A FILTER-AND-REFINE INDEXING METHOD FOR FAST SIMILARITY SEARCH IN MILLIONS OF MUSIC TRACKS

Dominik Schnitzer
Austrian Research Institute for Artificial Intelligence
dominik.schnitzer@ofai.at

Arthur Flexer
Austrian Research Institute for Artificial Intelligence
arthur.flexer@ofai.at

Gerhard Widmer
Johannes Kepler University Linz, Austria
gerhard.widmer@jku.at

ABSTRACT

We present a filter-and-refine method to speed up acoustic audio similarity queries which use the Kullback-Leibler divergence as similarity measure. The proposed method rescales the divergence and uses a modified FastMap [1] implementation to accelerate nearest-neighbor queries. The search for similar music pieces is accelerated by a factor of $10-30$ compared to a linear scan but still offers high recall values (relative to a linear scan) of $95-99\%$.

We show how the proposed method can be used to query several million songs for their acoustic neighbors very fast while producing almost the same results that a linear scan over the whole database would return. We present a working prototype implementation which is able to process similarity queries on a 2.5 million songs collection in about half a second on a standard CPU.

1. INTRODUCTION

Today an unprecedented amount of music is available online. As of April 2009, the Apple iTunes music store alone lists more than 10 million downloadable songs in its catalog. Other online music stores like Amazon MP3 still offer a 5 million songs catalog to choose from. With the catalog numbers constantly reaching new record highs, the need for intelligent music search algorithms that provide new ways to discover and navigate music is apparent.

Unfortunately many of the intelligent music processing algorithms that have been published do not easily scale to the millions of music pieces available in an online music store. In particular, this is true for music recommendation algorithms which compute acoustic music similarity using a Gaussian timbre representation and the Kullback-Leibler divergence, as in [2], [3] or [4].

Especially the Kullback-Leibler divergence, as it is used in the referenced works, poses multiple challenges when developing a large scale music recommendation system: (1) the divergence is very expensive to compute, (2) it is not a metric and thus makes building indexing structures around it very hard and (3) in addition, the extracted acoustic music similarity features have a very high degree of freedom, which too is a general problem for indexing solutions ("curse of dimensionality") [5].

But on the other hand, systems using this technique regularly rank in the very top places in the yearly MIREX Automatic Music Recommendation evaluations [1], which makes them a tempting but challenging target for broad usage in real applications.

1.1 Related Work

The idea of using FastMap-related techniques for computationally heavy non-metric similarity measures and nearest neighbor retrieval was first demonstrated by Athitsos [6]. They use BoostMap [7] to improve the speed of classifying handwritten digits. Cano et al. [8] use FastMap to map the high dimensional music timbre similarity space into a 2-dimensional space for visualization purposes.

Roy et al. [9] present a music recommendation system which uses a Monte-Carlo approximation of the Kullback-Leibler (KL) divergence as similarity measure. The Monte-Carlo approximation of the KL divergence is far more expensive to compute and less accurate than the closed form of the KL divergence which is used in our paper and recent music similarity algorithms. To speed up a similarity query, they narrow the number of nearest neighbor candidates by incrementally increasing the accuracy of the Monte-Carlo sampled divergence measure.

Another interesting approach, which was pursued by Garcia [10], is to compute computationally expensive similarity measures on modern graphics processors (GPUs). Modern GPUs offer high floating-point performance and parallelism. As an example Garcia shows how a linear brute force nearest neighbor scan using the Kullback-Leibler divergence can be accelerated on a GPU compared to computing it on a standard CPU. The idea to use the GPU to process similarities could be combined with the methods presented in this paper.

With mufin.com there also exists a commercial music recommendation service which computes acoustic audio similarities for a very large database of music (6 million tracks as of April 2009). However, their website gives no hint on how their service works [2].

[1] http://www.music-ir.org/mirexwiki/
[2] http://www.mufin.com/us/faq.html

1.2 Contributions of this paper

The contribution of this paper is three-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 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.
- 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. PRELIMINARIES

2.1 Data

Throughout this paper we use a collection of 2.5 million songs to evaluate the performance and to show the practical feasibility of our approach. The 2.5 million tracks consist of 30 second snippets of songs gathered by crawling an online music store offering free audio preview files.

2.2 Similarity

We extract timbre features from the snippets and compute a single Gaussian timbre representation using the method proposed by Mandel & Ellis [2]. We compute 25 Mel Frequency Cepstrum Coefficients (MFCCs) for each audio frame, so that a Gaussian timbre model x finally consists of a 25-dimensional mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. For performance reasons we also precompute and store the inverted covariance matrix $\boldsymbol{\Sigma^{-1}}$.

To compute acoustic timbre similarity we use the symmetrized version (SKL) of the Kullback-Leibler divergence (KL, [11]), defined between two multivariate normal distributions $x_1 \sim \mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})$ and $x_2 \sim \mathcal{N}(\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$:

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

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.

2.3 Nearest neighbor recall

To compare the effectiveness of the nearest neighbor retrieval variants evaluated, we used 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 number of true nearest neighbors (NN_{true}). The true nearest neighbors are found by a full linear scan.

$$recall = \frac{|NN_{found} \cap NN_{true}|}{|NN_{true}|} \quad (2)$$

3. THE METHOD

To build our filter-and-refine method for fast similarity queries we use an adopted version of FastMap [1], a Multidimensional Scaling (MDS) technique. MDS [12] is a widely used method for visualizing high-dimensional data. It takes the distance matrix of a set of items as input and maps the data to vectors into an arbitrary-dimensional Euclidean space. 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.

3.1 Original FastMap

The original FastMap [1] algorithm uses a simple mapping formula (Equation 3) 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, first two pivot objects from the feature database have to be selected for each of the k dimensions. The original algorithm uses a simple heuristic to select those pivot objects: for each dimension ($j = 1..k$), (1) chose 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 estimating $F_j(x)$ for each dimension ($j = 1..k$):

$$F_j(x) = \frac{D(x, x_{j,1})^2 + D(x_{j,1}, x_{j,2})^2 - D(x, x_{j,2})^2}{2D(x_{j,1}, x_{j,2})} \quad (3)$$

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 [7].

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

3.2 A Filter-And-Refine Method using FastMap

To use FastMap to quickly process music recommendation queries, we initially use it to map the Gaussian timbre models to k-dimensional vectors. In a two step filter-and-refine process we then use those vectors as a prefilter: first

we *filter* the whole collection in the vector space (with the squared Euclidean distance) to return a number (*filter-size*) of possible nearest neighbors, then we *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.

Since the complexity of a single SKL comparison is much higher than a simple vector comparison, the use of the squared Euclidean distance to prefilter the data results in large speedups compared to a linear scan over the whole collection using the SKL. 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 prefilter candidate nearest neighbors.

Divergence	$flops$	$flops/flops_{SKL}$
SKL	3552	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 $flops$) of computing the squared Euclidean distance (d^2) is, even for high mapping dimensions like $k = 60$, much lower than the costs of computing a single SKL comparison. *Note:* We already use an optimized implementation of the SKL exploiting matrix symmetry and the sequence of matrix operations [13].

Unfortunately, as we show in the next section (3.3), applying FastMap to the problem without any modifications yields very poor results.

3.3 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 the center of our interest.

3.3.1 Rescaling

Before mapping the objects $x_i \in X$ to a k-dimensional vector (Equation 3), we propose to rescale the original symmetric Kullback-Leibler divergences (SKL) by taking the square-root:

$$D(x_1, x_2) = \sqrt{SKL(x_1, x_2)}. \quad (4)$$

This rescaling has the effect of making the SKL behave more like a metric. As the SKL already has the important properties of being symmetric and non-negative, it only fails to fulfill the triangle inequality. Taking the square root has the effect to partly fix the divergence, making it more metric [14]. Another more common way is to rescale the SKL with $e^{\lambda SKL()}$ (see [3] or [2]).

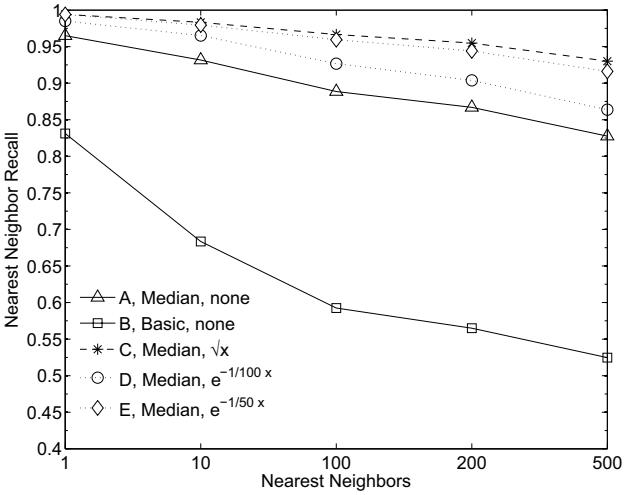

Figure 1. Nearest neighbor (NN) recall of two pivot object selection methods (*median*: the proposed pivot object selection heuristic, *basic*: the original FastMap heuristic) in combination with three divergence rescaling methods (*no-rescaling*, $e^{\lambda x}$, $\sqrt{x}$). NN recall is averaged over 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*.

We have experimentally verified the effect of rescaling on a collection of 100.000 randomly drawn Gaussian timbre models (Table 2), by checking the triangle inequality. The table clearly shows that exponentiating indeed reduces the number of cases where the triangle inequality is violated, but it does not work as well as taking the square root, which makes the SKL obey the triangle inequality in more than 99% of the cases in our experimental setup.

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.

3.3.2 Pivot Object Selection

To select the pivot objects which are 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 3.1). 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 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 also particularly interested in optimally mapping the near neighbors and not the whole divergence space, this strategy should also help in preserving the neighborhoods.

3.3.3 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.

4. 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 using Equation 3 once. Our implementation saves the pivot objects for each dimension and the vector mappings of processed objects 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*.

4.1 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 prefilter steps are increased.

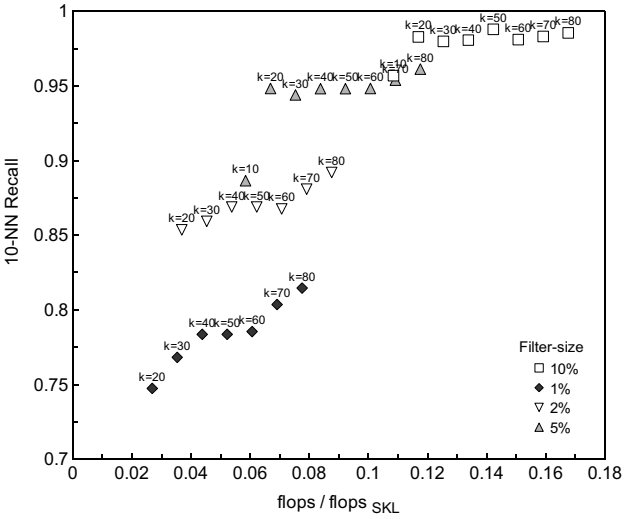

Figure 2. Plot relating the nearest neighbor recall and the floating point operations resulting from different filter-and-refine parameter combinations, to a full linear scan ($flops/flops_{SKL}$). Recall is computed for the 10 nearest neighbors for different parameter combinations of k and *filter-size* in a collection of 100.000 songs. A good combination (good recall, low computational cost) would be mapped to the upper left corner of the plot.

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).

The figure shows that a parameter combination of $k = 20$ and *filter-size*$= 5\%$ can be selected to achieve about 95% 10-NN recall. That combination would take only 7% of the query time required by a linear scan with the SKL. If a 10-NN recall of 85% is acceptable a parameter combination requiring only 3.5% the computational cost of a linear scan is possible ($k = 20$ and *filter-size*$= 2\%$). Almost perfect 10-NN recall values ($> 98 - 99\%$) can be reached when setting *filter-size* to about 10% of the collection size, which still requires only 10% of the time a linear scan would need.

This evaluation shows how a good parameter combination for a collection should be selected. In Section 5 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.

4.2 Errors

Another aspect which is of interest is how falsely reported nearest neighbors (false positives) affect the average quality of music recommendations. We have done a 1-NN genre evaluation (with artist filter, see [3]). This is a standard evaluation to test the quality of a music recommendation algorithm.

We tested four different collections (three in-house collections and the *Ismir 2004* Magnatune music collection

which is freely available for testing purposes [3]). Table 3 summarizes the results. It appears that the errors which are being made do not affect the classification accuracy in an adverse way. Classification accuracy decreases only by about 0.1% for the two larger collections and by about 0.5% for the two small collections.

$Collection, size$	Genres	F&R	Full Scan
$\#1, N = 16781$	21	30.17%	30.28%
$\#2, N = 9369$	16	28.55%	28.66%
$\#3, N = 2527$	22	28.27%	28.78%
Ismir 2004, $N = 729$	6	64.47%	64.88%

Table 3. 1-NN genre evaluation results (with artist filter) on four 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. Parameters: $k = 40$, *filter-size*=5%

5. PROTOTYPE PERFORMANCE

To show the practical feasibility of using this filter-and-refine method with large music databases we use the method on the 2.5 million song collection and build a prototype music recommendation system. The system should be able to answer queries for the 100 nearest neighbors with high speed and recall.

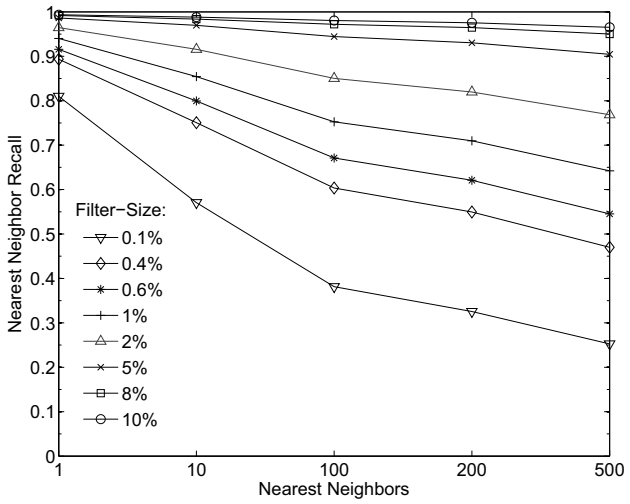

Figure 3. NN recall with different *filter-sizes* evaluated on 1% (= 25.000) of the 2.5 million songs collection. With a *filter-size* of 5% one can achieve 95% 100-NN recall and 98% 10-NN and 99% 1-NN recall. $k = 40$.

To select the optimal parameter we ran an experiment to determine the best *filter-size*, k was set to 40. Figure 3 shows the recall values for different NN and *filter-sizes*. It can be seen that the true 1-NN and 10-NN are retrieved almost always if the *filter-size* is set to 5%, 8% or 10% of the collection size.

[3] http://ismir2004.ismir.net/genre_contest/index.htm

In a second experiment (Figure 4) we compare the actual query response times of three different *filter-size* settings (*filter-size*= $8\%, 5\%, 2\%$, $k = 40$) to a full linear scan. It can be seen that the system running on a single standard CPU core is capable of answering music recommendation queries in half a second while returning about 95% of the correct 100 nearest neighbors compared to a linear scan which would take about $7.8sec$ on the system.

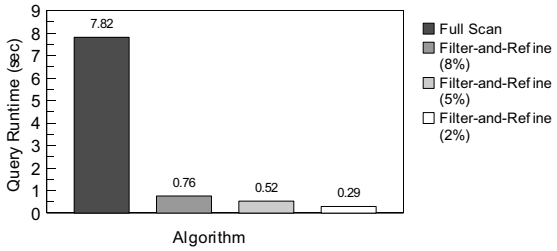

Figure 4. Comparison of the time it takes to query a 2.5 million song collection for nearest neighbors using a full scan compared to a scan using the filter-and-refine method proposed. The PC used a standard Intel Core Duo CPU (2.5GHz) and had all Gaussian models loaded to RAM.

6. CONCLUSIONS

We have described a filter-and-refine method for fast approximate music similarity search in large collections. The method is designed for Gaussian music timbre features using the symmetric Kullback-Leibler divergence to compute acoustic similarity, but could be generalized to other distance measures. A prototype implementation of our method handling 2.5 million tracks is able to answer music similarity queries in about half a second on a standard desktop CPU.

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

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.

7. REFERENCES

[1] C. Faloutsos and K.I. 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.

[2] 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 (ISMIR05), London, UK*, 2005.

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

[4] T. Pohle and D. Schnitzer. Striving for an improved audio similarity measure. *4th Annual Music Information Retrieval Evaluation Exchange*, 2007.

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is" nearest neighbor" meaningful? *Lecture Notes in Computer Science*, pages 217–235, 1999.

[6] V. Athitsos, J. Alon, and S. Sclaroff. Efficient nearest neighbor classification using a cascade of approximate similarity measures. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 1, 2005.

[7] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. BoostMap: A method for efficient approximate similarity rankings. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2.

[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. Roy, J.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 (ISMIR05), London, UK*, 2005.

[10] 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.

[11] W.D. Penny. Kullback-Leibler divergences of normal, gamma, Dirichlet and Wishart densities. *Wellcome Department of Cognitive Neurology*, 2001.

[12] T.F. Cox and M.A.A. Cox. *Multidimensional scaling*. CRC Press, 2001.

[13] D. Schnitzer. Mirage – High-Performance Music Similarity Computation and Automatic Playlist Generation. *Master's thesis, Vienna University of Technology*, 2007.

[14] K.L. Clarkson. Nearest-neighbor searching and metric space dimensions. *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59, 2006.