University of Canterbury

Department of Computer Science

Optical Music Recognition

Progress Report 1

David Bainbridge

Supervisor: Dr. Tim Bell

March 4, 1994

Contents

1 Introduction

The purpose of writing this report is to record and comment on the work done over the last year. The report will also summarise my main insights into the problem and outline future work.

There were three main areas of work: staff line detection; preparation for musical feature recognition and the recognition process itself. For each area, a program was developed that allowed the topic to me examined and investigated. Though work was carried out concurrently, they will be reported sequentially, in the order just given. The respective programs names were **xstaff**, **xsep**, and **xmsr**.

Having the right development environment to study Optical Music Recognition (OMR) was seen as an important facet to work. Time throughout the year was devoted to this. In particular, time was invested learning C++ (an object oriented programming language), Interviews (a toolkit for X Windows), Csound (an audio playing package) and Midi, as well as extending knowledge of Emacs, Motif and the Unix operating system.

2 Staff Line Detection

Detecting the staff lines is a crucial step in OMR. It is only after the gap between the staff lines has been found that the size of musical features, such as note heads and accidentals, can be determined. It is, therefore, a step that must be carried out early on in the OMR process.

The definition of the problem is succinct. You are solely given a bitmap (no additional information) and must find the gap between the staff lines. The only operation you may carry out is to ask if a co-ordinate in the bitmap is set or clear. Most of the work carried out in this area relies upon horizontal projection algorithms to locate staff lines ([Gla89] and [KI90] are the only known exceptions). For a description of projection methods see [BHY92]. Common to all these algorithms is the problem of skew. Surely some vertical scan based approach would be better suited to measuring the staff gap and would not have to circumvent the problem of skew since this would be naturally incorporated.

The **xstaff** program was developed to study a variety of vertical scan line algorithms. The structure of the application was modular, called *stages*, so the user could specify the number of major vertical scan lines, the location strategy for the major scan lines, the number of minor scan lines to cluster around each major scan line and which staff gap identification algorithm to use. Processor timing information was also displayed. Using this design a 'pick and mix' approach could be adopted to analyse the suitability of algorithms by discovering what was required to make a proposed staff line identification algorithm work reliably and how 'expensive' this was.

A *shrink canvas* was developed to display a crude approximation of the desired image on the screen, with the added functionality that a selected part of the image could be shown actual size by pressing the middle mouse button,

The three staff identification algorithms devised were: frequency count, Fourier transform and correlation.

The *frequency count* algorithm worked by tabulating the number of vertical runs of white found in the scan line. The highest frequency was chosen as the staff gap. The *Fourier transform* algorithm chose the largest frequency component as the frequency staff lines occurred. The measure was based on the magnitude of the complex frequency components. From the value chosen the staff gap could be computed. Finally the *correlation* based method checked a set of ideal staff line segments of increasing spacing against the given image. The best match was taken to be the staff gap.

The strategy choices for major scan line location were *regular* and *home in*. *Regular* spacing simply divided the width of the image into equal region. The *home in* method first used a vertical projection of the image to decide where the staff systems started and stopped and then divided up the located region into equal sections. Variations built on top of these two approaches were *intelligent* and *average*. *Intelligent* moved a chosen scan line to the left or the right to the lowest local minimum turning point of the vertical projection. *Average* moved the chosen scan line alternatively to the left and right, gradually moving further away from the original choice, until a scan line was found that had a vertical projection entry that was less than or equal to the average of the vertical projection.

Given the raw power of the Fourier transform, it was surprising to learn that it was the least successful approach. A substantial proportion of time was devoted to closely examining this algorithm. By selecting a higher verbose diagnostic level from the user interface, **xstaff** would form a Unix pipe to **gnuplot**, through which the frequency components of the Fourier transform were graphically shown, incrementally adding frequency components in order of magnitude, largest to smallest.

From this it was learnt that the staff gap frequency was always near the top of the order, however, greater or smaller harmonic frequencies of this value could also occur first. Since it was possible for a greater frequency or a smaller frequency to occur before the sought frequency, it was impossible to tailor the algorithm to track through the harmonic values until the last one was found. Using **gnuplot** it was also discovered that a vertical scan line that passed through a long segment of black, say a note beam, could cause large low frequency components - obliterating the delectate harmonics of the staff lines.

Attempts to develop algorithms to counter these problems, such as a low pass filter, were all fundamentally flawed due to scale. That is to say, because staff line identification is the first step in OMR, there is no prior information relating to size. Hence what constitutes a low frequency?

I am sure the information indicating the staff gap was there amongst the frequency coefficients and could be reliably exposed by 'massaging' the data, but I was unable to discover a way to reveal this. Perhaps it was a case of too much information obscuring the desired results.

The correlation algorithm proved to be the most expensive algorithm. Work during my honours project showed a series of improvements that could be made, as well as some undesirable properties of correlation for monochrome pattern matching that could be changed with no added expense. These alterations would certainly help performance. It should also be remembered that the fast Fourier transform, which lies at the hard of mathematical correlation, is also available in hardware.

The problem of scale also had ramifications on the correlation algorithm. The algorithm had to test example templates from a one pixel alterate black and white template, right through to a single staff

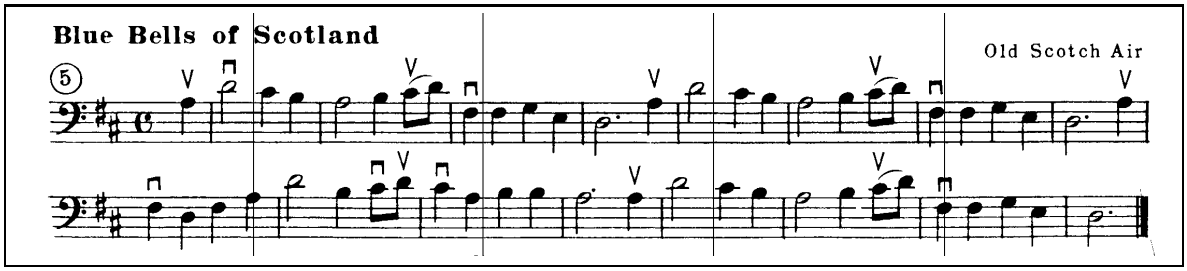


Figure 1: Frequency count.

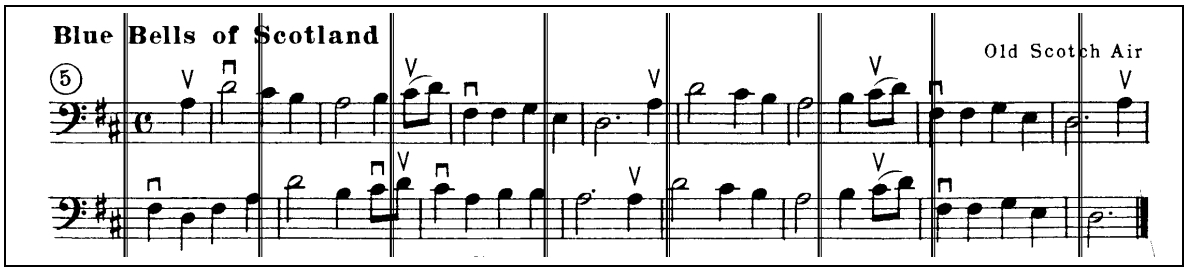


Figure 2: Fourier transform.

that fitted the entire height of the image. At that point in the process there is no way of knowing if you have an entire page of staffs, or just one.

The frequency counting algorithm proved almost as reliable as the correlation method, but only a fraction of the processing time.

2.1 Results

Figures 1 - 3 show the location of the scan lines chosen. Table 1 and Table 2 tabulate input and output respectively. By visual inspection using the utility program **xv**, the staff gap was measured to be 19 pixels and the staff line thickness 3 pixels.

The possible avenues that resulted from this piece of work were not pursued. The effort in locating good major vertical scan lines and taking a few minor scan lines about them, if viewed from a different perspective could be seen as taking small horizontal projects. Even without this observation, the extra

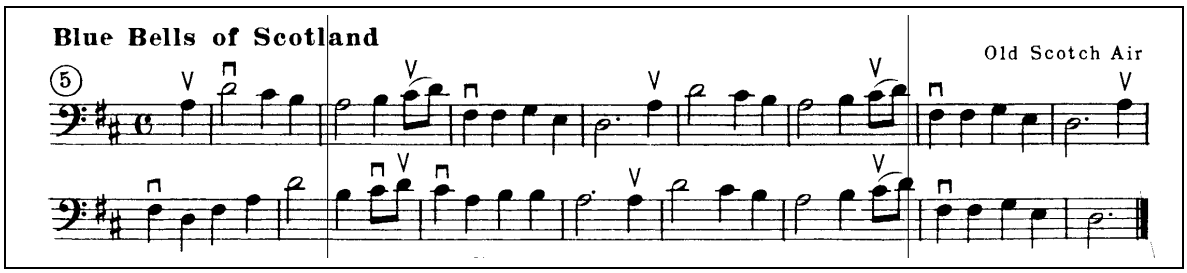


Figure 3: Correlation.

Algorithm	Major Scan Lines	Minor Scan Lines	Location Strategy
Frequency Count	4	1	Intelligent Regular
Fourier Transform	8	3	Average Home in
Correlation	2	1	Average Home in

Table 1: Input settings for **xstaff**.

Algorithm	Processor Time	Staff Gap	Staff Line Thickness
Frequency Count	20 msec	19 pixels	3.5 pixels
Fourier Transform	140 msec	18.46 pixels	1 pixel
Correlation	26012 msec	19 pixels	5 pixels

Table 2: Output information from **xstaff**.

work necessary to allow algorithms to work reliably certainly detracted from the potential elegance promised by the original concept. It was also around this time, that I started to realise that simply finding the staff lines was not the whole picture.

3 Preparation before Recognition

The purpose of **xsep** was to provide a framework to investigate the early stages of OMR. The work aimed to compare and contrast different algorithms for finding, straightening and removing staff lines. Work on **xstaff** directed me to the conclusion that simply detecting staff lines was an insufficient step. A piece of music has much more structure than staff lines: staff systems, even pages of music should be catered for. A page of music selected from a work cannot be interpreted correctly in isolation. For example, unless the time signature changes later on in the work, it is not repeated after the first line of each instrument. One could argue that a smart person/computer could infer the time signature from the consistent note durations in each measure, but what visible difference is there between a piece the $\frac{3}{4}$ and $\frac{6}{8}$? Ties and slurs crossing a page would also be impossible to interpret correctly in every situation, also repeat marks, 'to code' etc.

One idea would be to concatenate the pages of music together to make one long piece of music. Given the memory requirements of one page of A4 music scanned at 300 dpi (about one megabyte using the standard eight bits to one byte data structure), this is clearly impractical. A different complication that was highlighted by the **xstaff** work, was that a page of music can intentionally have different sized staffs. This is common, for example, in a piano accompaniment for a solo instrument, where the melody line is provided above each piano line, written on a smaller staff. An example of such a score is shown in Figure 4.

With these issues in mind, the **xsep** program was designed. The 'pick and mix' style user interface designed for **xstaff** had proved useful, hence was utilised in **xsep**. The five *stage* algorithms were: staff system location; staff area location; correction for skew (optional), staff line removal and correction for

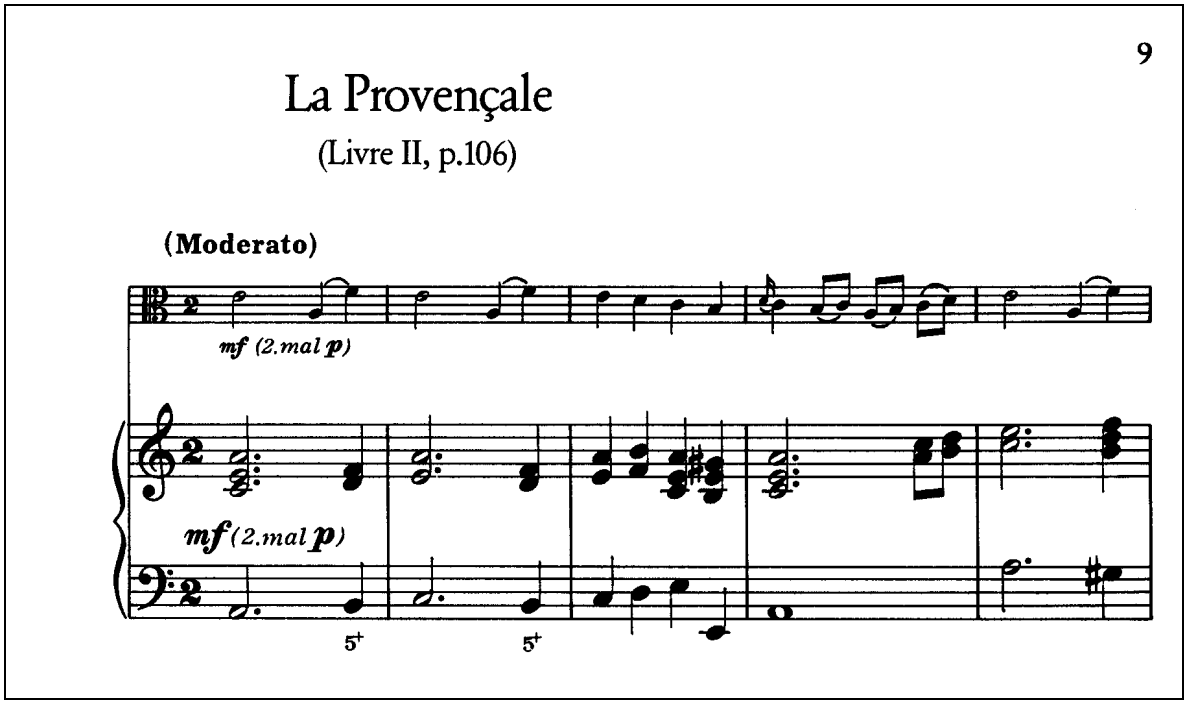


Figure 4: Excerpt from a piano accompianment for a violin.

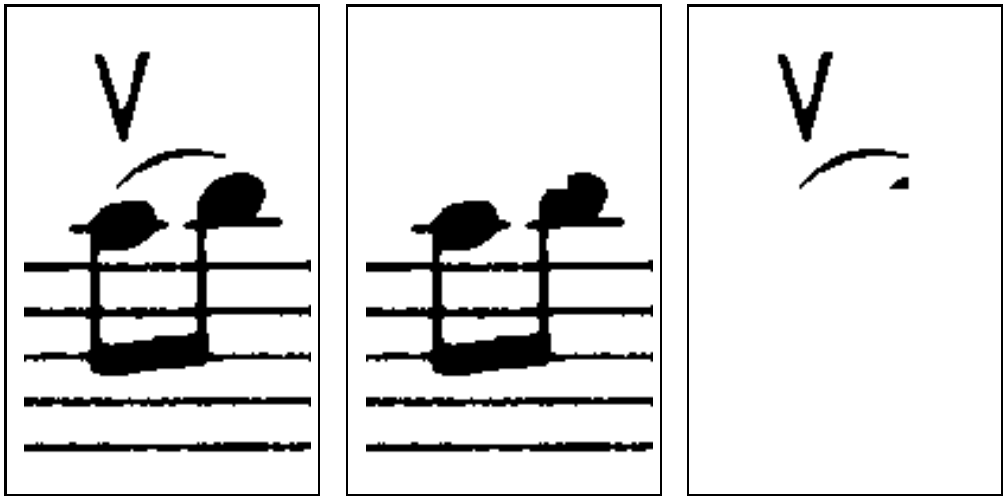
deformation (optional). Each stage had a choice of different algorithms to use. Similarly to **xstaff**, processor timing information was displayed. The *shrink canvas* that was originally written for **xstaff** was improved and extended for **xstaff**.

xsep was designed to be run on one or more files, where it was assumed that the order of the files represented the sequential order of a piece of music. *Next* and *Previous* buttons exposed this functionality.

For the stages of OMR covered by **xsep**, each page could in fact be (and was) treated as a separate isolated image. An improvement would be to make use of the staff information located in the first page: staff height, staff gap and staff line thickness, to locate staffs in later pages. Contrast this with the classified problem that **xstaff** set out to solve. Namely an image with no other information known. The algorithm would only fall back on starting from scratch if nothing of the specified size was found.

3.1 Staff System Location

Staff system location was based on isolating every graphical shape in the image and then deciding if it was a staff system. The set of criteria that was used to decide if a given shape formed a staff system, was not accurate. More specific conditions were required by the staff area location algorithm (discussed in Section 3.2) which could result in an object being rejected as a staff system and therefore added to the non staff system list. The work was structured in this fashion so different isolation algorithms could be evaluated. All isolated shapes were stored in lists with a well formed order (x then y) defined over them.



(a) Excerpt of Bitmap. (b) Clipped object. (c) Clipped fragments turn up elsewhere.

Figure 5: An example of clipping caused by contour tracing.

The crude requirement for a staff system was that the isolated object should be larger than $\frac{3}{4}$ inch wide and more that 13 pixels high. The justification for these measurements was that, when considering a piece of music, the smallest conceivable staff system would be a coda which must include a clef and a key signature before any notes. A lower limit of $\frac{3}{4}$ inch in width seemed a reasonable requirement. No example collected so far has proved this estimate too optimistic. The program had provision for expressing the dpi scan resolution of the work, so $\frac{3}{4}$ inch was well defined.

The y threshold of 13 pixels was based on the observation that the minimal score a computer could be expected to recognise would be a staff consisting of five one pixel thick staff lines at a spacing of 2 pixels apart. This restriction is no longer favoured since it excludes non-pitch percussion instruments, which have only one staff line. This restriction is not necessary if text is recognised first and removed from the image. This subject is discussed again in Section 5.

The two main forms of object isolation were *flood fill* and *counter tracing*. It was my expectation that contour tracing would prove significantly faster than a flood fill approach, but experimental data showed only a marginal improvement. Contour tracing used the extents of the outlined shape for extraction, but this could result in the 'clipping' of nearby objects. This effect is shown by Figure 5. It would be possible to correct this defect, however it was decided that the contour trace was not sufficiently faster than the more elegant and clean flood fill approach to merit the work.

Two variants on the flood fill algorithm were developed: two bit connectivity and one byte connectivity. The basic algorithm 'recursively' visits all the neighbours of the current pixel that are set and have not previously been visited, until there are no pixels left [FJDF90]. The two pixel adaptation considered neighbours up to two pixels away in the horizontal direction, to compensate for potentially broken staff

lines. Two pixels proved insufficient. The idea could be extended to n pixels, but by choosing $n = 8$, byte operations could be utilised instead of repetative bit extraction, resulting in a significantly faster algorithm. The theoretical speed up should have been x8 but only around x3 was observed, possibly limited by the initialisation code required, regardless of which flood fill method is used. In the future I plan to seperate the processing time information to verify this.

For the one byte algorithm, neighbouring bytes to the current byte were visited if their value was non zero. Under this scheme objects that were not physically joined by black could be located as a single object. This was not an adverse problem when locating staff systems, in fact it was a desired property. In a piece of music scanned at 150 dpi, generally agreed to be the lowest resolution that OMR can be reasonably expected to go [1], the largest jump a one byte algorithm could make is 14 white pixels. This is equivalent to $\frac{1}{10}$ inch. It would be hard to imagine a piece of work where staff systems were placed this closely. No test piece collected so far has staff systems that close.

3.2 Staff Location

Now that the music was structured into pages and staff systems, the staffs themselves could be identified. As was mentioned earlier, this is the stage where most existing OMR systems start. The staff location algorithm was developed from my existing staff location algorithm, written for my honours project. A brief outline of the original method is as follows.

Using a horizontal projection of the page, clusters of large valued peaks were isolated and identified as staff 'areas'. A staff area was a rough approximation for a staff. All that it said was that a staff existed within the confines of the delimited region. Further work was performed to precisely specify the staff. This was achieved by using a vertical projection of the staff area to identify sections of the staff with 'low noise' i.e. only staff lines. Suitable segments on the left and the right of the area were chosen and a horizontal projection of these segments taken. From this, the position of the staff lines on the left hand side and the right hand side of the page could be determined, hence the angle inferred.

The algorithm developed for **xsep** worked in much the same way but with a few changes. The algorithm was no longer applied to the entire page, but to each staff system. The original algorithm could not correctly identify an image with only one staff. This deficiency was corrected. The algorithm then proceeded as before, locating segments on the left and right hand side of the staff area. The old algorithm also only looked for staffs with five lines. The new algorithm generalised this. Unfortunately it was found that parts of text letters could cause false matches for left and right horizontal segments, the serifs on letter were a particular promenant cause. The highest number of lines a letter could cause was three [2] (the letter 'E'). A restriction was added that insisted that a segment must have at least four horizontal lines. With this adaptation and the (on average) reduced number of staffs the algorithm was applied over, the new algorithm was insufficiently robust. Further changes were made to increase

[1] This is not actually very helpful, since it is the size of the music on the page that is important.
[2] Assuming a European alphabet

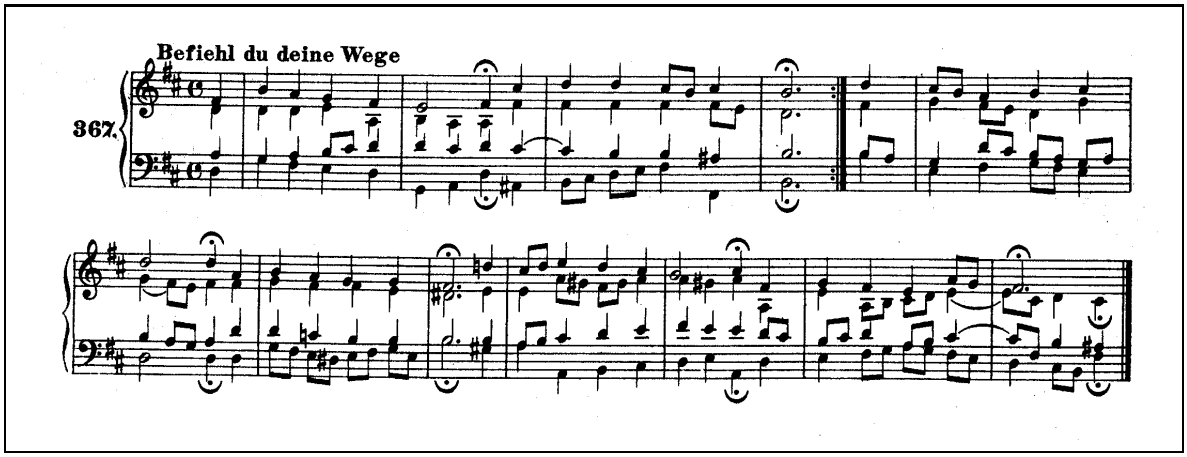


Figure 6: Choral bumber 367 by J.S. Bach.

its reliability. It is suspected that additional modifications will be made in the future as more scanned examples are tested.

With this work complete, the information about staff line thickness, staff gap height and staff height were finally exposed. From early on in the project, this was highlighted as a pivotal step in the work, but eventually realised not the first objective. This work also calculated the angle of each staff. All this data was totalled up, averaged and displayed to the user through the user interface, but that did not mean that local level data was thrown away. It was stored within each staff object. This maintains flexibility in future software. For example should it be discovered that staff systems are not printed parallel, then the individual information on skew can be utilised to locally correct each staff system. Given the background reading on laying out printed music, there is no reason to believe that they would be *perfectly* parallel. Also it caters for the previously described condition of different sized staffs within the same piece.

Currently no cross checks are made between the number of staffs found in each staff system. This is likely to remain so, since it is common practice to save space in scores by dropping instruments from a particular staff system if they are not playing.

3.2.1 Results

The image shown in Figure 6 was feed to **xsep**, and the program was directed to locate the staff systems and then the staff areas, respectively shown in Figure 7 and Figure 8. The program indicates the segments of the staff area that it used to obtain the information about the staff. These appear as boxes in Figure 8.

A comparison of the processor time of the various methods is tabulated in Table 3. From this work, information about the staffs was obtained: the average staff height was 66 pixels, the average staff gap height was 13 pixels, the average staff line thickness was 3 pixels and the skew of the staff lines was on average -0.322 degrees from the horizontal.

To verify the correctness of the work, a hand crafted blank staff system was generated using the

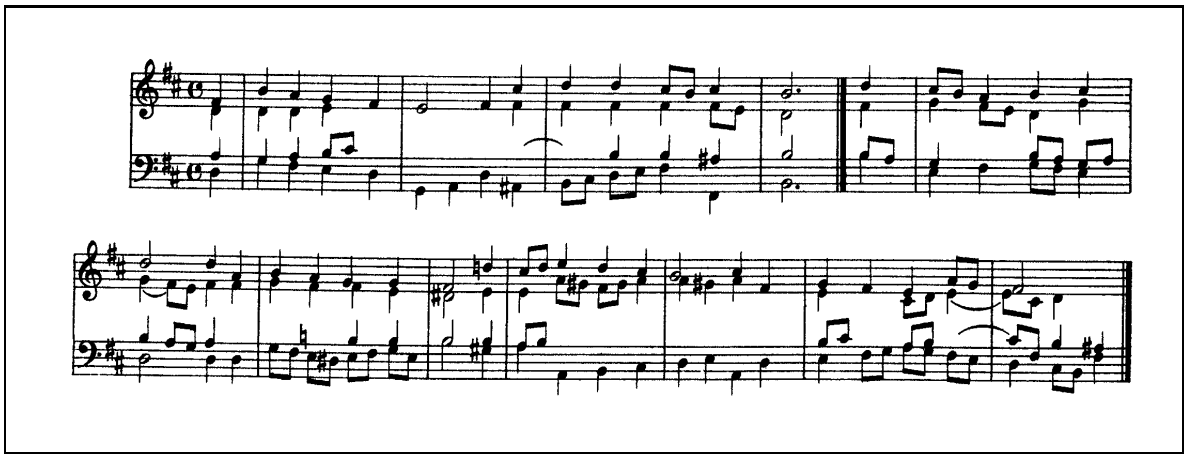


Figure 7: Staff systems located in choral number 367 by J.S. Bach.

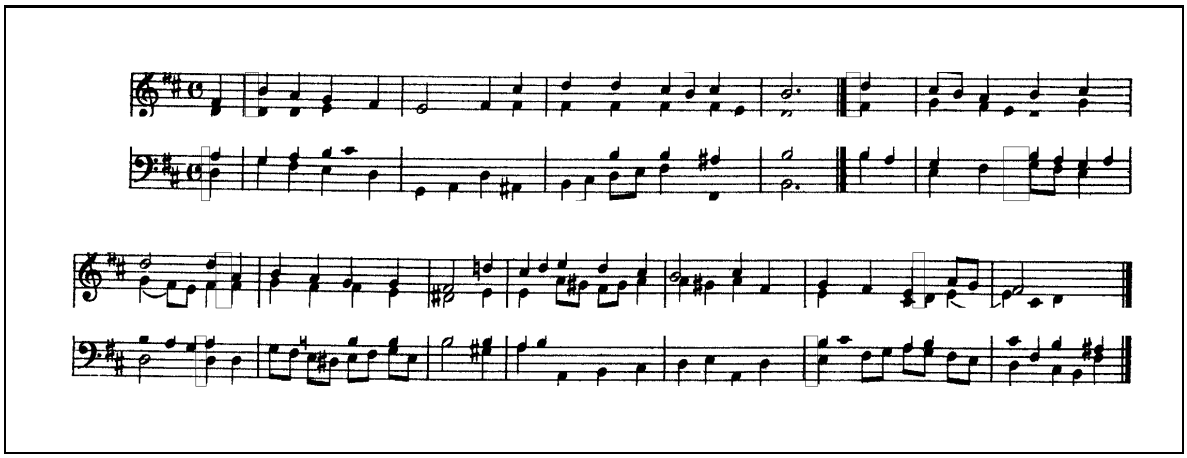


Figure 8: Staff areas located in choral number 367 by J.S. Bach.

Algorithm	User Processor Time	System Processor Time
1-bit flood fill	28410 msec	210 msec
2-bit flood fill	28660 msec	150 msec
1-byte flood fill	8680 msec	110 msec
Contour trace	17500 msec	120 msec

Table 3: Processor timing information from **xsep**.

PostScript graphics drawing language, and a precise angle of skew intentionally added. This was then coverted into the *portable bitmap* format required by the program. The work did indeed correctely extract the staff information.

3.3 Correction For Skew

There was no new innovative step involved in correcting the skew of the scanned image. Skew would be corrected using existing algorithms: namely rotation and shearing. What was desired was a more quantitative analysis of the pros and cons of each approach, than had appeared in the literature. The processor timing display could be used to measure the expense of each algorithm. The quality of the resultant image is a more subjective matter. It is intended to test this by correcting the skew of an image by rotation and shearing, then comparing the number of mistakes made by the recognition phase.

Having worked closely on the details of implementing these algorithms, some interesting points came to light.

The standard image processing trick of performing a transform in reverse to reduce 'holes' appearing in the transformed image, due to rounding errors, was used for both correction methods. That is to say that each integer coordinate in the destination image is transformed back to the original to see what intensity to make it. Performing the transform in reverse also permits the easy introduction of anti-aliasing techniques [FJDF90] to further reduce rounding problems, should the need arise.

The performance of the rotation routine can be improved by pre-computing the expensive mathematical calculations for $rcos\theta$ and $rsin\theta$ where r ranges $0..width\ of\ bitmap$. Each new line in the transformed image starts at a new coordinate. The pre-computed values are added to calculate its point of origin. The same optimisation of pre-computation can be carried out for shearing, only this time arithmetic is performed entirely on integers. Another improvement that can be made to the shearing algorithm is to carry out byte operations instead of pixel operations.

A potential improvement is to only rotate/shear the isolated objects. Rotating and shearing operations on a page of music spend a high proportion of time transforming large areas of 'nothing'. Obviously this adaptation moves the time complexity away from a constant, dependent on the size of the image to become dependent on the variable number of objects found in the page. Indications are that in the context of pages of music, the ratio of 'nothing' to 'interesting' is sufficiently great to make object transformation faster.

A rotated image increases both its x dimension and its y dimension. A sheared image only increases

its y dimension. Carefully choosing the corner of the image to perform the rotation or shear, dependent on the angle of correction (positive or negative) greatly simplifies code.

The same expansion of dimensions also leads to small segments of the original image being omitted from the transformation if a reverse technique is employed. These small segments must be treated as special cases. It could be argued that when transforming an entire page of music, it is acceptable to 'loose' such small wedges at the top or bottom of the piece. This is an unacceptable argument if transforming each object in the page separately.

The Sun SPARC architecture performs floating point arithmetic in hardware. Perhaps the traditional view of floating point operations as expensive and to be avoided was out dated for this particular task. The timing display shows rotation is slower than shearing, so the floating point arithmetic does carry an overhead, though not as bad as was originally feared. It should be noted that I have been advised to re-run the test with the software compiled with the optimiser option on.

Shearing makes staff lines as level as a rotation would, however the skew of any vertical line remains unchanged. Angles between the two are corrected proportionally. Concern is cast over the suitability of shearing in OMR. The most common occurrence of overlapping is that of staff lines (horizontal) with other musical features. The next largest category is the overlapping of slurs, ties, 'hairpins', etc. with bar lines (vertical). A degradation in the alignment of such lines could compound problems when separating shapes.

A solution might be to additionally apply the 'shear force' in an orthogonal direction. Whether this is a better solution than rotation depends on the answer to the question: Are two shears more or less expensive than one rotation? A substantial overhead is accessing the million or so pixels in a scanned image. This fact favours a rotation over two shears. It is planned to investigate this further.

Consider Figure 9. Are two shears really the same as a rotation? The answer is yes it is a rotation, however a scale factor is introduced.

From Figure 10:

$$\begin{aligned} \cos\theta &= \tfrac{h_n}{h_s} \\ \Rightarrow \quad h_s &= \tfrac{h_n}{\cos\theta} \\ \Rightarrow \quad sf &= \tfrac{1}{\cos\theta} \end{aligned} \tag{1}$$

Where sf is the scale factor.

Similarly for the y-direction. Through 'similar triangles' the shear force required to improve the image is the same for each direction. Thus the sheared image is scaled by the same factor. Since the correction for skew is undoing the two shearing forces, the corrected image will in fact be smaller. For the small angles encountered in OMR, this should not cause a problem. The skew in Figure 6 is 0.322. This would produce a scale factor of 0.9999842 i.e. an error of 0.00158%. A desirable property of the double shear is that the aspect ratio is preserved.

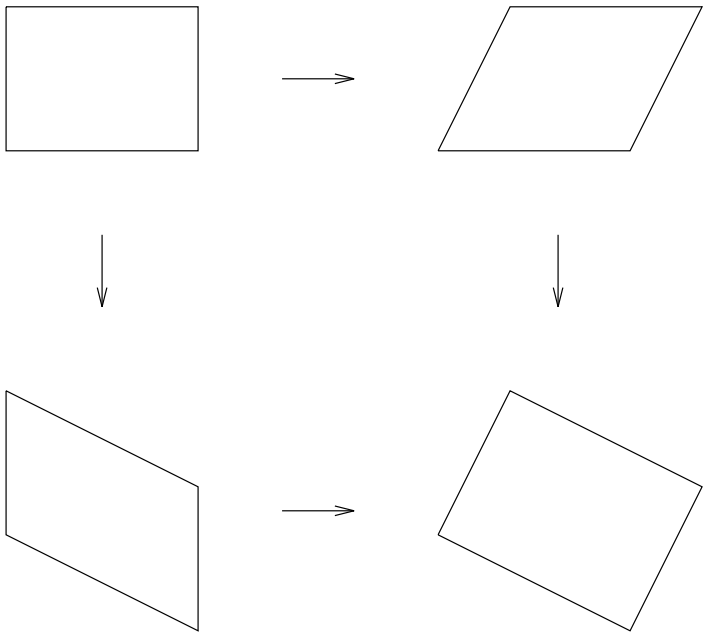


Figure 9: Applying 2 shear forces of equal magnitude in orthogonal directions.

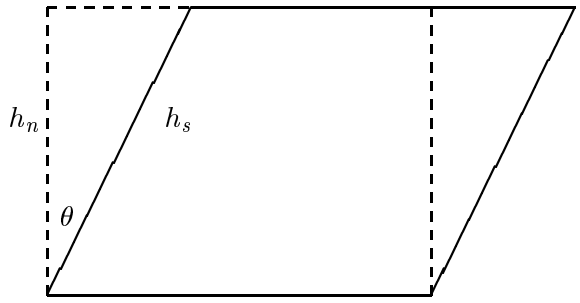


Figure 10: Calculating the scale factor caused by a shearing force.

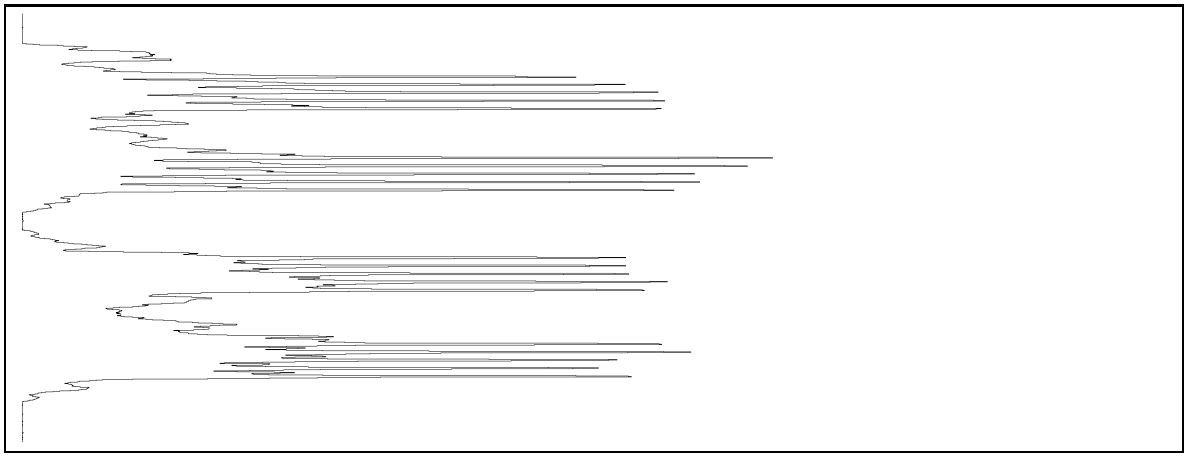


Figure 11: Horizontal projection of scanned image.

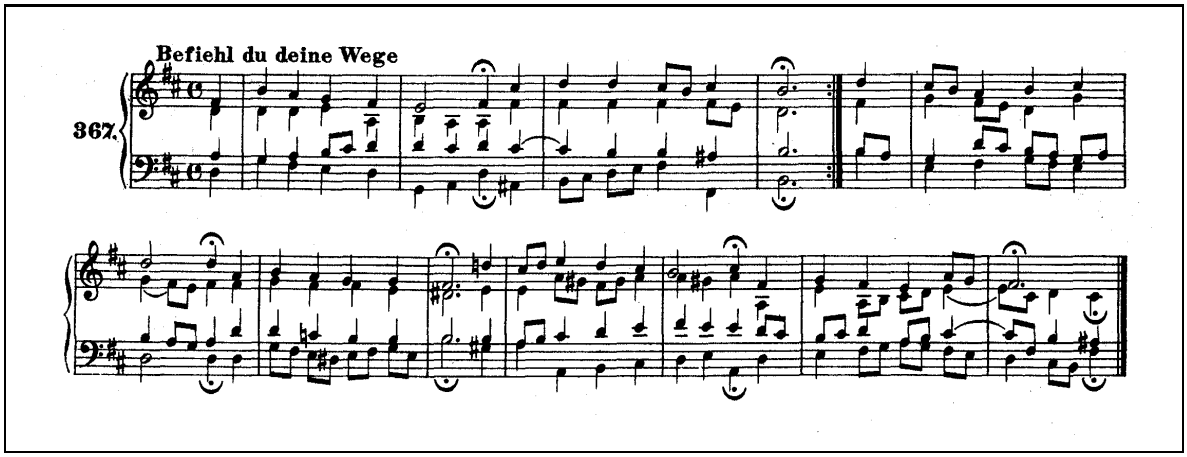


Figure 12: Corrected image using rotation

3.3.1 Results

The horizontal projection of Figure 6 is shown in Figure 11. The scanned image was corrected for skew using a rotation and a shear. The resultant images are shown in Figure 12 - 15.

The timing performance of the various alternative forms for correcting skew are tabulated in Table 4. It was noticed that requesting the program to perform the same task would produce similar timing values, but not precisely the same. The documentation available on the system calls used by the program to obtain processor time information was insufficient to discover the reason. In Figure 15 the system processor time values are all around the value 100. The variation in not significant.

It is interesting to note that shearing objects using byte operations is slower than shearing the entire page using byte operations. When shearing a line of an object there is an overhead caused by the first and last byte of the line that must be treated separately. A lot of the objects found off the staff are relatively small but numerous compared to the number of staffs contained in a page. It would appear this overhead starts to dominate the task, when individually shearing each object in the image. I intend

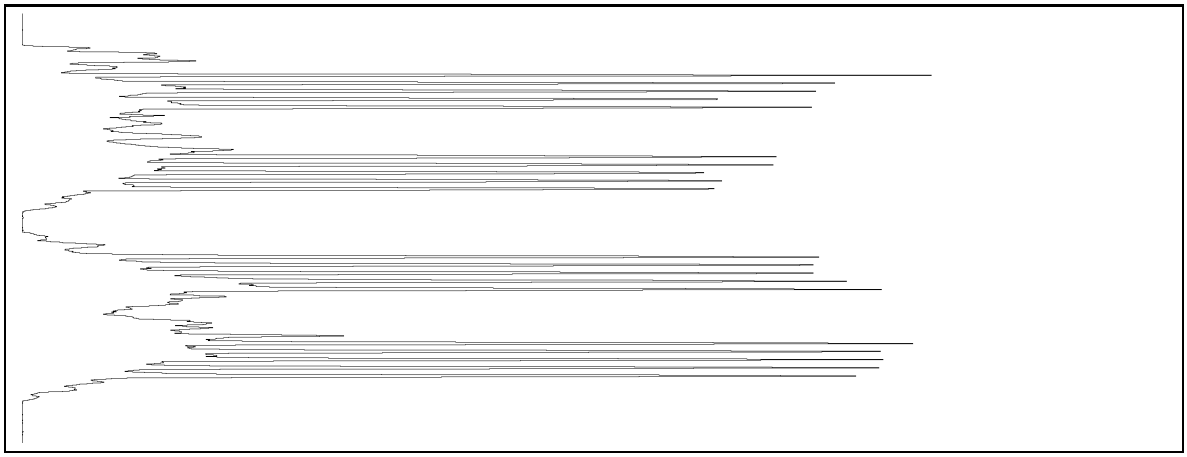


Figure 13: Horizontal projection of corrected image using rotation

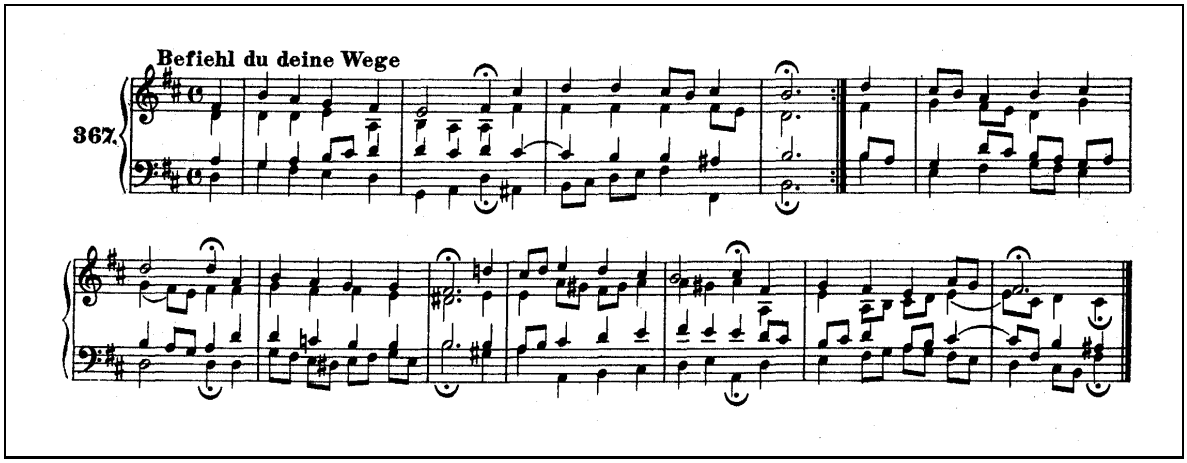


Figure 14: Corrected image using shearing

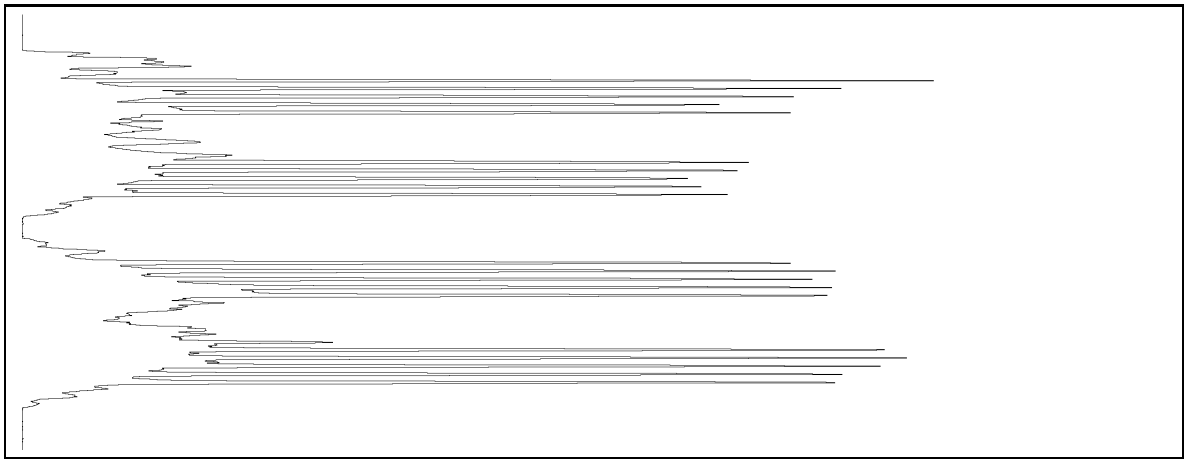


Figure 15: Horizontal projection of corrected image using shearing

Algorithm	User Processor Time	System Processor Time
Rotate page	39140 msec	120 msec
Shear page (bit ops)	28780 msec	100 msec
Shear page (byte ops)	3020 msec	80 msec
Rotate objects	28940 msec	110 msec
Shear objects (bit ops)	22110 msec	140 msec
Shear objects (byte ops)	6960 msec	90 msec

Table 4: Processor timing information for correcting skew from **xsep**.

to clarify this is indeed the cause in future work.

3.4 Extracting Staff Objects

The algorithm for my honours project isolated musical features by finding objects between the staff lines, then extracting the shapes using a modified flood fill algorithm that was adapted to 'jump' over staff lines if there was sufficient evidence that the object continued out the other side. Specific issues to my honours project complicated matters. It was also hard at times to envisage how the jumping rules effected the flood fill algorithm. A well documented and more common approach is to remove the staff lines [AC82], [CBT88a], [CBT88b], [KKII89], [SCEC89], [FAPH91], [Fuj92], [Sic92]. The honours work was done to explore an alternative route. I had its good points and so will not be dismissed entirely.

For the current project, I returned to the more traditional approach of removing the staff lines. The basic strategy [CBT88a] is to work along each staff line, independently deciding whether or not to remove this particular slither of staff line. Fragmentation of musical components is a well known consequence of this approach. A promising enhancement is described by [MB91b]. Instead of looking above and below the staff line for the existence of an object, all chords formed from a continuous set pixels leaving the staff line on a radial line to a specified circle are tabulated and used to decide whether or not a slither of staff line is contained within a musical component or not.

A variety of algorithms were written, based on these two methods. The reported methods seemed to assume that staff lines were straight, hence they must rely on $y = mx + x$ to follow each staff line. In fact they usually assumed the image was corrected for skew, hence $m = 0$, reducing the formula to $y = c$. I do not believe this is flexible enough. Music scanned from a book may result in the bowing of staff lines. The same effect can be noticed in photo copied books, where the lines of text acutely curve near the spine. Also an elongated one pixel up one pixel down stepping phenomenon in staff lines was observed in images that had been corrected for skew. It is a consequence of aliasing. To quote from Foley and Van Damn [FJDF90]:

> "This visual artifact is a manifestation of a sampling error called *aliasing* in signal processing theory; such artifacts occur when a function of a continuous variable that contains sharp changes in intensity is approximated with discrete shapes"

The usual way to treat an aliasing problem is to temper the intensity of pixels by the intensities of

Algorithm	User Processor Time	System Processor Time
Basic	7731 msec	90 msec
Basic wobble	8721 msec	150 msec
Basic track	10411 msec	80 msec
Chord based wobble	128812 msec	400 msec
Chord based track	1285473 msec	310 msec

Table 5: Processor timing information for removing staff lines.

their surround pixels, effectively 'blurring' the image. Any sharp intensity gradient in the image will acquire pixels at in between values close to the edge, thus reducing the 'jaggedness'. This particular approach to anti-aliasing is not well suited to the given problem since it would require a change from a monochrome image to a gray-scale image.

Different anti-aliasing techniques could be employed to reduce this error, but the bowing problem would still remain. The oscillation of staff lines could be viewed as a special case of bowing. A solution to bowing would, therefore, naturally encompass the stepping error, making any anti-aliasing work redundant.

Increasing the scanning resolution will not eradicate the problem. The stepping effect will always be one pixel. However, increasing the scanning resolution will reduce the relative error ($\frac{1}{staff\ line\ thickness}$) this phenomenon has on the staff lines.

My work, for this section of OMR, therefore, focussed on how best to follow these less than perfect staff lines. The two flexible stages developed were called *wobble* and *track*. They were combined with the standard and chord based procedures to produce enriched versions.

wobble used the y-extent of the previous staff line slither to locate the next slither. *Track* used $y = mx + c$ as a starting point for a staff line slither, then looked above and below that point for a possible candidate, should no slither be found at the starting point.

Just as it was desired to study the success of recognition work on an image that had been corrected for skew by rotation or by shearing, so too was it desired to pursue the path of recognition of an uncorrected image. If satisfactory results can be achieved with vertical lines being left uncorrected (as is the case with shearing), they why not both horizontal and vertical? All staff line removal algorithms incorporated a gradient parameter to make this option available.

A conversation held with Professor Witten is worthy to note on this section of work. Professor Witten had recently supervised a student who had used the Hough transform [BT88] to locate staff lines. This could be classified as a global method, whereas my algorithms had a distinctive local information approach to them. *Track* (developed after Professor Witten's visit) is really a hybrid model, hopefully capturing the beneficial qualities of each methodology. Future work on this area might expand the global content of the current algorithm.

3.4.1 Results

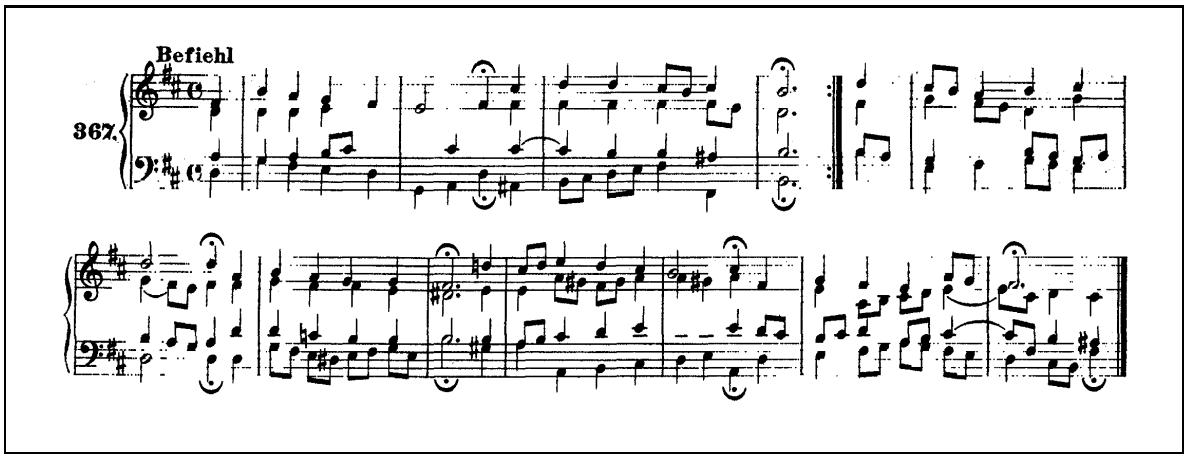


Figure 16: Straight staff line removal.

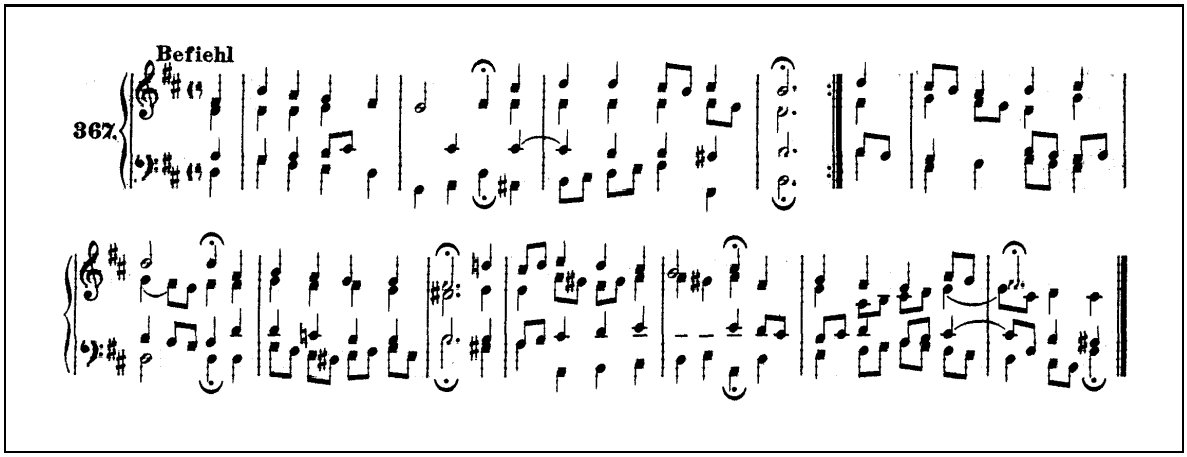


Figure 17: Wobble staff line removal.

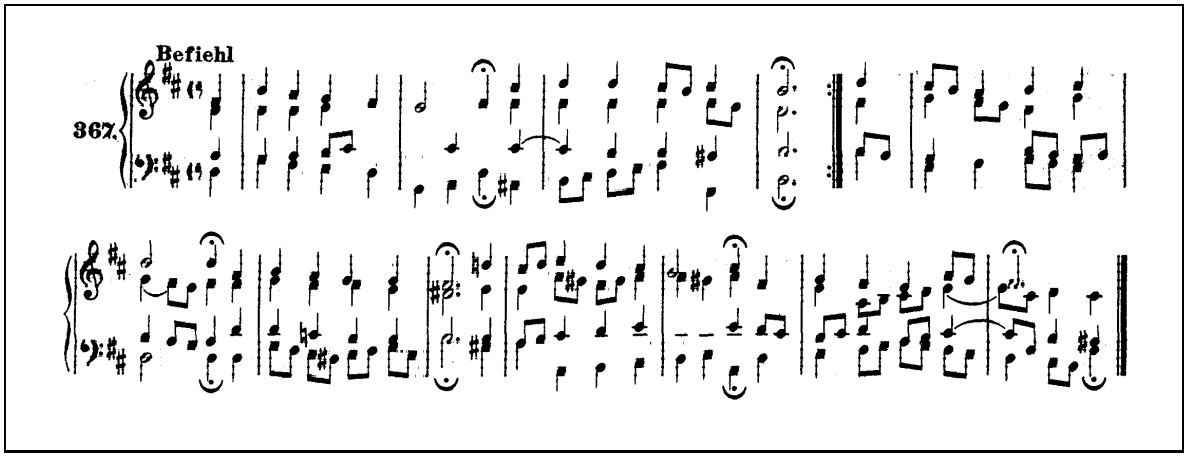


Figure 18: Track staff line removal.

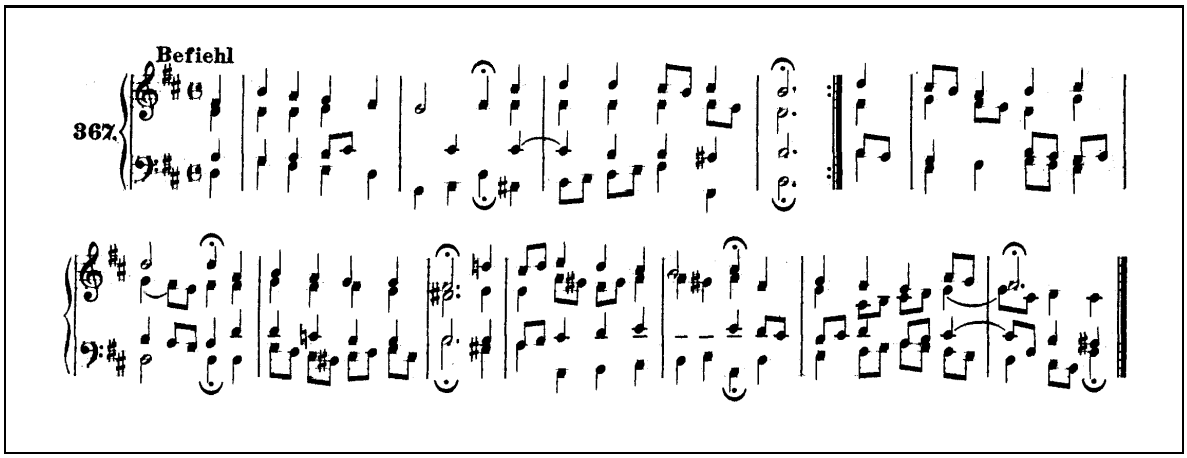


Figure 19: Wobble circle staff line removal.

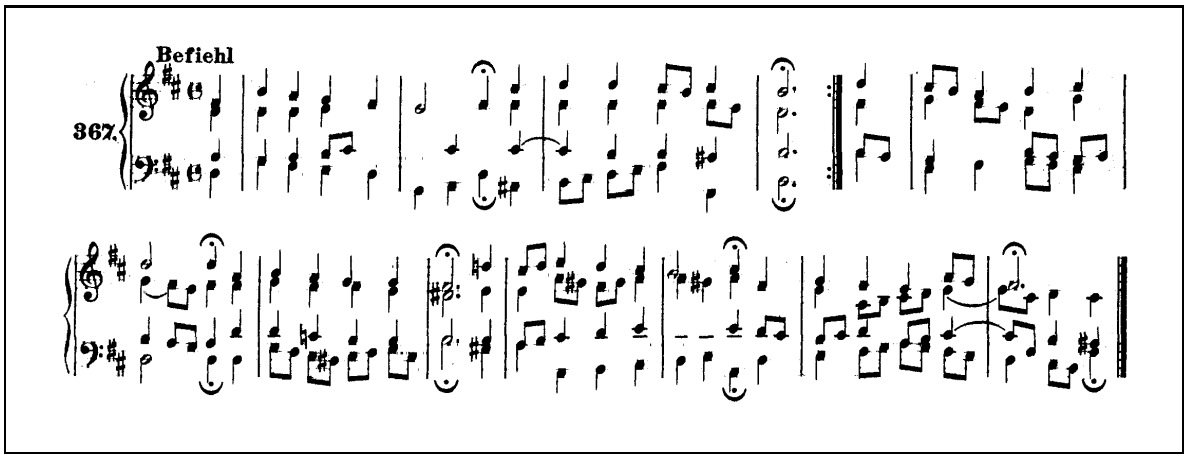


Figure 20: Track circle staff line removal.

Figure 21: Chord based removal causes correctly and incorrectly joined of objects.

The applications of the various staff line remove algorithms are shown in Figures 16 - 20.

Wobble in its present form is too flexible and in certain situation can wind up following the wrong thing; for example a slur that crosses a staff line is sometimes removed rather than the intended staff line. This was the principal motivation behind *track*, which used where the ideal line should be as a constraint to prevent the removal process wandering off.

The chord based removal algorithm did reduce the cases of fragmentation, however breakages did still occur. This fact was also reported in the published paper. Figure 21 shows a bass clef still connected. The chord based method was intended to improve isolation of musical components, but in a way, the situation worsened since examples of incorrectly joined objects were found: key signatures being the most common case observed. This is also shown in Figure 21. The algorithm also left more of the staff line attached to the musical components than the standard method. These appeared as thin worm like lines projecting horizontally from the sides of a musical component (Figure 22). The authors proposed a cleaning up phase to tidy up the objects, however I believe their claim to be premature. If it were possible to distinguish which thin lines on the object that existed where staff lines had been were safe to remove and those that should be left alone, then one would have solved the fragmentation problem in its entirety. The cleaning up stage is needed, however important line may be removed that leave objects fragmented just as much as before. Another factor to consider is the vast computational increase. From Table 5 it can be seen that the chord based method requires almost 15 times more processor time. Using the chord technique, a later phase in the OMR process must not only worry about fragmented objects, but would additionally have to tackle the issue of joined objects.

3.5 Correction of Deformation

It was known that staff lines in a scanned image could curl near the edges, but how could this corrected? It was observed that a staff line is intended to be horizontal, therefore any defect introduced by scanning is reflected by what happens to the staff lines. Extracting the staff lines as they are found in the image, provides a discrete mapping between the intended horizontal lines and the lines as they were scanned.

The algorithm developed to correct the deformation collated all the points in common with each staff line. This was then electronically thinned down to one pixel in height and any gaps filled. The data

Figure 22: Chord based removal causes worm like horizontal lines projections.

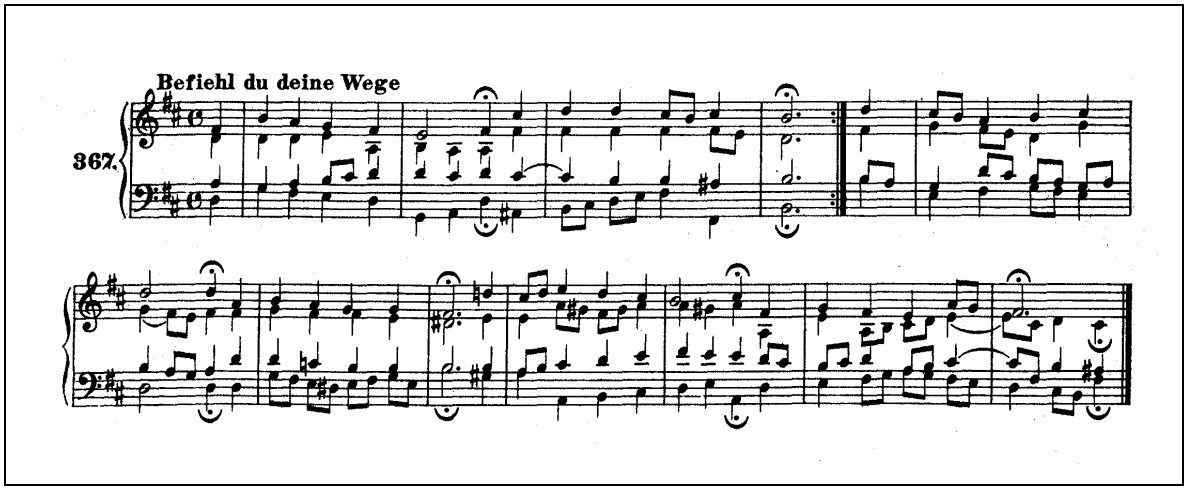


Figure 23: Deform corrected image.

was used to provide the discrete transformation. Each x co-ordinate value in the width of the scanned image, mapped to the y co-ordinate value that it corresponded to in the thinned line.

In a way, the transform can be thought of as a shear of variable load over x. In fact any error of skew introduced by scanning is incorporated in the transformation, making the shear operation redundant. Figure 24 and Figure 23 show the Bach choral (Figure 6) transformed using the correct deformation algorithm. It operation took 8190 msec of processor time.

This work is only recently finished and there has not yet been sufficient time to accumulate results from a variety of pieces.

4 Recognition of Isolated Shapes

At the start of the project it was felt that music pattern recognition routines had not been developed sufficiently under the honours project. Thus it would be ill-advised to propose, as in other stages, an iteration of the design cycle when the body of pattern recognition routines was incomplete. It seemed best to plan work that extended the existing system. This would also permit me to re-acquaint myself

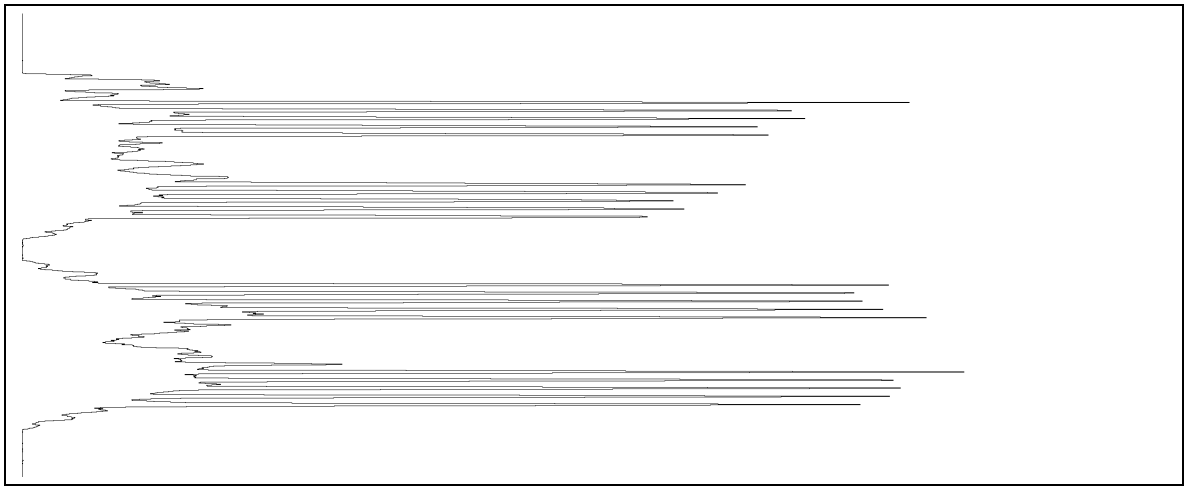


Figure 24: Horizontal projection of deform corrected image.

with the software as well as re-immerse myself in the problem after a two year break and think once more about the issues at stake. Two enhancements that increased the functionality of the software and have proved significant benefits to the development environment were a 'shrink canvas' and break points.

The original system displayed a scanned image (and any derived images requested) on the screen, only if the full size version fitted. A *portable bitmap* (pbm) file was written for each requested screen, when the image was too large. This file could then be viewed using utility programs like **xv**, which took care of scaling. Any scanned piece of music wider then three inches would result in the generation of files, and the frustrating delay in viewing them. Unlike **xstaff** and **xsep** where the shrink canvas was an integral part of the design, the shrink canvas idea was added retrospectively to **xmsr**. All screens plotting routines were modified to 'lossy' algorithms, which only plotted every n^{th} point, where $n = max(\frac{imagewidth-1}{screenwidth}, \frac{imageheight-1}{screenheight}) + 1$ An additional 'life size' routine was also provided for each screen drawing routine.

Even with the substantial quantity of information describing the matching process the system was going through to recognise musical shapes, it was possible to find incorrectly identified musical features, where there was no apparent reason for the mistake. Break points were included to increase the investigative powers of the system. Just as a break point in a debugger halts the program at a specified line, setting a break point within **xmsr** by pressing a mouse button over the component in the image in question stopped the reasoning strategy at the point where it was trying to match that shape. Using a standard debugger, the values of variables within **xmsr** could be printed, the code stepped through or any of the other facilities a debugger provides, to shed light on the problem before continuing the reasoning strategy.

The claim of my honours work was that the methodology devised was extendible. This was put to the test by extending the class of common music notation (CMN) recognised by the system.

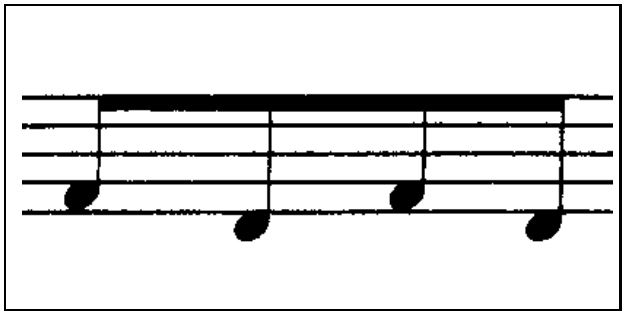


Figure 25: An example of a simply beamed group of notes.

Chords were added first. All that was required to realise this was to add the necessary entries in the musical feature description file and enhance the audio extraction stage to look at the x coordinate of each note head to decide if it formed a chord with a previously encountered note head, or the start of a new note. It had already been noted in my honours report that adding chords using the existing musical feature description language would be verbose.

Adding slurs was a more involved piece of work. The entry in the musical feature description file was straightforward and developing a 'connectivity' (see [Bai91]) based pattern recognition routine, simple enough. Complications arose in the ordering of musical components. For the original set of CMN recognised, it was safe to order the shapes on their x component, followed by their y component, since no musical entry in the set could interfere with another musical entry. The ordering allowed a simple, efficient linear matching routine that compared the bounding boxes found in the image with the permissible layout of bounding boxes in the musical feature description file. The introduction of slurs negated this non-interference property. The matching routine was replaced with a more convoluted, recursive alternative.

Slurs also effected the audio extraction phase of the program. The range of notes a slur extended over had to be found. If the range was only two notes in length and both notes heads had the same pitch, then the notes were tied together, rather than slurred. This function was also recursive to deal with slurs that were broken over one or more lines. Not only did the function have to determine the range of notes a slur effected, but which note heads in chords were involved. It was decided to bind the end of a tie to its nearest note head and that a slur effected all notes in a chord. This is yet to be confirmed as the correct treatment for all situations.

Permitting split voices on the same staff was already dealt with correctly by the graphical recognition phase. Again it was the audio extraction section that underwent change - noticeably the slur application function and note head extraction.

The original system only correctly interpreted simply beamed notes. By simply beamed I mean that a beam exists for the entirety of the shape, such as the example shown in Figure 25. This was generalised to complex beaming. An example of complex beaming is presented in Figure 26. The duration of a note in a group of beamed notes was determined by the number of beams entering or leaving its stem -

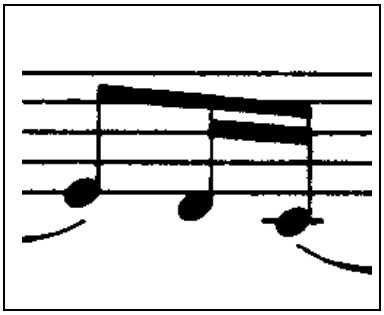


Figure 26: An example of a complex beamed group of notes.

which ever is the greater number. If a hemisemiquaver was the shortest duration and it was represented by the value of one, a semiquaver by two, a quaver by four etc, then the formula is:

$$duration = 1 << (4 - no\ of\ beams) \tag{2}$$

The alto clef was added to the existing clefs (treble and bass) along with the graphical location of all the related key signatures. The tolerance necessary to permit the correct bounding box match to be found where lax enough for a tenor key signature entry *and* a treble key signature entry to be produced as potential matches in some situations. Recognising the primitives found within the bounding boxes would not help distinguish the situation since the entries of the key signatures in question would have identical primitives. It was therefore necessary to add code that 'up-graded' a key signature if it did not match the current clef for that given staff.

The original system only had an entry for the $\begin{smallmatrix}4\\4\end{smallmatrix}$ time signature. By developing pattern recognition routines that identified the values 2,3,4,5,6,7 and 8, the set of permissible time signatures was widened. Even thought the original system only 'knew' about $\begin{smallmatrix}4\\4\end{smallmatrix}$, its duration field in the musical feature description file was utilised correctly. This meant no other changes were necessary to correctly interpret the musical significance of different time signatures. The recognition routines for time signature primitives are weak and needs more attention.

There were numerous small additions made to the musical feature description file. A few of the more prominent ones were hemisemiquavers, triplets, staccato and stressed notes.

Text recognition was added to **xmsr**. A copy of an OCR package written at the Trinity Collage, Dublin was obtained and installed. There are many different approaches to OCR techniques, the Dublin software was based on template matching. It was not seen as an issue for my work to use the most technologically advanced OCR system, just that a reasonably reliable library call existed that could be trusted to state if the provided bitmap was a text character and if it was a text character, which character.

A problem not yet mentioned in connection with slur recognition was super-imposing. Slurs can quite

legitimately cross bar lines and some poorer quality works printed slurs touching note heads. According to background reading, the latter is disliked by printed music layout experts and can always be avoided. Staff line removal dealt with the occlusion of objects by horizontal lines. This situation is an occlusion of an object with a vertical line. It is not the only occurrence, 'hair pin' volume marks can also cross bar lines, for example. A temporary solution was written to address this problem which worked in much the same way as staff line removal. First objects were constrained to only those that had a top or bottom extent that matched up with where the staff areas were. For each of these remaining objects, a vertical projection was taken. Pixels to the left and right of any thin spike found in the vertical projection were check in the object to determine if another object (i.e. a slur or hair pin) passed through that point. The routine is considered insufficient and should be replaced by future work. A property that might prove useful in identifying shapes that need to be separated would be a high bounding box area compared the the number of set pixels in the shape.

As the selection of music presented to **xmsr** grew, it became apparent that some template pattern recognition routines were insufficiently general to correctly match the variations in style (a bit like different fonts in text) of some shapes found in music, This was particularly noticeable in sharps, flats and naturals.

To address this problem, 'targeting' was introduced to **xmsr**. For a troublesome shape, a good quality specimen was extracted from the scanned image and saved as an X11 bitmap. This could then be touched up if desired. Once examples of all the style variant items had been collected, the X11 bitmaps were incorporated into the program and a new option was created for the group and added to the *target* menu in the user interface. The program was then recompiled.

Just as it was common to find slurs touching note heads, it was also common to find accidentals touching note heads. Similarly, this was unnecessary. A 'Sloppy' option was added to the user interface. If highlighted when the recognition phase was invoked, the residue of an object classified as a note was checked for accidentals.

4.1 Results

Figures 27 - 34 show various 'snapshots' of the **xmsr** system recognising "The Promenade" from "Pictures at an Exhibition" by Mussorski.

5 Thoughts for the Future

I believe a musical feature description language is fundamental to OMR work. It is an abstraction that induces clarity, leaving behind complications caused by the domain image as well as avoiding muddled code. A language exposes the raw essence of the problem. The flexibility available is such that extensions to the recognition of handwritten music, guitar tablature music and other music layout schemes are realistic goals. For example, to recognise handwritten music there would be no need to change the musical description file, since there is no change in the musical layout. The only module that

Wladimir Wassiljewitsch Stassow gewidmet

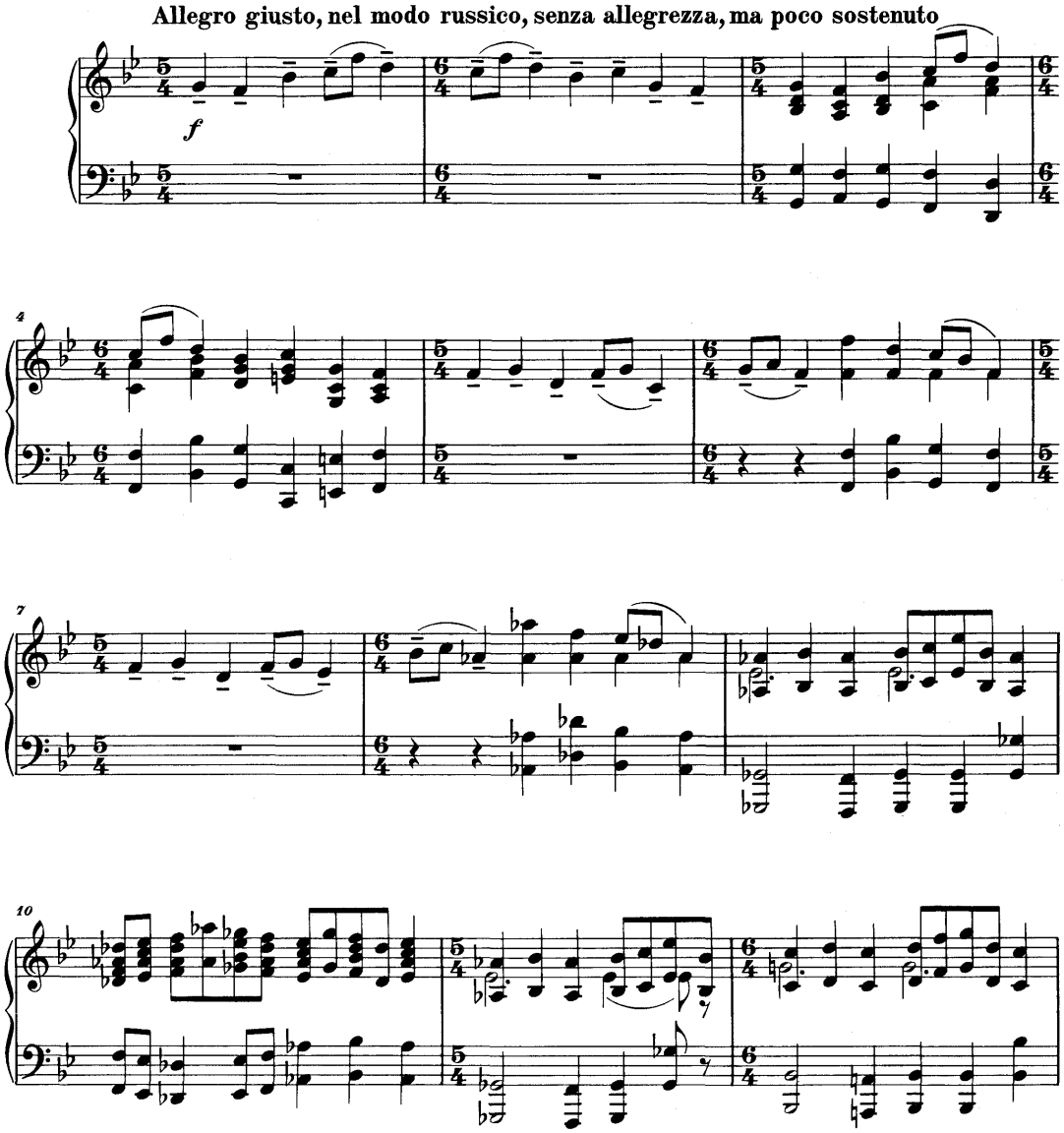
Bilder einer Ausstellung

Erinnerung an Viktor Hartmann

1874

Promenade

Allegro giusto, nel modo russico, senza allegrezza, ma poco sostenuto



Urtext Edition No. 50076

Figure 27: Pictures at an Exhibition, page one.

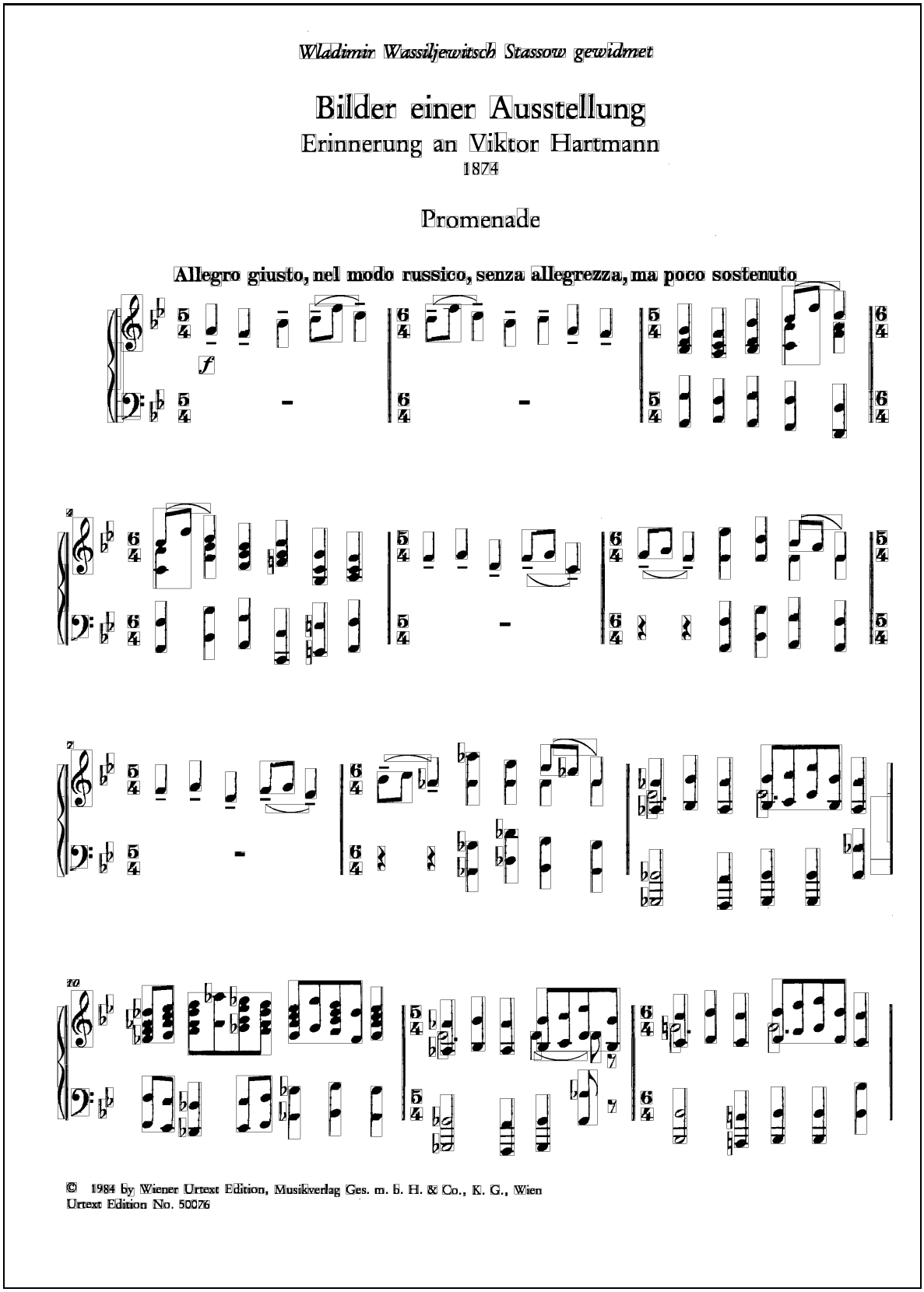


Figure 28: Page one with staff lines removed.

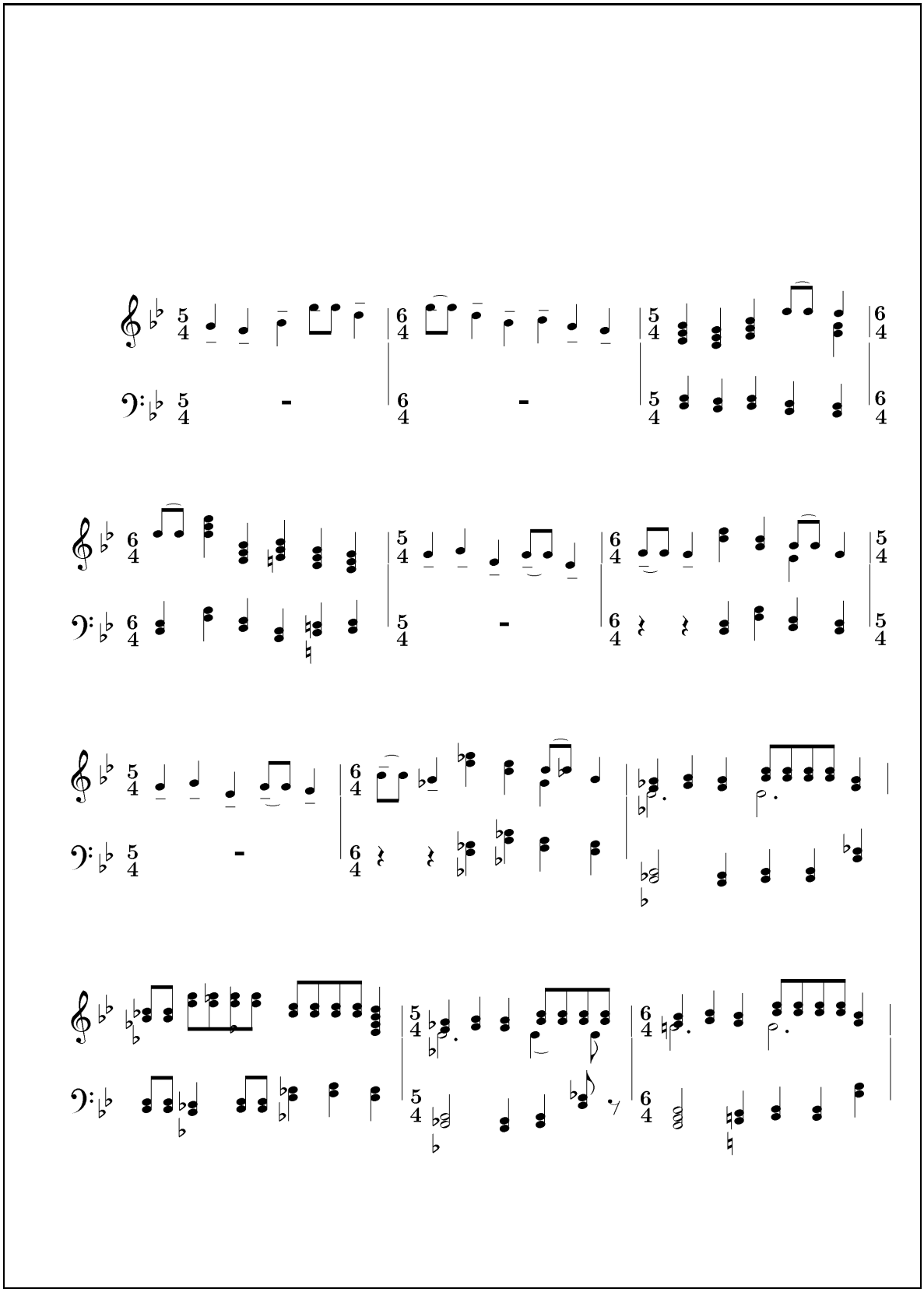


Figure 29: Library pictures of matched objects from page one.

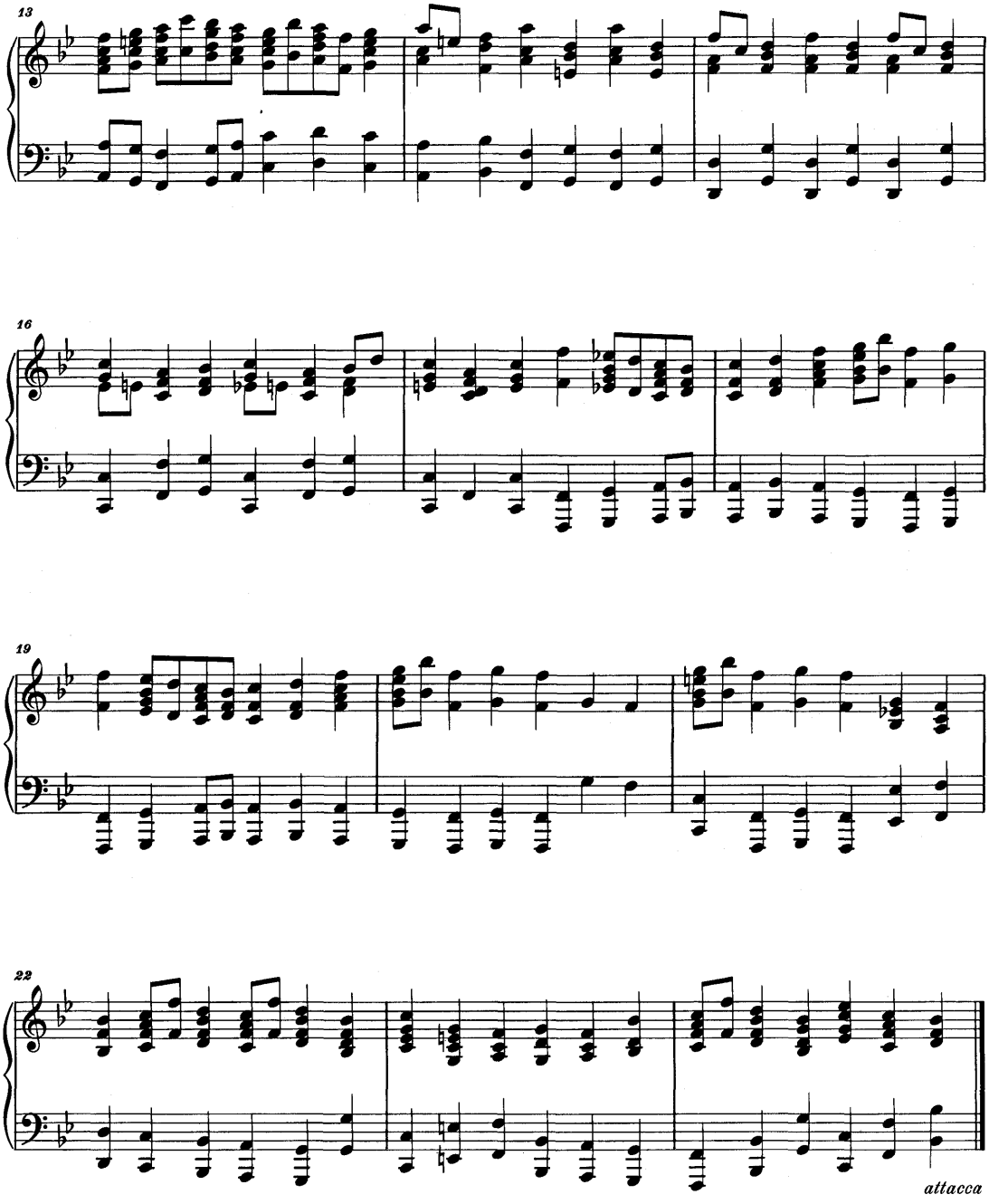
```
1.1:  |  |  |  |
4G(1.0),4F(1.0),4Bb(1.0),5C(0.5),5F(0.5),5D(1.0) |
5C(0.5)\_,5F(0.5)_,_/5D(1.0),4Bb(1.0),5C(1.0),4G(1.0),4F(1.0) |
{3Bb 4D 4G}(1.0),{3A 4F 4C}(1.0),{4Bb 4D 3Bb}(1.0),
{5C 4A 4C}(0.5)\_,5F(0.5)_,_/5D(1.0)&{4A 4F}(1.0) |  |
1.2:  |  |  |  |
rest(5.0) | rest(6.0) | {2G 3G}(1.0),{3F 2A}(1.0),{2G 3G}(1.0),{3F 2F}(1.0),
{3D 2D}(1.0) |  |

2.1:  |  |  |  |
{5C 4A 4C}(0.5)\_,5F(0.5)_,_/{5D 4Bb 4F}(1.0),{4Bb 4D 4G}(1.0),{4G 5C 4E}(1.0),
{4G 3G 4C}(1.0),{3A 4F 4C}(1.0) | 4F(1.0),4G(1.0),4D(1.0),
4F(0.5)\_,4G(0.5)_,_/4C(1.0) | 4G(0.5)\_,4A(0.5)_,_/4F(1.0),{5F 4F}(1.0),
{5D 4F}(1.0),5C(0.5)\_&4F(1.0),4Bb(0.5)_,_/4F(1.0) |  |
2.2:  |  |  |  |
{2F 3F}(1.0),{2Bb 3Bb}(1.0),{2G 3G}(1.0),{2C 3C}(1.0),{3E 2E}(1.0),
{3F 2F}(1.0) | rest(5.0) | rest(1.0),rest(1.0),{2F 3F}(1.0),{3Bb 2Bb}(1.0),
{3G 2G}(1.0),{3F 2F}(1.0) |  |

3.1:  |  |  |
4F(1.0),4G(1.0),4D(1.0),4F(0.5)\_,4G(0.5)_,_/4Eb(1.0) |
4Bb(0.5)\_,5C(0.5)_,_/4Ab(1.0),{4Ab 5Ab}(1.0),{4Ab 5F}(1.0),
5Eb(0.5)\_&4Ab(1.0),5Db(0.5)_,_/4Ab(1.0) | {4Ab 3Ab}(1.0)&4Eb(3.0),
{3Bb 4Bb}(1.0),{4Ab 3Ab}(1.0),{3Bb 4Bb}(0.5)&4Eb(3.0),{4C 5C}(0.5),
{4Eb 5Eb}(0.5),{3Bb 4Bb}(0.5),{3Ab 4Ab}(1.0) |
3.2:  |  |  |
rest(5.0) | rest(1.0),rest(1.0),{2Ab 3Ab}(1.0),{4Db 3Db}(1.0),{3Bb 2Bb}(1.0),
{2Ab 3Ab}(1.0) | {1Gb 2Gb}(2.0),{2F 1F}(1.0),{1Gb 2Gb}(1.0),{1Gb 2Gb}(1.0),
{2Gb 3Gb}(1.0) |

4.1:  |  |  |
{4Db 4F 4Ab 5Db}(0.5),{4Eb 4Ab 5C 5Eb}(0.5),{4F 4Ab 5Db 5F}(0.5),
{4Ab 5Ab}(0.5),{4Gb 4Bb 5Eb 5Gb}(0.5),{4F 4Ab 5Db 5F}(0.5),
{4Eb 4Ab 5C 5Eb}(0.5),{4Gb 5Gb}(0.5),{4F 4Bb 5Db 5F}(0.5),{4Db 5Db}(0.5),
{5Eb 4Eb 4Ab 5C}(1.0) | {4Ab 3Ab}(1.0)&4Eb(3.0),{4Bb 3Bb}(1.0),{4Ab 3Ab}(1.0),
{4Bb 3Bb}(0.5)&4Eb(1.0)\_,{4C 5C}(0.5),{5Eb 4Eb}(0.5)&_/4Eb(0.5),
{3Bb 4Bb}(0.5) | {5C 4C}(1.0)&4G(3.0),{5D 4D}(1.0),{5C 4C}(1.0),
{4D 5D}(0.5)&4G(3.0),{4F 5F}(0.5),{4G 5G}(0.5),{4D 5D}(0.5),{5C 4C}(1.0) |
4.2:  |  |  |
{2F 3F}(0.5),{2Eb 3Eb}(0.5),{3Db 2Db}(1.0),{2Eb 3Eb}(0.5),{2F 3F}(0.5),
{2Ab 3Ab}(1.0),{2Bb 3Bb}(1.0),{2Ab 3Ab}(1.0) | {1Gb 2Gb}(2.0),{2F 1F}(1.0),
{1Gb 2Gb}(1.0),{2Gb 3Gb}(0.5),rest(0.5) | {3C 2Bb 1Bb}(2.0),{2A 1A}(1.0),
{1Bb 2Bb}(1.0),{1Bb 2Bb}(1.0),{2Bb 3Bb}(1.0) |
```

Figure 30: Internal text representation for audio extraction from page one.



UT 50076

Figure 31: Pictures at an Exhibition, page two.

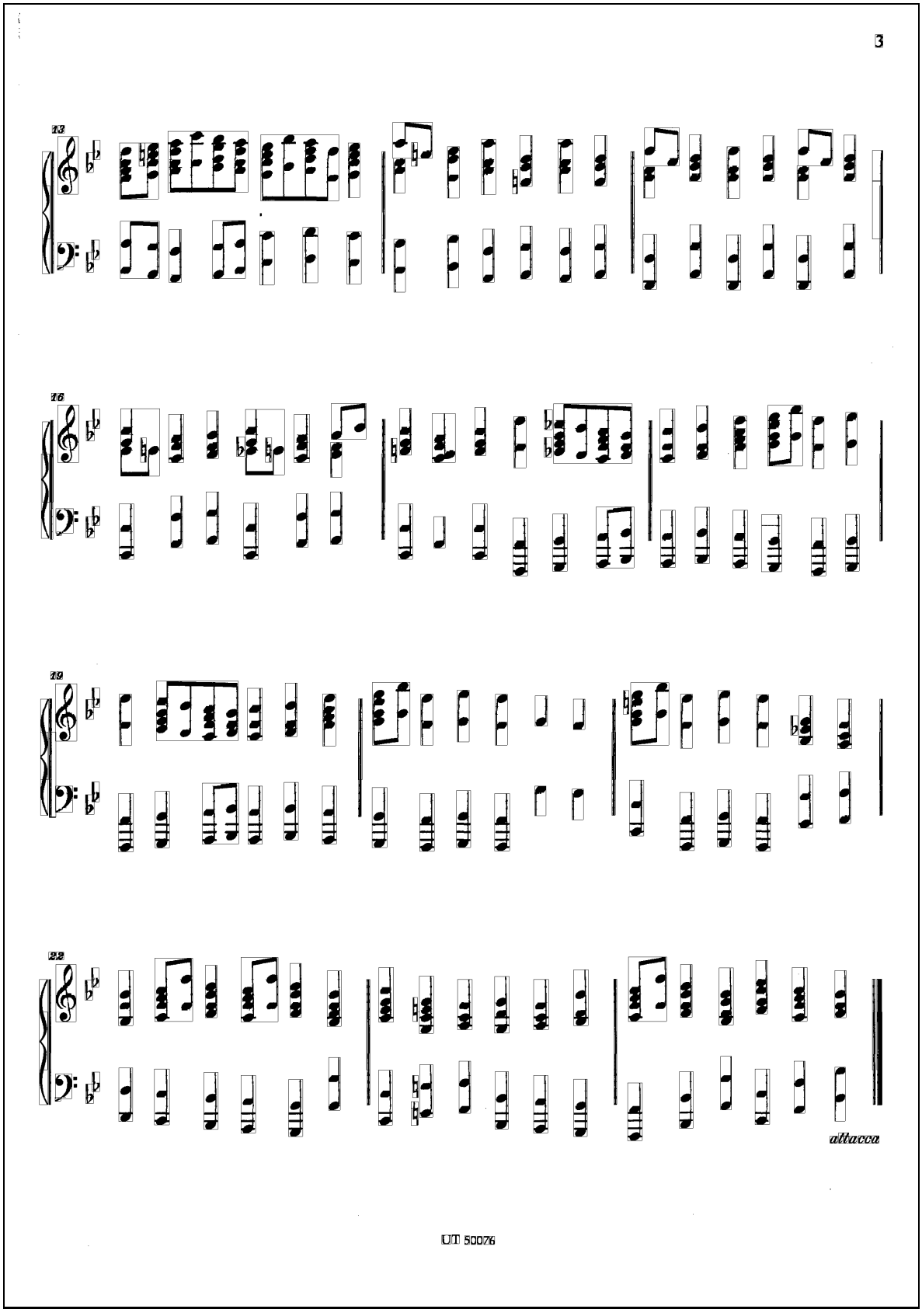


Figure 32: Page two with staff lines removed.

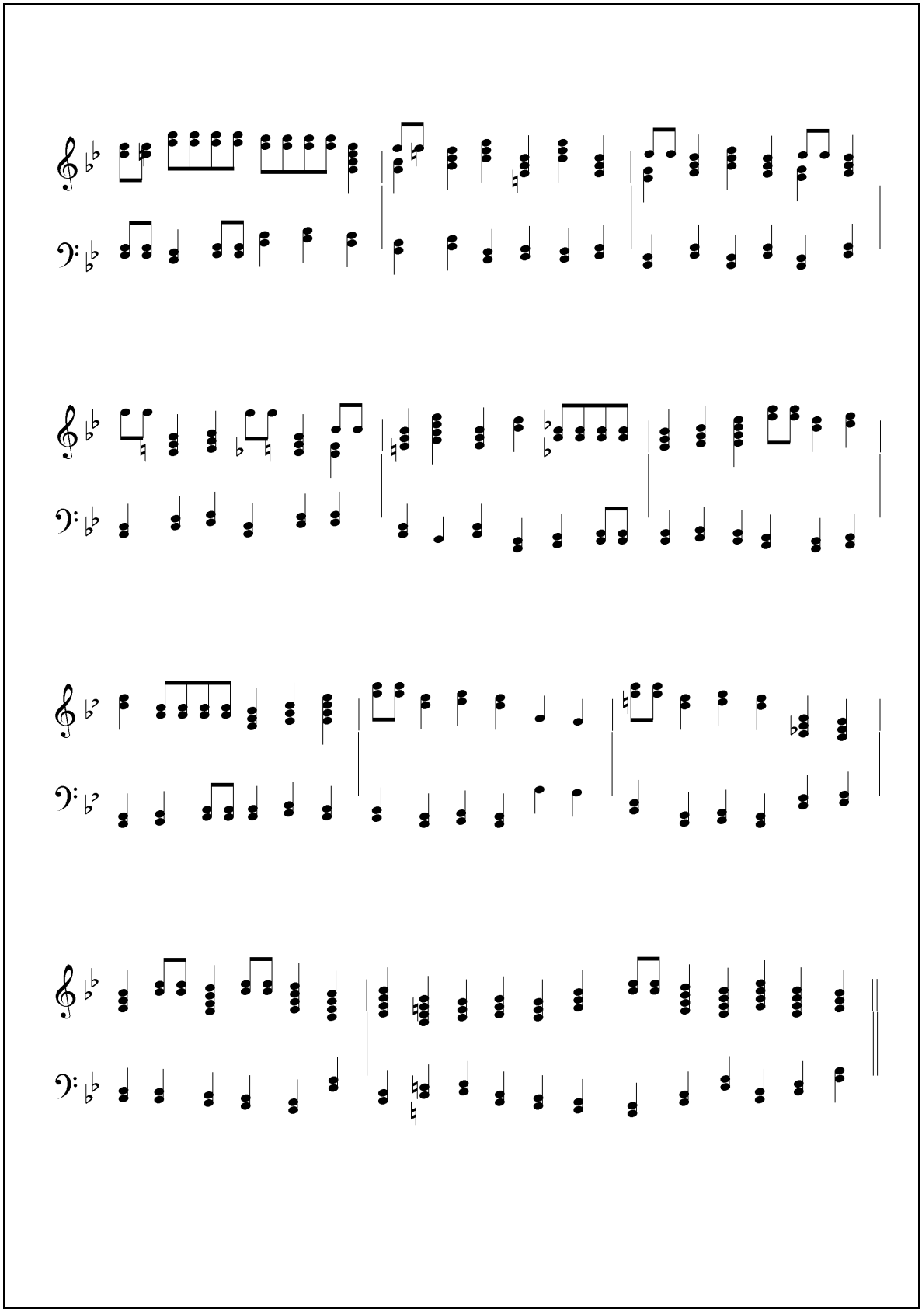


Figure 33: Library pictures of matched objects from page two.

```
1.1:  |  |  |
{4F 4A 5C 5F}(0.5),{4G 5C 5E 5G}(0.5),{4A 5C 5F 5A}(0.5),{5C 6C}(0.5),
{4Bb 5D 5G 5Bb}(0.5),{4A 5C 5F 5A}(0.5),{4G 5C 5E 5G}(0.5),{4Bb 5Bb}(0.5),
{4A 5D 5F 5A}(0.5),{4F 5F}(0.5),{4G 5C 5E 5G}(1.0) | 5A(0.5)&{5C 4A}(1.0),
5E(0.5),{5D 4F 5F}(1.0),{4A 5C 5A}(1.0),{4E 4Bb 5D}(1.0),{4A 5C 5A}(1.0),
{4E 5D 4Bb}(1.0) | 5F(0.5)&{4A 4F}(1.0),5C(0.5),{5D 4F 4Bb}(1.0),
{4A 4F 5F}(1.0),{5D 4Bb 4F}(1.0),5F(0.5)&{4A 4F}(1.0),5C(0.5),{4Bb 5D 4F}(1.0)|
1.2:  |  |  |
{2A 3A}(0.5),{2G 3G}(0.5),{3F 2F}(1.0),{2G 3G}(0.5),{2A 3A}(0.5),{3C 4C}(1.0),
{3D 4D}(1.0),{3C 4C}(1.0) | {2A 3A}(1.0),{2Bb 3Bb}(1.0),{2F 3F}(1.0),
{2G 3G}(1.0),{2F 3F}(1.0),{2G 3G}(1.0) | {3D 2D}(1.0),{2G 3G}(1.0),
{3D 2D}(1.0),{3G 2G}(1.0),{3D 2D}(1.0),{3G 2G}(1.0) |

2.1:  |  |  |
{4Eb 4G 5C}(0.5),4E(0.5),{4A 4F 4C}(1.0),{4F 4Bb 4D}(1.0),{4Eb 4G 5C}(0.5),
4E(0.5),{4A 4F 4C}(1.0),4Bb(0.5)&{4F 4D}(1.0),5D(0.5) | {4E 5C 4G}(1.0),
{4C 4F 4A 4D}(1.0),{5C 4E 4G}(1.0),{4F 5F}(1.0),{4Eb 4G 4Bb 5Eb}(0.5),
{4D 5D}(0.5),{4C 4F 4A 5C}(0.5),{4D 4F 4Bb}(0.5) | {4F 5C 4C}(1.0),
{4D 4F 5D}(1.0),{4A 5F 4F 5C}(1.0),{4F 4A 5D 5F}(0.5),{4A 5A}(0.5),
{4F 5F}(1.0),{4G 5G}(1.0) |
2.2:  |  |  |
{3Cb 2Cb}(1.0),{2Fb 3Fb}(1.0),{2G 3G}(1.0),{2Cb 3Cb}(1.0),{2Fb 3Fb}(1.0),
{2G 3G}(1.0) | {3Cb 2Cb}(1.0),2Fb(1.0),{3Cb 2Cb}(1.0),{1Fb 2Fb}(1.0),
{2G 1G}(1.0),{1A 2A}(0.5),{1B 2B}(0.5) | {2A 1G}(1.0),{1B 2B}(1.0),
{2A 1A}(1.0),{1G 2G}(1.0),{1E 2Fb}(1.0),{1G 2G}(1.0) |

3.1:  |  |  |
{5F 4F}(1.0),{4Eb 4G 4Bb 5Eb}(0.5),{4D 5D}(0.5),{4C 4F 4A 5C}(0.5),
{4D 4F 4Bb}(0.5),{5C 4F 4C}(1.0),{5D 4D 4F}(1.0),{4A 4F 5C 5F}(1.0) |
{4G 4Bb 5Eb 5G}(0.5),{4Bb 5Bb}(0.5),{5F 4F}(1.0),{4G 5G}(1.0),{4F 5F}(1.0),
4G(1.0),4F(1.0) | {4G 4Bb 5D 5F}(0.5),{4Bb 5Bb}(0.5),{4F 5F}(1.0),
{4G 5G}(1.0),{4F 5F}(1.0),{4Eb 3Bb 4G}(1.0),{3A 4F 4C}(1.0) |
3.2:  |  |  |
{2F 1F}(1.0),{2G 1G}(1.0),{1A 2A}(0.5),{1Bb 2Bb}(0.5),{2A 1A}(1.0),
{2Bb 1Bb}(1.0),{2A 1A}(1.0) | {2G 1G}(1.0),{2F 1F}(1.0),{1G 2G}(1.0),
{2F 1F}(1.0),3G(1.0),3F(1.0) | {2C 3C}(1.0),{2F 1F}(1.0),{1G 2G}(1.0),
{2F 1F}(1.0),{3Eb 2Eb}(1.0),{3F 2F}(1.0) |

4.1:  |  |  |
{4Bb 3Bb 4F}(1.0),{4C 4F 4A 5C}(0.5),{4F 5F}(0.5),{4D 4Bb 5D 4F}(1.0),
{4C 4F 4A 5C}(0.5),{4F 5F}(0.5),{5D 4D 4F 4Bb}(1.0),{4Bb 4F 4D 3Bb}(1.0) |
{4Eb 5C 4C 4G}(1.0),{4C 4G 4E 3G}(1.0),{3A 4F 4C}(1.0),{4D 4G 3G}(1.0),
{3A 4F 4C}(1.0),{4Bb 3Bb 4D}(1.0) | {4C 4F 4A 5C}(0.5),{4F 5F}(0.5),
{4D 4F 4Bb 5D}(1.0),{4Bb 3Bb 4G 4D}(1.0),{5Eb 5C 4Eb 4G}(1.0),
{5C 4A 4F 4C}(1.0),{4D 4F 4Bb}(1.0) |
4.2:  |  |  |
{3D 2D}(1.0),{3C 2C}(1.0),{2Bb 1Bb}(1.0),{1A 2A}(1.0),{1G 2G}(1.0),
{3G 2G}(1.0) | {3C 2C}(1.0),{3E 2E}(1.0),{3F 2F}(1.0),{1Bb 2Bb}(1.0),
{1A 2A}(1.0),{2G 1G}(1.0) | {2F 1F}(1.0),{1Bb 2Bb}(1.0),{2G 3G}(1.0),
{3C 2C}(1.0),{3F 2F}(1.0),{2Bb 3Bb}(1.0) |
```

Figure 34: Internal text representation for audio extraction from page two.

would need replacing would be the primitive detection routines that would need to be flexible enough to recognise the less geometrically precise shapes. As far as I am aware, my work is the only OMR project that uses a descriptive language.

The design of a new language, with the lessons I have learnt from the existing language, is still very much held in my head. The process of laying these thoughts out on paper is seen as an important step. One that must be carried out in the near future.

Musically interpreting the significance of the text found in a page of music as a post-processing stage is no longer viewed as a satisfactory scheme. It would result in numerous ad hoc routines, more or less one routine per text group (e.g. instrument name, volume, etc). Each routine would be explicitly managing the two-dimensional relationships between one particular text group and the surrounding music. This would not be a satisfactory, extensible approach. As part of the design of the new language, I will be considering ways that would permit the expression of such two-dimensional relationships.

From work over the last year it was also seen that the OMR problem would be significantly simplified if all the text and other isolated, fixed sized shapes were recognised early on in the recognition process and removed. Recognising and removing text before the main OMR process starts would also free the staff locations strategy from its current y constraints (see Section 3.2). Even if circumstances did not permit this work to be implemented, it would be possible to remove text by hand, so the results of musical recognition could be observed unaffected by other issues.

Music contains redundant information. For example, the duration of notes in a measure and the time signature for the staff duplicate information. Another example is the position of the key signature which reaffirms the clef of the staff. Currently such contextual information is only utilised if there is a problem or conflict in the system. For example 'bar line' synchronisation was introduced to improve audio play back, and the previously mentioned key signature up-grading addition was used to resolve a matching conflict. A more systematic tabulation of this 'redundant' information is sought. It is planned to use this information as a re-conformation mechanism, to increase the confidence of the matching strategy. Ideally this information should be expressible in the new language.

It is now seen that the technique of matching isolated graphical shapes is problematic. Complications arise when two different musical features touch or if a thin part of the musical feature is broken. Breaking a page of music down into primitive shapes (note heads, stems, tails etc) is thought to be more beneficial. Work can be carried out at a more uniform level.

The next immediate planned step is to build a database of all the constituent parts found in music. This data resource can then be used to obtain statistically based claims, rather then statements based on intuition and feelings. The database could answer queries such as what was the steepest angle for a beam or which shapes vary too much in style and thus prevent reliable template matching.

To conclude, I am reasonable pleased with the work completed, thought the unforgiving nature of OMR problem is at times frustrating.

References

[AC82] A. Andronico and A. Ciampa. On automatic pattern recognition and acquistion of printed music. In *Proceedings of the International Computer Music Conference*, pages 245–278, Venice, Italy, 1982. [ETHICS 735 988, OMR3] (*).

[APFB88] B. Alphonce, B. Pennycook, I. Fujinaga, and N. Boisvert. Optical music recognition: A progress report. In *Proceedings of the Small Computers in the Arts*, pages 8–12, 1988. (*).

[AT82] H. Aoyama and A. Tojo. Automatic recognition of printed music. TG PREL82-5, Institute of Electronics and Communications Engineers of Japan, 1982. (japanese; 1-page english version [TA82]).

[BAGS92] A. Bulis, R. Almog, M. Gerner, and U. Shimony. Computerized recognition of hand-written musical notes. In *Proceedings of the International Computer Music Conference*, pages 110–112, San Jose, 1992. [OMR1].

[Bai91] David Bainbridge. *Preliminary experiments in musical score recognition.* BSc thesis, Dpt of Computer Science, Univ. of Edinburgh, The Kings Buildings, Mayfield Road, Edinburgh, GB, June 1991. Undergraduate project, University of Edinburgh (*).

[Ban88] S. Bandyopadhyaya. *Techiniques of Sitar (The Prince Among All Musical Instruments of India).* B.R. Publishing Corporation, 1988. MT 649.B214 (*).

[BB92] Dorothea Blostein and Henry S. Baird. A critical survey of music image analysis. In H. S. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis.* Springer, 1992. [OMR3] (*).

[BD92] S. Baumann and A. Dengel. Transforming printed piano music into midi. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition (Proceedings of International Workshop on Structural and Syntactic Pattern Recognition, Bern, CH)*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 363–372. World Scientific, 1992.

[BH91] Dorothea Blostein and Lippold Haken. Justification of printed music. *Communications of the ACM*, 34(3):88 – 99, March 1991. (*).

[BHY92] H. Baird H.S, Bunke and K. Yamamoto, editors. *Structured Document Image Analysis.* Springer-Verlag, 1992. (*).

[Bri83] A. P. Brinkman. A design for a single-pass scanner for the darms coding language. In *Proceedings of the International Computer Music Conference*, pages 7–30, Rochester, New York, 1983.

[BT88] R.D. Boyle and R.C. Thomas. *Computer Vision: A First Course*. Artificial Inteligence Texts. Blackwell Scientific Publications, 1988. (*).

[Car89] N. P. Carter. *Automatic Recognition of Printed Music in the Context of Electronic Publishing*. PhD thesis, University of Surrey, February 1989.

[Car92a] Nicholas P. Carter. A new edition of walton's façade using automatic score recognition. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition (Proceedings of International Workshop on Structural and Syntactic Pattern Recognition, Bern, CH)*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 352–362. World Scientific, 1992.

[Car92b] Nicholas P. Carter. Segmentation and preliminary recognition of madrigals notated in white mensural notation. *Machine Vision and Applications*, 5(3):223–230, 1992. [OMR3].

[CB90] N. P. Carter and R. A. Bacon. Automatic recognition of music notation. In *Proceedings of the International Association for Pattern Recognition Worskshop on Syntactic and Structural Pattern Recognition*, page 482 ff., Murray Hill, NJ, June 1990. (*).

[CB91] Nicholas P. Carter and Richard A. Bacon. Automatic recognition of printed music. Dept. of Physics, University of Surrey, GB. Preprint of an article, 1991. [OMR2] (*).

[CBM88] N. P. Carter, R. A. Bacon, and T. Messenger. The acquisition, representation and reconstruction of printed music by computer: A survey. *Computers and the Humanities*, 22:117 ff., 1988. [OMR1] (*).

[CBT88a] A. T. Clarke, B. M. Brown, and M. P. Thorne. Inexpensive optical character recogniton of music notation: A new alternative for publishers. In *Proceedings of the Computers in Music Research Conference*, page 84 ff., Bailrigg, Lancaster, April 1988. (*).

[CBT88b] A. T. Clarke, B. M. Brown, and M. P. Thorne. Using a micro to automate data acquisition in music publishing. *Microprocessing and Microprogramming*, 24:549–554, 1988. (*).

[CBT89] A. T. Clarke, B. M. Brown, and M. P. Thorne. Coping with some really rotten problems in automatic music recognition. *Microprocessing and Microprogramming*, 27:547–550, 1989. (*).

[CBT90] A. T. Clarke, B. M. Brown, and M. P. Thorne. Problems to be faced by developers of computer based automatic music recognisers. In *Proceedings of the International Computer Music Conference*, Glasgow, September 1990. (*).

[Cop91] David Cope. *Computers and Musical Style*, volume 6 of *The Computer Music and Digital Audio Series*. A-R Editions, Inc., 1991. MT 723.C782 (*).

[CYD93] Chen R. Chen Y.L. and Hung D.C. The segmentation of analysis of handwritten musical notes. Department of Computer and Information Science, New Jersey Institute of Technology (*), 1993.

[Eri75] Raymond F. Erickson. "the darms project": A status report. *Computers and the Humanities*, 9:pp 291 – 298, 1975. (*).

[F+85] Yoichi Fujimoto et al. The keyboard playing robot WABOT-2. *Bulletin of Science and Engineering Research Laboratory*, 112, 1985. [ETHICS P410 752:112] (*).

[FAP89] Ichiro Fujinaga, Bo Alphonce, and Bruce Pennycook. Issues in the design of an optical music recognition system. In *Proceedings of the International Computer Music Conference*, pages 113–116, Ohio State University, November 1989. (*).

[FAPB89] Ichiro Fujinaga, Bo Alphonce, Bruce Pennycook, and Natalie Boisvert. Optical recognition of music notation by computer. *Computers in Music Research*, 1:161–164, 1989. (*).

[FAPD92] Ichiro Fujinaga, B. Alphonce, B. Pennycook, and G. Diener. Interactive optical music recognition. In *Proceedings of the International Computer Music Conference*, pages 117–120, San Jose, 1992. [OMR2] (*).

[FAPH91] Ichiro Fujinaga, Bo Alphonce, Bruce Pennycook, and K. Hogan. Optical music recognition: Progress report. In *Proceedings of the International Computer Music Conference*, pages 66–73, Montreal, 1991. (*).

[FB91] H. Fahmy and D. Blostein. A graph grammar for high-level recognition of music notation. In *Proceedings of First International Conference on Document Analysis*, volume 1, pages 70–78, Saint-Malo, France, 1991. [OMR1] (*).

[FB92] H. Fahmy and D. Blostein. Graph grammar processing of uncertain data. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition (Proceedings of International Workshop on Structural and Syntactic Pattern Recognition, Bern, CH)*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 373–382. World Scientific, 1992.

[FJDF90] Feiner Steven K. Foley James D., van Dam Andries and Hughes John F. *Computer Graphics: Principles and Practice.* Addison-Wesley Publishing Company, second edition edition, 1990.

[FPA89] Ichiro Fujinaga, Bruce Pennycook, and Bo Alphonce. Computer recognition of musical notation. In *Proceedings of the First International Conference on Music Perception and Cognition*, pages 87–90, 1989. (*).

[FPA91] Ichiro Fujinaga, Bruce Pennycook, and Bo Alphonce. The optical music recognition project. *Computers in Music Research*, 3:139–142, 1991. (*).

[Fuj88] Ichiro Fujinaga. Optical music recognition using projections. Master's thesis, McGill University, Montreal, CA, 1988. (*).

[Fuj92] Ichiro Fujinaga. An optical music recognition system that learns. In Jacek Maitan, editor, *Enabling Technologies for High-Bandwidth Applications*, pages 210–217, SPIE 1785, 1992.

[Gla89] S. Glass. Optical music recognition. Undergraduate project report, University of Canterbury, New Zealand, 1989. (*).

[GP87] Rafael C. Gonzalez and Wintz P. *Digital Image Processing.* Addison-Wesley Publishing Company, 1987. (EL) TA 1632.G643 1987 (*).

[GW92] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1992. (EL) TA 1632.G643 1992 (*).

[Hay92] Joseph Haydn. *Werke (Works), Reihe II, Concertante.* G. Henle Verlag, 1792. qM 3.H 415 Ser. 2.

[Heu87] George Heussenstamm. *The Norton Manual of Music Notation.* W. W. Norton and Company, 1987. MT 35.H595 (*).

[Hin49] Paul Hindemith. *Elementary Training for Musicians.* Associated Music Publishers, Inc., second edition, 1949. MT 35.H662 (*).

[HO87] K. Hachimura and Y. Ohno. A system for the representation of human body movements from dance scores. *Pattern Recognition Letters*, 5:1–9, January 1987.

[IHIO90] Takebumi Itagaki, Shuji Hashimoto, Masayuki Isogai, and Sadamu Ohteru. Automatic recognition on some different types of musical notation. In *Proceedings of the International Association for Pattern Recognition Worskshop on Syntactic and Structural Pattern Recognition*, page 488 ff., Murray Hill, NJ, June 1990. (*).

[Jai86] Anil K. Jain. *Fundamentals of Digital Image Processing.* Information and System Sciences. Prentice Hall, 1986. (EL) TA 1632.J25 (*).

[Jal89] François Jalbert. *MuTeX User's Guide*, version 1.1 edition, 1989. (*).

[Joh91] Margaret L. Johnson. Toward an expert system for expressive musical performance. *Computer*, pages pp 30 – 34, July 1991. (*).

[Kas70] M. Kassler. An essay toward specification of a music-reading machine. In B. S. Brook, editor, *Musicology and the Computer.* CUNY Press, New York, 1970. (*).

[Kas72] M. Kassler. Optical character recognition of printed music: A review of two dissertations. *Perspectives of New Music*, 11(2):250–254, 1972. (*).

[KG90] Kent Kennan and Donald Grantham. *The Technique of Orchestration*. Prentice Hall, fourth edition, 1990. MT 70.K34 1990 (*).

[KI88] Hirokazu Kato and Seiji Inokuchi. Automatic recognition of printed piano music based on bar unit processing. *Transactions of I.E.C.E. J71-D*, 5:894–901, 1988. (japanese).

[KI90] Hirokazu Kato and Seiji Inokuchi. The recogntion system for printed piano music using musical knowledge and constraints. In *Proceedings of the International Association for Pattern Recognition Worskshop on Syntactic and Structural Pattern Recognition*, pages 231–248, Murray Hill, NJ, June 1990. [OMR1] (*).

[KKII89] H. Katayose, H. Kato, M. Imai, and S. Inokuchi. An approach to an artificial music expert. In *Proceedings of the International Computer Music Conference*, pages 139–146, Ohio State University, November 1989. (*).

[KKII90] H. Katayose, H. Kato, M. Imai, and S. Inokuchi. Expression extraction in virtuoso music performances. In *Proceedings of the Tenth International Conference on Pattern Recognition*, pages 780–784, Atlantic City, New Jersey, June 1990. (*).

[Kli73a] Vernon L. Kliewer. *Music Reading: A Comprehensive Approach*, volume 1. Prentice-Hall, Inc., 1973. MT 35.K65 vol 1 (*).

[Kli73b] Vernon L. Kliewer. *Music Reading: A Comprehensive Approach*, volume 2. Prentice-Hall, Inc., 1973. MT 35.K65 vol 1 (*).

[Kwo88] Paul C. K. Kwok. A thinning algorithm by contour generation. *Communications of the ACM*, 31(11):pp 1314 – 1324, November 1988. (*).

[LC91a] I. Leplumey and J. Camillerapp. Comparison of region labelling for musical scores. In *Proceedings of First International Conference on Document Analysis*, volume 2, pages 674–682, Saint-Malo, France, 1991. [OMR1, faint!] (*).

[LC91b] Ivan Leplumey and Jean Camillerapp. Coopération entre la segmentation des régions blanches et des régions noires pour l'analyse de partitions musicales. In *AFCET, 8e Congress Reconnaissance des Formes et Intelligence Artificielle, Vol. 3*, pages 1045–1052, Lyon - Villeurbanne, France, November 1991. (french) [OMR2].

[Lie83] Fredric Lieberman. *A Chinese Zither Tutor*. University of Washington Press, 1983. MT 654.C5.M499 (*).

[Lip89] Stanley B. Lippman. *C++ Primer*. Addison-Wesley Publishing Company, 1989. (PSL) QA 76.73.C015.L766 (*).

[Mah82] J. V. Mahoney. Automatic analysis of musical score images. BSc thesis, Department of Computer Science and Engineering, MIT, Cambridge, May 1982.

[Mar87] Neil G. Martin. Towards computer recogniton of the printed musical score. B.Sc. project report, Thames Polytechnic, Woolwich, London, May 1987.

[Mar89] Philippe Martin. Reconnaissance de partitions musicales et réseaux de neurones: une etude. In *Actes 7ème Congrès AFCET de Reconnaissance des Formes et Intelligence Artificielle*, pages 217–226, Paris, 1989. (french), [OMR1].

[Mat85] Toshiaki Matsushima. Automated high speed recognition of printed music (WABOT-2 vision system). In *Proceedings of the 1985 International Conference on Advances Robotics*, page 477 ff., Japan Industrial Robot Association (JIRA), Shiba Koen Minato-ku, Tokyo, 1985.

[Mat88] Toshiaki Matsushima. Automatic printed-music-to-braille translation system. *Journal of Information Processing*, 14(4):249–257, 1988. (*).

[MB91a] P. Martin and C. Bellissant. Low-level analysis of music drawing images. In *Proceedings of First International Conference on Document Analysis*, volume 1, pages 417–425, Saint-Malo, France, 1991.

[MB91b] P. Martin and C. Bellissant. Neural networks at different levels of a musical score image analysis system. In *Proceedings of 7th Scandinavian Conference on Image Analysis*, pages 1102–1109, Aalborg, Denmark, 1991. (*).

[McL92] Graeme I. McLean. Music recognition. BSc thesis, Dpt. of Computer Science, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, GB, 1992. [OMR2] (*).

[MHS+85] T. Matsushima, T. Harada, I. Sonomoto, K.Kanamori, A. Uesugi, Y. Nimura, S. Hashimoto, and S. Ohteru. Automated recognition system for musical score - the vision system of WABOT-2. *Bulletin of Science and Engineering Research Laboratory, Waseda University*, 112:25–52, September 1985. [OMR3] (*).

[MM89] W. F. McGee and Paul Merkley. Optical recognition of music using page straightening. Abstract of paper submitted for the International Computer Music Conferecne, 1989.

[MM91] W. F. McGee and Paul Merkley. The optical scanning of medievel music. *Computers and the Humanities*, 25(1):47–53, 1991. (*).

[MRHS92] Bharath R. Modayur, Visvanathan Ramesh, Robert M. Haralick, and Linda G. Shapiro. Muser - a prototype musical score recognition system using mathematical morphology. Intelligent Systems Laboratory, EE Dept, FT-10, University of Washington, Seattle WA 98195, June 1992. [OMR2] (*).

[NP73] G. Nelson and T. R. Penney. Pattern recognition in musical score - project no M88. *Computers and the Humanities*, 8:50–51, 1973. (*).

[NSI78] Y. Nakamura, M. Shindo, and S. Inokuchi. Input method of [musical] note and realization of folk music data-base. Institute of Electronics and Communications Engineers of Japan (IECE), 5-8 Sh, 1978. (japanese).

[O+84] S. Ohteru et al. A multi processor system for high speed recogniton of printed music. NATL.CONV. (Rec.), IECE, JAPAN, 1984. (japanese?).

[OIT79] M. Onoe, M. Ishizuka, and K. Tsuboi. Experiment on automatic music reading. In *Proceedings of 20th IPSJ National Conference 6F-5*, 1979. (japanese).

[OM85] S. Ohteru and T. Matsushima. Automatic recognition of printed music. *Japan Acoustics Society Journal*, 41(6), 1985. (japanese).

[Ost88] Berge Ostenstad. Oppdeling av abjektene i et digitalt notebilde i klassifiserbare enheter. Institute of Informatics, P.O.Box 1080 Blindern, N-0316 Oslo 3, Norway, 1988. (norwegian).

[Pen90] Bruce Pennycook. Towards advanced optical music recognition. *Advance Imaging*, pages 54–57, April 1990.

[Pre70] D. S. Prerau. *Computer Pattern Recognition of Standard Engraved Music Notation*. PhD thesis, MIT, September 1970.

[Pre71] D. S. Prerau. Computer pattern recognition of printed music. In *Proceedings of the Fall Joint Computer Conference*, Montvale, NJ 39, November 1971. AFIPS Press. (*).

[Pre75] D. S. Prerau. Do-re-mi: A program that recognizes music notation. *Computer and the Humanites*, 9(1):25–29, January 1975. (*).

[Pru66] D. Pruslin. *Automatic Recognition of Sheet Music*. ScD dissertation, MIT, June 1966.

[Rea74] Gardner Read. *Music Notation: A Manual of Modern Practice*. Victor Gollanacz Ltd, 1974. MT 35.R282 (*).

[Roa86] Curtis Roads. The tsukuba musical robot. *Computer Music Journal*, 10(2):39–43, 1986. (*).

[Ros70] Ted Ross. *The Art of Music Engraving and Processing*. Hansen Press, 1970. z6810.r821 (*).

[Rot92] Martin Roth. OMR - optical music recognition. diploma thesis, Swiss Federal Institute of Technology, Institute for theoretical computer science, ETH Zürich, CH-8092 Züric, October 1992. (german, 1 page english abstract) [DA] (* abstract).

[RT88] J. W. Roach and J. E. Tatem. Using domain knowledge in low-level visual processing to interpret handwritten music: an experiment. *Pattern Recognition*, 21(1):33–44, 1988. [ETZ, OMR2] (*).

[Rum90] Francis Rumsey. *MIDI Systems and Control.* Focal Press, 1990. MT 723.R938 (*).

[Rus92] C. Russ, John. *The Image Processing Handbook.* CRC Press, 1992. (EL) qTA 1632.R958 (*).

[Rut91] Alan Ruttenberg. Optical reading of typeset music. Master's thesis, MIT, February 1991. [OMR3] (*).

[SCEC89] S.M. Stratford C. Embury and Rowett C.J.C. The well tempered scanner. Honours thesis, University of Kent, 1989. (*).

[SHM+85] I. Sonomoto, T. Harada, T. Matsushima, K. Kanamori, M. Konuma, A. Uesugi, Y. Nimura, S. Hashimoto, and S. Ohteru. Automated recognition system of printed music for playing keyboards. TG MA84-22, pp.17-22, Acoustical Society of Japan, Ikeda Bldg., 77- Yoyogi,, 1985. (japanese).

[Sic92] Etienne Sicard. An efficient method for the recognition of printed music. In *Poceedings of 11th International Conference on Pattern Recognition (IAPR) [?]*, pages 573–576, Den Haag (Netherlands), 1992. [OMR2] (*).

[Sma91] Alan Smail. Basic knowledge representation ideas. *Lecture Notes*, pages 13 – 23, 1991. (*).

[TA82] A. Tojo and H. Aoyama. Automatic recognition of music score. In *6th International Conference on Pattern Recogniton*, page 1223 ff., Munich, Germany, 1982. (english abstr. of [AT82]), [OMR1].

[Tho88] Elise Thorud. Analyse av notebilder. Institute of Informatics, P.O.Box 1080 Blindern, N-0316 Oslo 3, Norway, August 1988. (norwegian).

[Ton86] S. Tonnesland. Symfoni: System for notekoding. Institute of Informatics, P.O.Box 1080 Blindern, N-0316 Oslo 3, Norway, November 1986. (norwegian).

[Viv82a] Antonio Vivaldi. *The Autumn (L'Autunno).* Concerto for Violin, Strings and Basso continuo (F Major) Op 8/3 (P257). Ernst Eulenburg Ltd., 1982. min score (*).

[Viv82b] Antonio Vivaldi. *The Autumn (L'Autunno).* Concerto for Violin, Strings and Basso continuo (F Major) Op 8/3 (P257). Ernst Eulenburg Ltd., 1982. min score (*).

[Viv82c] Antonio Vivaldi. *The Spring (La Primavera).* Concerto for Violin, Strings and Basso continuo (E Major) Op 8/1 (P241). Ernst Eulenburg Ltd., 1982. min score (*).

[Viv82d] Antonio Vivaldi. *The Summer (L'Estate).* Concerto for Violin, Strings and Basso continuo (G Minor) Op 8/2 (P336). Ernst Eulenburg Ltd., 1982. min score (*).

[WCAK92] J. Wolman, J. Choi, S. Asgharzadeh, and J. Kahana. Recognition of handwritten music notation. In *Proceedings of the International Computer Music Conference*, pages 125–127, San Jose, 1992. [OMR1].

[Wit73] G. Wittlich. Project SCORE. *Computational Musicology Newsletter*, 1(1):6, 1973. abstract of paper from International Conference on Computers in the Humanities, University of Minnesota.

[Wit74] G. E. Wittlich. Non-physics measurements on the PEPR system: Seismograms and music scores. In *Report to the Oxford Conference on Computer Scanning*, page 487 ff., Oxford Nuclear Physics Laboratory, 1974.

[WLSC85] Myung Woo Lee and Jong Soo Choi. The recogniton of printed music score and performance using computer vision system. *Journal of the Korean Institute of Electronc Engineers*, 22(5):429–435, September 1985. (korean and english translation).

[Wol77] Anthony B. Wolff. Problems of representation in musical compution. *Computers and the Humanities*, 11:pp 3 – 12, 1977. (*).

[YPBD+92] O. Yadid-Pecht, E. Brutman, L. Dvir, M. Gerner, and U. Shimony. Ramit: Neural network for recognition of musical notes. In *Proceedings of the International Computer Music Conference*, pages 128–131, San Jose, 1992. [OMR1].