ABSTRACT
We report a system which retrieves piano exercises automatically to match patterns input by a user. The system constructs a database of exercises, indexed by representative patterns, which are extracted from MIDI files by a repeated pattern detection algorithm. An input pattern is compared with these indices using the longest common subsequence method, and the most highly-ranked matches are retrieved.

1. INTRODUCTION
There are many ways of teaching the piano to beginners (e.g. the Alfred Method, the Bastien Method, Play by Choice, etc.). But advanced students who have mastered basic fingering techniques develop their ability by learning works for the piano, and through fingering exercises. These included the études for the piano by Carl Czerny, such as “The School of Velocity” (Op. 299) and “The Art of Finger Dexterity” (Op. 740). These exercises provide practice in specific fingering patterns, such as those required to play diatonic and chromatic scales, arpeggios, and octaves. For example, the first exercise in Czerny’s “The School of Velocity” develops the C major diatonic scale in the right hand (Figure 1). These exercises are used by experienced pianists, as well as advanced students, to strengthen particular aspects of their technique.

However, it can take a long time to find an appropriate exercise by searching through books and sheet music; and the range of piano exercises that any individual may be expected to own is limited. While the études of Czerny and Cramer are staple pieces in the training of pianists, many lesser-known exercises are also well written. And many artistic études such as those by Chopin or Liszt also provide good exercises. It is especially difficult to find exercises for a specific purpose among these études.

We have constructed a computer-based system which automatically retrieves pattern-specific exercises from the piano exercise literature in response to an input pattern. The system builds a database of pattern-specific exercises indexed by representative patterns from each exercise, which are found by an algorithm that detects repeated patterns. The system responds to a query by comparing the input pattern with the indices of representative patterns, and the exercises with the most accurate matches are retrieved.

The representation of the indices and the input pattern is the key to accurate and efficient searching, and we suggest heuristic techniques such as an adaptive interval tolerance to improve the quality of retrieval.

Various music retrieval systems based on queries to a database already exist [1, 2]. But most of those systems are designed to find songs from a fragment of melody. The problem of locating piano exercises requires a quite different approach.

2. PATTERN FINDING
Becuase piano exercises are designed to provide intensive practice of a particular musical figure, the notes form patterns which are repeated several times. We identify these patterns using a repeated pattern detection algorithm. There have been many approaches to finding repeated patterns in symbolic music, (e.g. [3, 4]), but we used the method due to Yoo[5], which is particularly strong in its support for user interaction. We have modified this method in several respects to suit our particular application.

2.1. Interval Distance Profile
Yoo’s method[5] exploits the observation that similar patterns contain similar sets of intervals, which are separated by similar periods of time. Our system starts analyzing a piece of music by finding all the intervals \( I = \{ I_i \}, i = 1, \ldots, N - 1 \) between adjacent notes, where \( N \) is the total number of notes in the piece. Then it locates a set of pairs \( A \):

\[
A = \{(I_p, I_q), \text{ where } |I_p - I_q| < \theta \}.
\]

The threshold \( \theta \) controls the degree of robustness, and must be set adaptively to accommodate different fingerings; we will discuss this later. The time period between the two intervals in each pair \( A_i (\in A) \) is then calculated and the numbers of similar distances are counted. Repeated patterns become apparent as clusters in this dis-
2.2. Generating Representative Patterns

Because all the interval pairs are aggregated together without any information, interval pairs from unrelated parts of the piece of music can have the same distance values. But we can extract temporally coherent components by preserving the location of each pair in the music, and clustering temporally related pairs. The following pseudocode describes the clustering algorithm:

```
procedure simple_clustering (newItrPair)
    find the temporally closest interval pair
    if their separation ≤ threshold then
        add newItrPair to the cluster C containing the closest interval pair
    else
        create new cluster with newItrPair
end
```

The threshold used in this clustering affects the number and size of clusters. We have found it effective to set this threshold to correspond to the most frequent note duration in the piece of music. Let \( Hist(dur(Nt)) \) be a histogram of durations of all notes and their counts. Then the most frequent note duration \( (MFD) \) can be determined as follows:

\[
MFD = \text{mode}(Hist(dur(Nt))).
\]

It is likely that many patterns belonging to different temporal clusters will be similar, because the representative patterns in a pattern-specific exercise are continuously repeated. A pattern-specific exercise may also have two or more different representative patterns. For example, an exercise in the diatonic scale is likely to have representative patterns corresponding to both ascending and descending scales. We therefore refine the clusters by eliminating:

1. Clusters with a small number ( < 5) of interval pairs.

2. The cluster with the smaller interval pair, out of two similar clusters.

The similarity between clusters is determined by comparing their interval distributions. Let \( C^i_A(i) \) and \( C^i_B(i) \) be the count of interval \( i \), calculated between adjacent notes in clusters \( A \) and \( B \) respectively. Each interval can be represented as a number of semitones. For example, the interval between \( C_4 \) and \( D_4 \) is 2, and the interval between \( C_4 \) and \( G_3 \) is \(-5\). \( N_i^A \) is the total number of intervals between adjacent notes in cluster \( A \), and \( N_i^B \) is the same total for cluster \( B \). Then, their dissimilarity can be determined as follows:

\[
D(A, B) = \sum_{i=-12}^{12} \frac{(C_A(i)/N_A - C_B(i)/N_B)^2}{N_A}
\]

No that we truncate intervals to the range \([-12, -12]\]. Two clusters \( A \) and \( B \) are considered to be similar if \( D(A, B) < 1.0 \). This process determines the representative patterns of each exercise, to be used as indices of the database.

2.3. Adaptive Interval Tolerance

Musically, the three sequences (a), (b) and (c) in Figure 3 would be considered to be the same pattern. But they are different if we use an exact comparison. We therefore use an adaptive (nonlinear) quantization in our comparison between different intervals. Table 1 shows the tolerance used for each interval. Intervals within these tolerances are considered to be the same. Additionally, if both intervals are more than an octave (P8) apart, they are assumed to be the same.

2.4. Selecting Major Peaks

Generally, several high peaks appear in the interval distance profile and the choice of peak crucially influences the resulting clusters. We select the highest peak corresponding to a temporal distance longer than a crotchet.

\[\text{Figure 2. The interval distance profile of the first exercise in Czerny’s “New Studies in Technique”, Op. 849.}\]

\[\text{Figure 3. Three different patterns which must be regarded as the same in musical terms.}\]

\[\text{Table 1. Adaptive tolerances for each interval. Intervals within this tolerance are considered to be same (P = perfect, m = minor, M = major, Aug = augmented).}\]
We make this assumption because the most frequently repeated patterns are not always the most important musically. Our system provides candidate peaks that may contain important patterns from which the user can make a selection.

2.5. Identifying Right and Left Hand Practice

Every piano exercise can be classified as a right-hand, left-hand or both-hands exercise. To allow the user to select an exercise of the appropriate type, our system automatically clarifies the exercises in its database into right-hand, left-hand, or both-hands exercises and attaches this data to the indices. All the MIDI files for the exercises in our database have two tracks, each of which contains the notes for one hand (track 1 has the notes for the right hand, and track 2 those for the left). When an exercise is added to the database, the note density \( ND \) is calculated for each track of the exercise:

\[
ND(T) = \sum_{i=1}^{n} \left( \frac{1}{dur(Nt_i(T))} \right)
\]

where \( n \) is the number of notes in track \( T \) and \( dur(Nt_i(T)) \) is the duration of the \( i \)th note in \( T \). The major track is determined by the following method:

\[
\text{procedure finding_major_track} \\
\text{if ND(Trk1)/(ND(Trk1)+ND(Trk2))} \leq \text{threshold then} \\
\text{this exercise is for right hand} \\
\text{else if ND(Trk2)/(ND(Trk1)+ND(Trk2))} \leq \text{threshold then} \\
\text{this exercise is for left hand} \\
\text{else} \\
\text{this exercise is for both hands} \\
\text{end}
\]

This procedure is based on the assumption that there are more note in the passages that constitute the functional part of the exercise than elsewhere, and in these more important passages each note is therefore of short duration. We set this threshold to 0.6.

3. PATTERN REPRESENTATION AND MATCHING

3.1. Pattern Representation

A common approach to content-based music retrieval, based on extracting sets of features from symbolic music data or audio signals, is to represent music as a string of tuples. The representative patterns of each exercise in our system consist of strings of tuples containing the following four elements:

1. R or L : right or left hand data.
2. The interval between adjacent notes
3. The duration of a crotchet is 120 and the most frequent duration of a note is assumed to be 60.
4. The inter-onset interval (IOI) between adjacent notes, divided by the MFD (see Section 2.2).
5. The duration of the earlier of two adjacent notes, divided by the MFD.

For example, the shaded notes in Figure 4 would be represented as follows:

\[
(R -5 3 2)|(R -3 1 1)|(R 3 3 2)|(R 4 1 1)|(R -4 3 2)|(R -5 1 1)|(R 5 3 2)
\]

A user provides the pattern that he or she wants to search for, together with the hand or hands to be exercised. A user query is represented as a string of four-tuples and compared with the indices of each exercise, which are represented in the same form.

3.2. Matching Method

Well-known approximate string matching methods can be used to compare a user query with each index because both of them are represented as strings. We use the longest common subsequence algorithm[6] for matching. It provides an solution to the problem of finding the longest subsequence common to a set of all sequences. The cost of matching a user query and an index can be expressed as follows:

\[
Cost(Q, idx) = LCS(E(Q), idx)/L(idx)
\]

where \( LCS \) is the length of the longest subsequence common to an extended user query \( E(Q) \) and an index \( idx \). An extended user query \( E(Q) \) is generated by repeating a user query \( Q \) until its length is the same as that of the index \( L(idx) \). Note that the adaptive tolerance value (Section 2.3) is used in comparing the intervals in two sequences.

4. EXPERIMENTAL RESULTS

Figure 5 shows the search results for the query (L 7 1 1)|(L -3 1 1)|(L 3 1 1)|(L -7 1 1). The results page allows users to see the whole score of each exercise (in PDF format) and listen to the MIDI file.

We compared the time required for searches using our system with manual searching. Three users were asked to find exercises with a given pattern in three books of Czerny Etudes (Ops.299, 740 and 849). The time for a search was 27 minutes and 37 seconds. The same searches on our
system took less than one minute each including the encoding time of an input pattern. The results achieved were similar to those from manual searching and the system sometimes found additional results, which met the user’s requirement in a non-obvious way.

5. CONCLUSIONS AND FUTURE WORK

We have introduced a system that automatically retrieves piano exercises to match a user’s requirements. We expect this system will contribute to music education by allowing the selection of exercises that meet the needs of individual students more precisely.

We are currently adding more scores and MIDI files to our database, and elaborating our system to support fingering practice and exercises in more complicated piano practice techniques.

In the future, we intend to undertake more systematic testing, including a comparison of the melody indexing and retrieval methods used in our system and other approaches. In addition, we believe that our system could easily be extended to support other instruments.

Acknowledgements

This research is the result of the promotion project for culture contents technology research center supported by Korea Culture & Content Agency (KOCCA).

6. REFERENCES


