INDEXING AND RETRIEVAL OF MUSIC DOCUMENTS THROUGH PATTERN ANALYSIS AND DATA FUSION TECHNIQUES

Giovanna Neve
University of Padova
Department of Information Engineering

Nicola Orio
University of Padova
Department of Information Engineering

ABSTRACT

One of the challenges of music information retrieval is the automatic extraction of effective content descriptors of music documents, which can be used at indexing and at retrieval time to match queries with documents. In this paper it is proposed to index music documents with frequent musical patterns. A musical pattern is a sequence of features in the score that is repeated at least twice: features can regard perceptually relevant characteristics, such as rhythm, pitch, or both. Data fusion techniques are applied to merge the results obtained using different features. A set of experimental tests has been carried out on retrieval effectiveness, robustness to query errors, and dependency on query length on a collection of Beatles' songs using a set of queries. The proposed approach gave good results, both using single features and, in particular, merging the rank lists obtained by different features with a data fusion approach.

1. INTRODUCTION

The research work described in this paper focuses on indexing of music documents aided at efficient Music Information Retrieval (MIR). The presented methodology is similar to approaches based on the retrieval of music documents in symbolic notation – MIDI, GUIDO [5], or SMDL – through a query-by-example paradigm. The research work on this area of MIR can be roughly divided in two categories: on-line searching techniques, which compute a match between a representation of the query and a representation of the documents each time a new query is submitted to the system; and indexing techniques, which extract off-line, from music documents, all the relevant information that is needed at retrieval time and perform the match between query and documents indexes.

Both approaches have positive and negative aspects. On the one hand, on-line search allows for a direct modeling of query errors by using, for instance, approximate pattern matching techniques that deal with possible sources of mismatch, e.g. insertion and/or deletion of notes. This high flexibility is balanced by high computational cost, because the complexity is at least proportional to the size of the document collection and, depending on the technique, to the documents length. On the other hand, indexing techniques are more scalable to the document collection, because the index file can be efficiently accessed through hashing and the computational complexity depends on query length. The high scalability is balanced by a more difficult extraction of document content, with non trivial problems arising in case of query errors that may cause a complete mismatch between query and document indexes. Both approaches have given interesting and promising results. Yet, indexing approaches need to be investigated in more detail because of the intrinsic higher computational efficiency.

Previous work on on-line search has been carried out following different strategies. The work presented in [2] is based on the use of pattern discovery techniques, taken from computational biology, to compute occurrences of a simplified description of the pitch contour of the query inside the collection of documents. Another approach, reported in [7], applies pattern matching techniques to documents and queries in GUIDO format, exploiting the advantages of this notation in structuring information. Approximate string matching has been used by [6]. Markov chains have been proposed in [3] to model a set of themes that has been extracted from music documents, while an extension to hidden Markov models has been presented in [16] as a tool to model possible errors in sung queries. The work presented in [8] reports a comparison of different approaches to on-line search, with a discussion on computational complexity and scalability of four different techniques based on dynamic time warping.

An example of research work on off-line document indexing has been presented in [4]. In that work melodies were indexed through the use of N-grams, each N-gram being a sequence of N pitch intervals. Experimental results on a collection of folk songs were presented, testing the effects of system parameters such as N-gram length, showing good results in terms of retrieval effectiveness, though the approach seemed not be robust to decreases in query length. Another approach to document indexing has been presented in [11], where indexing has been carried out by automatically highlighting music lexical units, that have been called musical phrases. Differently than the previous approach, the length of indexes was not fixed but depended on the musical context. That is musical phrases were computed exploiting knowledge on music perception, in order to highlight only phrases that had a musical meaning. Phrases could undergo a number of different normalization, from the complete information of pitch intervals and duration to the simple melodic profile.

Most of the approaches, and the one presented in this paper as well, focus on the melody, while other music dimensions, such as harmony, timbre, or structure, are not taken into account. This choice may become a limitation depending on the way the user is allowed to interact with the system and on his personal knowledge on music language. When the query-by-example paradigm is used, the effectiveness of a system depends on the way a query is matched with documents: If the user may express his information need through a query-by-humming interface, the melody is the most likely dimension that he will use. Moreover, for non expert users, melody and rhythm (and lyrics) are the more simple dimensions for describing their information needs.

2. MELODIC PATTERNS FOR INDEXING AND RETRIEVING MUSIC

The basic idea underlying our work is that a music document can be effectively described by excerpts of its melodic features. The main goal then becomes the automatic extraction of relevant excerpts from an unstructured flow of notes. Differently from previous work carried out on musical phrases extraction [13], it is proposed that melodic excerpts, or melodic patterns, can be computed by the simple analysis of the repetitions of substrings of note features inside the melody; this way no prior knowledge on music perception or music structure is required. The proposed approach is similar to the one presented in [17], where frequent phrases are used to index a music database. Unfortunately, [17] focused on computational efficiency and did not present an analysis on retrieval effectiveness, and thus results cannot be compared.

2.1. Terminology

Before introducing the methodology, it can be useful to describe in detail the terminology used in the following sections.

A **feature** is one of the characteristics that describe subsequent notes in a score. A note feature can be: the pitch, the pitch interval with the previous note (PIT), a quantized PIT, the duration, the interonset interval with the subsequent note (IOI), the ratio of IOI with the previous note, and so on. All the features can be normalized or quantized. In our approach, features are related to pitch and rhythm that, though usually correlated, can be treated independently. For example, many songs can be guessed only by tapping the rhythm of the melody while other ones can be easily recognized even if played with no tempo or rubato.

A **string** is a sequence of features. Any sequence of notes in a melody can be considered a string. It can be noted that strings can be used as representative of a melody, which is the idea underlying many approaches to MIR, but the effectiveness by which each string represents a document may differ. For instance, it is normally accepted that the first notes of a melody play an important role in recognition, or that strings that are part of the main theme or motif are good descriptors as well. String length is an important issue: Long strings are likely to be effective descriptors, yet they may lead to problems when the user is request to remember long parts of a melody for querying a MIR system. In our experiments, strings shorter than three notes have been discarded, because they were not considered significant descriptors.

A **pattern** is a string that is repeated at least twice in the score. The repetition can be due to the presence of different choruses in the score or by the use of the same music material (e.g., motifs, rhythmical cells) along the composition. Each pattern is defined by the string of features, by its length n and by the number of times r it is repeated inside the score. All patterns that appear only inside longer patterns have been discarded. The computation of patterns can be done automatically using well known algorithms for pattern discovery.

Patterns can be considered as content descriptors of music documents. Depending on the features, patterns carry different information about document content. It is possible to take advantage from alternative descriptors by applying a **data fusion** approach. This approach is usually carried out in the research area of Meta Search Engines [9], where the results obtained by different indexing and retrieval methodologies are combined – or *fused* – together according to a predefined weighting scheme.

In the present work, three kinds of features are used for the pattern selection step – the interonset interval (IOI) normalized to the quarter note, the pitch interval (PIT) in semitones, and both (BTH) – and two data fusion approaches are exploited – the combination of IOI and PIT (Fuse2) and the combination of all the features (Fuse3).

2.2. Document Indexing

Document indexing is a mandatory step for textual information retrieval. Through indexing, the relevant information about a collection of documents is computed and stored in a format that allows easy and fast access at retrieval time. Document indexing is carried out only when the collection is created or updated, when users are not yet accessing the documents, and then the problems of computational time and efficiency are usually less restrictive. Indexing speeds up retrieval time because it is faster to search for a match inside the indexes than inside the complete documents.

According to the terminology introduced in the previous section, each document has a number of patterns of different length and with different multiplicity. It is proposed that patterns are effective descriptors for document indexing, that is a document can be indexed by the patterns of its melodies. The first step of document indexing consists in the automatic computation of the patterns of each document. As previously mentioned, the features used to compute the patterns are IOI, PIT, and BTH. Pattern computation is carried out with an ad-hoc algorithm that computes exhaustively all the possible patterns, and stores them in a hash table.

An exhaustive pattern discovery approach highlights a high number of patterns that have little or no musical meaning; for instance, a pattern that is repeated only two or three times in a document is likely to be computed by chance just because the combination of features is repeated in some notes combinations. Moreover, some patterns related to scales, repeated notes, or similar musical gestures, are likely to appear in almost all documents and hence to be poor discriminants among documents. In general, the degree by which a pattern is a good index may vary depending on the pattern and on the document. This is a typical situation of textual information retrieval, where words may describe a document to a different extent. For this reason it is proposed to apply the classical $tf \cdot idf$ measure [1]. A document is described by a sparse array, where each element is associated to a different pattern in the collection. The value of each element is given by the $tf \cdot idf$ value that is, the number of times a pattern appears in the document (the tf term) is multiplied by the inverse of the fraction of documents that contain that pattern, computed in log scale (the idf term).

The index is built as an inverted file, where each term of the vocabulary is a different pattern in a given notation (i.e., a text string). Each entry in the inverted file corresponds to a different pattern, and can efficiently be computed in an expected time $O(1)$ with an hashing function. Given the different sets of features, three inverted files are built, respectively for features IOI, PIT, and BTH. Inverted files can be efficiently stored in memory, eventually using compression, and accessed at retrieval time [1]. The size of the inverted file and the implementation of the hashing function depend on the number of different patterns of the complete collection.

It may be useful to fix the maximum allowable pattern length to improve indexing. In fact, it is likely that very long patterns are due to repetitions of complete themes in the score and taking into account also them will give a quite sparse inverted file. Moreover, it is unlikely that a user will query the system singing a complete theme. These considerations suggest that long patterns could be truncated when they are over a given threshold. Truncated patterns can be substituted by the set of their subpatterns. The choice of the threshold can be done experimentally, using a test collection as presented in Section 3.2.

2.3. Query Processing

For the query processing step, it is assumed that users interact with the system according to a query-by-example paradigm. In particular, users should be able to describe their information needs by singing (humming or whistling), playing, or editing with a simple interface a short excerpt of the melody that they have in mind. The development of a user interface is out of the scope of the present paper, but it has been the subject of previous research by one of the authors both on audio query transcription [15] and on Web interfaces for playing/editing symbolic queries [12]. For the aims of this work, it is assumed that queries are translated in a sequence of features by an external preprocessing module.

The string that represents the translated query has to be processed. Query processing allows for extracting a number of descriptors that can be used to match the query with potentially relevant documents. In this work, it is assumed that a query is likely to contain strings that characterize the searched document, either because they appear very often inside its theme or because they are peculiar of that particular melody. In other words, it is assumed that a query contains relevant patterns of the searched document, which may have a high tf – frequent inside the document – and/or a high idf – infrequent across the collection.

The automatic detection of *relevant* strings cannot be carried out as for document indexing, because normally queries are too short to have repetitions and hence to contain patterns. A simple approach to extract relevant strings, or potential patterns, from a query consists in computing all its possible substrings. That is, from a query of length q notes are automatically extracted $q - 2$ strings of three notes, plus $q - 3$ strings of four notes, and so on until the maximum allowable length for a pattern is reached. This approach can be considered similar to query expansion in textual information retrieval, which is known to increase recall at the risk of lowering precision. On the other hand, it is expected that most of the arbitrary strings of a query will never form a relevant pattern inside the collection, and then the negative effects on precision could be bounded.

2.4. Ranking Relevant Documents

At retrieval time, the strings are automatically extracted from the query and matched with the patterns of each document. The computation of potentially relevant documents is performed using the Vector Space Model [1], which allows computing the distance between the vector of strings representing the query and the vector of patterns representing each document. Hence, for each document a Retrieval Status Value (RSV) is calculated, the higher the RSV, the closer the document with the query. A rank list of potentially relevant documents is computed from RSVs, obtaining three parallel rank lists according to the three features used.

In general the orderings of documents in the rank lists differ. Differences may be due to many factors, as the diverse importance of rhythm and melodic profile, the effect of errors in the query, the kind of melodic excerpt chosen for the query. It is expected that BTH ranking will give high scoring to the relevant documents when the query is sufficiently long and correctly played, because BTH patterns are a closer representation of the original melody. On the other hand, IOI and PIT are robust to query errors in

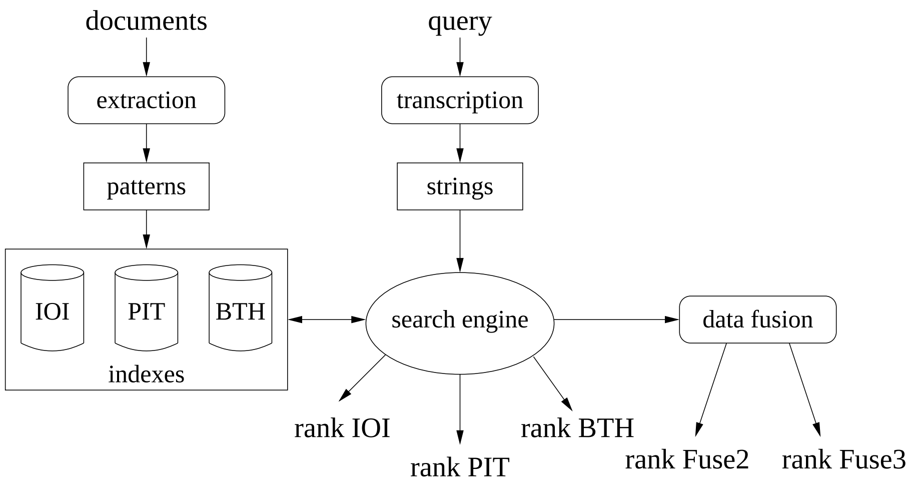

Figure 1. The phases of the proposed methodology: Indexing, retrieval, and data fusion

melodic profile and rhythm, respectively. Moreover, simple representations as IOI and PIT are expected to be less sensitive to query length because of the possible presence of subpatterns of relevant motifs.

It is possible to take advantage from the existence of different rank lists by fusing together the results, in order to give the user a single rank list which takes into account the results of the three parallel approaches. As introduced in Section 2.1, this is a typical problem of data fusion [9], which is usually faced by merging the rank lists obtained by different search engines. In this case, the fusion can be carried out directly using the RSVs, because they are all based on the same $tf \cdot idf$ scheme. In particular, a new RSV is computed as a weighted sum of RSVs of single features obtaining a new rank list. A number of tests with different combinations of weights have been carried out to find the optimal setting for a MIR system.

The complete methodology is shown in Figure 1, where steps undertaken at indexing time are shown on the left, while the operations that are performed at retrieval time are shown on the right. From Figure 1 and the above discussion, it is clear that the computational complexity depends on the query length – i.e., the number of strings that are computed from the query – while it is scalable on the number of documents. This is an important characteristic, because the time needed to reply to a query can be reasonably low also for large collections of documents.

3. EXPERIMENTAL EVALUATION

The evaluation has been carried out on a small test collection according to the Cranfield model for information retrieval, which is used at the Text REtrieval Conference (TREC) [18]. A test collection consists in a set of documents, a set of queries, and a set of relevance judgments that match documents to queries. The creation of a common background for evaluation is still an open issue in the Music Information Retrieval community [14], hence we created a small test collection from scratch. A number of experimental tests has been carried out to evaluate the effectiveness of the methodology and the robustness to common problems that may affect retrieval effectiveness, as errors in the queries and very short queries.

3.1. The Test Collection

A small test collection of popular music has been created using 107 Beatles' song in MIDI format downloaded from the Web. As for any test collection, documents may contain errors. In a preprocessing step, the channels containing the melody have been extracted automatically and the note durations have been normalized; in case of polyphonic scores, the highest pitch has been chosen as part of the melody. After preprocessing, the collection contained 107 complete melodies with an average length of 244 notes, ranging from 89 of the shortest melody to 564 of the longest. Even if a number of approaches for performing automatic theme extraction has been already proposed in the literature [10], our methodology relies on indexing of complete melodies, because repetitions of choruses and verses can be taken into account by the $tf \cdot idf$ measure.

A set of 40 queries has been created by randomly selecting 20 themes in the dataset and using the first notes of the chorus and of the refrain. The initial note and the length of each query were chosen to have recognizable motifs that could be considered representative of real users' queries. The queries had an average length of 9.75 notes, ranging from 4 to 21 notes. Only the theme from which the query was taken was considered as relevant. Using this initial set of correct queries, an alternative set has been created by adding errors on pitch, duration, and both, obtaining a new set of 120 queries. A simple error model has been applied, because errors were uniformly distributed along the notes in the queries, with a probability of about 13.3%. As for many approaches to approximate string matching, an error can be considered the result of a deletion and an insertion, thus these alternative sources of errors have not been explicitly modeled. Tests on robustness to query length were carried out by automatically shortening the initial queries by an increasing percentage, disregarding the fact that query would not sound musical. In this way, 160 more queries with decreasing length have been automatically generated. For all the modified queries, only the theme of initial query was considered as relevant. In the following, we will refer to the only relevant document with the term *r-doc* for all the experiments.

3.2. Truncation of Patterns

All the experimental analyses, whose results are shown in the following sections, have been carried out after truncating patterns longer than a given threshold t. When a pattern $[f_1 \dots f_n]$ had a length of $n > t$, it has been replaced (in the indexing step) by all its subpatterns of exact length t, that is the $n - t + 1$ subpatterns $[f_1 \dots f_t]$, $[f_2 \dots f_{t+1}]$, and so on until $[f_{n-t} \dots f_n]$, where some of the subpatterns may be already extracted, because they were part of other motifs.

With the aim of computing the optimal threshold for the test collection, five different thresholds have been tested, respectively 5, 7, 10, 15, and 20 notes. Results in terms of *average precision* [1] are reported in Figure 2 for the mean of the three single features and the two data fusion approaches, using the 40 correct queries. The retrieval effectiveness decreased with high values of the threshold, meaning that a compact representation of patterns can be more effective than longer ones. This result is consistent with the findings reported in [4]. The average precision was approximately constant when thresholds higher than $15 - 20$ notes were applied, probably because the number of different patterns longer than 20 notes is less than 8% and with a low value of r. The use of short patterns can be a useful way to control the increase of the index when new documents are added to the collection. Due to simple combinatorial reasons, the number of different patterns is bounded by the pattern length; on the other hand, the use of short patterns has the drawback of a higher number of patterns that are in common among documents, which may lower precision.

It is interesting to note that data fusion approaches gave consistently better results than single approaches, how can be seen both from the comparison of absolute values and of curve slopes in Figure 2. This behavior has been found in all our experiments, which are presented in the following sections, where results are shown only for $t = 5$.

3.3. Retrieval Effectiveness

The first detailed analysis regarded the retrieval effectiveness with the set of 40 correct queries. Results are shown in Table 1, where the average precision (Av.Prec.), the percentage queries that gave the r-doc within the first k positions (with $k \in \{1, 3, 5, 10\}$), and the ones that did not give the r-doc at all ("not found"), are reported as representative measures. As it can be seen, IOI gave the poorest results, even if for 90% of the queries the r-doc were among the first three retrieved. The highest average precision using a single feature was obtained by BTH, with the

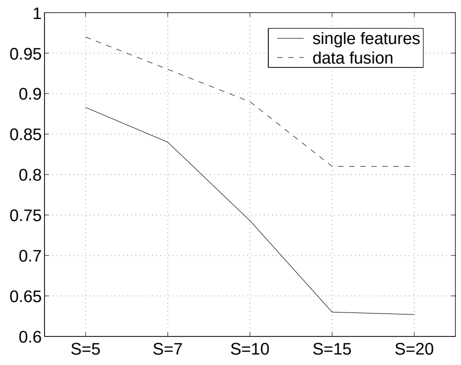

Figure 2. Average precision for the mean of single features (solid line) and data fusion (dashed line) with different truncation thresholds

drawback of an on-off behavior: either the r-doc is the first retrieved or it is not retrieved at all (2.5% of the queries). PIT gave good results, with all the queries that found the r-doc among the first three documents.

	IOI	PIT	BTH	Fuse2	Fuse3
Av.Prec.	0.74	0.93	0.98	0.96	0.98
$= 1$	57.5	87.5	97.5	92.5	95.0
≤ 3	90.0	100	97.5	100	100
≤ 5	95.0	100	97.5	100	100
≤ 10	97.5	100	97.5	100	100
not found	0	0	2.5	0	0

Table 1. Retrieval effectiveness for correct queries

Fuse2 gave an improvement in respect to the separate features – IOI and PIT – with an average precision of 0.98, hence with values comparable to BTH and without the drawback of not retrieving the r-doc (in all cases the r-doc is among the first three retrieved). It could be expected that adding BTH in the data fusion would not give further improvements, since BTH is already a combination of the first two. The set of BTH patterns is a subset of the union of set of IOI and PIT patterns, while it can be shown that set BTH includes the intersection of sets IOI and PIT, because of the choice of not considering subpatterns that have the same multiplicity of longer ones. Given these considerations, it is clear that BTH does not introduce new patterns in respect to IOI and PIT. Yet, as can be seen from column labeled with Fuse3 in Table 1, the use of all the three features allowed for reducing the drawbacks of the three single rankings. This result can be explained considering that BTH had different $tf \cdot idf$ scores, which were somehow more selective than simple IOI and PIT and which gave a very high $tf \cdot idf$ score to the r-doc in case of a good match.

The best results for Fuse2 and Fuse3 have been obtained assigning equal weights to the single ranks. When the $tf \cdot idf$ scores had different weights an improvement was still observed in respect to single rankings, though to a minor extent. For this reason, results for Fuse2 and Fuse3 are presented only when equal weights are assigned.

3.4. Robustness to Errors in the Queries

Users are likely to express their information needs in an imprecise manner. The query-by-example paradigm is error prone because the example provided by the user is normally an approximation of the real information need. In particular, when the user is asked to sing an excerpt of the searched document, errors can be due to imprecise recall of the melody, problems in tuning, tempo fluctuations, and in general all the problems that untrained singers have. Moreover, transcription algorithms may introduce additional errors in pitch detection and in melody segmentation. The robustness to errors has been tested on an experimental setup. Since indexing is carried out on melodic contour and on rhythm patterns, the errors that may affect the retrieval effectiveness regard the presence of notes with a wrong pitch and a wrong duration. As mentioned in Section 3.1, a set of queries with automatically added errors has been generated in order to test the robustness of the approach in a controlled environment.

The results obtained for queries with errors in the rhythm are presented in Table 2. As expected, the performances of IOI dropped, with a decrease of the average precision – from 0.74 to 0.51 – and a considerable amount of queries that did not retrieve the r-doc. The same considerations apply to BTH, with an even bigger drop in the performances – average precision at 0.77 and more than a fifth of the queries that did not retrieve the r-doc. As expected, PIT was insensible to this kind of error. It is interesting to note that data fusion allowed for compensating the decreases in performances of single ranks, giving for both Fuse2 and Fuse3 an average precision equal to the one obtained without errors.

	IOI	PIT	BTH	Fuse2	Fuse3
Av.Prec.	0.51	0.93	0.77	0.97	0.98
$= 1$	35.0	87.5	75.0	95.0	97.5
≤ 3	65.0	100	77.5	100	100
≤ 5	70.0	100	77.5	100	100
≤ 10	72.5	100	77.5	100	100
not found	17.5	0	22.5	0	0

Table 2. Results for queries with errors in rhythm

The results obtained for queries with errors in pitch are presented in Table 3. Dually to the previous case, IOI is not sensible to this kind of errors, while PIT and BTH had a drop in performances, both in terms of average precision and in the percentage of queries that did not retrieve the r-doc. Also Fuse2 showed a negative trend in retrieval effectiveness, with the main characteristics that the r-doc is retrieved more seldom as the first document – from 92% to 80.0% – while it is still retrieved within the first 3 documents in most of the cases. Fuse3 showed to be particularly robust also to this kind of errors, with almost exactly the same retrieval effectiveness obtained with correct queries.

	IOI	PIT	BTH	Fuse2	Fuse3
Av.Prec.	0.74	0.65	0.71	0.88	0.99
$= 1$	57.5	60.0	67.5	80.0	97.5
≤ 3	90.0	67.5	72.5	97.5	100
≤ 5	95.0	70.0	72.5	100	100
≤ 10	97.5	75.0	72.5	100	100
not found	0	20.0	27.5	0	0

Table 3. Results for queries with errors in pitch

The results obtained for queries with errors both in pitch and in duration are presented in Table 4. The two types of error had independent uniform distributions over the query notes. As expected, performances of IOI and PIT are completely comparable with the ones presented in Table 2 and Table 3 respectively, while performances of BTH have a considerable drop down to 0.41%. Also Fuse2 and Fuse3 showed a decrease in terms of average precision, which in both cases is about 10% higher than the best average precision of single features. It has to be noted that this kind of errors affects also the percentage of queries that did not retrieve the r-doc; moreover, data fusion approaches allowed for obtaining a percentage smaller than the one given by the best of the single features.

	IOI	PIT	BTH	Fuse2	Fuse3
Av.Prec.	0.51	0.65	0.41	0.74	0.75
$= 1$	35.0	60.0	37.5	67.5	67.5
≤ 3	65.0	67.5	42.5	77.5	80.0
≤ 5	70.0	70.0	42.5	82.5	85.0
≤ 10	72.5	75.0	42.5	85.0	85.0
not found	17.5	20.0	60.0	12.5	12.5

Table 4. Results for queries with errors in both pitch and duration

As it can be seen from these results, Fuse3 gave a considerable improvement in respect to the single rankings contribution. A query-by-query analysis showed that this behavior is due to the fact that the sum of $tf \cdot idf$ scores of the single features gave always a new ranking where the r-doc was at the same level of the best of the three separate ranks; that is, if one of the three gave the r-doc as the most relevant document, also Fuse3 had the r-doc in first position. Moreover, for some queries, the fused rank gave the r-doc at first position even if none of the three single ranks had the r-doc as the most relevant document. These improvements can be explained by two factors: First, when the r-doc was retrieved at top position by one of the features, it had a very high $tf \cdot idf$ score that gave an important contribution to the final rank; Second, the r-doc was often retrieved with a high rank by two or three of the features, while in general other documents were not considered as relevant by more than one feature. Similar considerations apply, though at a minor extent, also to Fuse2.

The small increment of average precision for queries with errors – Fuse2 in Table2 and Fuse3 in Table 3 – was due to the change of a single document from the second to the first position, and can be considered negligible.

These results can be compared with other approaches applied to test collections of similar size. For instance, [16] tested a HMM-based indexer, obtaining 41.7% of times the r-doc had the highest rank. In the same paper, a simple string matcher is used as a baseline, giving a value of 16.7% for the same experiment. The approaches based on Dynamic Time Warping [8] gave, in the best case, the 54.0% of times of the r-doc retrieved at top rank. The results presented in [3] showed that in 60.0% of the cases the r-doc was the nearest neighbor of the query. In carrying out the comparison with our approach, it has to be considered that all of this previous work used real queries, whose effect may decrease significantly system performances.

3.5. Dependency to Query Length

A final analysis has been carried out on the effects of query length to the retrieval effectiveness. It is known that users of search engines do not express their information needs using much information. The community of information retrieval had to face the problems of finding relevant information also with vague or short queries. To some extent, a similar problem applies to MIR because users may not remember long excerpts of the music documents they are looking for. Moreover, untrained singers may not like to sing for a long time a song that they probably do not know very well. The effects of query length on a MIR system should then be investigated.

Tests on the dependency to query length have been carried out on a set of queries that were obtained from the original set of queries by shortening the number of notes from 90% to 60% of their original lengths. The average length, its standard deviation (STD), and the minimum and maximum lengths for each new group of queries is reported in Table 5. With this approach, queries may become very short, for instance a query of two notes cannot retrieve any document because patterns shorter than three notes are not taken into account.

Fraction	100%	90%	80%	70%	60%
Mean	9.8	8.8	7.8	6.9	5.8
STD	3.5	3.3	2.8	2.5	2.1
Min	4	4	3	3	2
Max	21	19	17	15	13

Table 5. Query lengths when the original queries were reduced to a decreasing percentage of their number of notes

Results on retrieval effectiveness with increasingly shorter queries are depicted in Figure 3, where the average precision is reported for the three single features and the two data fusions. As it can be seen from Figure 3, there was a general decrease of performances. The drops in average precisions were accompanied by an increasing percentage of queries that did not retrieve the r-doc, which in the worst cases, BTH and IOI for a reduction to 60%, reached 27.5% and 22.5% respectively.

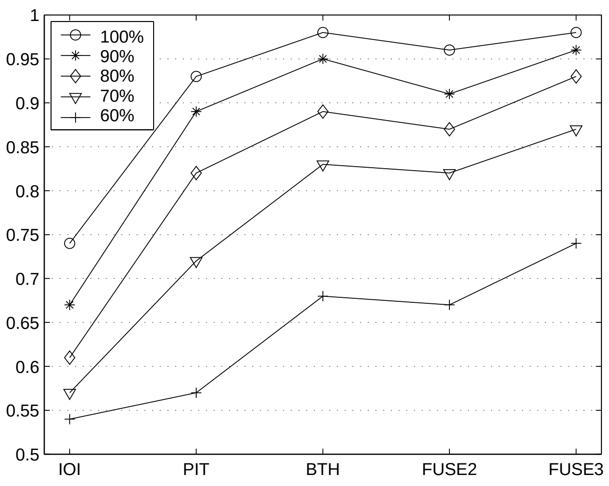

Figure 3. Average precision with different query lengths

Consistently with previous results, Fuse3 gave the best performances and showed a higher robustness to decrease in query length. The trend for Fuse3 alone is reported in Table 6. Also in this case results showed that the data fusion approach was enough robust to changes in the initial queries. As mentioned in Section 3.1, each initial query has been created selecting a number of notes that allowed to recognize the theme by a human listener. Moreover, each query was made by one or more musical phrases – or musical gestures or motifs – considering that a user would not stop singing his query at any note, but would end the query in a position that have a “musical sense”. For this reason, tests on query length can give only a general indication on possible changes in retrieval effectiveness. An analysis of the effect of query length has been proposed in [2], where it is estimated that in the worst case a user should sing up to 18 notes to uniquely identify a music document, even though this number can be reduced when only melodies are indexed.

Fraction	100%	90%	80%	70%	60%
Av.Prec.	0.98	0.96	0.93	0.87	0.74
$= 1$	95.0	92.5	90.0	82.5	65.0
≤ 3	100	97.5	95.0	90.0	80.0
≤ 5	100	100	97.5	92.5	85.0
≤ 10	100	100	97.5	95.0	87.5
not found	0	0	2.5	2.5	7.5

Table 6. Retrieval effectiveness for Fuse3 when query lengths are reduced by different fractions

4. CONCLUSIONS

A methodology for indexing and retrieving music documents in symbolic format has been presented. The methodology is based on the automatic extraction of patterns, computed from information on rhythm, pitch, and the combination of the two. Data fusion techniques have been applied to improve retrieval effectiveness. The approach can be extended including different features, for instance based on the quantization of pitch contour and note durations, that could be more robust to query errors. Additional features can replace the ones that have been tested in the present work, or can be added to the data fusion.

The methodology has been tested on a small test collection, giving encouraging results, even if a standard test collection of documents and real queries would be advisable to compare these results with the ones obtained with other techniques. Though results need to be confirmed by more extensive test on larger databases, there are some characteristics that are likely to apply in general. Data fusion techniques improve the performance of a music information retrieval system. For instance, the fusion of two separate features (Fuse2) seemed to be more robust to errors than the combination of the two (BTH). Other results that can be generalized regard the higher performances given by short patterns compared to longer ones.

The best results have been obtained by fusing three different indexing schemes. Yet simple indexing, like the one purely based on IOI, showed to be enough effective to develop a query-by-tapping MIR system, in particular for music genres in which the rhythm is more important, or easier to remember, than melody. Experimental tests showed also that retrieval effectiveness obtained with pitch information, can be improved using rhythmic information, especially in the case of short or error prone queries.

5. REFERENCES

[1] Baeza-Yates, R., Ribeiro-Neto, B. *Modern Information Retrieval*, ACM Press, New York, 1999.

[2] Bainbridge, D., Nevill-Manning, C. G., Witten, I. H., Smith, L. A., McNab, R. "Towards a Digital Library of Popular Music", *Proc. of the 4th ACM Conference on Digital Libraries*, Berkley, USA, pp. 161–169, 1999.

[3] Birmingham, W.P., et al. "MUSART: Music Retrieval Via Aural Queries", *Proc. of the ISMIR*, Bloomington, USA, pp. 73–82, 2001.

[4] Downie, S., Nelson, M. "Evaluation of a Simple and Effective Music Information Retrieval Method", *Proc. of the ACM-SIGIR Conference*, Athens, Greece, pp. 73–80, 2000.

[5] GUIDO The GUIDO Music Notation Format Homepage, `http://www.salieri.org/guido/`, visited on July 2004.

[6] Haus, G., Pollastri, E. "A Multimodal Framework for Music Inputs", *ACM Multimedia 2000 Conference*, Plymouth, USA, pp. 282–284, 2000.

[7] Hoos, H.H., Renz, K., Görg, M. "GUIDO/MIR – an Experimental Musical Information Retrieval System Based on GUIDO Music Notation", *Proc. of the ISMIR*, Bloomington, USA, pp. 41–50, 2001.

[8] Hu, N., Dannenberg, R.B. "A Comparison of Melodic Database Retrieval Techniques Using Sung Queries", *Proc. of the ACM/IEEE JCDL*, Portland, USA, pp. 301-307, 2002.

[9] Lee, J.H. "Analysis of Multiple Evidence Combination", *Proc. of ACM-SIGIR Conference*, Philadelphia, USA, pp. 267–275, 1997.

[10] Meek, C., Birmingham, W. "Automatic Thematic Extractor", *Journal of Intelligent Information Systems*, Kluwer Academic Publishers, Vol. 21(1), pp. 9–33, 2003.

[11] Melucci, M., Orio, N. "Musical Information Retrieval Using Melodic Surface", *Proc. of 4th ACM Conference on Digital Libraries*, Berkeley, USA, pp. 152–160, 1999.

[12] Melucci, M., Orio, N. "SMILE: A System for Content-based Music Information Retrieval Environments", *Proc. of International Conference RIAO*, Paris, France, pp. 231-246, 2000.

[13] Melucci, M., Orio, N. "Evaluating Automatic Melody Segmentation Aimed at Music Information Retrieval", *Proc. of the ACM/IEEE JCDL*, Portland, USA, pp. 310-311, 2002.

[14] The MIR/MDL Evaluation Project White Paper Collection, `http://music-ir.org/evaluation/wp.html`, visited on July 2004.

[15] Orio, N., Sisti Sette, M. "A HMM-Based Pitch Tracker for Audio Queries", *Proc. of the ISMIR*, Baltimore, USA, pp. 249-250, 2003.

[16] Shifrin, J., Pardo, B., Meek, C., Birmingham, W. "HMM-Based Musical Query Retrieval", *Proc. of the ACM/IEEE JCDL*, Portland, USA, pp. 295-300, 2002.

[17] Pienimäki, A. "Indexing Music Database Using Automatic Extraction of Frequent Phrases", *Proc. of the ISMIR*, Paris, France, pp. 25–30, 2002.

[18] TREC. Home page of Text REtrieval Conference, `http://trec.nist.gov/`, visited on July 2004.