Muugle: A Modular Music Information Retrieval Framework

Martijn Bosma, Remco C. Veltkamp, Frans Wiering

Utrecht University

Padualaan 14, De Uithof

3584 CH Utrecht, The Netherlands

{mbosma, remco.veltkamp, frans.wiering}@cs.uu.nl

Abstract

Muugle (Musical Utrecht University Global Lookup Engine) is a modular framework that allows the comparison of different MIR techniques and usability studies. A system overview and a discussion of a pilot usability experiment are given. A demo version of the framework can be found on *http://give-lab.cs.uu.nl/muugle*.

Keywords: Music Information Retrieval, Framework

1. Introduction

Recently, many MIR systems have been developed with many differences between them [7]. For example, the representation of the music data on which they operate varies and the collections on which they are tested are different. Not all systems handle polyphonic data and some ignore note durations. They thus operate under different circumstances. To make a methodical comparison of their performance possible, the underlying techniques of these systems need to operate in the same framework.

Frameworks such as RUBATO [5] and M2K [3] are component based, which in principle makes them suitable for the comparison of different techniques. However, RUBATO is designed for the generation of performance data from score and not for MIR purposes. Therefore it contains no matching components. M2K is mainly a development environments for the rapid prototyping of MIR systems. In [4] an experimentation framework for comparing similarity algorithms is described. However, this system and M2K are not designed as frameworks for usability experiments.

To our knowledge, there is still no framework that allows the comparison of different feature extraction and similarity methods. Our aim with Muugle (Musical Utrecht University Global Lookup Engine) is to provide such a framework.

2. Architecture

A retrieval process starts with the query formulation in the user interface and it leads to the presentation of the most
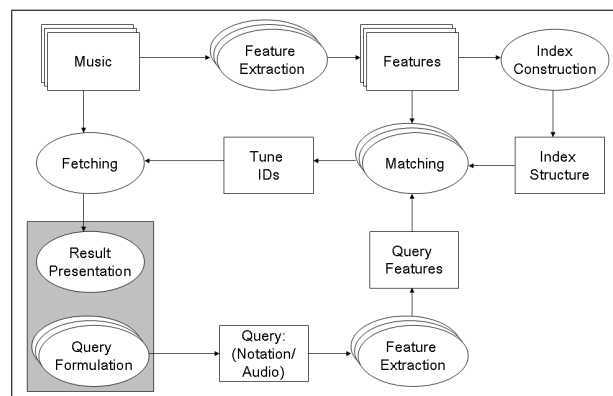
Figure 1. Overview of the Muugle-system architecture.

similar music present in the database. Figure 1 shows how this process is modelled in the architecture of the Muugle framework. The arrows in the diagram represent flows, the ellipses processes and the boxes data.

The data set can be a local music collection on hard-disk, or music found online by a web-crawler. As a first instantiation we have filled a database with 2 music collections, namely 815 ringtones, and 476.621 incipits of the RISM collection [2].

Features are extracted and stored in the database's tables. Each matching component may require a specific feature set to be extracted from the data. We extracted chunks, each consisting of six consecutive notes in the ringtones and five in the RISM collection. Chunks overlap resulting in 28.483 chunks in the ringtone and 4.664.702 in the RISM collection. For each note the onset, pitch and duration are stored.

Comparing the features of the query with those of all tunes in the database can be a time-consuming task. Therefore an index data structure is constructed. We have indexed the chunks with the vantage indexing method [9]. The distances between all database tunes and some vantage objects are pre-calculated. The set of tunes that have approximately the same distance to the vantage objects as the query has to these vantage objects, contains also those objects that have about the same distance to the query object. At query time a matching component only has to calculate the possibly complex distance between the query and the vantage objects, after which an efficient range search among the pre-computed distances can be done.
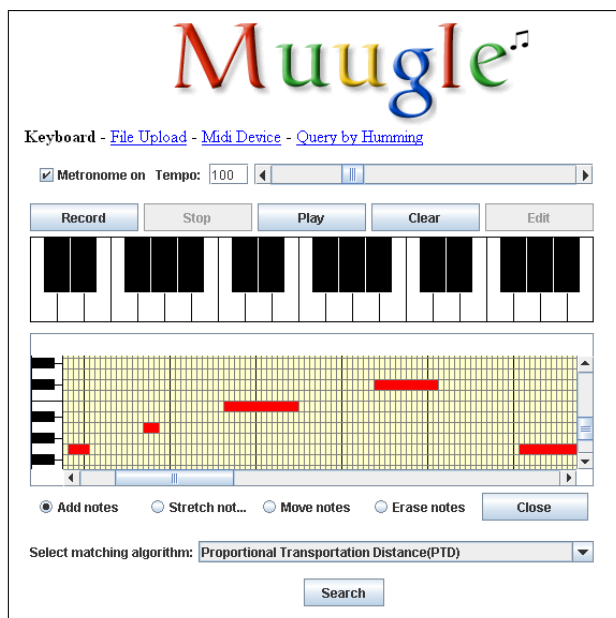
Figure 2. Muugle interface with piano-roll editor.

Generally, queries are in an audio or symbolic format. Muugle currently provides four different interfaces designed for query formulation by playing a software keyboard, uploading a MIDI file, playing an external MIDI device, or by Query By Humming. Figure 2 depicts the software keyboard interface which can be played with the mouse. The query can be modified (or generated from scratch) by means of a piano-roll editor.

Appropriate features must be extracted from the query for matching. The user-selected matching component uses the features of the query and those of the tunes in the database to arrive at a similarity judgment. Currently, four matching components are implemented in Muugle, which all operate on the note level. Three of these components use transportation distances, which are the *Earth Mover's Distance* (EMD), the *Proportional Transportation Distance* (PTD) [6] and the *Combined EMD-PTD*. The combined EMD-PTD component calculates the EMD and the PTD distance and returns the minimum of the two. The fourth component uses the *Maximum Overlap* which is one of the geometrical algorithms developed for the C-Brahms project [8].

The output of the matching component is a list of tune ids, ordered according to the tunes' similarity. The fetching module receives this list and retrieves the data of the corresponding tunes. Finally, an ordered list of tune references, with for every tune a notation of its first few bars is presented to the user.

The programming languages used for this instantiation of Muugle are MySQL (database), Java (user interfaces), C and C++ (matching and indexing), Perl (feature extraction) and PHP (fetching and result presentation).

3. Discussion and Future Work

Muugle is a framework that integrates the study of several aspects of MIR. Its modular architecture is designed for testing and comparing input methods, result presentations and MIR algorithms. Currently the focus is on symbolic representations of music, but it is possible to incorporate audio components.

We performed a pilot experiment to compare the different input methods [1]. The subjects had to formulate a query that was similar to a melody they just had heard. For each query they used one of the input methods described above. The results indicated that the Query by Humming method was slightly better, and that subjects with prior knowledge of MIDI benefited from the piano-roll editor. A larger experiment is needed to give support to these indications.

We believe that the performance of a MIR system will improve if it operates on higher-level features that are relevant to human music perception and cognition. Although these features, such as key and metric structure, are important, extracting them from the raw query does not seem to be feasible. It seems that people first of all try to capture the basic contour of the melody they are searching. Therefore we have started investigating the possibility of relevance-feedback by the user. The idea is that feedback is given on the results of a rough contour based search. In a second search, key and metric structure are used.

References

[1] M. Bosma, R.C. Veltkamp, and F. Wiering. Muugle: A music retrieval experimentation framework. In *ICMPC*, 2006.

[2] Repertoire International des Sources Musicales RISM. 2002. http://rism.stub.uni-frankfurt.de.

[3] J.S. Downie, J. Futrelle, and D. Tcheng. The international music information retrieval systems evaluation laboratory: Governance, access and security. In *ISMIR*, 2004.

[4] J. Garbers. An environment for testing and comparing music analysis algorithms. In *ISMIR*, 2006.

[5] G. Mazzola and O. Zahorka. The RUBATO performance workstation on NEXTSTEP. In *ICMC*, 1994.

[6] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R. van Oostrum. Using transportation distances for measuring melodic similarity. In *ISMIR*, 2003.

[7] R. Typke, F. Wiering, and R.C. Veltkamp. A survey of music information retrieval systems. In *ISMIR*, 2005.

[8] E. Ukkonen, K. Lemström, and V. Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *ISMIR*, 2003.

[9] J. Vleugels and R. C. Veltkamp. Efficient image retrieval through vantage objects. *Pattern Recognition*, 35(1), 2002.