Noname manuscript No.
(will be inserted by the editor)

A Fast Audio Similarity Retrieval Method for Millions of Music Tracks

Dominik Schnitzer · Arthur Flexer · Gerhard Widmer

Received: date / Accepted: date

Abstract We present a filter-and-refine method to speed up nearest neighbor searches with the Kullback-Leibler divergence for multivariate Gaussians. This combination of features and similarity estimation is of special interest in the field of automatic music recommendation as it is widely used to compute music similarity. However, the non-vectorial features and a non-metric divergence make using it with large corpora difficult, as standard indexing algorithms can not be used.

This paper proposes a method for fast nearest neighbor retrieval in large databases which relies on the above approach. In its core the method rescales the divergence and uses a modified FastMap implementation to speed up nearest-neighbor queries. Overall the method accelerates the search for similar music pieces by a factor of $10 - 30$ and yields high recall values of $95 - 99\%$ compared to a standard linear search.

Keywords Audio · Indexing · Music Recommendation

1 Introduction

Today an unprecedented amount of music is available on-line. As of January 2010, the Apple iTunes music store alone lists more than 11 million songs in its catalog. Other

Dominik Schnitzer
Austrian Research Institute for Artificial Intelligence (OFAI), Freyung 6/6, Vienna, Austria
Tel.: +43-1-5336112-21
Fax: +43-1-5336112-77
E-mail: dominik.schnitzer@ofai.at

Arthur Flexer
Austrian Research Institute for Artificial Intelligence (OFAI), Freyung 6/6, Vienna, Austria
Tel.: +43-1-5336112-25
Fax: +43-1-5336112-77
E-mail: arthur.flexer@ofai.at

Gerhard Widmer
Department of Computational Perception, Johannes Kepler University, Altenberger Str. 69, Linz, Austria
Tel.: +43-732-2468-1511
Fax: +43-732-2468-1520
E-mail: gerhard.widmer@jku.at

on-line music stores like Amazon MP3 offer a 9 million song catalog to choose from. With the catalog numbers constantly reaching record highs, the need for intelligent music search algorithms that provide new ways to discover, navigate and recommend music is critical.

Automatic content-based music recommendation systems usually operate with acoustic music similarity algorithms and work as a query-by-example system: (1) the user selects a song she/he likes, (2) the system searches its databases for similar songs, (3) according to the similarity measure the n-nearest neighbors are returned to the user as possible recommendations. This setup defines the very basic scenario this paper aims to address.

But to use music recommendation algorithms on these large databases, the search strategy usually needs to be adjusted to scale. General approaches to search high dimensional data utilize Binary space partitioning (BSP) trees, like Kd-Trees [4] or Vantage-Point Trees [32]. These work well for moderately high-dimensional features using common metrics. For very high-dimensional data Locality Sensitive Hashing (LSH, [1]) should be used as the afore mentioned algorithms are likely to perform worse or equal than a linear scan with very high dimensional data ("curse of dimensionality", [5]). LSH is an approximate nearest neighbor retrieval algorithm for the metric spaces L_1 and L_2, it scales sub-linearly in the number of items and comes with a probabilistic guarantee on accuracy. It is possible to use LSH for other non-metric divergences if the features can be embedded in the $L_{1,2}$ spaces.

A common method to compute acoustic music similarity uses timbre features represented as multivariate Gaussian distributions (cf. [22], [24] or [27]). Similarity is computed using the non-metric Kullback-Leibler (and related) divergences defined for Gaussian distributions.

Although this method for automatic acoustic music similarity is shown to have weaknesses [20], still systems using it are currently seen as the de-facto state of the art as they ranked first in the last four Music Information Retrieval Evaluation Exchange (MIREX[1], [12]) evaluations for *Audio Music Similarity and Retrieval*[2]. MIREX is a yearly community-based framework for the formal evaluation of Music Information Retrieval (MIR) systems and algorithms.

The high ranks in the listening tests and evaluations make the method a tempting and challenging target for broad usage in real applications.

1.1 Contributions

The contributions of this paper are four-fold:

- First, we present a filter-and-refine method based on FastMap which allows quick music similarity query processing. It is designed to work with very large music databases which use Gaussian timbre models and the symmetric Kullback-Leibler divergence as music similarity measure.
- Second, we show how a rescaling of the divergence values and a new FastMap pivot object selection heuristic substantially increase the nearest neighbor recall of the algorithm.

[1] http://www.music-ir.org/mirexwiki/

[2] See http://www.music-ir.org/mirex/200{6,7,8,9} for detailed results.

- Third, we show how to use the proposed method with recent music similarity measures which linearly combine the Kullback-Leibler divergence with rhythmic similarity measures.
- Finally, we present an implementation of a music recommendation system using the proposed techniques which handles a 2.5 million tracks evaluation collection in a very efficient way.

2 Related Work

Scalable music recommendation systems are already the subject of a number of publications. The first content-based music recommendation system working on large collections (over 200 000 songs) was published by Cano et al. [9] in 2005. They present a music recommendation system relying on a diverse range of audio features including rhythmic as well as timbre parameters. Together these features form a music similarity feature vector. They do not report on special indexing techniques.

In 2007 Cai et al. [7] presented a music recommendation system which uses LSH to scale their acoustic music similarity algorithm. A single song is represented by about 30 high-dimensional vectors. Those are obtained using techniques from fingerprinting algorithms [6]. Their music similarity algorithm is evaluated using a ground truth of twenty playlists but is never compared with other established methods. They report an average query time of 2.5 seconds on a collection of about 100 000 tracks, which is rather high, as the LSH index needs to be searched multiple times to compute similarity using their method.

Rafailidis et al. [28] propose to embed the audio similarity space into a low-dimensional Euclidean space. To do so they adopt non-linear dimensionality reduction techniques, and use multidimensional indexing structures to reduce query times.

Casey and Slaney [10] describe a large scale system to scan collections for possible derivate work. They represent their song features as high dimensional vectors and use, like Cai et al. [7], LSH to scale their system. Although their system was not designed for music recommendation, it shows how LSH can be effectively applied to build large scale MIR systems.

Roy et al. [29] were the first to present a music recommendation system which could be used for large databases using Gaussian timbre features. They use a Monte-Carlo approximation of the Kullback-Leibler (KL) divergence to measure music similarity. In principle, their method also resembles a filter-and-refine method similar to the one proposed here. To pre-filter their results, they steadily increase the sampling rate of the Monte-Carlo approximation. As the divergence values converge they are able to reduce the number of possible nearest neighbors. In comparison to the closed form of the KL divergence, which is used in recent music similarity algorithms, this method is far more expensive to compute and yields worse results [23].

Another method to better cope with multivariate Gaussian timbre models was proposed in a publication by Levy and Sandler [21] a year later in 2006. They propose to use Gaussians with a diagonal covariance matrix, instead of a full one to compute music similarity. They report a ten-fold speedup compared to the full Kullback Leibler divergence. However, the quality of this simpler similarity measure is degraded resulting in worse genre classification rates.

With mufin.com there also exists a first commercial content-based music recommendation service that computes acoustic audio similarities for a very large database

of music (6 million tracks as of April 2009). Their website gives no information on how their service works[3].

Besides these approaches to build large scale music recommendation systems, a number of general methods from the class of distance based indexing methods are relevant for indexing the Gaussian features. These methods usually just require a dissimilarity function. A member of this class are Vantage-Point (VP) Trees [32], which build a tree structure that can be searched efficiently, but require a metric distance measure. Another interesting and novel distance-based indexing method, Distance-Based Hashing (DBH), was presented in 2008 by Athitsos et al. [3]. They use FastMap to randomly embed arbitrary features in the Euclidean Space and use LSH to index that mapping. As the effectiveness of this method depends on the quality of the mapping FastMap produces, the findings of our paper could be combined with DBH to omit the linear scan in the filter step of the method presented here.

The idea of using FastMap-related techniques for computationally heavy, non-metric similarity measures and nearest neighbor retrieval was already demonstrated by Athitsos at al. [2] in 2004 to speed up classification of handwritten digits. In MIR research, FastMap was also already used by Cano et al. [8] to map the high dimensional music timbre similarity space into a 2-dimensional space for visualization purposes.

Finally, we like to mention an approach to deal with high dimensional statistical distance measures pursued by Garcia [18]. He uses modern graphics processors (GPUs) to compute the divergences, as they offer very high floating-point performance and parallelism. Garcia shows how a linear brute force nearest neighbor scan be accelerated on a GPU by a factor of about 40, compared to a plain C implementation on a standard CPU.

3 Preliminaries

3.1 Data

Throughout this paper we use a collection of 2.5 million songs to evaluate the performance and show the practical feasibility of our approach. The 2.5 million tracks consist of 30 second snippets of songs. The tracks are assigned one or more of 22 musical genres and were gathered by crawling through an online music store offering free audio previews. The database is representative of music that is listened to in the western hemisphere. More in-depth information about this collection is presented in [17], where a random subsample of the collection was used and evaluated.

3.2 Similarity

We extract timbre features from the snippets and compute a single Gaussian timbre representation using the method proposed by Mandel & Ellis [22].

We compute 25 Mel Frequency Cepstrum Coefficients (MFCCs) for each audio frame (cf. [27]), so that a Gaussian timbre model x finally consists of a 25-dimensional mean vector $\boldsymbol{\mu}$, and a 25×25 covariance matrix $\boldsymbol{\Sigma}$. For performance reasons we also precompute and store the inverted covariance matrix $\boldsymbol{\Sigma^{-1}}$.

[3] http://www.mufin.com/us/faq.html

To compute acoustic timbre similarity we use the symmetrized variant (SKL) of the Kullback-Leibler divergence (KL, [26]). The Kullback-Leibler divergence is an asymmetric information theoretic divergence measure. It is a measure of difference between two probability distributions. For two D-dimensional, multivariate normal distributions $x_1 \sim \mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})$ and $x_2 \sim \mathcal{N}(\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$ the closed form Kullback-Leibler divergence (KL) and the symmetrized variant (SKL) are given by:

$$KL(x_1, x_2) = \frac{1}{2}\left[\log\left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right) + tr(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) - (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - D\right] \quad (1)$$

$$SKL(x_1, x_2) = \frac{1}{2}KL(x_1, x_2) + \frac{1}{2}KL(x_2, x_1). \quad (2)$$

A query for similar songs is processed in a linear scan by computing the SKL between the Gaussian x_1 of the seed song and all other songs in the database. The songs with the lowest divergence to the seed song are its nearest neighbors and possible recommendations.

3.3 Nearest neighbor recall

To compare the effectiveness of the nearest neighbor retrieval variants evaluated, we use what we call nearest neighbor (NN) recall. We define it as the ratio of true nearest neighbors found by some algorithm (NN_{found}) to the real number of true nearest neighbors (NN_{true}) as computed by an exhaustive search.

$$recall = \frac{|NN_{found} \cap NN_{true}|}{|NN_{true}|} \quad (3)$$

4 The Method

To build our filter-and-refine method for fast similarity queries we use an adapted version of FastMap [14], a Multidimensional Scaling (MDS) technique. MDS [11] is a widely used method for visualizing high-dimensional data. FastMap takes the distance matrix of a set of items as input and maps the data to vectors in an arbitrary-dimensional Euclidean space. It was developed in the mid 90s and was since then extended in various ways like BoostMap [2] or MetricMap [31]. These extensions were designed to improve the quality of the mapping of the objects. For our purposes, the original FastMap algorithm produces excellent results.

FastMap is straightforward to use even for large databases since it only needs a low and constant number of rows of the similarity matrix to compute the vector mapping. However, FastMap requires the distances to adhere to metric properties.

4.1 Original FastMap

The original FastMap [14] algorithm uses a simple mapping formula (Equation 4) to compute a k-dimensional projection of objects into the Euclidean vector space.

The dimension k is arbitrary and can be chosen as required. Usually higher dimensions yield a more accurate mapping of the original similarity space.

To project objects into a k-dimensional Euclidean vector space, $2k$ pivot objects from the collection are selected. The original algorithm uses a simple random heuristic to select those pivot objects: for each dimension ($j = 1..k$), (1) choose a random object x_r from the database, (2) search for the most distant object of x_r using the original distance measure $D()$ and select it as the first pivot object $x_{j,1}$ for the dimension, (3) the second pivot object $x_{j,2}$ is the object most distant to $x_{j,1}$ in the original space.

After the $2k$ pivot objects have been selected, the vector representation of an object x is computed by calculating $F_j(x)$ for each dimension ($j = 1..k$):

$$F_j(x) = \frac{D(x, x_{j,1})^2 + D(x_{j,1}, x_{j,2})^2 - D(x, x_{j,2})^2}{2D(x_{j,1}, x_{j,2})} \tag{4}$$

This method depends on metric properties of D to produce meaningful mappings. However, it has been noted that FastMap works surprisingly well also for non-metric divergence measures [2].

FastMap only requires a distance function D and pivot objects to compute the vector mapping. Therefore it can be instantly applied to map the Gaussian timbre models with the SKL as the distance function to the Euclidean vectors (ignoring the fact that the SKL is not metric).

4.2 A Filter-And-Refine Method using FastMap

To use FastMap to quickly process music recommendation queries, we initially map the Gaussian timbre models to k-dimensional vectors. In a two step filter-and-refine process we then use those vectors as a pre-filter: given a query object we first *filter* the whole collection in the vector space (with the squared Euclidean distance) to return a number (*filter-size*) of possible nearest neighbors. We then *refine* the result by computing the exact SKL on the candidate subset to return the nearest neighbors. By using the SKL to refine the results, the correct nearest neighbor ranking is ensured. We set the parameter *filter-size* to a fraction of the whole collection.

The complexity of a single SKL comparison is much higher than a simple vector comparison, so using the squared Euclidean distance to pre-filter the data results in large speedups compared to a linear scan. Table 1 compares the computational cost (in floating point operations, flops) of the SKL to the squared Euclidean distance d^2 using different vector dimensions (k) to pre-filter nearest neighbor candidates.

Unfortunately, as we show in the next section (4.3), applying FastMap to the problem without any modifications yields very poor results.

4.3 Modifications

In our implementation we have included two important modifications which improve the quality of FastMap mappings for nearest neighbor retrieval. The modifications are centered around two thoughts: (1) a metric divergence measure would produce better vector mappings, and (2) a more specialized heuristic for pivot object selection could produce better mappings especially for the near neighbors, which are at the center of our interest.

Divergence	*flops*	$flops/flops_{SKL}$
$SKL, mfcc = 25$	3 552	1
$d^2, k = 20$	60	0.017
$d^2, k = 40$	120	0.034
$d^2, k = 60$	180	0.051

Table 1

The computational complexity (in floating point operations, flops) of computing the squared Euclidean distance (d^2) is, even for high mapping dimensions like $k = 60$, much lower than the cost of computing a single SKL comparison. To compute the squared Euclidean distance between two k dimensional vectors $3k$ flops are required, whereas computing the SKL between two m-dimensional Gaussian distributions requires $\frac{1}{2}(11m^2 + 9m + 4)$ flops ($= 3\,552$ flops for the SKL between two 25-dimensional Gaussians). The SKL is computed using an optimized implementation [30].

4.3.1 Rescaling

The SKL already has the important metric properties of being symmetric and non-negative, but fails to fulfill the triangle inequality. We try to correct this by rescaling the SKL. The SKL (Equation 2) is very similar to the Jensen-Shannon Divergence (JSD, Equation 5), another symmetrized and smoothed (but very expensive to compute) version of the Kullback-Leibler divergence:

$$JSD(x_1, x_2) = \frac{1}{2}KL(x_1, x_{1,2}) + \frac{1}{2}KL(x_2, x_{1,2}) \tag{5}$$

$$x_{1,2} = \frac{1}{2}(x_1 + x_2). \tag{6}$$

As the square-root of the JSD is proven to be a metric divergence measure [13], we were hoping for similar corrective effects for the SKL. Before mapping the objects $x_i \in X$ to a k-dimensional vector (Equation 4), we propose to rescale the original symmetric Kullback-Leibler divergences (SKL) with the square-root:

$$D(x_1, x_2) = \sqrt{SKL(x_1, x_2)}. \tag{7}$$

We have experimentally verified the effect of rescaling on a collection of 100 000 Gaussian timbre models (Table 2). The models were computed using randomly selected songs from the 2.5 million songs collection and the triangle inequality was checked for all possible divergence triples. The table shows that taking the square-root makes the SKL obey the triangle inequality in more than 99% of the cases. The table also includes a comparison of other common ways to rescale the SKL using $e^{\lambda SKL()}$ (see [24] or [22]).

4.3.2 Pivot Object Selection

To select the pivot objects needed to map an object x to a vector space, the original algorithm uses two objects for each dimension which lie as far away from each other as possible (see Section 4.1). In contrast to the original heuristic we propose to select the pivot objects using an adapted strategy: (1) first we randomly select an object x_r and

Divergence	% triangle inequality
$SKL()$	91.57%
$1 - e^{\lambda SKL()}, \lambda = -\frac{1}{100}$	93.71%
$1 - e^{\lambda SKL()}, \lambda = -\frac{1}{50}$	95.60%
$\sqrt{SKL()}$	99.32%

Table 2

Percentage of Gaussian object triples fulfilling the triangle inequality ($D(x,z) \leq D(x,y) + D(y,z)$) with and without rescaling. The triangle inequality was checked for all possible triples in a collection of 100 000 randomly selected Gaussian timbre models.

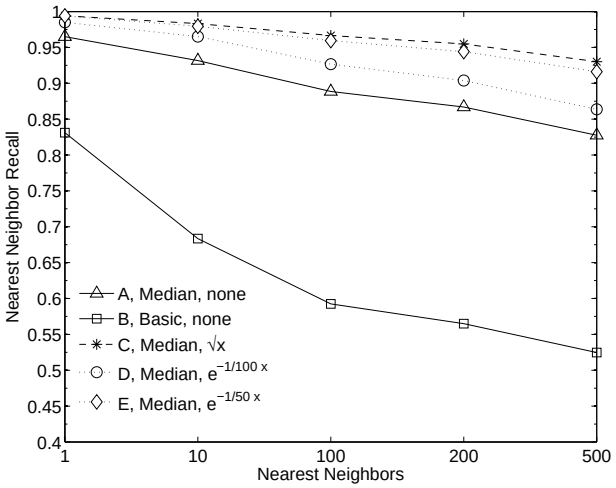

Fig. 1 Nearest neighbor (NN) recall of two pivot object selection methods (*median*: the proposed pivot object selection heuristic, *basic*: the original Fastmap heuristic) in combination with three divergence rescaling methods (*no-rescaling*, $e^{\lambda x}$, $\sqrt{x}$). NN recall is averaged over five independent evaluation runs (10 000 queries per run), each time using a new random collection. Parameters: $k = 40$, *filter-size* $= 10\%$, *collection size*$= 100\,000$.

compute the distance to all other objects; (2) we then select the first pivot object x_1 to be the object lying at the distance median, i.e. the object at the index $i = \lfloor N/2 \rfloor$ on the sorted list of divergences; (3) likewise, the second pivot object x_2 is selected to be the object with the distance median of all divergences from x_1 to all other objects.

By using pivot objects at the median distance we avoid using objects with extremely high divergence values which often occur in the divergence tails when using the SKL. Since we are particularly interested in optimally mapping the near neighbors and not the whole divergence space, this strategy should also help in preserving the neighborhoods.

4.3.3 Improvements

Finally, we measure how these modifications improve the filter-and-refine method by experimentally computing the nearest neighbor (NN) recall of each change on a 100 000

songs collection. Figure 1 shows the result of the experiment. A huge improvement in the nearest neighbor recall can be seen for all strategies which use the median pivot object selection heuristic (A, C, D, E) compared to the original FastMap heuristic (B). The figure also shows that rescaling the SKL values helps to further increase the NN recall. The suggested strategy (C) using the median pivot object selection strategy together with square-root-rescaling gives the best results.

5 Combined Music Similarity Measures

Besides using single Gaussian timbre models and the SKL to compute acoustic music similarity, recent approaches enhance the quality of a music similarity measure by mixing multiple similarity components. For example Pampalk [24] and Pohle [27] combine the single Gaussian timbre models with rhythmic structures to better capture more aspects of similarity.

In this section we want to show the applicability of the previously proposed method to a similarity measure which linearly combines the SKL with rhythmic similarity using Fluctuation Patterns (FPs) [25]. Fluctuation Patterns describe the amplitude modulation of the loudness per frequency band. We closely follow the implementation outlined in [24] to compute the FPs: (1) cutting a MFCC spectrogram into three second segments, (2) using a FFT to compute amplitude modulation frequencies of loudness (range $0 - 10$Hz) for each segment and frequency band, (3) weighting the modulation frequencies based on a model of perceived fluctuation strength, (4) applying filters to emphasize certain patterns and smooth the result. The resulting FP is a 12 (frequency bands according to 12 critical bands of the Bark scale [15]) times 30 (modulation frequencies, ranging from 0 to 10Hz) matrix for each song. The distance between two FPs x_i and x_j is computed as the Euclidean distance (ED):

$$ED(x_i, x_j) = \sqrt{\sum_{k=1}^{12}\sum_{l=1}^{30}\left(x_i^{k,l} - x_j^{k,l}\right)^2} \tag{8}$$

To linearly combine the two similarity measures in a meaningful way, each distance has to be normalized to unit variance. To do so we compute the standard deviation for each distance measure ($\sigma_{\sqrt{SKL}}$ and σ_{ED}) on an independent test collection. The final combined similarity measure is defined as:

$$SKL\&ED(x_1, x_2) = 0.7\frac{\sqrt{SKL(x_1, x_2)}}{\sigma_{\sqrt{SKL}}} + 0.3\frac{ED(x_1, x_2)}{\sigma_{ED}} \tag{9}$$

The weights of the combined measure (0.7 for the timbre- and 0.3 for the rhythmic-component) were taken from [24]. In a genre classification experiment we show that the combined measure indeed performs better (see Table 3, column: *Combined*).

A genre classification experiment is a standard evaluation to test the quality of a music recommendation algorithm. It assumes that each song in a music collection is labeled with a genre. A nearest neighbor classifier is then used to compute the genre classification accuracy, the ratio of correctly classified tracks to the total number of tracks. In each query we exclude songs from the same artist, as this leads to overly optimistic genre classification results (see [16] for a discussion).

To quantify the effects, we have done nearest neighbor classification experiments with and without the filter and refine method using six different collections (three in-house collections and three public datasets: the *Ismir 2004* Magnatune music collection[4], the *Homburg* collection [19] and the Ballroom collection[5]. Table 3 summarizes the results. It appears that the decrease due to the $F\&R$ method is negligible. Classification accuracy decreases only by about 0.1% for the two large collections and by about 0.5% for the two small collections.

Music genre classification accuracy increases in all four tested collections by up to 3.5% compared to the results of the previous evaluation using only the SKL as the similarity measure (Table 3, column: *Single Gaussian*).

		Single Gaussian		Combined	
Collection, size	Genres	F&R	Full Scan	F&R	Full Scan
$\#1, N = 16\,781$	21	30.17%	30.28%	31.21%	31.29%
$\#2, N = 9\,369$	16	28.55%	28.66%	31.97%	32.10%
$\#3, N = 2\,527$	22	28.27%	28.78%	31.16%	31.04%
Ismir 2004, $N = 729$	6	64.47%	64.88%	65.57%	65.57%
Ballroom, $N = 698$	8	53.86%	53.43%	55.73%	55.73%
Homburg, $N = 1851$	9	42.84%	42.78%	43.27%	43.21%

Table 3

1-NN genre classification experiment results (with artist filter) on six different collections. The table compares the genre classification accuracy of the filter-and-refine (F&R) approach presented in this paper with a full exact linear scan using two different similarity measures (the *Combined*/improved features and the standard *Single Gaussian* features). It can be seen that the F&R approach yields the same results as the full scan. Parameters for the F&R method: $k = 40$, *filter-size* $= 5\%$

5.1 Filter-And-Refine

The proposed F&R method can be applied to the combined similarity measure $SKL\&ED$ without any modifications. We use the previously introduced median pivot object selection strategy to select good pivot objects for FastMap using the afore mentioned combined similarity measure. Additionally, we use a rescaled variant of the SKL in the linear combination (Equation 9) to increase the cases where the SKL - and thus the $SKL\&ED$ - behaves like a metric. This property is discussed in the next paragraphs.

The triangle inequality is defined between the distances of three points: $D(x, z) \leq D(x, y)+D(y, z)$. It is a basic property of every metric distance measure. This inequality also holds for any linear combination of two metric distance measures D_1 and D_2:

[4] http://ismir2004.ismir.net/genre_contest/index.htm
[5] http://mtg.upf.edu/ismir2004/contest/rhythmContest/

$$\alpha_1 D_1(x,z) + \alpha_2 D_2(x,z) \leq \alpha_1 (D_1(x,y) + D_1(y,z)) + \alpha_2 (D_2(x,y) + D_2(y,z)). \quad (10)$$

Therefore a linear combination of a non-metric (i.e. SKL) and a metric distance (i.e. Euclidean) could only violate the triangle inequality where the non-metric measure does. Since we have experimentally shown that rescaling the SKL by taking the square root makes the divergence almost metric, the linear combination inherits this property and should deliver comparable results with FastMap.

In a genre classification experiment (see Table 3, column: *Combined*) we confirm that the F&R approach indeed works well with the combined similarity measure. Like in Table 3 (column: *Single Gaussian*) it can be seen that the genre classification accuracy decreases only very little with the F&R method (0.1%); in one collection the approximative method even returns better results than a full exact search.

We conclude that the combined similarity measure $SKL\&ED$ can be used alongside the proposed filter-and-refine method, improving the quality of the music recommendations.

6 Implementation

The implementation of the filter-and-refine music recommendation engine is straightforward: in an initial step the whole collection is preprocessed with the proposed mapping method, transforming the database objects into a k-dimensional vector space. This is a linear process since only $2k$ pivot objects have to be selected and each object in the database is mapped to a vector once, using Equation 4. Our implementation saves the pivot objects for each dimension and the vector mappings to disk. This allows fast restarting of the system and easy processing of new objects.

To query for similar objects we use the previously described filter-and-refine method, filtering out a predefined number (*filter-size*, a percentage of the collection size) of nearest neighbor candidates using the vector representation and refining the result with the exact SKL.

This outlines the general method we propose, but obviously two parameters which have a huge impact on the retrieval quality (nearest neighbor (NN) recall) and the query speed have not been discussed yet: the number of vector dimensions k and the *filter-size*.

6.1 Recall and Speed

It is obvious that a larger *filter-size* results in better NN recall values but higher computational costs. Likewise, a higher k used for the vector mapping results in a more accurate mapping of the divergence space, but with each dimension the computational costs to compute the squared Euclidean distance in the pre-filter steps are increased.

Figure 2 evaluates different parameter combinations of k and *filter-size* and their impact on nearest neighbor recall and computational cost (and thus query speed). The diagram was compiled using a collection of 100 000 Gaussian timbre models. It shows the 10-NN retrieval recall and query speed (computational cost in terms of flops).

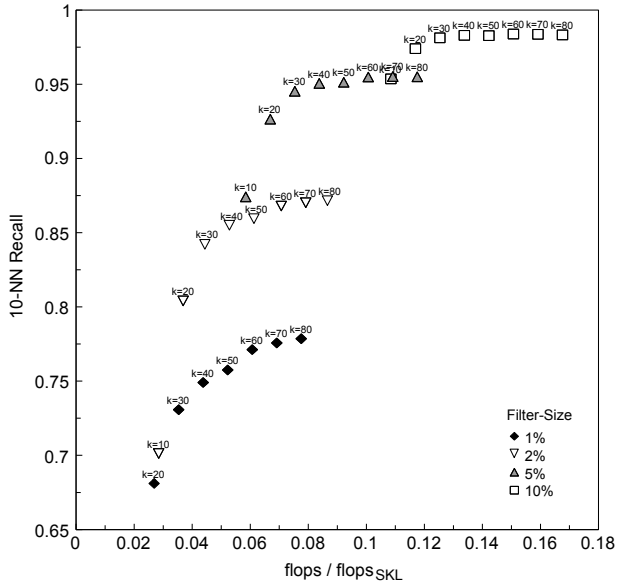

Fig. 2 Plot relating the nearest neighbor recall and the floating point operations resulting from different filter-and-refine parameter combinations to a full linear scan ($flops/flops_{SKL}$). Recall is computed for the 10 nearest neighbors for different parameter combinations of k and *filter-size* in a collection of 100 000 songs. The experiment was run 10 times, recall is averaged. A good combination (good recall, low computational cost) would be mapped to the upper left corner of the plot.

The figure shows that a parameter combination of $k = 20$ and *filter-size* $= 5\%$ can be selected to achieve about 93% 10-NN recall. This combination would take only 7% of the query time required by a linear scan with the SKL. If a 10-NN recall of 81% is acceptable a parameter combination requiring only 3.5% of the computational cost of a linear scan is possible ($k = 20$ and *filter-size* $= 2\%$). Almost perfect 10-NN recall values ($> 98\%$) can be reached when setting *filter-size* to about 10% of the collection size, which still requires only 11% of the time a linear scan would need.

This evaluation shows how to select a good parameter combination. In Section 6.2 we plot a similar diagram (Figure 3) to select the best parameters for a 2.5 million song collection, achieving 99% 1-NN, 98% 10-NN and 95% 100-NN recall on the collection with a combined music similarity measure.

6.2 Prototype

To demonstrate the practical feasibility of using the proposed method with very large music databases, we built a prototype music recommendation system working on a music collection of 2.5 million songs. The system should be able to answer queries for the 10 nearest neighbors (10-NN) with high speed and accuracy compared to a full linear scan. It should be easy to browse and explore.

Before building the prototype system, we first decided to use the combined but computationally more expensive music similarity measure $SKL\&ED$ as it produces better music recommendations. We then ran an evaluation to determine the best *filter-*

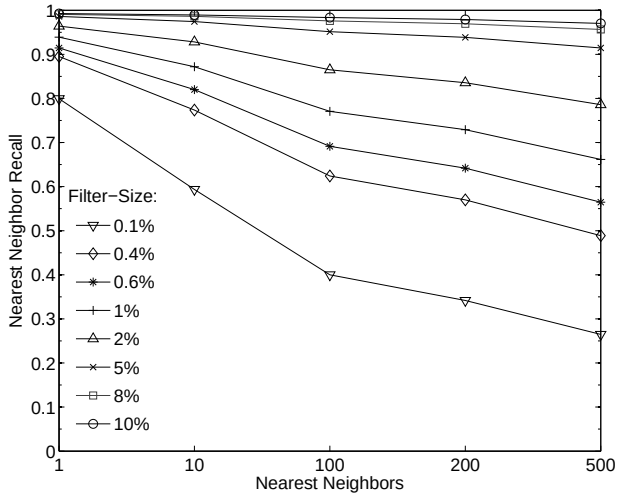

Fig. 3 NN recall with different *filter-sizes* evaluated with 25 000 songs on the 2.5 million songs collection using the combined similarity measure. With a *filter-size* of 5% one can achieve 96% 100-NN recall and 98% 10-NN and 99% 1-NN recall. $k = 40$.

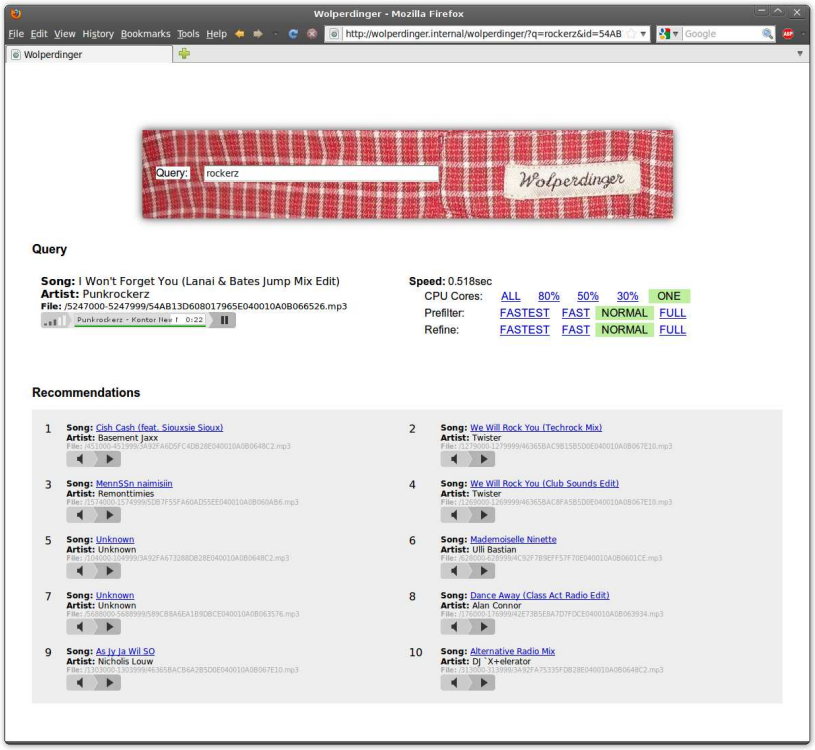

Fig. 4 The prototype web music-recommendation system displaying ten recommendations computed on a collection of 2.5 million tracks. The recommendations listed are the eight nearest neighbors found by the audio similarity measure for the seed song by the *Punkrockerz*.

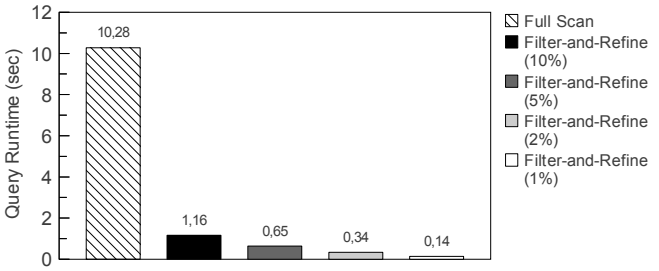

Fig. 5 Comparison of the time it takes to query a 2.5 million song collection for nearest neighbors using a full scan compared to a scan using the proposed filter-and-refine method. A single CPU core of a standard Intel Core Duo CPU (2.5GHz) was used and all Gaussian timbre models/Fluctuation Patterns were loaded to RAM.

size for the described use-case (see Figure 3). In the figure we see that even for a *filter-size* of only 5%, the true 1-NN and 10-NN are retrieved almost every time on a 2.5 million song collection. A *filter-size* of 2% still returns the true 10-NN in 94% the time.

We finally assemble all components which were gradually introduced in this paper and build a simple music recommendation web application for the collection. Figure 4 shows a screenshot of the application: To get a music recommendation a user first has to search the database for a seed song by using the query text-field. After the user has selected the desired seed song by clicking it, ten music recommendations are computed instantly using the proposed filter-and-refine method. The screenshot shows the recommendations the system returned for the seed song *I Won't Forget You* by the *Punkrockerz*. The recommendations can be listened to by clicking the Flash music player. It is possible to browse the recommendation system by clicking on a suggested song to make it the seed for a new query.

6.3 Performance

As high performance is a key objective, we measured the actual query response times of the system. Figure 5 compares the query response times of three different *filter-size* settings (*filter-size*$= 1\%, 2\%, 5\%, 10\%$, $k = 40$) to a full linear scan using the $SKL\&ED$. The final system is capable of answering music recommendation queries in 0.34 seconds on a 2.5 million songs collection while returning about 94% of the correct 10 nearest neighbors compared to a linear scan, which takes 10.28 seconds on the same system.

As the parameter *filter-size* is statically set in the web application prototype, the time taken to answer a query is constant for every request. Systems deployed for real use could very easily implement a system-load dependent strategy and dynamically vary *filter-size* with the number of concurrent pending requests. Implementing that strategy would make the system capable of answering queries under high load with constant speed at the cost of decreased accuracy (see Figure 3).

7 Conclusions

We have described a filter-and-refine method for fast approximate music similarity search in large collections. The method is primarily designed for the widely used Gaussian music timbre features using the symmetric Kullback-Leibler divergence to compute acoustic similarity. We show that the method can also be used with more recent algorithms which linearly combine multiple music similarity measures.

Based on the described method we built a prototype music recommendation web service which works on a collection of 2.5 million tracks. The system is able to answer music similarity queries in about half a second on a standard desktop CPU.

By accelerating similarity queries by a factor of 10 to 30, we show how a large scale music recommendation service relying on recent music information retrieval techniques could be operated today.

Acknowledgments

This research is supported by the Austrian Research Fund (FWF) under grant L511-N15, and by the Austrian Research Promotion Agency (FFG) under project number 815474-BRIDGE.

References

1. A. Andoni, P. Indyk, and C. MIT. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468, 2006.
2. V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. BoostMap: A method for efficient approximate similarity rankings. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 2004.
3. V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios. Nearest neighbor retrieval using distance-based hashing. In *IEEE 24th International Conference on Data Engineering, 2008. ICDE 2008*, pages 327–336, 2008.
4. J. Bentley. *Multidimensional binary search trees used for associative searching.* ACM New York, NY, USA, 1975.
5. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When Is 'Nearest Neighbor' Meaningful? In *Proceedings of the 7th International Conference on Database Theory*, pages 217–235. Springer-Verlag London, UK, 1999.
6. C. Burges, J. Platt, and S. Jana. Distortion discriminant analysis for audio fingerprinting. *IEEE Transactions on Speech and Audio Processing*, 11(3):165–174, 2003.
7. R. Cai, C. Zhang, L. Zhang, and W. Ma. Scalable music recommendation by search. In *Proceedings of the 15th international conference on Multimedia*, pages 1065–1074. ACM New York, NY, USA, 2007.
8. P. Cano, M. Kaltenbrunner, F. Gouyon, and E. Batlle. On the use of FastMap for audio retrieval and browsing. In *Proc. Int. Conf. Music Information Retrieval (ISMIR)*, pages 275–276, 2002.
9. P. Cano, M. Koppenberger, and N. Wack. An industrial-strength content-based music recommendation system. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 673–673. ACM New York, NY, USA, 2005.
10. M. Casey and M. Slaney. Song intersection by approximate nearest neighbor search. In *Proc. ISMIR*, pages 144–149, 2006.
11. T. Cox and M. Cox. *Multidimensional scaling.* CRC Press, 2001.
12. J. S. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

13. D. Endres and J. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information theory*, 49(7):1858–1860, 2003.
14. C. Faloutsos and K. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 163–174. ACM New York, NY, USA, 1995.
15. H. Fastl and E. Zwicker. *Psychoacoustics: facts and models*. Springer-Verlag New York Inc, 2007.
16. A. Flexer. A Closer Look on Artist Filters for Musical Genre Classification. In *Proceedings of the International Symposium on Music Information Retrieval, Vienna, Austria*, 2007.
17. A. Flexer and D. Schnitzer. Effects of Album and Artist Filters in Audio Similarity Computed for Very Large Music Databases. *Computer Music Journal*, 34(3):20–28, 2010.
18. V. Garcia, E. Debreuve, and M. Barlaud. Fast k nearest neighbor search using GPU. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPR Workshops 2008*, pages 1–6, 2008.
19. H. Homburg, I. Mierswa, B. M
"oller, K. Morik, and M. Wurst. A benchmark dataset for audio classification and clustering. In *Proceedings of the International Conference on Music Information Retrieval*, pages 528–31, 2005.
20. J. Jensen, M. Christensen, D. Ellis, and S. Jensen. Quantitative analysis of a common audio similarity measure. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(4):693–703, 2009.
21. M. Levy and M. Sandler. Lightweight measures for timbral similarity of musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 27–36. ACM New York, NY, USA, 2006.
22. M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005), London, UK*, 2005.
23. M. Mandel and D. P. Ellis. Labrosa's audio music similarity and classification submissions. In *Proceedings of the International Symposium on Music Information Retrieval, Vienna, Austria - Mirex 2007*, 2007.
24. E. Pampalk. Computational models of music similarity and their application in music information retrieval. *Doctoral dissertation, Vienna University of Technology, Austria*, 2006.
25. E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 570–579. ACM New York, NY, USA, 2002.
26. W. Penny. KL-Divergences of Normal, Gamma, Dirichlet and Wishart densities. *Wellcome Department of Cognitive Neurology, University College London*, 2001.
27. T. Pohle and D. Schnitzer. Striving for an improved audio similarity measure. *4th Annual Music Information Retrieval Evaluation Exchange*, 2007.
28. D. Rafailidis, A. Nanopoulos, and Y. Manolopoulos. Nonlinear dimensionality reduction for efficient and effective audio similarity searching. *Multimedia Tools and Applications*, pages 1–15.
29. P. Roy, J. Aucouturier, F. Pachet, and A. Beurive. Exploiting the tradeoff between precision and cpu-time to speed up nearest neighbor search. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005), London, UK*, 2005.
30. D. Schnitzer. Mirage – High-Performance Music Similarity Computation and Automatic Playlist Generation. *Master's thesis, Vienna University of Technology*, 2007.
31. J. Wang, X. Wang, D. Shasha, and K. Zhang. Metricmap: an embedding technique for processing distance-based queries in metric spaces. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(5):973–987, 2005.
32. P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 311–321. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 1993.