# Audio-Based Music Similarity and Retrieval: Combining a Spectral Similarity Model with Information Extracted from Fluctuation Patterns

**Elias Pampalk***
National Institute of Advanced Industrial Science and Technology (AIST)
IT, AIST, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan

October 7, 2006

## Abstract

This paper describes the implementation submitted by the author to the MIREX'06 (Music Information Retrieval eXchange) evaluation track on audio-based music similarity and retrieval. In addition, this paper summarizes the optimization of this implementation and its evaluation prior to submission. Finally, a detailed analysis and discussion of the MIREX results is presented. Overall, this implementation performed slightly better in terms of quality and computation time than the other implementations. However, the measured differences were not significant.

## 1. Introduction

The perception of music similarity is subjective, context-dependent, and multi-dimensional (including instrumentation, harmony, melody, rhythm etc.). Nevertheless, the basic approach of this implementation is one-size-fits-all. In particular, given any two songs, without any further context, one number is computed.

There are mainly two reasons for focusing on such obviously over-simplistic approaches. First, there are applications where one-size-fits-all can be applied such as automatic playlist generation. Second, evaluating models which change their similarity rankings depending on the respective context is significantly more complex.

### 1.1. Evaluation

The optimal approach to compare the performance of computational models of music similarity is to evaluate them within the context of their application. For example, one approach could be to ask users which similarity model generates the best playlists.

However, empirical results suggest that even without a specific application context similarity ratings can be evaluated consistently by human listeners. For example, Logan & Salomon [1] presented results from a listening test where two subjects were asked to judge if a given song is similar to another (yes/no). The subjects disagreed in only 12% of the cases.

A similar consistency of judgments is reported in [2]. In particular, a listening test was conducted where the subjects were given a song (X) and asked to rate its similarity to two other songs (A and B) on a scale from 1 to 9. Thus, each subject was asked for two numbers given three songs: the similarity of AX and the similarity of BX. The consistency of the ratings from different subjects were compared in terms of the difference $d_i = \mathrm{AX}_i - \mathrm{BX}_i$ which was computed for each subject $i$. The results showed that this difference was surprisingly consistent. In 26% of the cases two subjects $i$ and $j$ had the same values ($d_i = d_j$). In 32% of the cases the difference was only 1 point ($|d_i - d_j| = 1$). In about 19% of the cases the difference was 2 points ($|d_i - d_j| = 2$). Only in a few cases the listeners truly disagreed (in 15% of the cases ($|d_i - d_j| \geq 4$).

A simpler alternative to conducting listening tests is to use a genre classification approach (e.g. [1, 3, 4]). The basic assumption is that given a piece of music, very similar pieces can be found within the same genre. However, special precautions need to be taken such as applying an "artist filter" and using different collections with different taxonomies (for a detailed discussion see [2]).

### 1.2. Related Work

All details of this implementation are given in [2], where the implementation (Matlab source code), optimization, and evaluation (including a listening test) are described.

The approach of combining a spectral similarity model with information from fluctuation patterns is based on previous work presented in [4]. A similar version which was optimized with respect to computation time was submitted to the genre classification track of MIREX'05 [5] where it (despite only using a nearest neighbor classifier) outperformed a number of powerful classifiers (such as support vector machines).

The spectral similarity part of this implementation is based on the work of Mandel & Ellis [6]. However, alternative approaches developed by Logan & Salomon [1] or Aucouturier & Pachet [3] could be applied as well. The main advantage of the approach used by Mandel & Ellis is that it is computationally very fast.

The fluctuation patterns were presented in [7, 8] and are based on previous work by Früwirth & Rauber [9, 10]. The "gravity" descriptor extracted form the fluctuation patterns was presented in [4]. The "bass" descriptor is based on

*) Part of this work was done while the author was working at the Austrian Research Institute for Artificial Intelligence (OFAI).

work presented in [7] and was slightly modified as described in [2]. The specific implementation of the fluctuation patterns (e.g. using MFCCs) used for this implementation to compute the fluctuation patterns was first described in [5].

## 2. Implementation of G1C

G1C stands for "Single Gaussian Combined". The implementation submitted to MIREX is described in detail in [2] including the Matlab code. In this section the techniques are only roughly summarized.

First, for each piece of music the MFCCs are computed for a maximum of two minutes from the center of the piece. In particular, a 19-dimensional MFCC vector for every 23ms of the signal is computed. The distribution of these vectors is summarized using a single Gaussian (G1) with full covariance matrix. The distance between two Gaussians is computed using a symmetric version of the Kullback Leibler divergence.

The fluctuation pattern (FP) describes the modulation of the loudness amplitudes per frequency bands. To some extent it can describe periodic beats. A FP is a two-dimensional matrix where each row corresponds to a frequency-band and each column to a modulation frequency (in the range of 0-10Hz). The values of this matrix describe how strong the fluctuation of the loudness amplitude is within a specific frequency band and at a specific modulation frequency.

To compute the fluctuation patterns the Mel spectrogram (with loudness in dB) is used. The Mel spectrogram is obtained in an intermediate step when computing the MFCCs. In the next step the energy in each frequency bands is regrouped into 12 bands. This re-grouping is done such that variations in lower frequency bands are emphasized. The Mel spectrogram is then chopped into 3 second segments. For each segment, the loudness modulation in each frequency band is computed using an FFT. The modulation frequencies are analyzed in the range of 0-10Hz. The modulation amplitudes are weighted to emphasize modulation around 4Hz based on a model of fluctuation strength [11, 12]. Specific modulation patterns are emphasized using smoothing and edge detection filters. All fluctuation patterns computed for each 3 second window are combined by computing the median of all patterns. The patterns of two pieces of music are compared by computing the Euclidean distance (and first converting the matrix into a vector by concatenating the rows).

From the FP of each song two features are extracted. One is the "gravity" (FP.G). It is the center of gravity of the FP along the modulation frequency dimension. It roughly corresponds to the perception if a piece is slow or fast. The other is the "bass" (FP.B). It is computed as the fluctuation strength of the lower frequency bands at higher modulation frequencies. The distance of two songs for each of these descriptors is computed as the absolute difference of values.

Given the four distance values (for G1, FP, FP.B, and FP.G) the overall similarity of two pieces is computed as a weighted linear combination. The normalization and weights used are described in detail in [2]. If the inversion of the covariance necessary for G1 leads to numerical problems for any song, then the combination weight is set to zero when it is compared to any other song. The optimization and evaluation of the weights is briefly described in Section 3.

### 2.1. Computational Resources

The following computation times are measured running Matlab code (Windows XP) on a Pentium M 2GHz processor. Given an audio file in WAV format extracting the features for one piece of music takes about 2 seconds. Computing the spectral similarity of two songs takes about 0.1 milliseconds. The FP part of the similarity computation is much faster as it only requires computing the Euclidean distance of two vectors with 362 elements each. For each song a total of $362 + 2\times19\times19+19$ (=1103) values need to be stored. (In addition to the G1 covariance matrix also the inverted covariance matrix is stored so it does not need to be recomputed for each similarity computation.) For a comparison of computation times see Section 4.5 and Figure 6.

## 3. Optimization and Evaluation

To optimize the combination weights and to evaluate G1C a genre-based evaluation procedure was used. In particular, given a music collection containing pieces for which the genre and artist is known the following steps were computed: First, for each piece (query) all pieces from the same artist in the collection are removed. Second, the piece most similar to this query is found (according to the similarity model). If this piece is from the same genre, the score for the query piece is 1, otherwise 0. Finally, the average score for all queries is computed. This is identical to nearest neighbor genre classification, measuring the performance using leave-one-out cross-validation, and using an artist filter to ensure that training and test set contain non-overlapping sets of artists.

The combination weights of G1C were optimized using two music collections (DB-MS and DB-L described in [2]). The parameter space was evaluated using a grid search in combination with the genre-based evaluation approach. The combination which performed best in average on the two music collections, was evaluated using 4 collections (DB-S, DB-ML, DB-30, DB-XL). In addition a listening test was conducted to analyze how the genre-based results are related to judgments made by human listeners. The results of this test confirmed that improvements measured with the genre-based procedure are also measurable using a listening test. (For details see [2].)

## 4. MIREX'06 Results

The raw data and details of the evaluation procedure can be found on the MIREX pages.[1] This section briefly describes the listening test setup and analyzes the results.

[1] http://www.music-ir.org/mirex2006/index.php

### 4.1. Listening Test Setup

To evaluate the performance of the algorithms a user study was conducted by IMIRSEL.[2] First, each of the 6 algorithms submitted to the contest retrieved the 5 most similar songs form the database given the query. Songs from the same artist as the query were filtered. The main reason for filtering songs from the same artist is that the intention was not to evaluate how well the submissions perform in artist identification.[3] Overall 60 queries were used for the listening test. Each candidate and query pair was rated by three subjects.

Only researchers working on related topics participated in the listening test. This restriction was necessary due to legal issues.[4] The participants were given a query song and a list of 30 candidate songs and asked to rate the similarity of each candidate song on a scale from 0-10 using a slider (with a resolution of 0.1) where 0 corresponds to not similar and 10 corresponds to very similar (fine scale). Instead of using the slider the participants could directly enter a number into a text box. Overall, 32% of the fine score ratings the participants entered are whole numbers (0,1,2,...) and 16% are half numbers (0.5,1.5,...). In addition, the participants were asked to rate each of the 30 songs on a broad scale with the options: not similar (NS), somewhat similar (SS), and very similar (VS).

### 4.2. Official Ranking

The official ranking of the algorithms was computed using the data from the slider (fine scale). For each query and algorithm a score was computed. This score is the mean of the 15 (=3$\times$5) similarity ratings (each in the range 0-10) associated with each query/algorithm pair. In the next step the Friedman test is computed, the results of the Friedman test are then post-processed to find significant differences between algorithms. The corresponding Matlab code is:

```
[p,table,stats] = friedman(M);
multcompare(stats, ...
              'ctype', 'tukey-kramer', ...
              'estimate', 'friedman', ...
              'alpha', 0.05);
```

where `M` is a matrix with 60 rows (corresponding to the queries) and 6 columns (corresponding to the algorithms). The Friedman test was chosen because it is a non-parametric test which does not assume a normal distribution of the data.

For each query the Friedman test ranks the algorithms with respect to their scores. If an algorithm is consistently ranked higher than another one, then it is significant better. On the other hand, if the algorithm A scores better for half of the queries and algorithm B for the other half, then the difference is not significant. (Note that this is the case even if algorithm B has much higher scores in average.) Estimating the significance of differences is very important as some difference are very small and wrong conclusions could easily be drawn.

[2] http://www.music-ir.org/evaluation/

[3] For a discussion on the relationship between artist identification and music similarity see [5].

[4] For details see the email by Stephen Downie on the MIREX list on September 2, 2006.

| | Mean | Median |
|---|---|---|
| EP | 4.30 | 4.05 |
| TP | 4.23 | 4.05 |
| VS | 4.04 | 3.50 |
| LR | 3.93 | 3.70 |
| KWT* | 3.72 | 3.40 |
| KWL* | 3.39 | 2.95 |

Table 1. Statistics of the average fine score per algorithm. The minimum value is 0 (not similar) and the maximum is 10 (very similar).

The results are shown in Figure 1 (left side). Note that EP marks the G1C algorithm and "*" marks all submissions that contain bugs according to their authors. Most of the measured differences between algorithms are not significant according to the Friedman test (at p-level 0.05). Only KWL* performs significantly worse than some others.

### 4.3. Other Fine Scale Results

The right side of Figure 1 shows the results when the median is computed instead of the mean of the 15 ratings to obtain the score for each query/algorithm pair. For some query/algorithm pairs the differences between the two approaches can be very large. For example, the in contrast to the mean the median would not distinguish between 1,2,9 and 1,2,3. The advantage of using the median is that it is less sensitive to outliers. However, as can be seen the significance of the measured differences remains the same.

Figure 2 shows the results when using a balanced two-way ANOVA instead of the non-parametric Friedman test. Although the necessary assumptions are not perfectly met the results are very similar to those of the Friedman test.

Figure 3 shows the distribution of scores per query for each algorithm. As can be seen, when each of the 60 scores is computed as the mean of the respective 15 observations the normal distribution is a better approximation than in the case where the median is used.

Table 1 shows statistics of all ratings. The average ratings are lower than those reported in [2] where the mean ratings was 6.37 (one a similar scale from 1-9). The most likely reason for the difference is the that a different music collection was used. As shown in [2] the performances vary largely depending on the collection. In any case, the average values indicate that the overall quality of the similarity measures might not be satisfactory for users. To evaluate this would require tests within the application context.

### 4.4. Broad Scale Results

Figure 4 (left side) shows how the broad scale ratings are distributed per algorithm. Overall, G1C got the largest num-

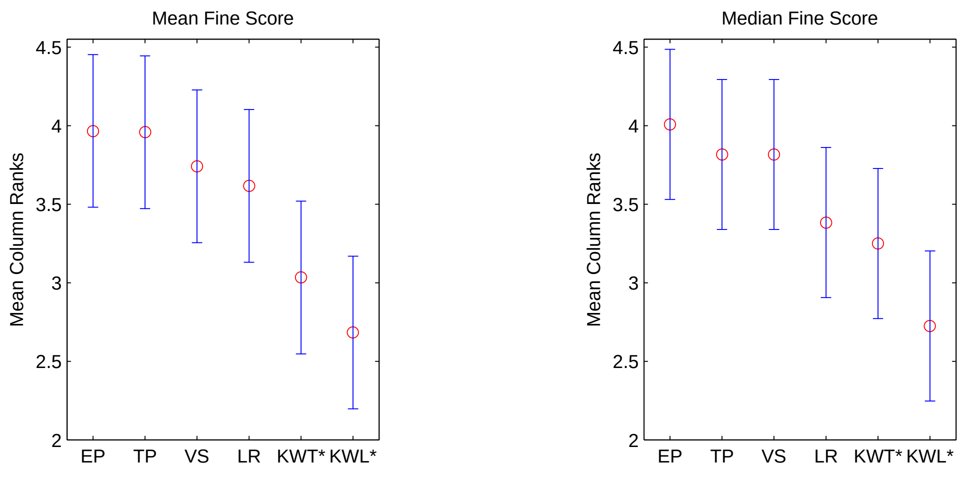


Figure 1. Evaluation results using the Friedman test. The left side shows the final ranking of the contest. The red circles mark the mean ranks computed using the Friedman test. The blue lines mark the significance bounds using a level of p=0.05. The right side differs from the left side with respect to how the score was computed per query/algorithm pair. In particular, the median was used instead of the mean.

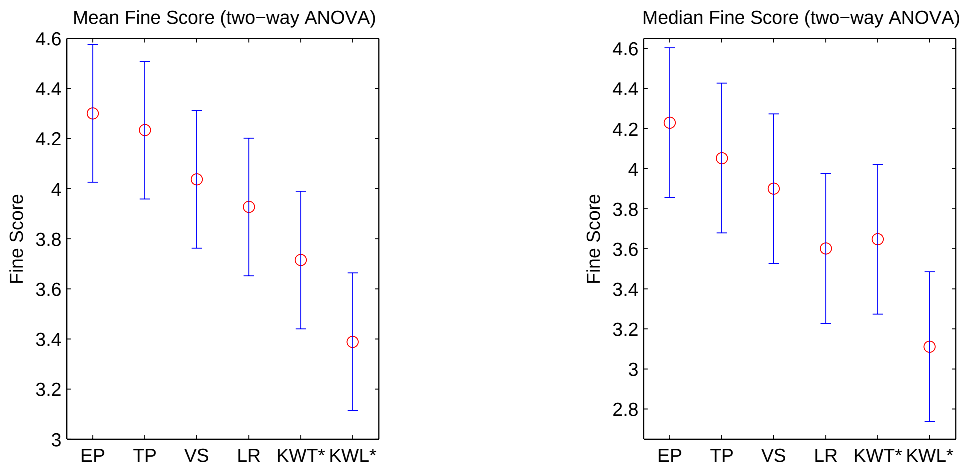


Figure 2. Results using a balanced two-way ANOVA instead of the Friedman test (see Figure 1). The left side uses the mean and the right side uses the median to compute the score for each query/algorithm pair given the 15 observations.

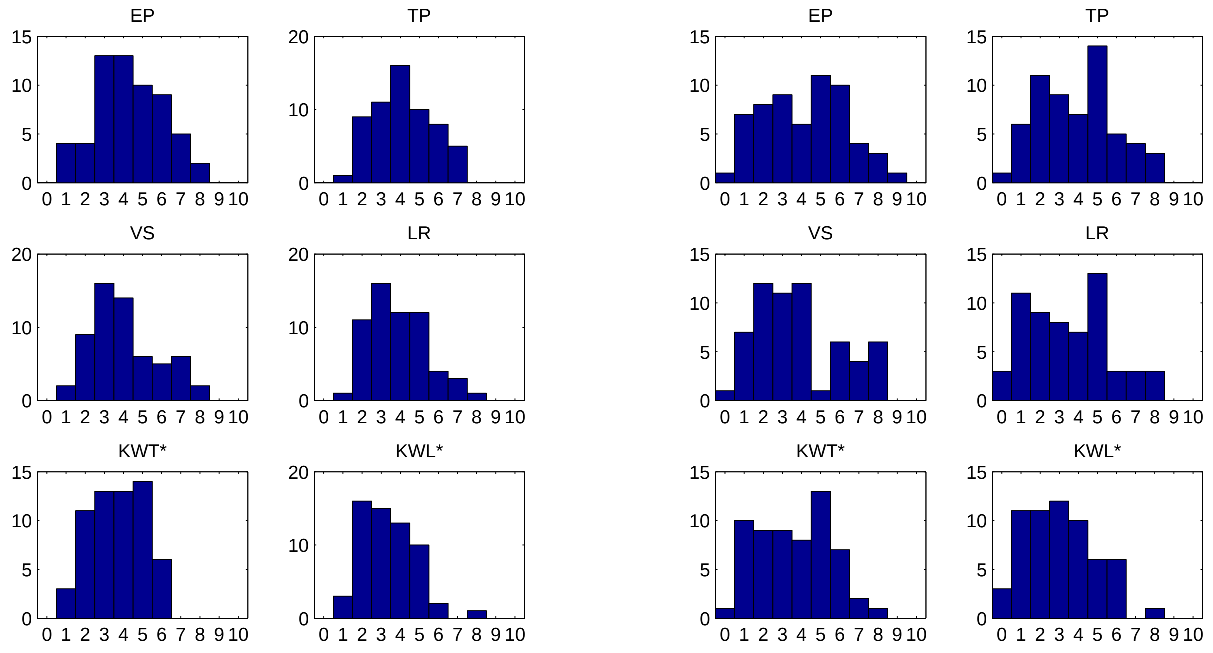


Figure 3. Histogram of the 60 scores per algorithm. On the left side the scores are computed as the mean of all 15 observations (subject ratings) and on the right side as the median.

ber of very similar ratings (VS) and about the same number of not similar ratings (NS) as TP. The right side of Figure 4 shows a similar visualization using the fine scale data. As can be seen, both scales have produced very similar results except that the fine scale resulted in more fine grained distinctions.

The MIREX webpage lists a number of statistics computed based on the broad scale data. There are a number of different ways to interpret the broad scale data. Each interpretation assigns a different number of points to each similarity category. The optimal interpretation depends very much on the final application. In the same way the broad categories are assigned to different points the fine scale could be mapped to a non linear scale to emphasize certain areas of the scale differently.

Given a specific interpretation the average of points assigned to an algorithm per query is used for further analysis. The statistics are listed in Table 2 and Figure 5 shows the corresponding significance boundaries using the Friedman test (and the mean to compute the scores).

### 4.5. Other Statistics

In addition to the results of listening test a number of other statistics were computed. These are based on a collection of 5000 pieces which included 330 cover songs. For some of the statistics reported here, the cover songs were filtered from the collection. In general the statistics are based on computing the compute distance matrix (5000$\times$5000 values).

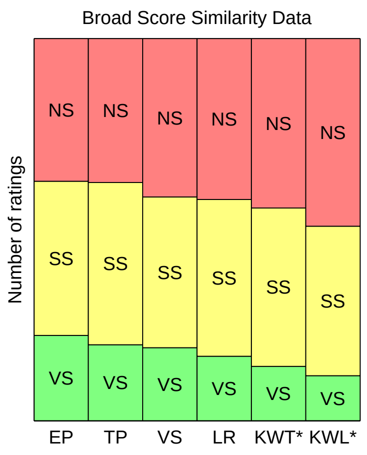
#### *4.5.1. Computation Times*

Figure 6 shows the computation times for some of the submissions. When interpreting the results it is necessary to consider that some of the algorithms have not been optimized with respect to computation times. However, the measured computation times are rough indicators of the complexity of each algorithm.

Overall G1C is fastest to extract the features from the 5000 songs and compute the complete distance matrix. G1C requires only about half the time to extract the features than the next fastest submission. Lidy & Rauber (LR) submitted by far the fastest algorithm with respect to the distance computation time. Depending on the task this can be a significant advantage. However, to improve the distance computation time of G1C and TP, for example, M-trees [13] and other indexing strategies could be used. In particular, for most applications it is not necessary to compute a complete distance matrix.

#### *4.5.2. Characteristics of the Similarity Spaces*

Most similarity space indexing algorithms make assumptions about the similarity space. To study the properties of the similarity space three aspects were analyzed for those algorithms which allowed computation of the complete dis-



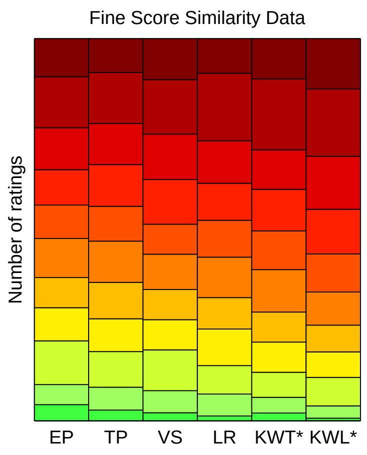


Figure 4. The left side shows the number of times per algorithm the songs were rated with each of the broad similarity scale categories. The categories on the broad scale are: not similar (NS), somewhat similar (SS), and very similar (VS). The right side uses the data from the fine scale. For each algorithm the lowest block (green) corresponds to a rating of 10, the highest block (red) to a rating of 0. All ratings are rounded to the nearest whole number.

| | Points per Category | | | Scores | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Abbreviation | NS | SS | VS | EP | TP | VS | LR | KWT* | KWL* |
| Greater0 | 0 | 1 | 1 | 62.7 | 62.3 | 58.6 | 57.9 | 55.7 | 50.9 |
| Psum | 0 | 1 | 2 | 42.5 | 41.1 | 38.8 | 37.4 | 34.9 | 31.3 |
| Wcsum | 0 | 1 | 3 | 35.8 | 34.0 | 32.3 | 30.6 | 28.0 | 24.8 |
| Sdsum | 0 | 1 | 4 | 32.4 | 30.5 | 29.0 | 27.1 | 24.6 | 21.6 |
| Greater1 | 0 | 0 | 1 | 22.3 | 19.9 | 19.1 | 16.9 | 14.2 | 11.8 |

Table 2. Evaluation scores using the broad scale data for different interpretations of the data. The values are normalized so that the scale ranges from 0 (complete failure) to 100 (perfect). The general tendency is the same regardless of the points per category.

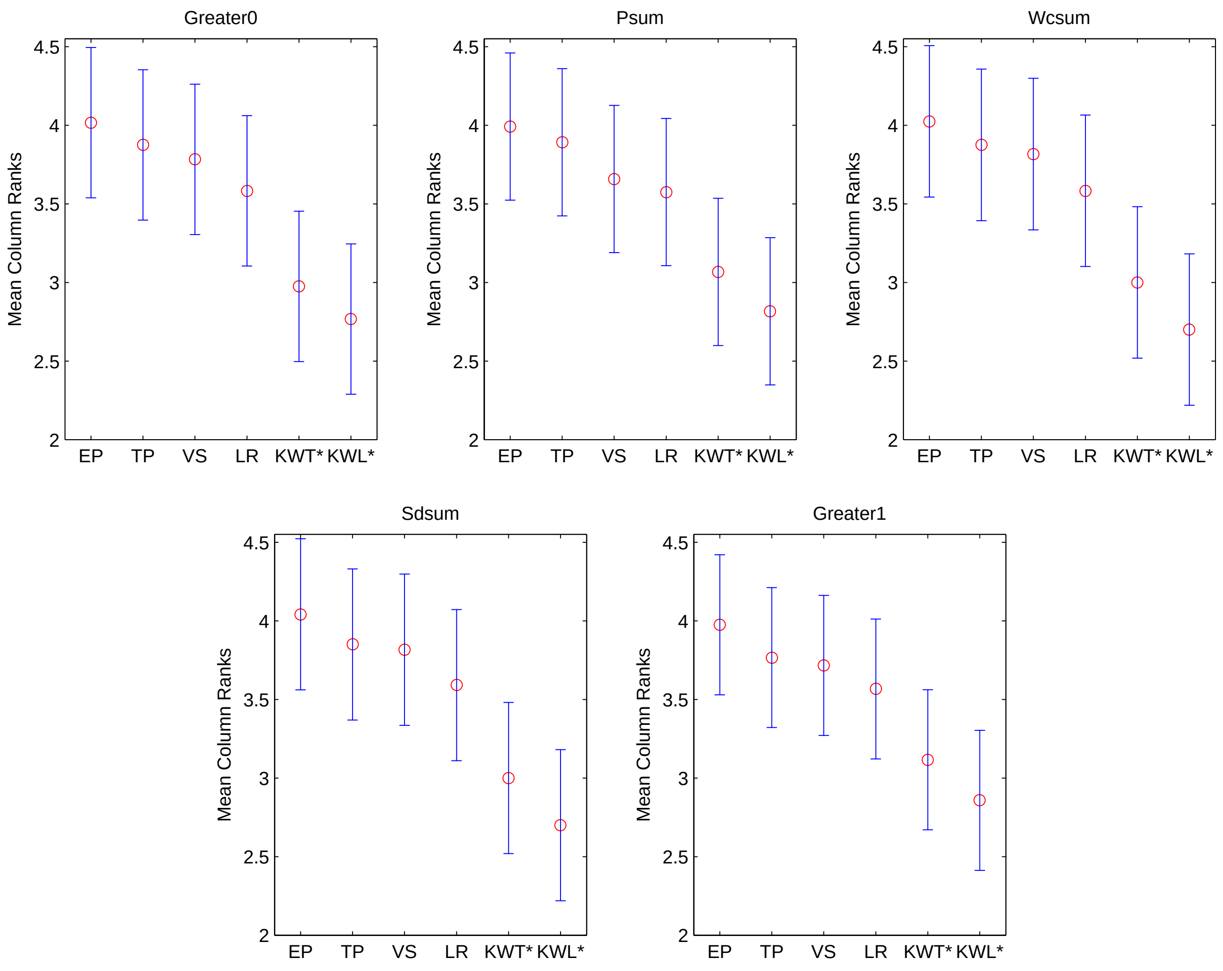


Figure 5. The same evaluation results as shown in Figure 1 except that the broad score (with different interpretations) is used instead of the fine score. In terms of the insignificance of the differences the results are similar to those using the fine score data.

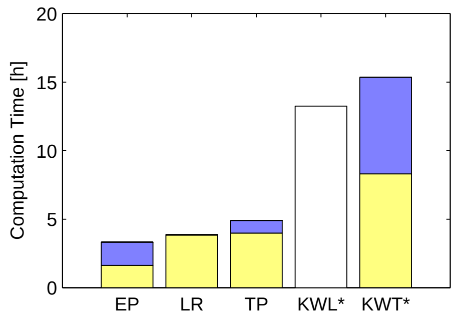


Figure 6. Computation times for some of the submissions. The lower part of each bar (yellow) is the feature extraction time (for 5000 songs). The upper part (blue) is the distance computation time for the complete distance matrix (which requires computing the distance of 12.5 million song pairs). For KWL* the times for the individual parts were not recorded. The VS submission was not able to compute the full distance matrix within a reasonable amount of time. The times were measured on a machine with: Dual AMD Opteron 64, 1.6 GHz, 4 GB RAM, running Linux (CentOS).

tance matrix within reasonable time. [5]

First, the problem of "always similar songs" (also known as "hubs") was analyzed. Hubs in music collections were first reported by Auctouturier & Pachet [14]. A detailed discussion can be found in [15]. A hub is a song that (according to the computational model of similarity) is very similar to a large number of other songs. However, this computational similarity does not correspond to perceptual similarity.

Some similarity measures (such as those based on spectral similarity) are affected. However, the number of these hubs is usually very low, and in a collection of several thousand pieces only few can be observed. Hubs are easy to detect when analyzing a distance matrix, however, they are difficult to detect when only computing a number given two songs. Tim Pohle's submission uses an interesting approach to prevent extreme hubs.

Table 3 shows the maximum number of times a song appeared in the top 5 ranks. The main observation is that in the subset of cover songs one "always similar" song appeared for G1C but not for the other submissions. The existence of "always similar" outliers for G1C was also documented in [2].

Very closely related to the analysis of "always similar" songs is the question if there are any songs which never occur in the top 5 rankings. In the collection used for this contest, no cases of "always dissimilar" songs were found. However, as shown in [2] songs which are dissimilar to almost all songs in the collection (including songs which sound similar) can occur using G1C.

Of interest is also if the triangular inequality holds in the

[5] The analysis was implemented by Kris West.

| Collection Size | EP | TP | LR | KWT* | KWL* |
|---|---|---|---|---|---|
| 4670 | 48 | 62 | 42 | 61 | 24 |
| 5000 | 1753 | 62 | 42 | 61 | 24 |

Table 3. Maximum number of times a song was ranked in the top 5 most similar songs of all songs. The lowest possible value is 5. The highest possible value equals the size of the collection minus 1.

similarity space the submission defines. The triangular inequality states that the sum of the distances AX and BX is larger or equal to the distance AB (in our case A, B, and X are songs). This inequality is an important characteristic of metric spaces and a number of algorithms (especially indexing algorithms) rely on it. To measure the degree to which the submissions fulfill the inequality a random sample of triangles is drawn from the distance matrix. Each of these triangles is tested whether the inequality is fulfilled. For the submissions G1C, KWT*, and LR the inequality held for all samples drawn. For TP the inequality held in about 32% of the cases, and for KWT* in about 54%.

In [2] a different music collection is used and G1C fulfills the inequality only in 36% of the cases. One possible explaination could be that the submitted version of G1C is not exactly the same as the one used on [2]. The only difference is that the contribution of the spectral similarity is set to zero if nummerical problems occur when computing the inverse covariance matrix. However, nummerical problems only occur very seldomly.

#### 4.5.3. *Genre-based Evaluation*

Previous work has shown that evaluations based on genre data correspond to evaluations based on similarity ratings gathered in listening tests [2]. However, the genre data available for the music collection used in this contest was not reliable. For example, Britney Spears and Depeche Mode are assigned to the same genre (rock). Furthermore, the distribution of songs per genre is very unbalanced. This is reflected in the results computed using an artist filter which do not reflect the results from the listening test.

The results without using an artist filter are more useful, because they basically measure the artist identification performance. However, identifying artists and finding similar pieces are not the same tasks. For example, an algorithm that can identify recording environments or other production effects might perform very well for artist identification but not for music similarity. for a more detailed discussion on this see [5, 2].

## 5. Discussion

There are two reasons why the measured differences (using the listening test data) were not significant. First, the differences between the algorithms are very small. Second, the evaluation procedure was not adequate to measure these small differences.

### 5.1. Glass Ceiling

Aucouturier and Pachet pointed out a glass ceiling for spectral similarity [14]. The results of this listening test have confirmed this ceiling. In particular, G1C is only marginally better than the author's submission to the MIREX 2005 and ISMIR 2004 genre classification contests [2]. The main improvements from 2004 and 2006 have been in terms of computation time which has been reduced by several magnitudes.

Thus, it is not surprising that some of the submissions have reached the same glass ceiling. In particular, the overall difference between G1C and TP is very difficult to measure.

### 5.2. Evaluation Procedure

A very straightforward approach to increase the power of the test (and allow us to measure significant differences) would have been to use a larger number of queries. To reduce the overall load on the subjects, fewer subjects per rating and fewer candidates per algorithm and query could be used. Thus, more than 10 times as many queries could have been used in the evaluation without increasing the effort per subject.

Using fewer candidates per algorithm and query would also have the advantage that the size of the local context would be reduced. The local context is the context in which the subjects rate the songs. This context consists of the query, and the 30 candidates to be rated. A smaller local context is likely to lead to more consistent ratings.[6] For example, in [2] the local context consisted of only 3 songs and the consistency was higher. However, when using a larger context (e.g. when evaluating several algorithms) the subjects should be given tools to help them apply ratings consistently. One such option would be to implement a sort function for the ratings.

Figures 7 and 8 visualize the consistency of the ratings. Figure 9 shows the consistency of the ratings from the listening test described in [2]. The consistency is computed as follows. For each query and candidate pair ($60\times30$) there are 3 ratings (by three different subjects). We compute the absolute differences between these ratings. In total this results in $60\times30\times3$ absolute differences. In an ideal case all of these would be zeros. In the worst case (worse than random) a large proportion of these values would be 10 (which is the maximum possible disagreement on the fine scale from 0-10).

The consistency can be quantified and compared using the ratios of pairs with a very high consistency. In particular, the first quarter of bins (using the histograms in Figures 7-9) is considered to be highly consistent. In case of the broad

[6] It is important to note that splitting the candidates per query into two or more sets is not a solution. If the candidates are rated in a different local context than the ratings are not comparable. As a result a test such as the Friedman test could not be used to evaluate the results.

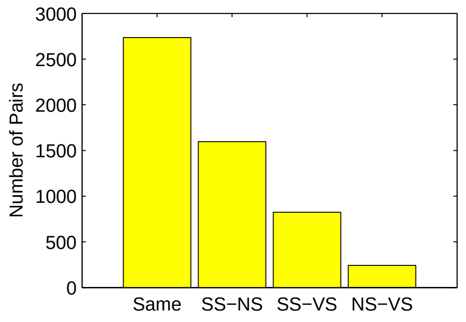


Figure 7. Histogram of absolute rating differences (broad score).

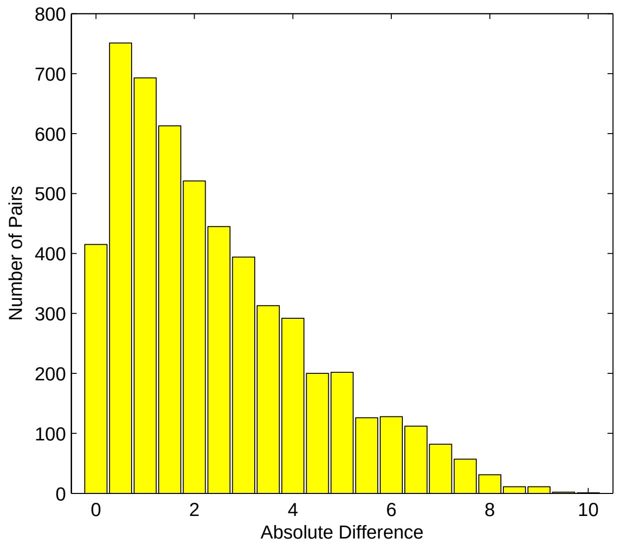


Figure 8. Histogram of absolute rating differences (fine score).

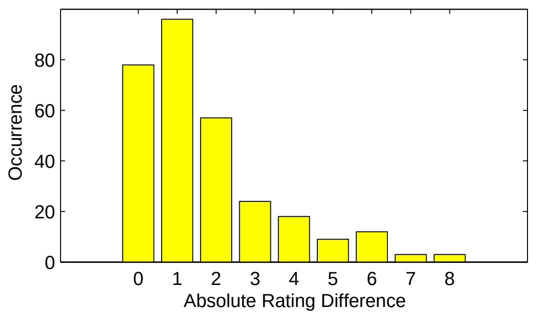


Figure 9. Histogram of absolute rating differences for the listening test reported in [2] where a scale from 1-9 was used.

| Evaluation Procedure | Consistent Pairs |
|---|---|
| Broad Scale | 51% |
| Fine Scale | 55% |
| Evaluation reported in [2] | 58% |

Table 4. Percentage of very consistent pairs of ratings for different evaluation procedures.

score data this means the ratio of exact same ratings compared to the number of all pairs. For the fine score data the first 5 bins are used. For the listening test conducted in [2] the first 2 bins are used. The results are given in Table 4. The results question the use of the broad scale and support the argumentation for using a smaller local context.

## 6. Conclusions

Future listening tests should use a larger number of queries. A fine scale is preferable to a broad scale because the ratings are more consistent. Furthermore, the size of the local context should be reduced to increase the consistency of the ratings.

However, even with improved evaluation procedures the differences between algorithms which have reached the "glass ceiling" are very marginal and might not be relevant for most applications. Evaluations within the context of applications are clearly desirable. A question of particular interest is if the similarity measures in their current form (despite their obvious limitations) can be successfully applied in any applications.

### 6.1. Conclusions of the Evaluation Results

G1C was fastest overall and achieved the highest score. However, the measured differences were not significant. The submission by Lidy & Rauber is the only algorithm which uses a vector space. This results in extremely fast distance computations and also allows the application of their submission to a larger number of problems. Their submission is probably the most suitable for extremely large collections (containing millions of pieces). Tim Pohle presented a very interesting approach which can also be applied to G1C to reduce the number of "always similar" outlier songs. Furthermore, the distance computation time for his submission is only about half of that for G1C which can be a major advantage for some applications.

## References

[1] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'01)*, 2001.

[2] E. Pampalk, "Computational models of music similarity and their application in music information retrieval," Docteral dissertation, Vienna University of Technology, Austria, March 2006.

[3] J.-J. Aucouturier and F. Pachet, "Music Similarity Measures: What's the Use?" in *Proc. of ISMIR*, 2002.

[4] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *Proceedings of the ISMIR International Conference on Music Information Retrieval*, 2005.

[5] E. Pampalk, "Speeding Up Music Similarity," in *Proceedings of the MIREX Annual Music Information Retrieval eXchange*, 2005.

[6] M. I. Mandel and D. P. W. Ellis, "Song-Level Features and Support Vector Machines for Music Classification," in *Proc. of ISMIR*, 2005.

[7] E. Pampalk, "Islands of Music: Analysis, Organization, and Visualization of Music Archives," MSc thesis, Vienna University of Technology, Department of Software Technology and Interactive Systems, 2001, http://www.ofai.at/˜elias/music/thesis.html.

[8] E. Pampalk, A. Rauber, and D. Merkl, "Content-Based Organization and Visualization of Music Archives," in *Proceedings of the ACM Multimedia*, 2002.

[9] M. Frühwirth, "Automatische Analyse und Organisation von Musikarchiven (Automatic Analysis and Organization of Music Archives)," MSc thesis, Vienna University of Technology, Austria, 2001.

[10] M. Frühwirth and A. Rauber, "Self-Organizing Maps for Content-Based Music Clustering," in *Proceedings of the Twelfth Italian Workshop on Neural Nets (WIRN01)*, 2001.

[11] E. Terhardt, "Über akustische Rauhigkeit und Schwankungsstärke (On the acoustic roughness and fluctuation strength)," *Acustica*, vol. 20, pp. 215–224, 1968.

[12] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*, 2nd ed., 1999.

[13] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces," in *Proceedings of the International Conference on Very Large Data Bases*, M. Jarke, M. Carey, K. R. Dittrich, F. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, Eds., 1997.

[14] J.-J. Aucouturier and F. Pachet, "Improving Timbre Similarity: How high is the sky?" *Journal of Negative Results in Speech and Audio Sciences*, vol. 1, no. 1, 2004, http://journal.speech.cs.cmu.edu/articles/2004/3.

[15] J.-J. Aucouturier, "Ten experiments on the modelling of polyphonic timbre," Docteral dissertation, University of Paris 6, Paris, France, May 2006. [Online]. Available: http://www.jj-aucouturier.info/papers/PHD-2006.pdf