PICASSO – To Sing you must Close Your Eyes and Draw [*]

Aleksandar Stupar
Saarland University
Saarbrücken, Germany
astupar@mmci.uni-saarland.de

Sebastian Michel
Saarland University
Saarbrücken, Germany
smichel@mmci.uni-saarland.de

ABSTRACT

We study the problem of automatically assigning appropriate music pieces to a picture or, in general, series of pictures. This task, commonly referred to as soundtrack suggestion, is non-trivial as it requires a lot of human attention and a good deal of experience, with master pieces distinguished, e.g., with the Academy Award for Best Original Score. We put forward PICASSO to solve this task in a fully automated way. PICASSO makes use of genuine samples obtained from first-class contemporary movies. Hence, the training set can be arbitrarily large and is also inexpensive to obtain but still provides an excellent source of information. At query time, PICASSO employs a three-level algorithm. First, it selects for a given query image a ranking of the most similar screenshots taken, and subsequently, selects for each screenshot the most similar songs to the music played in the movie when the screenshot was taken. Last, it issues a top-K aggregation algorithm to find the overall best suitable songs available. We have created a large training set consisting of over 40,000 image/soundtrack samples obtained from 28 movies and evaluated the suitability of PICASSO by means of a user study.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: [abstracting methods, indexing methods]; H.3.3 [**Information Search and Retrieval**]: [search process, selection process]; H.5.5 [**Sound and Music Computing**]: [methodologies and techniques]; I.4 [**Image Processing and Computer Vision**]: Applications

[*]The original quote "To draw you must close your eyes and sing" by Pablo Picasso has been reverted to become the main theme of our approach.
This work has been supported by the Excellence Cluster on Multimodal Computing and Interaction (MMCI).

General Terms

Algorithms, Experimentation, Human Factors

Keywords

Automatic Music Selection, Background Music, Slide Show, Soundtrack Recommendation

1. INTRODUCTION

Presenting the most impressive pictures from the last vacation on a Greek island can be spiced up with a well chosen background music that underpins the beauty and ease of life enjoyed in the last summer. Being faced with these problems from time to time, unfortunately not often enough with pictures of Greek islands, we experience that choosing the right music is everything but trivial. A good solution to this problem must have a big repository of music to choose from to allow for a high level of diversity to avoid playing the same old "born to be wild" song for all pictures somehow related to motorbikes, for instance. While the requirement of having many songs is relatively easy to fulfill, and in fact is fulfilled already with the amount of, and often also diverse, songs on the personal mp3 players, the challenge of selecting the right song, out of hundreds or even thousands of songs, becomes difficult. To tackle the problem, we have devised a fully automated approach, coined PICASSO, which is not only the name of the famous painter, Pablo Picasso, but also an acronym for *PIcture CAtegorization for Suggesting SOundtracks.* PICASSO makes use of training data obtained by sampling screenshot/soundtrack pairs from popular common movies, to learn an appropriate mapping from images to music at first hand, given the expertise of professional movie directors.

Sketch of the approach: Given a set of movies, we take random samples of them. For a given sample point, we investigate if the corresponding sound surrounding the sample by a couple of seconds resembles a piece of music. If so, we take the screenshot and the music piece and consider it as training data to our approach. We do this for very many samples, for dozens of movies. The set of considered movies should be large enough and also diverse such that the training base is big enough and also the resulting categorization captures all (or almost all) situations appearing in pictures people take.

At query time, we obtain an image, uploaded to our Web application, and, roughly speaking, in a naïve approach, find first the most similar screenshots taken (in the training phase) and then pick the most similar music piece in our song database compared the soundtrack at the time the screenshot was taken.

Admittedly, we tamper a bit with Pablo Picasso's quote *"To draw you must close your eyes and sing."*, trying to figure out if the inverse holds, too, namely, if we manage to find music (singing) by looking at pictures (drawings).

1.1 Contribution and Outline

With this paper we make the following contributions:

(i) We present an algorithm for soundtrack recommendation for an arbitrary wide range of user generated images as input.

(ii) We show how a training dataset to achieve this goal can easily be extracted out of popular common movies, capturing the knowledge of experienced movie directors.

(iii) Two cases of soundtrack recommendation, for single image and for multiple images, with grouping of multiple images, are described.

(iv) We report on the results of a user study evaluating the suitability of our approach.

This paper is organized as follows. Section 2 gives an overview of the related work. Section 3 presents our framework and briefly recaps basic techniques and similarity measures used in our approach. Section 4 presents the core concepts of PICASSO. Section 5 shows experiments results. Section 6 concludes the paper and gives an outlook on ongoing work.

2. RELATED WORK

Although the problem of soundtrack suggestion has been addressed in a couple of recent publications, it received relatively low attention. This is surprising given the vast amount of potential users, in particular with the increasing number of digital cameras and the big success of portals like Youtube [30] and Flickr [9].

Approaches based on language models, built for both the image and the soundtrack (song), are proposed in [29, 3]. By design, these approaches are only applicable to the songs with known lyrics, hence, ignore a huge area of classical music and all other instrumental music pieces. In addition, these approaches are limited in the way the language models are created, as the initial dataset of image annotations is very sparse. While the approach in [3] is able to recommend only images obtained from web, using publicly available image search engines, the approach in [29] is able to infer image annotations to personal images using an image similarity measure based on low level features. However, none of these approaches address the problem of recommending the soundtrack to a given image(s), instead, they start with the given soundtrack and then find the set of images to accompany the soundtrack.

The problem of recommending a soundtrack for a set of images and combining them into a slide show is addressed in [15]. This work focuses on impressionism paintings, matching the emotions in soundtracks with the emotions in a given painting, hence, does not address a general set of images (e.g., photographs). Emotions are discovered using a trained classifier on low level features for both the paintings and the music.

The approach in [5] considers the problem of suggesting music, generated from midi files, to car drivers, given a video of the scenery, taken by a camera installed in the car. The correlation, learned from a set of documentary movies, between a limited number of features (three for videos and three for music pieces) using only three quantization levels for each of the features, were explored and used later on for the recommendation. In contrast to their work, we aim at supporting a larger variety of images and aim at providing music suggestions based on arbitrary music collections, such as the users' personal mp3 collection.

Research on automatic video generation systems has been presented in [12, 13, 28]. The main focus of these works is on aligning the video transitions with the transitions in a given music piece. Although the ultimate aim of the proposed work is different to ours, the similarity with our work comes from a fact that both video (image) and audio content are explored for purpose of alignment (recommendation). Our approach could serve as a way to generate input to this automatic video generation, as recommended soundtracks can be feed into these systems for alignment and video generation.

We base our approach on low level features for both image-to-image and song-to-song similarity measures. The best proven low level features for image similarity used in object recognition are SIFT local image features [16]. However, the extraction of these features result in a large number of feature vectors per image and their comparison for image similarity is computationally more demanding than methods based on global image features (e.g., MPEG-7). Concerning this and the fact that we wanted to capture the "global" feeling of the image rather than individual localized features, we use standardized MPEG-7 [18, 4] color and texture low level features for computing the image-to-image similarity.

For the song-to-song similarity measure, we use both the spectral shape and temporal low level features. The spectral decomposition representation based on mel scale of human auditory system, named MFCC, has been used extensively in speech recognition [20], music genre classification [26], artist identification [17], etc. Beside MFCC, we use Chroma features [7], which represent the song's spectrum in intensity measures of each of twelve semitones of the musical scale. We also used spectral centroid, spectral rolloff, spectral flux, and time domain zero crossings music features described in [26]. All music low level features are extracted using the open source software framework Marsyas [27].

3. FRAMEWORK

PICASSO is based on the expertise of movie directors to select appropriate songs for specific scenes in their movies. This knowledge is made public, naturally, when movies are presented in cinemas, TV, or DVD, generating a huge knowledge base that perfectly fits our goal. We extract samples of screenshot/soundtrack pairs from a certain amount of these movies and feed them to our framework.

As a final training dataset we aim at having, for each of the screenshots taken, a list of songs (for instance, the user's private mp3 collection), which are available for recommendation, in decreasing order of likelihood to be recommended for a specific screenshot. Building this dataset is done in the following steps:

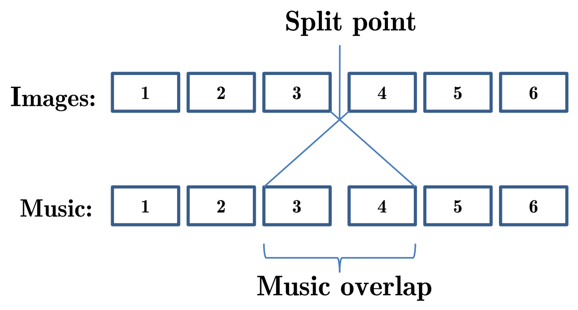


Figure 1: Splitting of the long scene

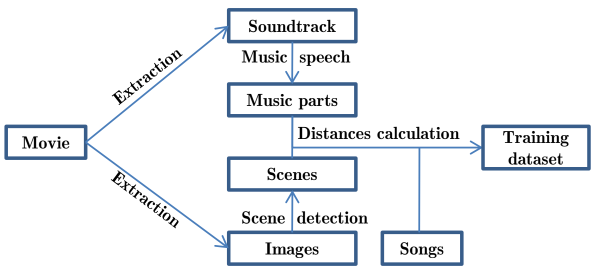


Figure 2: Training dataset building process

(i) the soundtrack of the movie is extracted
(ii) music/speech classification is done on the soundtrack
(ii) speech parts are discarded
(iv) screenshots, during the musical part, are taken
(v) parts of the same scene are detected
(vi) soundtrack is split according to the scenes
(vii) for each soundtrack part the distance to all songs is calculated
(viii) the song lists are sorted in increasing order of distance, resulting in our training set

Soundtrack and image extraction, from the movies, are easily achievable using one of many available tools and will not be covered here in greater depth. However, music/speech classification is not straight forward. For this task, we use the Naive Bayes [22] classifier, trained using the labeled dataset available at [19]. This training dataset for music/speech classification contains 64 speech samples, in different languages, and also 64 music samples of different genres. The already mentioned low level features (i.e., MFCC, Chroma, spectral centroid, spectral rolloff, spectral flux and time domain zero crossing) are used to train the classifier and for the later classification task. The Marsyas [27] tool is used for both the features extraction and the classification. The output of the classifier for a given soundtrack is a label (i.e., "music" or "speech") for each second of the soundtrack together with a confidence value. To be sure to use only the musical parts of the soundtrack, all parts that are classified as a music with the confidence value less than 95% and all speech parts are discarded. Musical parts of the soundtrack with length shorter than five seconds are also discarded, as they do not contain enough information to make the similarity measurement with the songs appropriate.

Screenshots are taken from the movies at each second of the remaining musical parts and, to obtain a more logical grouping, we divide these musical parts further into scenes. The scene detection is implemented by splitting the sequence of screenshots on positions where the image-to-image distance, measured as described in Section 3.1, is larger than a given threshold. The sequence of the screenshots from one split to the second one is considered a scene. This approach may not lead to exact movie scenes but, as we are only interested in image-to-image similarities for the further processing, it provides a reasonable grouping of similar images. To eliminate abrupt scene changes, for the same reason, we have eliminated short musical parts, i.e., we filter out and discard detected scenes whose length is shorter than 5 seconds.

Additionally, very long musical themes can cause problems as a too large amount of information can render them too specific and, hence, would not match to any parts of the available songs. This is why scenes, that are longer than 10 seconds, are split in multiple parts. Splitting these long scenes in multiple parts also improves the locality connection between the soundtrack part and the taken screenshot. To split the long scenes into smaller parts and to make musical parts of approximately the same length, we split the long scenes so that screenshots sequences are disjunctive, but with the possibility of having overlap between the music parts as illustrated in Figure 1. Screenshots together with links to theirs corresponding music parts are then saved for further processing.

After the musical parts are cut, first based on the music/speech classification and then based on scene detection, the distances between each of the musical parts of the soundtrack and all of the songs are calculated. The distance measure between the musical parts and the songs is described in Section 3.2. Songs are then ordered in increasing order of distance for each of the musical parts.

These ordered lists of songs together with the links from the taken screenshots represent our training dataset, which is later used for soundtrack recommendation, c.f., Figure 3 for an illustration. Each screenshot taken from the movie has a link to the part of the accompanying soundtrack which has a list of songs ordered by their distance to the soundtrack. The complete process of building the training dataset, from soundtrack extraction to song distance calculation, is illustrated in Figure 2.

3.1 Image Similarity Measure

For the image-to-image similarity measurement *we use the following MPEG-7 feature vectors* extracted from the images: scalable color, color structure, color layout, and edge histogram.

Scalable color describes all the colors found in the image by aggregating them in a color histogram. Each bin of the histogram represent one level of the color quantization. To lower the memory space needed for the histogram, the Haar transform [11, 25] is used for encoding. The standard defines three possible values for the number of Haar coefficients used: 128, 64, and 32.

The **color structure** descriptor is represented also by a histogram, where each bin represents the quantization of the color space and the value of the bin is a counter for the structuring elements, of 8×8 size, in which the respective color appears. Counting of color appearances is done while sliding the structural element over the whole image. The number of color quantization levels is a parameter for this descriptor and can by 184, 120, 64, and 32 as defined by the standard.

The **color layout** descriptor tries to summarize the distribution of colors in a given image, using YCbCr color space. First, the image is divided into 64 (8×8) equal sized parts and, in a second step, the average color for each of the parts is calculated. Discrete cosine transform (DCT) is performed on these 64 parts and only low frequency coefficients, for each component, are used.

The **edge histogram** descriptor is used for the spatial representation of the edges in the image. The image is divided into 16 (4×4) equal sized parts and the local edge histogram for each part is calculated. Each histogram has 5 bins representing 4 orientation directions and one used for non orientation specific edges. Concatenation of the local histograms with the histogram levels quantization produces the final descriptor.

In this work, we use common configurations of the MPEG-7 image descriptor coefficients, with 64 Haar coefficients for scalable color descriptor, 64 color quantization levels for color structure descriptor and 6 coefficients for Y component, 3 for Cr component and 3 for Cb component of the color layout descriptor.

After the distance calculation, for each of the descriptors, we need to combine them in one distance measurement for further comparison. We do this by first calculating the *standard score (z-score)* for each of the descriptors and then summing up all of the standard scores into a single score. The standard score is calculated by Formula 1, where μ is the expected value (mean) of the distances for the given descriptor and σ is the standard deviation of the distances for the same descriptor.

$$z = \frac{x - \mu}{\sigma} \qquad (1)$$

The mean μ and the standard deviation σ for the descriptors are approximated from the training dataset. The distances between each pair of the images, for each of the descriptors, are calculated, and then the mean and the standard deviation for the descriptor are approximated by the estimated mean and standard deviation of this set of calculated distances.

3.2 Music Similarity Measure

For the music similarity measures, the following low level musical descriptors are used: *MFCC*, *Chroma*, *spectral centroid*, *spectral rolloff*, *spectral flux*, and *time domain zero crossing*.

Most of the musical low level descriptors are based on the spectral representation of a given musical signal. The spectral representation is given by employing a short time Fourier transform on the signal. Each descriptor uses different characteristics of the spectral representations to emphasize certain features of a musical signal.

The **Mel Frequency Cepstral Coefficients (MFCC)** [20, 26, 17]descriptor is based on the mel scale, model of the human auditory system, emphasizing the spectral distribution of a signal in the manner perceived by humans. First, the spectral representation of a musical signal is transformed by the model and then the coefficients of the discrete cosine transform (DCT) are used as features. The DCT coefficients are used to lower the space requirement and save as much of the uncorrelated information as possible.

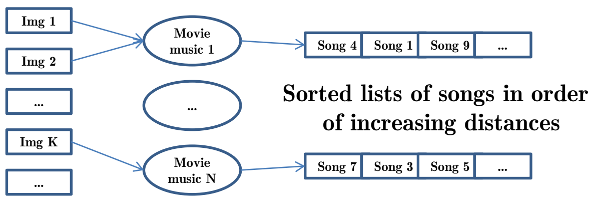


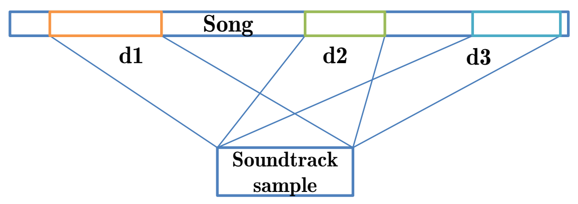
Figure 3: Training dataset structure

As the western music is based on a musical scales containing a subset of twelve basic musical semitones, the **Chroma descriptor** [7] represents the intensity of each of twelve semitones in the analyzed music part. It is a very valuable descriptor, used for musical similarity measurement, as it highly emphasizes melodical and harmonical characteristics of the analyzed musical piece.

The **spectral centroid** is defined as a center of gravity of a musical signal's spectral representation, while the **spectral rolloff** represents the frequency under which 85% of the signal's magnitude is located. The local spectral change, in the time domain, is represented by the **spectral flux** which is calculated as a squared difference between two successive spectral distributions. The **time domain zero crossing**, describing the noisiness of the signal, is defined as a number of times a musical signal crosses between positive and negative values in the time domain.

To calculate the similarity between two songs first the feature vectors of each of the descriptors are extracted for every 23.25 milliseconds of the song, resulting in 43 feature vectors per second (the default configuration for the Marsyas tool). Then, the pairwise similarity between the feature vectors is calculated and combined. The combination of distances of different descriptors into one distance measurement is done also using the sum of standard scores for each descriptor, as described in Section 3.1. The mean and the standard deviation for the standard scores are approximated using the mean and the standard deviation of the pairwise distances between the songs in the training dataset.

Calculating only pairwise distances between feature vectors is not enough as music also has a time dimension. One option to handle the time dimension is to calculate the sum of successive pairwise distances. As this option results in rigid comparison in time dimension, we use a more flexible option called Dynamic Time Warping [23]. **Dynamic Time Warping (DTW)** enables sequence matching with the variations in speed, making it possible to compare songs that are similar in content but with different speed measures. As speed change can be beneficial for the flexibility it also has to be limited such that the similarity between songs with different speed measures is less than between the songs with the same speed measure. This is also achievable using DTW by specifying the slope constrained condition and by specifying *Weighting Coefficient* for the case of speed change (time warp) and the case of no speed change (no time warp). In this work we use symetric variant of DTW with slope con-



d(song, soundtrack sample)= d1 + d2 + d3

Figure 4: Song to soundtrack sample distance

strained condition set to 1, and with the value of $\sqrt{2}$ for the *Weighting Coefficient* in case of speed change and value 1 for the case of no speed change.

Beside giving us a distance, as a result, DWT also gives us a part of the song that matches the best to the other song. Because soundtrack samples are a lot shorter than the songs in general, we use DWT to find *three* positions in the song that are most similar to the soundtrack sample. The sum of distances between the soundtrack sample and these three positions in the song is used as the resulting distance between the song and the soundtrack sample, as illustrated in Figure 4.

4. APPROACH

Once the training dataset is extracted we can use it for soundtrack recommendation. We consider two types of recommendations, single image recommendation and multiple images recommendation.

- In the single image recommendation, one image is submitted as a query and a list of songs is returned, in decreasing order of their suitability to the query image.

- In case of multiple images, we consider the images as the input for the slide show generation and try to recommend a grouping of the images together with the songs recommended for each of the groups. We base the grouping of the images on the similarity between the lists of recommended songs for each of the image.

They key idea of finding the most suitable song for a query image is to employ two phases of K-nearest neighbor searches, first, in the image domain, and second, between musical pieces. Selecting the K-Nearest Neighbors (KNN) means that in a multi-dimensional feature space, for a given feature vector the K nearest feature vectors are selected (cf., [24] for an overview).

Phase 1: When the query image is submitted, its distance to each of the images in training dataset is calculated. The images in the training dataset are then ordered in increasing order of distances to the query image and only the top-K images (i.e., the K-nearest neighbors) are used for the further recommendation process. We focus only on the effectiveness in this task and omit any optimizations or approximations of the K-Nearest Neighbors search to speed up the overall runtime. However, once runtime becomes an issue, we can benefit from the rich literature on the KNN problem, which has been extensively studied (cf., e.g., [2, 10, 1, 6]).

Phase 2: After the top-K images are found, the list of the songs together with their score for each image is retrieved. First, the name of the soundtrack sample for each image is retrieved and then the list of songs, ordered in increasing order of distance to the soundtrack sample, is retrieved for that soundtrack sample. As a result, each image, retrieved from the training set, has a list of songs in decreasing score order to that image. To avoid a negative impact of outliers to the recommendation process, smoothing is applied on the top-K retrieved songs for each of the top-K retrieved images.

For smoothing purposes, we first assign the score to each of the top-K retrieved images based on the similarity. The score is normalized to the $[0.1, 1]$ interval by Formula 2, where d is the distance of the image to the query image, m is the distance to the most similar image (i.e., the smallest distance), and M is the distance to the least similar image (i.e., the largest distance). Each of the image scores is then further assigned to the top-K songs in the list of corresponding images. An average score for each song is then calculated and used to reorder the songs. The ordered list of songs, by their new calculated score, is returned as the final list of recommendations.

$$z = 1 - 0.9 * \frac{d - m}{M - m} \tag{2}$$

For each of the K images we get K songs, leading to a final number of songs between K (in case of heavy redundancy) and K^2 (in case of no redundancy), which, after duplicate elimination and a final ranking, assembles the final result list. In our experimental evaluation, we use $K = 10$.

While a fully automated process requires only one suggestion, in a semi automated approach we could easily return many results to let users do the final assignment. To show that the ranking is meaningful, i.e., the rank 1 is preferred over lower ranks, we also evaluate the performance at rank K.

4.1 The Case of Multiple Images

When multiple images are given as input, we group these images using a clustering algorithm and recommend a soundtrack for each of the groups. To achieve this, we first perform the recommendation for each of the images. This results in the list of recommended soundtracks for each input image. The similarity between these recommendation lists is then used as the grouping criteria and is computed using a slightly modified Spearman's footrule distance measure [8], as follows.

First, we find the length of the shortest list as the length of the list depends on the number of unique songs in the top-K songs for each of the top-K similar images from the dataset. The lists are cut at the length of the shortest list to make the pairwise similarity measure comparable. The similarity between two lists l_1 and l_2 is calculated by Formula 3, where L is the length of the lists, and pos_1 and pos_2 are positions where the song s is located in each of the lists. If a song is found in only one of the lists, score is left unchanged, as specified by the formula.

As we can see, this kind of similarity measure takes into account not only the overlap between the lists but also an ordering of songs in the lists, imposing that the lists with the same ordering will be more similar than the ones with the same overlap but different ordering. As the Spearman's footrule distance is defined over full (i.e., complete) rankings, we adjust it to be used also for our partial rankings (top-K lists) (cf., [8]) by ignoring the songs not present in one of the lists. Also, the list length L is added to the formula with subtracted $L1$ distance, turning the measure into a similarity measure.

$$sim(l_1, l_2) = \sum_{s \in l_1 \wedge s \in l_2} L - |pos_1 - pos_2| \tag{3}$$

The similarity matrix, containing all pairwise similarities between the lists, is calculated and used for a *bottom-up hierarchical clustering* which produces a grouping of images as a result. We use a hierarchical clustering with the explicit number of clusters and the maximum cluster size specified. The process of clustering begins with creating a cluster for each image instance (i.e., list) and then iteratively merging the clusters until the required number of clusters is achieved. In each step, all pairs of clusters, except those whose cumulative size becomes larger than the specified cluster size limit, are considered for merging. Two clusters whose pairwise similarity is smallest, compared to other cluster pairs, are merged. The pairwise similarity between clusters is defined as the average pairwise similarity between their instances as shown in Formula 4, where c_1 and c_2 are clusters, i_1 and i_2 are their instances respectively. Cluster sizes, represented by $|c_1|$ and $|c_2|$, are defined as the number of instances in each of the clusters.

$$sim(c_1, c_2) = \frac{1}{|c_1| \cdot |c_2|} \sum_{i_1 \in c_1 \wedge i_2 \in c_2} sim(i_1, i_2) \tag{4}$$

In our experiments, we use the floor of the number of input images divided by 5 for the number of clusters, aiming at the clusters with average size of 5 images, and we use 7 as the specified cluster size maximum.

After the grouping of the images is finished a soundtrack needs to be recommended for each of the groups. We use two approaches for group recommendation, *average position* and *least misery*. First, we find all the songs that are present in all of the lists for a specific group. If there are no songs present in all of the lists, we find all of the songs present in $n - 1$ lists, where n is the number of the lists. If there are no songs present in $n - 1$ lists, we find the songs present in $n - 2$ lists, and so on.

- For the **average position** approach, we calculate the average position for each song found in the previous step. Songs are then reordered based on their average position in increasing order and the song with the smallest average position (best average ranking) is recommended for that group. The average ranking can sometimes be misleading for a group recommendation, as some group members can be highly disappointed by the recommendation but the average satisfaction is still high because of the high satisfaction of other group members.

- The **least misery** approach is used in group recommendation with intention to circumvent this problem. In least misery approach group recommendation is valued as a value gained by the group member who likes the recommendation the least. In our case, we find the maximum position (lowest rank) for each of the songs found in previous step and use this position as a score for recommendation. Like for the average ranking approach, songs are reordered on their new score in increasing order, and the song with the smallest score is recommended for that group.

5. EXPERIMENTAL EVALUATION

The training dataset used for the experiments is distilled, as described in Section 3, out of 28 movies of different genres. The movies were selected to have a high variance in content, with the intention to provide a high coverage of the training dataset, without having to process very many (random) movies. Processing these 28 movies resulted in 46,594 screenshots taken, with approximately 13 hours of accompanying soundtrack music.

We use 275 songs as potential soundtrack recommendations. As these songs are played to the users during the performed user study, we had to make sure to use only copyright free material. We obtained this dataset by downloading songs from the music2ten site [21], a site which contains only music given away by its creators for free use, as stated on the site: "All the music selections at music2ten.com are MP3s that are given away free with the artists' blessings.". As our songs dataset contains fairly unknown material, i.e., artists are not promoted by the big labels, it assures that there is no bias in users' assessments towards more popular songs.

5.1 Evaluation Set Up

To obtain a meaningful evaluation in absence of a publicly available benchmark, we conducted a user study for both cases of recommendation, the single image recommendation and the recommendation for the case of multiple images. We have asked users from our university environment to grade the suitability (aka., relevance, appropriateness) of a suggested soundtrack w.r.t. a given image or to the given slide show, in case of multiple images. Users were provided with the following options: "fits very good", "fits good", "fits ok", "does not fit" and "total miss". We assign grades from 5 to 1 for each of the option for the further numerical analysis of the results. Grade 5 is assigned to "fits very good", 4 to "fits good", and so on, having grade 1 for "total miss".

We include a random recommendation to the user study to help us interpreting the results. Additionally, to be able to see how much the ranking of the recommended soundtracks make sense, we include also the recommended soundtrack at rank 10 in the evaluation of the single image case. That means, during the evaluation, users were asked to grade the first ranked, the tenth ranked, and a randomly recommended soundtrack for each query image. To get consistent results for the random recommendation, a song was randomly chosen for every query image and then used consistently throughout the whole study (i.e., for all users).

The evaluation is performed using an online tool which we have developed, shown in Figure 5. This tool enables users to play three recommended songs for each query image, and to choose how well each song fits to the query. The user judgment on how well some soundtrack fits to the query may be influenced by the previously heard recommended soundtrack. That is why the evaluation tool places the songs on the web page in random order. The order of songs in which

	Average position	Least misery	Random
Average	3.69	3.31	2.38
Std. Dev.	0.95	0.95	1.26
With aggrement	3.90	3.49	1.89

Table 1: Multiple images grades

	First rec.	Tenth rec.	Random
Average	3.21	3.08	2.97
Std. Dev.	1.21	1.20	1.32
With aggrement	3.32	3.17	3.05

Table 2: Single image grades

they were placed on the web page is kept for the current session so that users are not confused with the new ordering when the web page is reloaded.

The evaluation for the single image recommendation is done on the following 12 pictures, shown in Table 4: (1) boats on the sea, (2) skeleton with a headache, (3) man with two guns, (4) formula one pit stop, (5) queue of four men, (6) red rock mountain, (7) satellite dish with stormy weather, (8) futuristic city buildings, (9) monks in orange dresses, (10) palm tree in front of the blue ocean, (11) stop sign on a crossing, and (12) stars in a galaxy. These pictures were selected to be very different in content, as we want to evaluate how well the recommendation performs in the general case, rather than in one or a few specific cases. We had to limit the number of query images to 12, as each user was asked to assess the suitability of 3 recommended soundtracks, for each of the query images, resulting in a tedious and time consuming task of listening to 36 different songs.

For the case of multiple images, we also add a random recommendation to the evaluation, together with the evaluation of the average position and the least misery approaches. We use 36 images taken indoors and outdoors, on various locations all around the world, for the slide show creation. The grouping of the images is done using hierarchical clustering, as described in Section 4.1. Three slide show videos are made with the recommended soundtracks: by average position approach, by least misery approach, and by the random selection of the songs as a soundtrack. The slide show videos are made with a light "dissolve" transitions between the images, inside the groups, and with the heavier "page fold" transition between the images from different groups. Each image is shown for five seconds, resulting in the total length of three minutes for each of the slide shows. The transition between the songs is done with the fade out effect on the end of each song and with the fade in effect on the beginning of each of the songs. The users are asked to rate how well the soundtrack fits to the slide show, for each of the three cases. This evaluation is also done with our online tool, where the slide shows were played and graded by the users. The placement of the slide shows on the web page is also in random order, for the same reason as for the song placement in the single image case.

5.2 Evaluation Results

A total of 13 users participated in the evaluation of the generated slide shows, i.e., the multiple images case. The aggregated results of the evaluation are shown in Table 1 where each option for the soundtrack's suitability was assigned a grade, as already explained in the previous Section. As we can see, the average grade for the average position approach is lot higher than the random soundtrack recommendation. The least misery approach has an average grade a lot higher than random, but slightly lower than the average position approach, indicating that users tend to grade the appropriateness of the soundtrack to a group of images on average rather than on the minimum suitability.

Besides having just a good average grade for the recommendation, it is also very important that people actually agree on how well the soundtrack fits to the image (or the set of images). The agreement between users is very important as high average grades could still have users with high disappointment, what we want to avoid. The agreement on the suitability of the recommended soundtrack is represented in Table 1 by the standard deviation of the results. The lower the standard deviation the larger the level of agreement between users. We see from the standard deviation that users agree more for the soundtracks recommended by our two approaches compared to the random recommendation.

For an easier comparison, a score that takes into account both the relevance measure and the agreement is constructed as a ratio between the average grade (the mean) and the standard deviation, aiming at giving higher grades to the evaluation on which people agree, and a lower score to the one on which people disagree. This combined score is presented in the last row of the Table 1. We see that the combined score for both of our approaches is a lot higher than the one for the randomly recommended songs.

Table 2 summarizes the results of the single image evaluation. Again, 13 users participated in the evaluation, but not all of them have evaluated the soundtracks for all of the 12 images. Each image is evaluated 10.83 times in average. We can see that the average grade for the first recommended song is higher than the random recommended song and that the tenth recommended is graded in between the first recommended and the random, showing that our ranking is indeed in line with the users' perceived ranking. As we can also observe, the difference in average score between the first recommended and the random one is lot smaller than the difference between the average position approach and the random one in case of multiple images evaluation. This kind of result is well expected as the slide show contains multiple images, accumulating the users perceived soundtrack relevance over all of the images.

For the case of single images the average agreement over all 12 tests, in terms of standard deviation, is better for the first recommended and tenth recommended song than for the random one. This shows that not only the first ranked and ten ranked recommended soundtrack have higher score but that users also agree more on these scores compared to randomly recommended song. The score, combining the relevance with the agreement, averaged over all 12 tests is shown in the last row of Table 2. We can see that the combined score for all three cases is slightly increased, compared to the average grade, but yields no significant impact.

Runtime & Scalability

Although we did not focus on the efficiency when designing PICASSO, we measured the runtime of the query processing

Thread(s)	4	3	2	1
Runtime (sec.)	86.13	99.19	132.09	213.84
Queries/sec.	9.75	8.46	6.35	3.92

Table 3: Scalability measurements

task to show that this approach can be used in real-time environments.

The average runtime for the 12 single image queries is 0.629 seconds per query image, with a standard deviation of 0.728. As the K-nearest neighbor search was done completely in main memory, it is yielding a very low runtime. Most of the running time was spent on extracting the image features for the query images, as these were not precomputed. The recommendation for the multiple image case was done in 14.933 seconds for our case of 36 input images. Here, again, most of the runtime was spent on the features extraction of each of the 36 images.

As all required indices, in total less than 500MB Java VM memory (in our setup with 28 movies), are kept in main memory, the runtime is CPU bound due to the feature extraction task. Given this small amount of required memory that nowadays every simple commodity PC provides, the indices can be replicated over many such PCs for scalability reasons. Additionally, the indices are read-only, creating even on a single machine with multiple cores no conflicts between memory accesses , rendering the approach "embarrassingly scalable", a common term for these kinds of parallelization. To show the scalability of the approach on a single machine, we measure the running times and the throughput (i.e., images per second), varying the number of threads used, for 840 query images, shown in Table 3. The measurements were done on the single machine with single four core processor. We did not measure the case of parallelizing the tasks on several separate machine, as the scale-up in throughput is obvious.

Once the memory consumption hits the limit of available memory, we can partition the indices on a per-movie basis. Then, for each machine, we obtain recommendations based only on a subset of all movies. These "local" results can get, however, easily merged by their similarity score in a final aggregation step (i.e., a very simple list merging task). We did not investigate this further. In a real world deployment one would install the approach directly in the cloud to benefit from the provided elasticity in terms of required computing power.

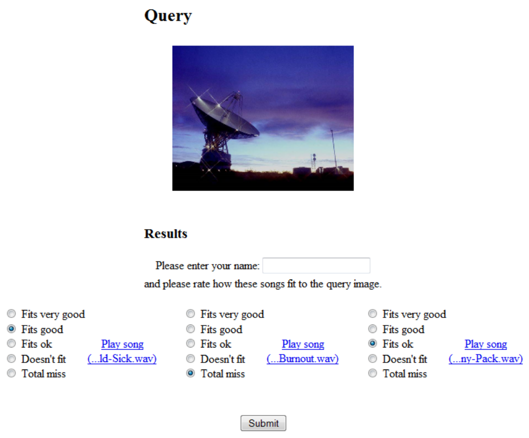
5.3 Lessons Learned

The lessons learned from the evaluation presented above can be put in the following statements. Naturally, in most cases these are not hard facts being universally true, rather reflecting the first insights obtained throughout developing and evaluating our approach.

- Publicly available movies contain expert knowledge on matching the soundtrack to the given images

- It is possible to extract that knowledge and use it for soundtrack recommendation

- Fairly simple methods for recommendation, in conjunction with the extracted training dataset, work quit well


Query

Results

Please enter your name:

and please rate how these songs fit to the query image.

○ Fits very good		○ Fits very good		○ Fits very good	
● Fits good		○ Fits good		○ Fits good	
○ Fits ok	Play song	○ Fits ok	Play song	● Fits ok	Play song
○ Doesn't fit	(...ld-Sick.wav)	○ Doesn't fit	(...Burnout.wav)	○ Doesn't fit	(...ny-Pack.wav)
○ Total miss		● Total miss		○ Total miss	

Figure 5: Screenshot of our evaluation tool

- Users find that the recommendation of the soundtracks, for single image or set of images, works well

- Users agree on the ranking of the recommended soundtracks

- The approach is scalable

6. CONCLUSION

We have presented PICASSO, an automated approach to recommend a soundtrack for a picture or a series of pictures. The main focus of the approach is the extraction of knowledge from popular, publicly available, common movies and how these information can be used for our specific task. We have shown how images and music can be matched together by identifying pairs of the most similar screenshots of the movies and the surrounding soundtrack music. PICASSO is based on the usage of low level features for similarity comparison between images and between songs, rendering it applicable for all cases of image/soundtrack matching problems as these features are always available. We have demonstrated the applicability of PICASSO by reporting on the results of a study involving user generated feedback.

As part of our ongoing and future work, we plan to address the problem of integrating user preferences, in terms of music and movie taste, to our approach. We believe that this can lead to an even higher level of user satisfaction. Potentially, we can benefit from existing information in Web portals, such as imdb [14] or youtube [30], to model user preferences.

7. REFERENCES

[1] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. In *SIGMOD Conference*, 1998.

[2] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *ICDT*, 1999.

[3] Rui Cai, Lei Zhang, Feng Jing, Wei Lai, and Wei-Ying Ma. Automated music video generation using web image resource. In *ICASSP*, 2007.

Table 4: The 12 query images used in our user study.

[4] Shih-Fu Chang, T. Sikora, and A. Purl. Overview of the mpeg-7 standard. In *IEEE Trans. Circuits Syst. Video Techn.*, June 2001.

[5] Marco Cristani, Anna Pesarin, Carlo Drioli, Vittorio Murino, Antonio Rodà, Michele Grapulin, and Nicu Sebe. Toward an automatically generated soundtrack from low-level cross-modal correlations for automotive scenarios. In *ACM Multimedia*, 2010.

[6] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, 2004.

[7] D.P.W. Ellis and G.E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *ICASSP*, 2007.

[8] Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. Comparing partial rankings. In *SIAM J. Discrete Math.*, 2006.

[9] Flickr. `http://www.flickr.com/`.

[10] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[11] Alfred Haar. *Zur Theorie der orthogonalen Funktionensysteme. Mathematische Annalen, 69.* 1910.

[12] Xian-Sheng Hua, Lie Lu, and HongJiang Zhang. Automatic music video generation based on temporal pattern analysis. In *ACM Multimedia*, 2004.

[13] Xian-Sheng Hua, Lie Lu, and HongJiang Zhang. Optimization-based automated home video editing system. In *IEEE Trans. Circuits Syst. Video Techn.*, 2004.

[14] Internet movie database. `http://www.imdb.com`.

[15] Cheng-Te Li and Man-Kwan Shan. Emotion-based impressionism slideshow with automatic music accompaniment. In *ACM Multimedia*, 2007.

[16] David G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, 2004.

[17] Michael I. Mandel and Dan Ellis. Song-level features and support vector machines for music classification. In *ISMIR*, 2005.

[18] B. S. Manjunath, Jens-Rainer Ohm, Vinod V. Vasudevan, and Akio Yamada. Color and texture descriptors. In *IEEE Trans. Circuits Syst. Video Techn.*, 2001.

[19] Marsyas - music/speech dataset. `http://marsyas.info/download/data_sets`.

[20] A. Martin, D. Charlet, and L. Mauuary. Robust speech/non-speech detection using lda applied to mfcc. In *ICASSP*, 2001.

[21] Music2ten. `http://music2ten.com`.

[22] Irina Rish. An empirical study of the naive Bayes classifier. In *IJCAI-01 workshop on "Empirical Methods in AI"*, 2001.

[23] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, February 1978.

[24] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures.* Morgan Kaufmann, 2006.

[25] Zbigniew R. Struzik and Arno Siebes. The haar wavelet transform in the time series similarity paradigm. In *PKDD*, 1999.

[26] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. In *IEEE Transactions on Speech and Audio Processing*, July 2002.

[27] George Tzanetakis. Music analysis, retrieval and synthesis of audio signals marsyas. In *ACM Multimedia*, 2009.

[28] Jinjun Wang, Changsheng Xu, Engsiong Chng, Lingyu Duan, Kongwah Wan, and Qi Tian. Automatic generation of personalized music sports video. In *ACM Multimedia*, 2005.

[29] Songhua Xu, Tao Jin, and Francis Chi-Moon Lau. Automatic generation of music slide show using personal photos. In *ISM*, 2008.

[30] Youtube. `http://www.youtube.com`.