (Wabot-2 Vision System). *Proceedings of the 1985 International Conference on Advances in Robotics.* 477–82.

Matsushima, T. 1988. Automatic printed-music-to-braille translation system. *Journal of Information Processing.* 11(4): 249–57.

Matsushima, T. 1992. Computerized Japanese traditional music processing system. *Proceedings of the International Computer Music Conference.* 121–4.

Matsushima, T., T. Harada, I. Sonomoto, K. Kanamori, A. Uesugi, Y. Nimura, S. Hashimoto, and S. Ohteru. 1985. Automated recognition system for musical score: The vision system of WABOT-2. *Bulletin of Science and Engineering Research Laboratory.* Waseda University. 112: 25–52.

McGee, W. F. 1994. MusicReader: An interactive optical music recognition system. *Computing in Musicology.* 9: 146–51.

McGee, W. F. and P. Merkley. 1991. The optical scanning of medieval music. *Computers and the Humanities.* 25(1): 47–53.

McLean, Graeme I. 1991. *Music recognition.* B.Sc. Thesis. Heriot-Watt University, Riccarton, Edinburgh.

Mitra, P. 1992. Answering Gabriel neighbour queries. *Pattern Recognition Letters.* 13: 557–60.

Miyao, H., T. Ejima, M. Miyahara, and K. Kotani. 1992. Symbol recognition for printed piano scores based on the musical knowledge (in Japanese). *Transactions of the Institute of Electronics, Information and Communication Engineers D–II,* J75D–II(11): 1848–55.

Miyao, H., T. Ejima, M. Miyahara, K. Kotani, and M. Miyahara. 1990. Recognition for printed piano scores (in Japanese). *NLC90–34, PRU90–74.* 39–46.

Modayur, B. R., V. Ramesh, R. M. Haralick, and L. G. Shapiro. 1992. MUSER—a prototype musical recognition system using mathematical morphology. *Intelligent Systems Laboratory, EE Dept, FT–10.* University of Washington.

Modayur, B. R., V. Ramesh, R. M. Haralick, and L. G. Shapiro. 1994. MUSER: a prototype musical recognition system using mathematical morphology. *Machine Vision and Applications.* 6(2–3):140–50.

Musitek Midiscan. 1994. *Keyboard.* 20(3): 136.

Nagy, G. 1989. Document analysis and optical character recognition. *Proceedings of the 5th International Conference on Image Analysis and Processing.* 511–29.

Nakamura, Y., M. Shindo, and S. Inokuchi. 1978. Input method of [musical] note and realization of folk music data-base (in Japanese). *Institute of Electronics and Communications Engineers of Japan (IECE)* TG PRL78–73: 41–50.

Narendra, P. M. and K. Fukunaga. 1977. A branch and bound algorithm for feature subset selection. *IEEE Transactions of Computers.* 26: 917–22.

Nelson, G. and T. R. Penney, 1973. Pattern recognition in musical score - Project no. M88. *Computers and the Humanities.* 8: 50–1.

Ng, K.-C. and R. D. Boyle. 1992. Segmentation of music primitives. *Proceedings of the British Machine Vision Conference.* 472–80.

Ohteru S. 1987. Automatic recognition of music score (in Japanese). *Bit* (special issue on Computer and Music). 92–100.

Ohteru S. 1988. Data entry and automatic recognition of music score (in Japanese). *Journal of the Information Processing Society of Japan.* 29(6): 586–92.

Ohteru, S., et al. 1984. A multi processor system for high speed recognition of printed music (in Japanese). *National Convention Records of I. E. C. E.* [n.p.].

Ohteru, S. and T. Matsushima. 1985. Automatic recognition of printed music (in Japanese). *Japan Acoustics Society Journal.* 41(6).

Onoe, M., M. Ishizuka, and K. Tsuboi. 1979. Experiment on automatic music reading (in Japanese). *Proceedings of 20th IPSJ National Conference.* 6F–5.

Ostenstad, B. 1988. Oppdeling av abjektene i et digitalt notebilde i klassifiserbare enheter (in Norwegian). *Institute of Informatics.*

Parthasarathy, G. and B. N. Chatterji. 1990. A class of new KNN methods for low sample problems. *IEEE Transactions on Systems, Man, and Cybernetics.* 20(3): 715–8.

Pennycook, B. 1990. Towards advanced optical music recognition. *Advanced Imaging.* 54–7.

Penrod, C. S. and T. J. Wagner. 1977. Another look at the edited nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics.* 7: 92–4.

Perrotti, F. A. and R. A. Lotufo. 1993. Pre-processamento, Exctracao de Atributos e Primeiro Nivel de Classiccao para un Sistema de Reconhecimento Otico de Simbolos Musicais. [Preprocessing, Feature Extraction, and First Classification Level for an Optical Recognition System.] in *VI Brazilian Symposium in Computer Graphics and Image Processing,* SINGRAPI. [n.p.]

Prerau, D. S. 1970. *Computer pattern recognition of standard engraved music notation.* Ph.D. Dissertation. MIT.

Prerau, D. S. 1971. Computer pattern recognition of printed music. *Proceedings of the Fall Joint Computer Conference.* 153–62.

Prerau, D. S. 1975. Do-Re-Mi: A program that recognizes music notation. *Computer and the Humanities.* 9(1): 25–9.

Prokop, R. J. and A. P. Reeves. 1992. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphical Models and Image Processing.* 54(5): 438–60.

Pruslin, D. 1966. *Automatic recognition of sheet music.* Sc.D. Dissertation. MIT.

Ramasubramanian, V. and K. K. Paliwal. 1992 An efficient approximation-elimination algorithm for fast nearest neighbour search based on a spherical distance coordinate formulation. *Pattern Recognition Letters.* 13: 471–80.

Randriamahefa, R., J. P. Cocquerez, C. Fluhr, F. Pépin, and S. Philipp. 1993. Printed Music Recognition. *Proceedings of the International Conference on Document Analysis and Recognition.* 898–901.

Read, G. 1979. *Music notation.* 2d. ed. Boston: Allyn and Bacon.

Roach, J. W. and J. E. Tatum. 1988. Using domain knowledge in low-level visual processing to interpret handwritten music: An experiment. *Pattern Recognition.* 21(1): 333–44.

Roads, C. 1986. The Tsukuba musical robot. *Computer Music Journal.* 10(2): 39–43.

Roth, M. 1992. *OMR-optical music recognition.* Diploma thesis. Swiss Federal Institute of Technology.

Roth, M. 1994. An approach to recognition of printed music. *Technical Report 210.* Department of Computer Science, Swiss Federal Institute of Technology.

Ruttenberg A. 1990. *Optical music reading.* M.S. Thesis. MIT Media Laboratory.

Sawada, H., T. Matsushima, T. Itakagi, and S. Ohteru. 1990. A practical bilateral translation system between printed music and braille. *Proceedings of 6th International Workshop on Computer Applications for the Visually Handicapped.*

Selfridge-Field, E. 1994. Optical recognition of musical notation: A survey of current work. *Computing in Musicology.* 9: 109–45.

Sicard, E. 1992. An efficient method for the recognition of printed music. *Proceedings of 11th International Conference on Pattern Recognition (IAPR).* 573–6.

Siedlecki, W. and J. Sklansky. 1989. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters.* 10: 335–47.

Smith, F. W. and M. H. Wright. 1971. Automatic ship photo interpretation by the method of moments. *IEEE Transactions on Computers.* 20(9): 1089–95.

Sonomoto, I., T. Harada, T. Matsushima, K. Kanamori, M. Konuma, A. Uesugi, Y. Nimura, S. Hashimoto, and S. Ohteru. 1985. Automated recognition system of printed music for playing keyboards (in Japanese). *Acoustical Society of Japan.* TG MA84–22: 17–22.

Takala, T., J, Hahn, L. Gritz, J. Geigel, and J. W. Lee. 1993. Using physically-based models and genetic algorithms for functional composition of sound signals, synchronized to animated motion. *Proceedings of the 1993 International Computer Music Conference.* 180–5.

Taxt, T., P. J. Flynn, and A. K. Jain. 1989. Segmentation of document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 11(12): 1322–9.

Teague, M. R. 1980. Image analysis via the general theory of moments. *Journal of Optical Society of America.* 70: 920–30.

Thorud, E. 1988. *Analyse av notebilder.* (in Norwegian). Institute of Informatics.

Tojo, A. and H. Aoyama. 1982. Automatic recognition of music score. *Proceedings of 6th International Conference on Pattern Recognition.* 1223.

Tomek, I. 1976a. Two modifications of CNN. *IEEE Transactions of Systems, Man, and Cybernetics.* 6: 769–72.

Tomek, I. 1976b. An experiment with edited nearest-neighbor rule. *IEEE Transactions of Systems, Man, and Cybernetics.* 6: 448–52.

Tonnesland, S. 1986. *SYMFONI: System for note coding* (in Norwegian). Institute of Informatics.

Vuori, J. and V. Välimäki. 1993. Parameter estimation of non-linear physical models by simulated evolution-application to the flute model. *Proceedings of the 1993 International Computer Music Conference.* 402–4.

Wagner, T. J. 1973. Convergence of the edited nearest neighbor. *IEEE Transactions on Information Theory.* 9: 696–9.

Wang, Q. R. 1987. A flexible tree design in an edit-partition scheme. *Pattern Recognition Letters.* 5: 261–5.

Weiss, S. M. 1991. Small sample error rate estimation for k-NN classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 13(3): 285–9.

Wilson, D. L. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions of Systems, Man, and Cybernetics.* 2: 408–20.

Wilk, R. 1995. *Converting graphic musical data to a machine playable form.* M.Sc. Thesis. McGill University.

Wittlich, G. E. 1973. Project SCORE. *Computational Musicology Newsletter.* 1(1): 6.

Wittlich, G. E. 1974. Non-physics measurements on the PEPR System: Seismograms and music scores. *Report to the Oxford Conference on Computer Scanning.* Oxford Nuclear Physics Laboratory: 487–9.

Wolman, J., J. Choi, S. Asgharzadeh, and J. Kahana. 1992. Recognition of handwritten music notation. *Proceedings of the International Computer Music Conference.* 125–7.

Yadid O., E. Brutman, L. Dvir, M. Gerner, and U. Shimony. 1992. RAMIT: Neural network for recognition of musical notes. *Proceedings of the International Computer Music Conference.* 128–31.

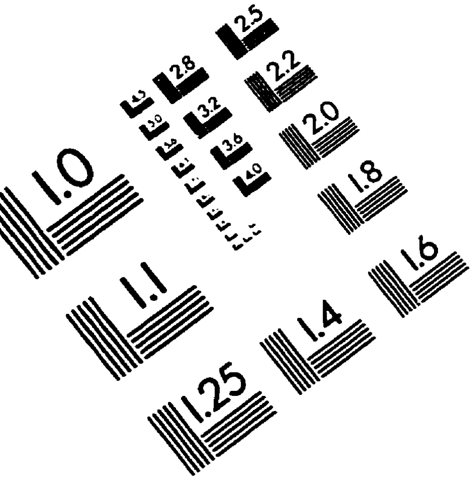
Vidal, E. 1986. An algorithm for finding nearest neighbours in (approximately) constant average complexity. *Pattern Recognition Letters.* 4:145–57.

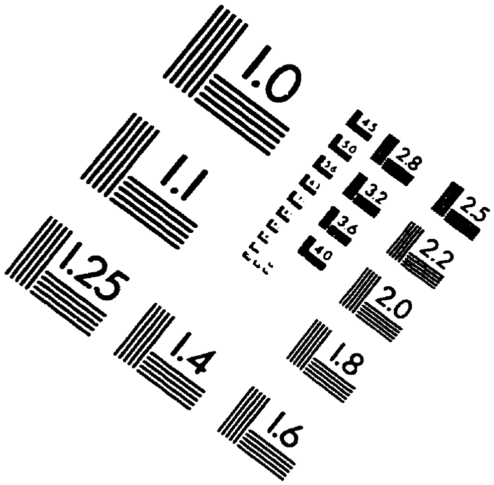
Yin, F., G. Qingshi, and Z. Xiang. 1989. Principle on designing the music reading system (in Chinese). *Mini-Micro Systems.* 10(12): 1–10.

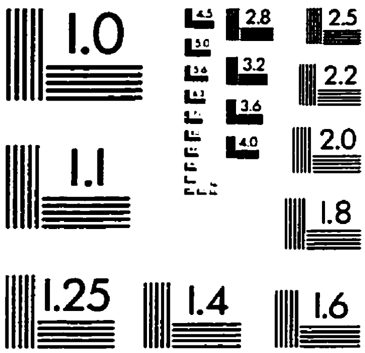
Yip-San Wong and A. Choi. 1994. A two-level model-based object recognition technique. *International Symposium on Speech, Image Processing and Neural Networks Proceedings.* 1: 319–22.

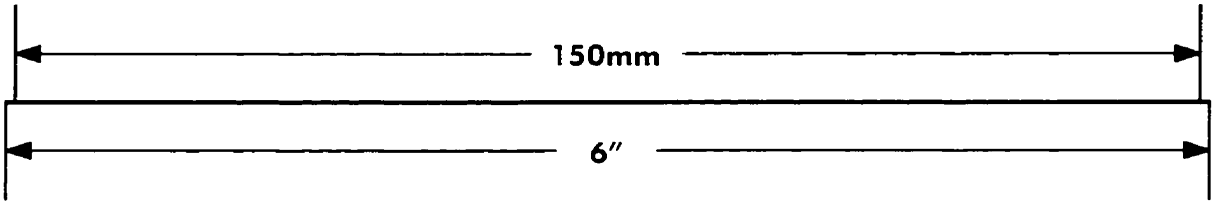
Zhang, T. Y. and C. Y. Suen. 1984. A fast a parallel algorithm for thinning digital patterns. *Communications ACM.* 27(3): 236–9.

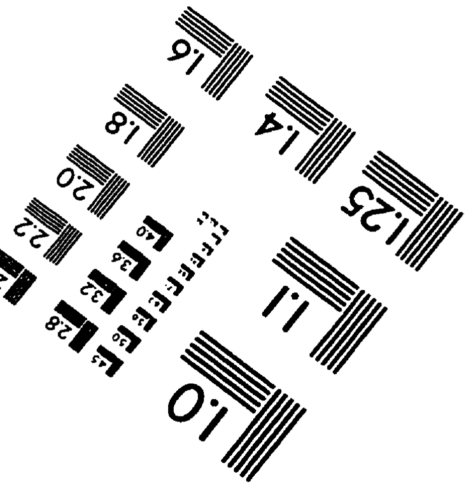
IMAGE EVALUATION TEST TARGET (QA-3)

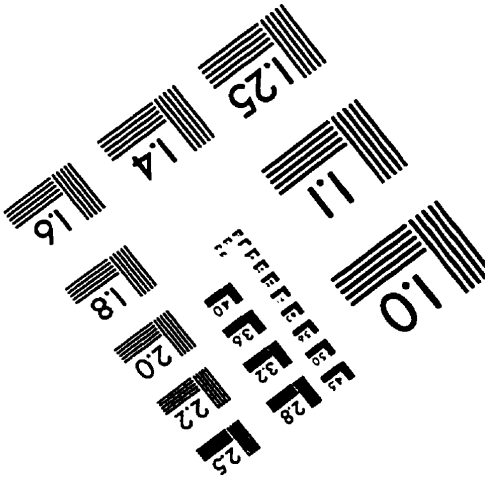












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989