Overview of Algorithms and Techniques for Optical Music Recognition

Quratul-Ain Arshad , Wazir Zada Khan, Zohair Ihsan

(brightsuccess_12, wazirzadakhan, zohairihsan}@hotmail.com
Department of Computer Science
COMSATS Institute of Information Technology, Wah Cantt, Pakitan

Abstract:
The systems for music score recognition are traditionally called OMR. In this paper we have presented an overview of different techniques and algorithms which can be used at various steps of music symbol recognition. In the very first, we have given some background for music. Then the details of the four basic steps usually used by OMR systems and then detailed analysis of the techniques used for Optical Music Recognition. OMR can be of two types i.e. recognition of printed music and handwritten music symbols. This paper addresses techniques that are used for recognition of printed music symbols.

1. INTRODUCTION:

The area of Optical Music Recognition (OMR) has long been overwhelmed by an inability to provide a definitive method for identifying and locating musical objects, which are superimposed on musical stave lines.

Optical Music Recognition (OMR) is form of structured document image analysis in which music symbols overlaid on the conventional five-line stave are isolated and identified so that music can be played through a MIDI system, or edited in a music publishing system. Traditionally OMR systems have had recognition techniques hard coded in software.

Optical Music Recognition (OMR) Systems allow people to scan printed music onto a computer and have it played by the computer, or brought up in music editor. OMR has many applications. For Example, a vocalist can practice with a computer playing an accompaniment, a musician can have a piece of music transposed to another key automatically; the notation of the music could be converted (for example, to tablature or Braille); or an old edition of the music could be re-edited.

Despite to the availability of several commercial OMRs including PIANISCAN, NOTESCAN, MIDISCAN, PhotoScore, SmartScore, SharpEye etc, none of these is satisfactory in term of precision and reliability.

In the recent years, OMR systems have also become available. Systems are either pure OMR ayatems (e.g MIDISCAN, Capella-scan) or part of a larger music editing system (e.g Finale). Common to all these systems is that they are designed to recognize printed sheet music scores where the contrast between music symbols and background is good, staff lines are straight, and symbols are printed clearly. However, they fail at the task of recognizing historic handwritten music scores because of degraded paper and ink, bent staff lines, notations that vary from writer to writer, symbols shinning through from the reverse page and so on [7].

The term OMR is tightly linked to OCR (Optical Character Recognition).Typically OCR techniques cannot be used in music score recognition. It is related to OCR but there comes some important extra challenges.One important difference is that music is two-dimentional (pitch vertically and time horizontally), while text is one-dimentional. Another significant complication in music is that the symbols are ussually overlayed on a five line stave, so its difficult to isolate symbols. Also unlike OCR, symbols are
made up of components that can be combined in many ways.

2. MUSICAL BACKGROUND:

There are some musical terms and symbols that we should be familiar with to better understand the Recognition of Musical symbols. In order to get familiar with them we have first given a prpoer nomenclature for musical terms.

2.1 *Staff:*

A staff is made up of five horizontal lines and four spaces. The height of a note on the page determines its pitch. Because people are unable to accurately judge relative heights over great distances, lines are drawn on the page to show the distinctions between heights. It is standard to draw 5 lines and they are called the *staff* or *stave* [18].

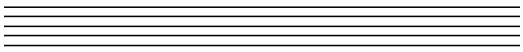


Fig 1: The Musical Staff (Five parallel, equidistant lines with spaces in between)

2.2 Pitches:

Pitches are named after the first seven letters of the alphabet (A B C D E F G) [18].

Fig 2: The positions of alphabets on a piano

2.3 Sounds in music:

A sound in music usually has a definite *pitch* that is described as "high" or "low". A musical sound, called a *tone,* is produced when something causes a series of vibrations that recur a certain number of times each second. For example, heavy wires that vibrate slowly, only 32.7 times a second, produce the lowest C on the piano. The thin wires that produce the highest C on the piano vibrate more than 4,000 times a second, or almost 130 times as fast as the lowest [19].

2.4 Notes and rests:

Musical tones also have other characteristics. For example, some tones are long and some short. This is called *duration* [19]. Notes are a double encoding of pitch and the duration for which that pitches should be played. Note duration is indicated by the number of flags a note has.
Rests indicate silence or no sound. They have various shapes, and have the same time values or duration as the notes they replace [19].

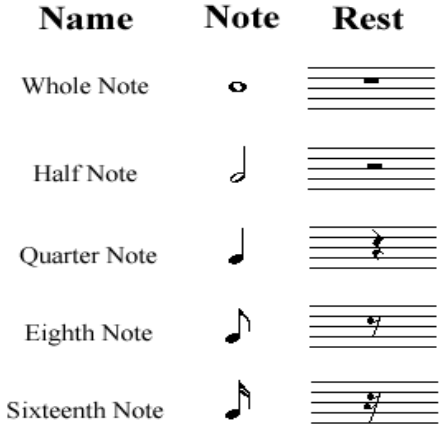


Fig 3: Notes and Rests.

2.5 Clefs:

Note pitch is heavily tied to the idea of clefs. The staff alone does not indicate which lines correspond to which pitches. A *clef* is used to indicate which lines correspond to which pitches. Clefs are symbols placed at the beginning of the staff that defines a pitch on a line [18].
There are two types: [20].

- Treble clef (G clef)
- Bass clef (F clef)

Fig 4: TREBLE Clef and BASS Clef

2.6 Bar lines:

Bar lines provide separation of measures. These symbols are simply vertical lines that stretch the height of the staff. Because of their shape, they resemble the stems of notes.

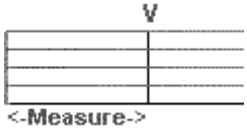


Fig 5: A vertical line placed in a STAFF to mark off MEASURES

2.7 BEAMS and SLURS:
The beam or hitch links two or more notes together, in order to build a rhythmic or melodic entity [21].

Fig 6: Beam

A slur is located between two notes of any pitch. If it is played, it behaves as a glissendo: the note pitch will vary smoothly or semitone by semitone from the first pitch to the second one. A slur can also be used, when underlining a whole group of notes, to specify "phrasing", i.e. indicate this group of notes has to be played in a single "sentence" (in a single blow for a wind instrument). A slur can be inserted between two notes belonging to different staves.

Fig 7: Slur

2.8 Accidentals:

The composer may place an *accidental* in front of a certain note. *Accidentals* are the signs for *sharp, flat, or natural* that shows a change from the key signature. A *key signature* appears at the right of the clef sign. By using sharp signs or flat signs, the composer indicates that certain notes should always be played sharp or flat. Any note not marked with sharp or flat is called *natural*.
Sharp (#): A symbol that raises a note by one half-step.
Flat (b): A symbol that lowers a note by one half-step.
Natural: A symbol that removes the effect of a sharp or flat sign, so that the note represents a pitch that is not, sharp or flat.

2.9 Scores:

Scores contain music written for several instruments, or for instruments and voices [19].

3. STEPS FOR RECONIZING MUSIC SYMBOLS:

OMR systems generally have the following four stages in their recognition process. [1]
These steps are shown in the Fig 8.

Stave line identification
The position of the staves is identified, and (usually) the lines are removed, leaving the superimposed musical symbols.

Musical object location
The symbols that were on and around the stave are located.

Symbol identification
The type of each symbol is determined.

Semantics of music notation
The relationship between symbols is determined and the information is stored in a form that programs such as sequencers or music editors can use.

4. TECHNIQUES USED AT VRIOUS STEPS OF RECOGNIZING MUSIC SYMBOLS:

The Fig: 9 shows a tree of techniques or algorithms that can be used for varous steps followed by OMR systems in order to recognize the music symbols and to interpret an image score. Table 1 shows time complexities of some of the techniques.

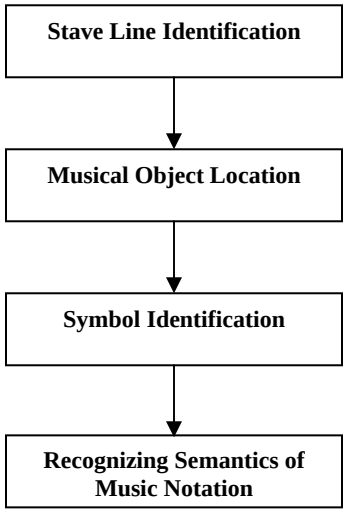


Fig: 8 Steps of OMR Systems

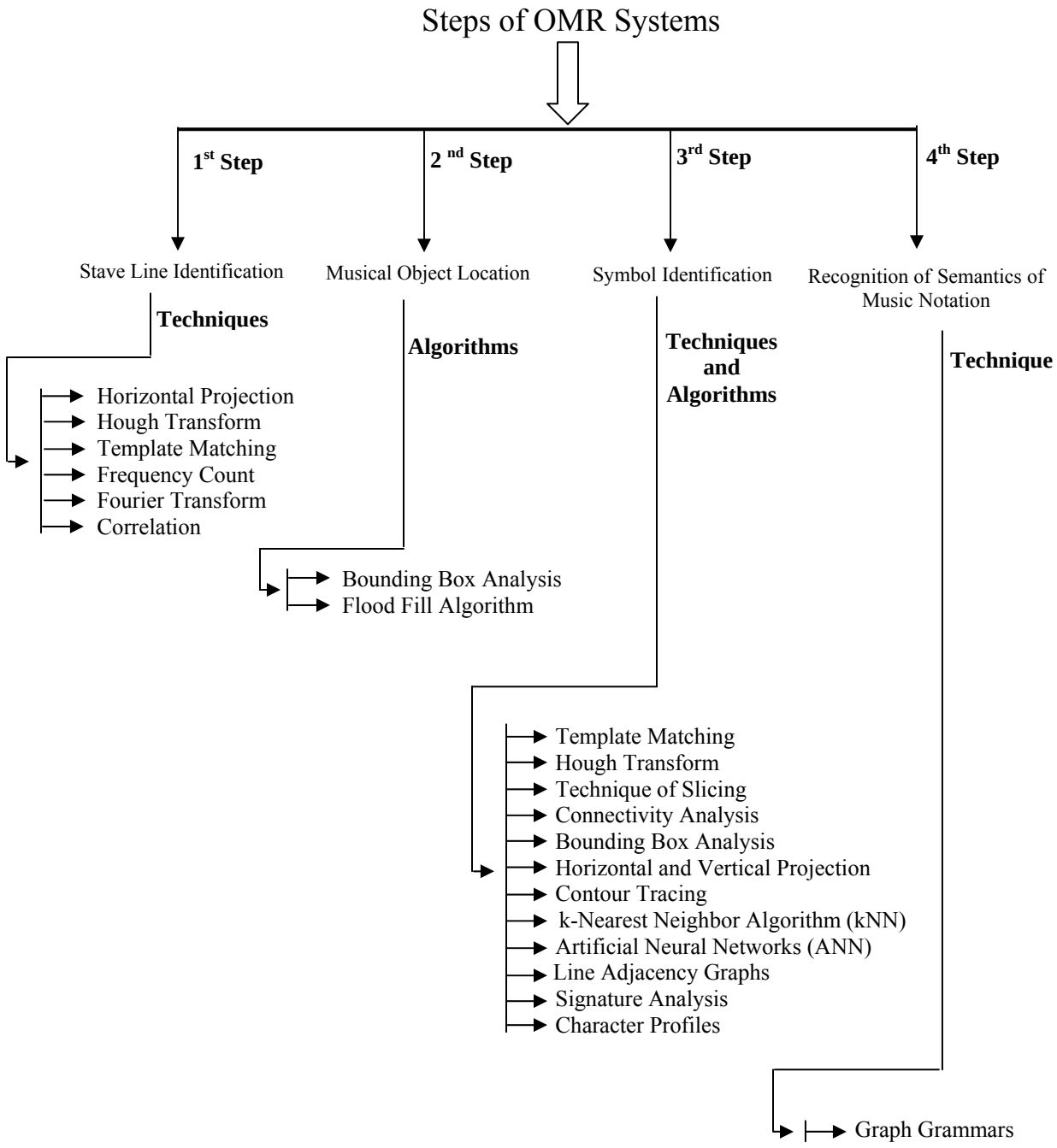


Fig: 9 Techniques and Algorithms used at various steps of printed music symbol recognition process

Techniques and Algorithms	**Time complexities and accuracies**
Hough Transform	Not robust for staff line identification Computationally expensive with time complexity of about O(n3)
Frequency Count Algorithm	Processor Time in identifying Staff lines is20msec It is proved almost as reliable as the correlation method but only fraction of the processing time
Fourier Transform	Processor Time in identifying staff lines is 140msec
Correlation	Processor Time in identifying staff lines is 26612msec
K-Nearest Neighbor	Time complexity id about O(log(n))

Table :1 Accuracies and time complexities of the algorithms and Techniques

5. DETAILED ANALYSIS OF TECHNIQUES USED AT VRIOUS STEPS OF RECOGNIZING PRINTED MUSIC SYMBOLS:

5.1 Identifying the Stave Lines:

The first stage in the process of recognizing musical symbols in OMR is identifying where the stave lines are. It is because the data they provide is essential for understanding the pitch of the notes. Identifying stave lines first provides the additional benefit of providing a measure of the width between stave lines, which is useful for determining the scale of the musical score. For example, note heads are almost always 1 stave line widths high, so finding this height before trying to identify notes is very important.

5.1.1 Horizontal Projection:

The most common method for identifying stave lines is *Horizontal Projection.* The horizontal prtojection takes a sheet of music, and counts the number of black pixels along each row of the image.This method shows 5 distinct peaks in the number of black pixels for each staff, identifying the positions of each stave line of that staff.

This technique is used in the project described in the paper by Scott Sheridan, Susan E. George [2]. They have described that as to one of the pioneers of the OMR field noted, the 5 stave lines are not exactly parallel, horizontal, equidistant, of constant thickness, or even straight. To help cope with this, scores are broken into smaller portions and the stave lines from each portion's beginning and end are joined after their identification.

In the paper written by David Bainbridgr and Tim Bell [1], the same technique is used by the CANTOR system described in this paper. According to them detection of stave lines is usually staright forward because the stave lines are very regular feature on the page and can be found relably by taking a horizontal projection of the number of black pixels on each line of the image. Stave lines are marked by peaks in the histogram of the projection.

As the first step in the process of recognizing musical symbols in OMR has previously been to either remove the stave lines, or ignore them. But According to Scott Sheridan, Susan E. Georg [2], removing stave lines leads to many problems of fragmented and deformed musical symbols, or in the case of ignoring them, a lowered chance of recognition. Most OMR systems attempt to correct these deficiencies later on in the process through many varied approaches including bounding box analysis, k-nearest-neighbor (k-NN) and neural network (ANN) classification schemes. All of these have a level of success, but none have provided nearly the desired level of accuracy.

The paper by Sheridan, Susan E. Georgem [2], aims to show that this remaoval of the stave lines before symbol recognition is not the only first step and may not be the best. Instead of removing stave lines, more should be added. This process is called 'defacing' since it adds stave lines to the score at a 1/2 stave line width, and actually overwrites the score , apparently complicating the recognition procedure. However, the addition of signal to the image means that subsequent symbol recognition is 'normalized' and a musical symbol will look the same whether it was above, below or on a stave line. As a result of this, a classification system trained with double stave lines should provide a higher level of accuracy than the traditional approaches of removing/ignoring the stave lines [2].

The technique of Horizontal Projection is also described by **R.** Randriamahefa*, J.P. Cocquerez**, C. Fluhr***, F. PCpin**, ***S.*** Philipp** [3]. They have explained that the most straight forward method to detect these lines is the horizontal projection. These lines appear as the projection maxima. This technique supposes that these lines are perfectly horizontal, unbroken and parallel, whereas they are skewed, split and present a slight curvature. These defects appear either at the image acquisition (slope of the score sheet) or at the score edition. Thin horizontal lines of perfect image disappear at the image acquisition

5.1.2 Hough Transform:

The Hough Transform has been developed by Paul Hough in 1962 and patented by IBM [8]. It became in the last decade a standard tool in the domain of artificial vision for the recognition of straight lines, circles and ellipses. The Hough Transform is particularly robust to missing and contaminated data. It can also be extended to non-linear characteristic relations and made resistant to noise by use of anti-aliasing techniques.

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the *classical* Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, *etc.* A *generalized* Hough transform can be employed in applications where a simple analytic description of a feature(s) is not possible.

Pros and Cons of Hough Transform Technique:
The Hough Transform can detect lines or curves that are very broken (after initial edge detection, for example). The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

The disadvantage of this technique is that Hough Transforms can only detect lines or curves that analytically specifiable, or that can be represented in a template-like form (GHT, Ballard).

Some defects are mentioned by **R. Randriamahefa**, **J.P. Cocquerez, C. Fluhr, F**. **PCpin, S. Philipp** in their paper [3]. Therefore, they have stated another technique taking into account the defects may be designed i.e. the Hough Transform. According to them using the Hough Transform on little part of image (to limit processing time) gives a satisfactory result for perfect scores but a bad result is obtained for slanted scores. When it was tested, the result for bowed or fragmented lines was not satisfactory.

Yun-Chung Chung and Greg C Lee have chosen the Hough Transform method for detecting Stave lines as well as Bar Lines in their paper [4]. According to them it is chosen because it is not very noise sensitive. As the orientation of the staves has been determined, the staves can then be rotated back to the upright position if needed. The parameters of the line equation,

$$x \cos\theta + y \sin\theta = \rho$$

are θ the angle formed by the normal vector of the line direction and the x-axis, and ρ the perpendicular distance from the origin to the line.

They have stated that there is also a problem with Hough Transform. The problem is that it is very time consuming. A fine partition of θ and ρ increases the accuracy but at the expense of long running time, while coarse partitions of θ and ρ yields unacceptable θ and ρ values.

5.1.3 Template Matching:

This technique was originally developed by Walter O'Dell PhD, wodell at Rochester.edu [9].

Template is a prototype object created by taking the mean of a large number of samples and is a general representation of that given class of objects.

Templates are mostly used for identification of printed characters, numbers and other small objects, generally in the bi-level images.

The matching process moves the template over the image one pixel by one pixel and replaces that pixel with the matching index that indicates how well that part of the image matches with the template.

To count the number of object pixels that match between the template and the image minus the number of object pixels that disagree.

K.Todd Reed and **J.R.Parker** [5] have described a complete optical music recognition system (Lemon), which uses a technique of template matching for staff line detection. The writers have stated the algorithm for staff detection which uses the fact that there are five staff lines that arte equally spaced in every staff. The algorithm operates by examining vertical columns from the image and recording all *staff samples* - five equally spaced, equally sized run-lengths of black pixels (Figure 10) below. The staff samples are grouped based on their spacing, thickness, and vertical position on the page. Within each group, the angles between all of the samples are computed, yielding a median angle α. Then the angles between adjacent samples are compared; if the angles between a sample and its neighbors disagree wildly with α then the sample can be discarded. The remaining samples in each group are fitted to a straight line. When a staff line cannot be represented by a single line, the staff line is piecewise approximated by dividing the staff line until each division is linear, within a small tolerance

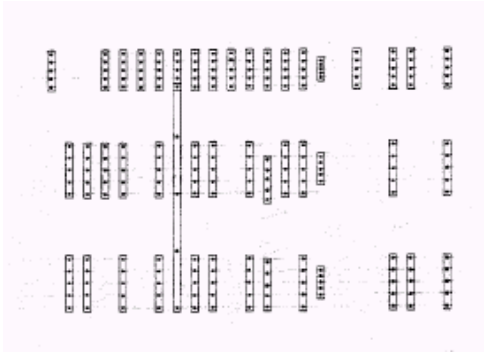


Fig 10: Staff samples used for detecting staff lines.

5.1.4 Frequency Count, Fourier Transform, Corelation:

Weighted Frequency Count (EFC) was introduced in 2001 by Sebastian Deorowicz [10].

Fourier Transform Algorithm was named after Joseph Fourier and is an Integral Transform [11].

The concept of Correlation was introduced by Francis Galton in 1888 [12].

According to David Bainbridge, detecting staff lines is a crucial step. In his Progress Report [5] he has stated that most of the work of this step relies upon projection algorithms to locate the staff lines. Common to all these algorithms is the problem of skew. His point of view is that surely some vertical scan based approach would be better suited to measuring the staff gap and would not have to avoid the problem of skew since this would be naturally incorporated. He devised three staff detection algorithms: Frequency Count, Fourier Transform and Corelation.

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 stave gap. The Fourier transform algorithm chose the largest frequency component as the frequency stave lines occurred. The measure was based on the magnitude of the complex frequency components. From the value chosen the stave gap could be computed. Finally the correlation based method checked a set of ideal stave line segments of increasing spacing against the given image. The best match was taken to be the stave gap.

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

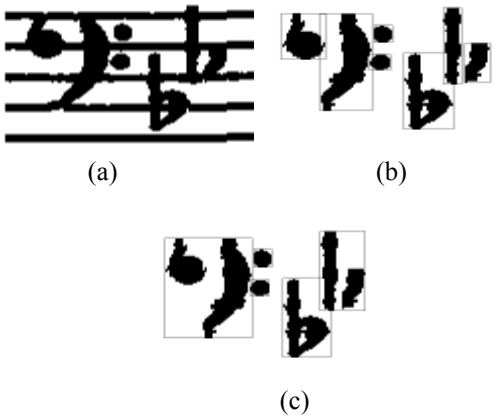
5.2 Loacting the Musical Objects:

This is the second step in OMR systems. In this step, after the staff line detection and then removal, the musical objects are located.

5.2.1 Bounding Box Analysis:

The bounding box is a rectangular box that encloses the solid model and fits exactly the model's upper and lower X, Y, and Z extents.

The outcome of stave line removal is the fragmented musical objects. **David Bainbridge** and **Tim Bell** have shown that one method of dealing with fragmented objects is to devise a heuristic that merges *Bounding Boxes* together by comparing adjacent bounding boxes in the vicinity of stave lines. It is illustrated by showing a bass clef that is broken up by stave removal (Fig 11a) but reconstructed by the joining of bounding boxes (Fig 11b). [1]



(a) (b)

(c)

Fig 11: Locating a bass clef (a) the original image (b) broken up by stave removal (c) reconstructed.

Scott Sheridan and **Susan E. George** have described that the historically and still the most prevalent first step in locating musical objects has been to remove the stave lines from the image, thus isolating the remaining shapes for identification. The method of removing the stave lines has drawbacks. The stave line removal results in a fragmented musical objects even with the best methods developed so far [2]. They have stated that one of the reasons that the stave lines were removed in the dawn of OMR was to help the primitive software and hardware locate the musical

objects using *Bounding Boxes. Bounding Boxes* required white space to determine where an object started and ended.

But they have further mentioned that this reason is no longer an issue, as more modern methods, even simple ones such as horizontal and vertical projections (the technique of counting how many black pixels in a row or column), can locate a musical object without having to remove the stave lines.

5.2.2 Flood Fill Algorithm:

Flood Fill Algorithm was developed by Bellman. This approach was developed from the idea of imagining water being poured into the maze [13].

Flood fill, also called **seed fill,** is an algorithm [22], that determines the area connected to a given node in a multi-dimensional array. It is used in the "bucket" fill tool of paint programs to determine which parts of a bitmap to fill with color, and in puzzle games such as Puyo Puyo, Lumines, Magical Drop, and some implementations of Tetris (but not Columns) for determining which pieces are cleared.

The flood fill algorithm takes three parameters: a start node, a target color, and a replacement color. The algorithm looks for all nodes in the array which are connected to the start node by a path of the target color, and changes them to the replacement color. There are many ways in which the flood-fill algorithm can be structured, but they all make use of a queue or stack data structure, explicitly or implicitly.

Implicit stack-based (recursive) flood-fill implementation (for a two-dimensional array) is esy to understand but but is impractical in languages and environments where stack space is severly constrained (e.g. Java applets).

Most practical implementations use a loop for the west and east directions as an optimization to avoid the overhead of queue management.

Flood Fill Algorithm can be used in OMR systems in order to locate the musical objects.

David Bainbridge and **Tim Bell** [1] have mentioned that the musical objects left after stave removal are loacted by searching for black pixels, and finding connected pixels using a *flood fill algorithm.*

5.3 Recognizing/Identifying the Musical Objects:

Scott Sheridan and **Susan E. George** have described that this is the step where the most variety comes into OMR. There have been many methods proposed for this step, such as *bounding box analysis*, *horizontal* and *vertical projection*, *template matching*, *contour tracing*, *k-nearest neighbor* and *Artificial Neural Networks* [2]. The writers have stated that the identification of musical objects is made difficult by the sheer number of ways that the note heads, stems, rests, holds, flags and all of the innumerable other musical symbols can be combined into a single musical object. It is also made more difficult by the differing styles of notation, and differing rules governing the sizes and appearance of the musical symbols in a diverse selection of music scores.

5.3.1 k-Nearest Neighbor (kNN):

In pattern recognition, the *k*-nearest neighbour algorithm (k-NN) is a method for classifying phenomena based upon observable features, similar to the nearest neighbor classification method. The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the features scales are not consistent with their relevance [23].

The kNN algorithm is extremely simple, yet powerful, used in many applications and can be safely applied in all sorts of data sets, real life and artificial ones, independently of size or time compromises, resulting into high quality scientific observations. kNN is also extremely suitable to use in cases where instances map to points in n _ , there are lots of training data into consideration and – after performing soft feature selection – less than 20 attributes per instance [24].

Specifically, the kNN algorithm assumes that all elements correspond to points in the n dimensional space Rn. The neighbors of an element are defined in form of some distance measurement. A variety of metrics can be used as distances in the algorithm, like Euclidean square distance:

$$d(a,b) = \sqrt{\sum_{i \in N_F} (a_i - b_i)^2}$$

Minkowsky distance:

$$d(a,b) = \sum_{i \in N_F} |a_i - b_i|$$

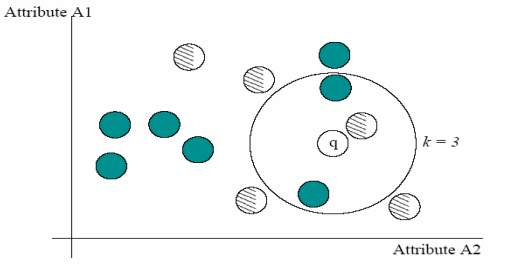
Minimax distance:

$$r(a,b) = \max \sum_{i \in N_F} |a_i - b_i|$$

Mahalanobis distance and others [24].

In instance-based methods, such as k-nearest-neighbors, the learning step is trivial: we simply store the dataset in the system's memory [17].
In the classification step, we are given an instance q (the query), whose attributes we will refer to as q:Ai and we wish to know its class. In kNN, the class of q is found as follows:
1. Find the k instances in the dataset that are closest to q.
2. These k instances then vote to determine the class of q.

Ties (whether they arise when finding the closest instances to q or when voting for the class of Q) are broken arbitrarily. In the following visualization of this method, we assume there are only two attributes A1 and A2, and two different classes (where circles with solid fill represent instances in the dataset of one class and circles with hashed fill represent instances in the dataset of the other class). Query q is here being classified by its 3 nearest neighbors [17].



All that remains is that we need a measure of distance in order to know who the neighbours are by using the above distance functions and formulas.

5.3.2 Artificial Neural Networks (ANN):

The history of ANNs stems from the 1940s, the decade of the first electronic computer. However, the first significant step took place in 1957 when Rosenblatt introduced the first concrete neural model, the perceptron. McCulloch and Pitts (1943) developed models of neural networks based on their understanding of neurology [14].

Neural Networks are a different paradigm for computing: [25]

- Von Neumann machines are based on the processing/memory abstraction of human information processing.
- Neural networks are *based on the parallel architecture of animal brains*.

Neural networks are a form of multiprocessor computer system, with

- simple processing elements
- a high degree of interconnection
- simple scalar messages
- adaptive interaction between elements

A biological neuron may have as many as 10,000 different inputs, and may send its output (the presence or absence of a short-duration spike) to many other neurons. Neurons are wired up in a 3-dimensional pattern.

The Neural Network Algorithm:

This algorithm is presented in [21].

The basic input variables are listed in Eq. 1, where the index **i** runs over the central element (**i = 0**) and its eight nearest neighbors.

$$E_{tot} = \sum_i E_i \quad e_i = \frac{E_i}{E_{tot}} \qquad (1)$$

A second set of input variables is based on a general cluster which is defined as a set of contiguous elements with non-zero energy deposition. The crystal barrel has axial symmetry which allows us to define displacements of elements in terms of angles θ and ϕ. The topological properties of the cluster are captured by the variables in Eq. 2, where the index **0** indicates

the element with the highest energy and the index **i** runs over the whole cluster.

$$d\theta_i = \theta_i - \theta_0 \quad d\phi_i = \phi_i - \phi_0 \quad \varepsilon_i = \frac{E_i}{E_{tot}}$$

$$E_{tot} = \sum_i E_i \qquad del\,\theta\theta = \sum_i \varepsilon_i d\theta_i |d\theta_i|$$

$$del\,\phi\theta = \sum_i \varepsilon_i d\phi_i |d\theta_i|$$

$$del\,\theta = \sum_i \varepsilon_i d\theta_i \qquad del\,\theta\phi = \sum_i \varepsilon_i d\theta_i |d\phi_i|$$

$$del\,\phi = \sum_i \varepsilon_i d\phi_i \qquad del\,\phi\phi = \sum_i \varepsilon_i d\phi_i |d\phi_i|$$

The neural network was trained to learn the position of impact relative to the central element of the cluster in the (θ, ϕ)-coordinate system. The structure of the network is a fully connected four-layer-perceptron with non-linear transfer functions

$$f(x) = tanh(x)$$

in the hidden and output layers. The number of neurons per layer is not a critical quantity. We have chosen numbers of neurons for the hidden layers which interpolate smoothly between the number of input and the number of output neurons.

The simulation package GEANT (version 3.13) has been used to generate data samples for the training and testing of the neural network algorithm. During the simulation, energy and angles of the photon were varied within the ranges required by the experiment.

The neural network was trained with the traditional back-propagation algorithm. The training and testing sets contained 1000 and 2000 events, respectively. The learning behavior was very stable and an acceptable solution could be reached after typically 20000 epochs.

Advantage:

In case of Optical Music Recognition, the recognition of the symbols by neural net is fast.

Pitfalls:

The net takes an appreciable time to learn symbols. Since its all like a black box, we don't have much control over the functioning of the net. Also if two symbols in the character set resemble each other too closely, there might me some error in recognizing these symbols.

5.3.3 Template Matching:

David Bainbridge and **Tim Bell** also have stated that various techniques can be used to recognize the musical objects after the staves have been removed. A commonly used method is ***template matching***. This method is to use a set of templates that are matched against objects that have been isolated. A measure is made of the similarity between an object being matched, and each available template, with the highest rating match being used, provided that it passes some threshold. [1]

Drawback of Template Matching:

The drawback with this method is that the shapes of objects can vary greatly from publisher to publisher, and even within a piece of music. Also, some shapes are very similar, such as bar lines and stems, and even hollow and filled note-heads, while other musical features appear in an infinite variety of variations, such as slurs and beams.

The most significant drawback of template metching is its poor run-time performance. Matching can be slow because each isolated object must be compared with every template (which can be particularly time-consuming if many variations of each symbol are stored, and symbols can be scaled.)

Alternatives: The writers **David Bainbridge** and **Tim Bell** have described that several alternatives have been proposed to either accelerate this process, or to make it more reliable.

Horizontal and Vertical projections are used to create a signature of an object, and match that with projections of previously identified objects.

Hough Transform is used in image processing to locate straight lines and curves and has proved useful for finding bar-lines, stems, and the curl of a bass clef.

Slicing Technique involves taking a cross-section through an object at a predetermined position, and counting the number of transitions from black to white pixels.

Connectivity Analysis is another pattern recognition technique that is particularly suited to classifying musical features that are dynamic in their size within one piece of music, such as slurs. Connectivity analysis answers the question, is there a path through the object from (x_1, y_1) to (x_2, y_2) (where $x_1<x_2$) with x monotonically increasing. Connectivity analysis is useful for detecting beams and slurs.

Rather thsn opting for just one of these techniques, CANTOR system described in this paper makes them all avaliable. They are invoked through a language designed as part of the CANTOR for this purpose called PRIMELA (PRIMitive Expression LAnguage), through which any combination of these technoques can be used to identifya primitive object in the input image.

5.3.4 Character Profiles:

Character Profiles is the development of 1970s [15].

The writers **K. Todd Reed** and **J.R Parker** have described the method of Character Profiles for the recognizing accidentals, rests, and clefs which are isolated by the staff line removal. Character Profiles are the functions that measure the perpendicular distance of the character's outer contour from some reference axis, usually the left or bottom edge of the character's bounding box. The right profile, for example, is obtained my measuring the horizontal distance between the left side of the character's bounding box (the reference axis) and the right edge of the character. Left *(L),* right *(R),* top *(T),* and bottom *(B)* profiles can be measured: often height and width profiles are computed from *T - B* and *R - L* respectively. [5]

Shortcoming of Charcter Profiles Technique:

The most significant shortcoming of this technique is that it requires the symbol to be isolated. Unfortunately, some symbols, especially accidentals, touch other symbols, preventing successful segmentation.

5.3.5 Mathematical Morphology:

The writers **Yun-Chung Chung** and **Greg C Lee** have described that detection of most musical symbols in the music scores can be done using *mathematical morphology*. Mathematical morphology is capable of detecting given symbol patterns by opening and closing operations.
Musical symbols in scores have fixed shapes. Thus features of symbols may be recognized by morphological operations. [4]

Mathematical morphology can be used to recognize clef signatures (treble and bass clefs), time signatures, heads of solid notes, some rest symbols (whole, half and quarter rests), heads of whole or half notes, and note tails. For each symbol, different feature patterns need to be designed to match distinct target symbol.

5.3.6 Signature Analysis:

The same writers have mentioned that some musical symbols are relatively small in size as compared to others in scores. Their size in low resolution is only about a few pixels wide and high. Thus, hardly any feature can be defined for morphological operations. A different method to differentiate the remaining small symbols is needed. The new method must have the property that does not need any fixed matching patterns. *Signature analysis* compares each symbol in the relative width, height, or profiles. These features help to distinguish them. The undetected symbols in scores are recognized by signature analysis. In this paper signature analysis is used to detect accidentals (sharps, flats, and naturals), key signatures, eighth-rest and sixteenth-rest symbols, beams, and dotted notes or rests in sequence. [4]

5.4 Interpretation of Musical Objects:

The main benefits of OMR are located in uderstanding the mening of a musical score.Without understanding of the score, an OMR system becomes just an overly complex photocopier. The system must understand the relationships between separate musical objects and their effect on the objects that follow in order to understand a score. This interaction between objects is quite complex, and sometimes the same semantic meaning can be arrived at by different ways. For example, a half note followed by a hold dot means exactly the same thing as a half note tied to a quarter-note.
The writers K Todd Reed and J.R. Parker have described the last step of musical recognition process i.e. Contextual Recognition. Once the individual symbols have been found, they are synthesized into more complex forms by using *Graph Grammars.*

The concept of Graph Grammars originated in the late 1960's and motivated by considerations about pattern recognition [16].

A disconnected graph is created from the primitive symbols in the score, and then "parsed" to perform high level recognition. For example, a sharp, two note heads, two stems, and a beam, under certain geometric/spatial constraints, can be combined into a single high level symbol, as illustrated in Figure 12. This technique provides the opportunity to use the context of symbols to determine their syntactic validity (using rules of music notation), and possibly correct some recognition errors. [5]

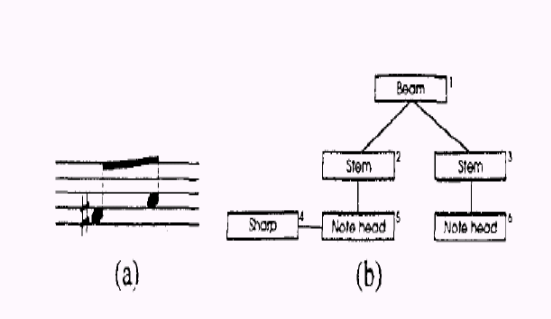


Fig 12: Using a Graph Representation to incorporate context. (a) A small sample containing six primitives. (b) The Graph Representation of (a).

6. CONCLUSION:

OMR is an image processing problem that requires locating and recognizing musical symbols in the image that are superimposed upon stave lines.

The difficulties that come in recognition process of printed music are magnified when the music symbols are not typeset but rather handwritten.Commercial systems are still unable to deal satisfactory with handwritten music.

Music recognition would be simple if music images had straight staff lines, clear printing, and a layout of symbols that followed the rules for music engraving. The reality is that the images are bent and distorted, have poorly formed symbols, and symbols that touch each other when they should not. Because the symbols are superimposed on a staff line, these problems are especially challenging, particularly when a symbol is fragmented or two symbols overlap when they should not.

By careful removal or avoidance of staff lines, and allowing for them to deviate from a straight line, the ``damage" done to symbols by staff line removal can be reduced. The symbol recognition can actually be enhanced if rather than removing the stave lines more lines are added. The next stage, matching the objects, is then designed to allow for the kinds of imperfections than can be expected both in the original image and after staff line removal, resulting in low error rates, which are necessary to make OMR a feasible technology.

7. REFERENCES:

[1] David Bainbridge, and Tim Bell. "*An Extensible Optical Music Recognition System"*

[2] Scott Sheridan, and Susan E. George, "*Defacing Music Scores For Improved Reognition."*

[3] R. Randriamahefa, J.P. Cocquerez, C. Fluhr, F. PCpin, and S. Philipp. "*Printed Music Recognition"*

[4] Yun-Chung Chung and Greg C Lee. "*Recognition of Printed Sheet Music Using Hough Transform and Morphology Operations"*

[5] K. Todd Reed and J. R. Parker, "*Automatic Computer Recognition of Printed Music"*

[6] David Bainbridge, "*Optical Music Recognition, Progress Report1"*

[7] Roland Gocke. "Building a System for Writer Identification on Handwritten Music Scores"

[8] http://www.ee.duke.edu/~jshorey/Hough/Page2.htm

[9] http://rsb.info.nih.gov/ij/plugins/template/matching.html

[10] http://www.data-uncompression.info/Algorithms/WFC/index.htm

[11] http://en.wikipedia.org/wiki/Fourier_transform

[12] http://en.wikipedia.org/wiki/sir_Francis_Galton

[13] http://ieee.ee.unv.edu/micromouse/resources.html

[14] http://koti.mbnet.fi/~podju/nenet/NeuralNetworks/NeuralNetworks.html

[15] http://faculty.ncwc.edu/toconnov/428/428kect01.htm

[16] http://www.ebooks.com/ebooks/book_display.asp?llD=183731

[17] www.cs.uuc.ie/~dgb/cources/tai/notes/handout5.pdf

[18] http://library.thinkquest.org/15413/theory/note-reading.htm

[19] *The World Book Encyclopedia, 1970,* "Music".

[20] http://Lesson Tutor Introduction to Music Notation.htm
[21] http://www.myriad.online.com/resources/docs/harmony/portugues/tieslurbeam.htm
[22] http://en.wikipedia.org/wiki/Flood_fill
[23] ge processing\more algos\ K-nearest neighbor algorithm - wikipedia, the free encyclopedia.htm
[24] www.image.ece.ntua.gv/papers/257.pdf
[25] http://www.cs.stir.ac.uk/~lss/NNIntro/In.Slides.html