

Feature Fusion for Efficient Content-Based Video Retrieval

Machiel Visser, *Student Member, IEEE*

Abstract—Content-based video retrieval is a complex task because of the large amount of information in single items and because databases of videos can be very large. In this paper we explore a possible solution for efficient similar item retrieval. In our experiments we combine relevant feature sets together with a learned Mahalanobis metric while using an efficient nearest neighbor search algorithm. The efficient nearest neighbor algorithms we compare are Locality Sensitive Hashing and Vantage Point trees. The two options are compared to several baseline systems in the general video retrieval framework. We used three sets of features to test the system: SURF features, color histograms and topics. The topics were extracted using a Latent Dirichlet Allocation topic model. We show that fusing the individual feature sets with a learned metric improves the performance upon the best individual feature set. The feature fusion can be combined with an efficient nearest neighbor search algorithm to reduce the number of exact distance computations with limited impact on retrieval performance.

Index Terms—Content-based video retrieval, feature fusion, metric learning, efficient retrieval, nearest neighbor search, locality sensitive hashing, vantage point trees

I. INTRODUCTION

To use a large database of videos effectively it is essential to have a good retrieval mechanism that enables the user to focus on only that what is important to him. Retrieval can be active querying of the database by the user, but it could also mean that the user gets recommendations based on his interests or search history.

Video retrieval has as goal to find and only find the relevant items to a query. It is a very broad area of research and has overlap with many other research areas such as information retrieval, speech recognition, pattern recognition and computer vision. Most of the techniques used for video retrieval are not unique to that domain.

The task of video retrieval can be divided in two video main categories: (1) exact copy retrieval and (2) similar item retrieval. The main problem of exact copy retrieval is to distinguish between similar items and manipulated versions of the same item. For similar item retrieval it is important to distinguish between similar and dissimilar items with respect to a chosen definition of similarity.

This paper considers the problem of using content-based features from videos in a similar item retrieval system using an example video as a query. We do not consider the detection of certain a priori concepts within a video. Content-based features are the features that are extracted from the video itself. The other common source of features is meta data, which is information that is attached to the video [13]. Both sources of information are suitable to base the similarity measure between videos on. Content-based features can be used in combination

with meta data to enhance the performance, but here we will look solely at the implications of building a retrieval system based on content-based features.

Multiple features can be extracted from a video, such as visual region descriptors or output from a speech recognizer [14]. To use those features for a similarity measure for videos, they need to be combined in a certain manner. Combining features for a classification or comparison task is called feature fusion. The features can be fused directly, which is called early fusion, or the decisions can be fused after using the individual feature sets for retrieval, which is called late fusion.

Using a similarity measure for retrieval can become slow when the database to retrieve from is large. For a useful retrieval system it is important that the time complexity does not scale linearly with the number of items in the database. To accomplish this there exist several more efficient near neighbour search algorithms that make use of efficient structures to reduce the number of exact distance computations.

In this paper we will show that it is possible to improve retrieval performance in an early fusion scheme with a learned metric over the feature space. Second, we will show that this system can be improved by an efficient nearest neighbour algorithm; in terms of the number of exact distance computations with limited impact on retrieval performance. It is important that the fusion scheme and the near neighbour algorithm are compatible.

This paper is organized in the following way. We will start with an overview of the related work about features, similarity measures, feature fusion and efficient near neighbour search. Then the specific techniques that we use are discussed in more detail. Followed by the experiments we have conducted to answer the questions raised, and the results of these experiments. The paper ends with a conclusion and discussion on the results of the experiments.

II. RELATED WORK

A. Similarity

There are two distinct information sources that can be used to derive a similarity measure for similar item retrieval: meta data and content-based information. Meta data is the information that is attached to an item such as title, tags, comments or usage statistics. Content-based information is information extracted from the item itself. The former is more common nowadays because the information is directly available without extraction techniques and is a more compact representation of the video compared to content-based features. The main problem with meta data is the manual annotation step that makes the features less reliable. Content-based retrieval can

be useful when large amounts of data need to be annotated automatically or to complement the meta data based retrieval to improve performance [26].

For content-based retrieval systems, the definition of similarity between items is very important. Sometimes there is a domain specific measure of similarity already available for the task. But in most situations the choice of similarity measure is not obvious [23].

The similarity of two items can be defined as a domain specific function over the items, or in a more general way as a measure of how close items are in the feature space. The similarity of two items is then related to how close they are in the feature space given a measure of distance. But closeness in a given space is not a well-defined property; there are multiple ways to quantify closeness or distance in a space [6].

Distances over vector spaces are typically defined by a metric, or the norm of the difference between two vectors. Most common is the Euclidean distance metric or L_2 -norm. Besides general metrics that can be applied to every $\mathbb{R}^d$ space, it is possible to use side information and construct a Mahalanobis metric from that specific to one $\mathbb{R}^d$ space. Side information can be class labels or a pair or triple wise similarity. There exist several methods to learn a metric based on side information [34].

B. Modalities and features

A video can be seen as a combination of several modalities of data. Modalities here can be understood as data observed through different senses; such as visual information and auditory information. Each modality contains information that can be relevant for the task [27]. To capture the information from a modality, one or multiple features can be extracted. Individual features are generally not able to capture all the relevant information from an item; therefore often multiple features are used. In the visual modality there are several classes of features such as structure, color and motion features. The classes of features in the auditory domain are speech, music and general audio features [10].

The optimal set of features is different for every application, and is constrained by the desired performance, computation time and storage space requirements. Although computation speed and storage space are increasing, they are not unlimited and therefore choices need to be made for practical applications [36].

C. Fusion of features

After the extraction of features, possibly from different modalities, the features need to be combined for the task. Features can be used directly for certain tasks, but it is often better to process the features first, using domain knowledge, to make them more suitable. Several fusion schemes are possible, but the main distinction is between early and late fusion. In an early fusion scheme the features from different modalities are first combined, in for example one feature vector, and then used for classification or comparison. A late fusion scheme uses the individual features first for classification or comparison, and then combines the results of this [28].

Other fusion schemes are possible, such as double fusion which is a combination of complete early and late fusion schemes [21]. It was also proposed to adapt the fusion scheme to the incoming query by grouping queries in several classes with their own optimal fusion scheme [18]. There is no consensus on whether early or late fusion is better in general; it really depends on the application at hand, the features that are used and the type of task one wants to accomplish [19]. Early fusion is capable of exploiting the feature correlations that will be lost after the decision stage, but late fusion is in general a lot simpler and therefore often preferred.

D. Metric learning

One way to do early fusion is with a learned metric because it scales the distances of features with different ranges. The general idea of metric learning is to construct a metric for a certain space using side information. By using side information it is expected that the metric reflects certain characteristics of the feature space better than standard metrics in the context of a specific task. And later in the decision stage, this should result in a lower error. A Mahalanobis distance metric is one such metric, and most common in this setting [17].

There are several types of side information possible for metric learners. Most ideal would be to have the real similarities for pairs of items as similarity constraints. But since the real similarity is often ill-defined and manual annotation would take too much time, constraints must be based on available information. Information that is available, or would be easy to obtain, are the class labels. Classes can be the categories or genres of the items. Most metric learners require pair wise similarity constraints, but others require triple wise constraints [34].

A Mahalanobis metric A can be used directly in a nearest neighbor setting to measure distance between items. But it can also be decomposed to a transformation G on the feature space such that $X_{new} = GX$. The original metric and the transformation are equivalent in that they both contain all information and are related as $A = G^T G$ [17]. The decomposition can be done for example with Singular Value Decomposition (SVD) as $A = USV^T$, and since metrics are symmetric, U and V are identical. Transformation G can then be obtained from the SVD as $G = (U * S)^T$.

Learning a metric based on a set of constraints is a complex problem, and given a set of constraints it is in general impossible to satisfy them all [9]. Because of this, metric learning algorithms are framed as optimization problems that go for approximations to the exact solution. The earlier approaches made use of semi definite programming methods such as [35], but often use eigenvalue decompositions to solve the problem which is quite expensive.

Other noteworthy methods are Relevant Component Analysis [25] that works under the assumption that class covariances are equal. The Large Margin Nearest Neighbour [32] algorithm uses triplets with a similar and a differently labelled example as constraints, and tries to create a margin between them. Learning algorithms exist also to directly learn the feature space transformation G such as Neighborhood Component

Analysis (NCA) [12].

More recent methods use iterative updates of the metric to satisfy the constraints, while minimizing a regularized loss function between the learned metric A and a standard metric A_0 such as the Euclidean distance. The Pseudo Metric Online Learning Algorithm (POLA) [24] still uses eigenvector decompositions in this setting. Other algorithms such as LogDet Exact Gradient Online (LEGO) use the exact gradient in the update step [16]. Information Theoretic Metric Learning (ITML) uses Bregmans cyclic projection algorithm to update the metric, which is quite efficient compared to other metric learners [9]. Therefore we will use the ITML metric learner in this paper.

E. Time complexity and reduction

For practical retrieval applications it is important that the answer to a query is retrieved in real time. There are two aspects of the retrieval process that can be optimized for efficient retrieval [22]. First, the number of items n that are compared is high if an exhaustive search conducted. There exist several efficient retrieval algorithms that reduce the number of comparisons, such as space partitioning methods and hashing methods. The other aspect is the feature space dimensionality d that influences the time a pair wise comparison takes. Several dimension reduction techniques exist.

Dimension reduction aims to reduce the original set of features to a smaller set of features. One way is to select the best n features and throw out the remaining features. Another way is to extract new features from the original features such that the new features contain as much of the relevant information as possible and not much irrelevant information. This transformation of the input data can be linear such as PCA, or non-linear.

For video retrieval it is more important to use space partitioning or hashing methods to reduce the time complexity because n is in real systems much larger than d , though it can be preceded by a dimension reduction step. In an exhaustive search algorithm the query is compared to every item in the database and the results are then ranked. This is almost intractable for practical applications. The order of the time complexity of a query search can be reduced significantly by using an appropriate indexing structure. Space partitioning structures are good for this, but hashing methods that use an approximate similarity measure are also common [1].

Space partitioning methods divide the space in a set of smaller spaces. This is often done hierarchically and the resulting structure is then a tree. Vantage Point trees which are a specific type of Binary Space Partitioning are an example of this. Hashing methods such as locality sensitive hashing create one or more hashes of the items [2]. It is important that those hashes capture the chosen definition of similarity approximately.

III. APPROACH

The main idea of our approach is to combine an early feature fusion scheme with an efficient nearest neighbour search algorithm for content-based video retrieval. We use the

Information Theoretic Metric Learner (ITML) for fusion of the features to transform the feature space after the concatenation to a single feature vector. With the fused feature vectors we do the retrieval using two different methods: (1) Locality Sensitive Hashing (LSH) and (2) Vantage Point (VP)-trees.

The first step of the entire system is processing a video before inserting it in the database or using it to query the database. Processing consist of segmentation and the extraction of several features from the visual and audio streams. The next step is the fusion of the different features into one set of features. This set of features can then be used for retrieval. The two systems are compared with each other and with a couple of baseline systems that will be explained in the experiments section.

A. Fusion with Information Theoretic Metric Learning

The idea of ITML is to satisfy distance constraints while minimizing a loss function. The purpose of the loss function is to ensure that the resulting metric generalizes well to other data. This is done by minimizing the difference of the metric A from a standard metric A_0 to prevent over-training.

J.V. Davis et al. [9] formulate the problem of metric learning as minimizing the differential relative entropy, also called Kullback-Leibler divergence, between two Gaussians parametrized by A and A_0 . The Gaussian corresponding to A is $p(x; A) = \frac{1}{Z} \exp(-\frac{1}{2} d_A(x, \mu))$. The differential relative entropy is a measure of the distance between the two Gaussians:

$$KL(p(x; A); p(x; A_0)) = \int p(x; A_0) \log \frac{p(x; A_0)}{p(x; A)} dx \quad (1)$$

This minimization problem is than subjected to the following similarity and dissimilarity constraints over items in the feature space; where S and D are the sets of similar and dissimilar items respectively, and u and l are thresholds:

$$d_A(x_i, x_j) \leq u \quad (i, j) \in S \quad (2)$$

$$d_A(x_i, x_j) \geq l \quad (i, j) \in D \quad (3)$$

In [9] it is shown that the problem of minimizing the relative entropy can in this case be expressed as minimizing a LogDet matrix divergence over A and A_0 , which is a convex function over the cone of positive-definite matrices. The constraints can be written as products over the feature space items and the metric. And to guarantee feasibility of the problem slack variables are introduced in the formulation of the problem; where γ regulates the trade-off between the minimization and the constraints. This leads to the following minimization problem:

$$\min_{A \succeq 0} D_{ld}(A, A_0) + \gamma D_{ld}(\text{diag}(\xi), \text{diag}(\xi_0)) \text{ s.t.} \quad (4)$$

$$\text{tr}(A(x_i, x_j)(x_i, x_j)^T) \leq \xi_{c(i,j)} \quad (i, j) \in S$$

$$\text{tr}(A(x_i, x_j)(x_i, x_j)^T) \geq \xi_{c(i,j)} \quad (i, j) \in D$$

To solve this problem iterative projections are made of the current metric onto one single constraint by Bregman

projections until a set convergence threshold is reached. One update using a single projection is as follows; where β is the Lagrange multiplier for the current constraint:

$$A_{t+1} = A_t + \beta A_t (x_i - x_j)(x_i - x_j)^T A_t \quad (5)$$

The parameters for this algorithm that are not automatically chosen by the algorithm are:

- The initial distance thresholds u and l for the positive and negative constraints.
- The trade-off γ between minimizing the divergence and satisfying the constraints.

To set the right parameters for the learning algorithm we compared the distributions of distances for the positive and negative constraints. Comparing different settings for the distance thresholds, the best results are obtained by settings them at the extremes of the original distances. This means that initially all constraints are violated. Finding the right gamma depends on the specific feature set the metric is trained with; therefore every new metric should be trained with different settings. See for more details [31].

B. Retrieval

We compare two options for efficient retrieval in the setting of nearest neighbor search. There are many more nearest neighbor algorithms possible. However, LSH is interesting because it has the possibility to integrate a metric directly. The other option we compared it with is Vantage Point trees, which is a way to do Binary Space Partitioning. VP trees are more suitable than other space partitioning structures because the partitioning is done based on distances instead of planes. Because of this the potential nearest neighbors can be found more efficiently.

1) *Locality Sensitive Hashing*: P. Indyk and R. Motwani [15] introduced the original idea and later it has been extended with different options. The goal was to find an approximate nearest neighbour algorithm that runs in sub-linear time. The general idea of Locality Sensitive Hashing (LSH) is to hash the items with several hash functions such that the locality of the items is approximately preserved. Items that are close in the original feature space should have high probability of collision after hashing, meaning that they end up with the same hash or in the same bucket:

$$\Pr_{h \in \mathcal{F}} [h(x) = h(y)] \approx \text{sim}(x, y) \quad (6)$$

Several different hash functions can be used in this setting. Most practical implementations make use of projection-based hash functions. A projection-based hash function projects items onto a randomly oriented line in the high-dimensional feature space. The standard-projection based functions divide the line in two and assigns one bit to the item x :

$$h_{(r,A)}(x) = \text{sign}(r^T x) \quad (7)$$

Another option is the family of p -stable distribution functions that divide the line in several bins with a chosen width w and assigns an integer to the item x :

$$h_{(r,A)}(x) = \left\lfloor \frac{(r^T x + b)}{w} \right\rfloor \quad (8)$$

For p we use 2 because we work in a Euclidean space and the b in the numerator is to randomize the bin boundaries. P -stable distributions have the desirable property that the p -norm of i.i.d. variables has the same distribution, up to scaling and location, as the individual variables.

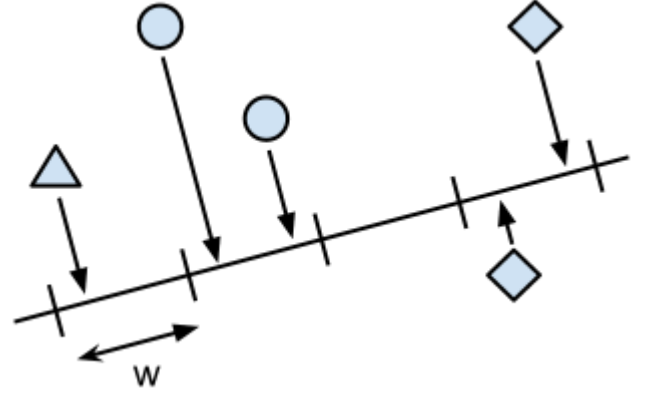


Fig. 1. LSH with p -stable hash function

One projection from a high dimensional space captures only a small portion of the discriminative information that the items have. To capture enough discriminative information in the hashes of items, LSH uses multiple hash functions and concatenates them into one hash g :

$$g(x) = [h_1(x), \dots, h_k(x)] \quad (9)$$

The probability that similar items end up in the same bucket gets smaller when k is larger. To counter this effect LSH uses also multiple tables with their own hash function. To retrieve from an LSH structure every table is queried and the results from all the tables are ranked together.

In the case of p -stable projection based hash functions, which we will use here, there are three parameters to be set before an LSH index can be constructed. The number of hash functions k that determines how discriminative the buckets are. The buckets should not be too small, otherwise nothing will be retrieved. The number of hash tables L that increases the probability of retrieving similar items, but only up to a certain level because otherwise it becomes an exact nearest neighbour search again. The last parameter is the bin width which strongly depends on the type of data.

Because only a small number of the buckets are non-empty, it is common to apply a second-level standard hash to the buckets. This way the LSH structure does not grow exponentially in memory with the number of hash functions k .

The transformation can be included in the random hash planes of projection-based hash functions. Due to integration within the hash functions, no transformation of the input features is required. Both the standard projection-based hash function, as well as the p -stable hash functions can be used with the transformation G :

$$h_{r,A}(x) = \text{sign}(r^T Gx) \quad (10)$$

$$h_{r,A}(x) = \left\lfloor \frac{r^T Gx + b}{w} \right\rfloor \quad (11)$$

To set the parameters for optimal retrieval performance we varied all three parameters, k , L and w , over the range of possible values. From this three-dimensional space we tested 100 configurations at random to find the best configuration [31].

2) *Vantage Point trees*: Peter N Yianilos [37] is the first to refer to this algorithm, but it was introduced a couple of years earlier. What makes Vantage Point trees more suitable for some applications than other Binary Space Partitioning methods is the spherical partitioning of the space. General Binary Space Partitioning algorithms divide the space recursively into two parts by dividing the space with a hyper plane. Because the partitioning is done with hyper planes the exact distances are not preserved in the structure, which is important for retrieval systems that rank the results according to their distance to the query.

Vantage Points trees partition the space by recursively taking a point and dividing the remaining region into two parts. One part is the region that is within a certain threshold from chosen point, and the remaining region is the other part of the binary partition.

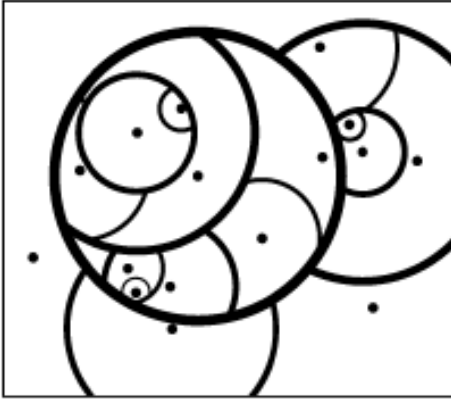


Fig. 2. 2 dimensional VP-tree [20]

A search through the structure for nearest neighbours with respect to a query is started at the root node. The depth of the search is controlled by parameter τ . Children of a node are considered as potential nearest neighbours if they lie on the right side of the threshold of the current node. The algorithm maintains a short-list of size k , and parameter τ is set at the distance of the last node in the short-list. As soon as the short-list is filled, parameter τ is lowered such that only nodes closer than the farthest in the short-list are considered.

The number of exact distance computations in this algorithm depends on how balanced the tree is and the position of the query in the tree. Because the algorithm guarantees to find k items given that there are at least k items available, the number of exact distance computations cannot be reduced up to zero. A minimum number of distance computations is required that

is always larger than k . A metric cannot be integrated in the algorithm itself, but a combination can be made by first transforming the features before applying the algorithm.

IV. EXPERIMENTS

A. Dataset

The experiments are performed on a dataset of videos from the on-line video sharing site blip.tv. The set consisted of 1967 videos with varying length and resolution; some videos were only a minute long while others were over an hour long. All videos came in the Ogg container format using Theora for video encoding and Vorbis for audio encoding. From the total set we took 50% for training and 50% for testing the systems. For testing particular configurations we drew random subsets from the test set. All videos came with genre labels, speech transcripts and meta data. The speech transcripts were extracted using a state of the art speech recognizer.



Fig. 3. Examples of extracted video frames

B. Features

Feature extraction was done using a couple of open source tools and libraries. The visual features were extracted using functions from the OpenCV libraries for C++ in combination with FFmpeg for extraction from the container format. And the audio stream was extracted and converted to the right format with FFmpeg.

To extract the visual features from a video we divided them into segments that have similar visual characteristics. The boundaries of these segments were detected where there was a low histogram correlation between the first and last frame

within a sliding window, which is a simple but effective way to detect segment boundaries [11]. We used the middle frame for feature extraction, but another option would be to use multiple frames and combine the extracted features.

1) *Region descriptors*: To describe local structures in the extracted frames we used 128 dimensional Speeded Up Robust Features (SURF) [3]. We used SURF features instead of Scale-Invariant Feature Transforms (SIFT) because it is significantly faster and can be done on the GPU.

Because of the dimensionality and the variation it is not possible to use SURF features directly or even by calculating a histogram from them. The common approach is to use the Bag-of-Words model so that the actual features are a histogram over clusters of SURF features. A Bag-of-Words model consists of a codebook that defines which SURF features are assigned to what cluster.

A lot of variations are possible in the Bag-of-Words approach with SURF features, such as different clustering algorithms, the number of clusters, soft versus hard cluster assignment, multiple scales or integration of color. We used a variant on mean-shift clustering for non-uniform clustering of a set of training features into 2000 clusters [29]. The features were only extracted from the grayscale layer and the codebook used soft cluster assignment.

2) *Colour histogram*: To complement the SURF features with a representation of the colors in the frames, we used a histogram of the hue values from the HSV color space with 180 bins. We chose to use only the hue values because they describe colors and are expected to be more discriminative and relevant to measure similarity. While the saturation and value layers of the HSV color model are less discriminative and might vary depending on post processing choices.

3) *Topics*: Although the given speech transcripts were of good quality, we chose another speech recognizer to extract slightly better speech transcripts in a more suitable format. The speech was extracted using the open source speech recognizer Sphinx4 [8] with the Gigaword trigram language model [30] using a 20k dictionary and the HUB4 acoustic model [7].

We used a topic model to determine the most prominent topics in the spoken content. A topic model describes the relations between words and a set of topics. Latent Dirichlet Allocation (LDA)[5] was used to build the topic model using approximately 3 million English Wikipedia articles for training [4]. The actual features extracted using this model from the videos are the histogram values of the topics. We used a model with 250 topics. Using fewer topics would reduce the performance while increasing the number of topics would not improve performance [31].

C. Baseline systems

For both aspects of our problem statement, fusion of the features and efficient nearest neighbour search, we need good baseline systems to compare them with. The most important baseline for the fusion of features with a metric is retrieval with just the concatenation of features in an exhaustive search. This way we show the effect of using a learned metric for feature fusion. The efficient nearest neighbour search algorithms are

tested with and without metric, and compared to the corresponding exhaustive search systems.

As an absolute baseline we did retrieval with random results using only the class priors. This is the absolute minimum that a system without any features will have. We compared this with the retrieval systems using the individual feature sets to show what can be done without any kind of fusion.

Besides just concatenating the features and early fusion with a learned metric, we also tested a system with a late fusion scheme. Late fusion is done by first converting the distances D_x obtained with the individual feature sets to probabilities of being similar using Bayes rule [33]; where x identifies the feature set:

$$P(S|D_x) = \frac{P(D_x|S)P(S)}{P(D_x)} \quad (12)$$

$P(S)$ can be taken out because it is independent with respect to the distances. The individual probabilities are then fused by multiplying them to obtain one score. $P(D_x|S)$ is estimated using the class depend distances and $P(D_x)$ is estimated using all distances.

D. Measures

For practical systems the precisions at k , where k is often set at 10, is a very useful measure because it takes only the results into account that are shown on the first page of results in a practical application. Recall is important because although a typical application can only show a certain number of results on the first page, we still want to retrieve as many of the relevant items as possible.

$$\begin{aligned} \text{Precision at 10} &= \frac{\# \text{Relevant items of first 10 retrieved}}{10} \\ \text{Recall} &= \frac{\# \text{Relevant items retrieved}}{\# \text{Total relevant items}} \end{aligned}$$

We combined the precision at 10 and the recall by taking their harmonic mean, or F_1 -measure, such that we can set it out against the number of exact distance computations. We use the number of exact distance computations as a measure of the how efficient the retrieval algorithms are.

V. RESULTS

We have analysed the two systems by comparing their performance to the baselines as mentioned in the previous section. For all systems that use an exhaustive search the number of exact distance computations is always the maximum. Therefore it is not useful to set them out against the number of distance computations unless they are compared to the efficient retrieval algorithms. Because all items are retrieved the recall will always be 1. The absolute baseline, random retrieval using the genre priors, has a harmonic mean of precision at 10 and recall of 0,220.

A. Early fusion with metric learning

Below in table I we report the performance of several systems that use an exhaustive search. We have separate system for the individual feature sets that show how well suited the features are for this task. Then we concatenate the features into one feature vector. For all these four systems we add a learned metric to evaluate the improvement due to the metric. For both cases, with and without metric, we reported the performance with the harmonic mean of precision at 10 and recall. In the last column we report the relative improvement of the system due to the metric.

The use of metric learning in the exhaustive search setting for transformation of the concatenated features, as well as for transformation of the individual feature sets improves the retrieval performance significantly. The improvement due to a learned metric on the concatenation of all features results in the largest relative improvement. But it is also the best system in absolute performance.

System	Baseline	With metric	Improvement
Topics	0.552	0.627	13.6%
Colour histogram	0.466	0.484	3.9%
SURF features	0.380	0.502	32.1%
Concatenated	0.456	0.670	46.9%

TABLE I
PERFORMANCE WITH AND WITHOUT METRIC

Not every feature set can benefit equally from a learned metric. The color histograms improve with only 3,9% while the SURF features improve 32,1%. This indicates that bins of the color histogram are equally discriminative while the clusters of SURF features are not equally discriminative. The topic features are by themselves the most discriminative, which is to be expected because they operate at the semantic level.

In the table of results we see that just concatenating the features has a performance that is a weighted average of the individual features. It cannot improve upon the best individual feature set, but is not worse than any of them either. By applying the metric on the set of concatenated features we get a performance that is better than the best individual feature set.

The metric learner we used has one disadvantage that could be a problem in some cases. The learning stage scales quadratic with the number of feature dimensions. We had 2506 feature dimensions in the system with concatenated features, but for practical systems with more different feature sets the number of dimensions could be up to 10 times more. In that case a different metric learner that scales linear with the feature dimensions is necessary.

B. Late fusion of retrieval results

The system with late fusion of the retrieval results of the individual feature sets performs not as good as the systems with concatenated features as can be seen in table II. Both with and without a metric it performs less than the concatenation of features. However, using learned metrics does improve the performance.

System	Baseline	With metric	Improvement
Late fusion	0.448	0.537	19.9%

TABLE II
PERFORMANCE WITH LATE FUSION

The advantage of late fusion for metric learning is the lower feature space dimensionality in the learning stage. This means that learning the metrics takes less time, but also that not all relations between features can be exploited by the metric learner. Without a metric there is no use for late fusion because it does not improve upon the concatenation of features.

Both with and without a metric the performance of the systems with late fusion is a weighted average of the results of the individual feature sets. Especially in the case with the metric the difference with the concatenation of features becomes clear, because that system can improve upon the individual features sets and late fusion not.

We have to note that we only tested one option for late fusion. In [33] it is shown that supervised late fusion of retrieval results can outperform unsupervised static fusion rules. They give several supervised late fusion schemes of which we only tested one. However, they also note that none of the fusion schemes, both supervised and unsupervised, can consistently outperform the individual retrieval results.

C. Efficient retrieval

Two options for efficient retrieval were tested: Locality Sensitive Hashing and Vantage Point trees. We compared them with and without metric. The number of exact distance computations of efficient nearest neighbour algorithms is not a fixed value, but can be varied indirectly by adjusting the parameters of the algorithms. For both algorithms we tested several configurations with different parameter settings. All features were used for these systems. The small variation between different configuration with almost the same number of exact distance computations is due to random selection of test queries.

In figure 4 below we see random retrieval as baseline (Large black dot), the exhaustive search without metric (Large red dot) and exhaustive search with metric (Large green dot). The small dots are the different configurations of efficient nearest neighbour algorithms, where the light red and blue dots are LSH and the darker dots the VP-trees.

The VP-trees algorithm reduces the number of exact distance computations by searching only through part of the tree structure that has potential neighbours. But because the structure can only be searched in a top down fashion starting from the root node and not starting at the query position, there is always a minimum number of exact distance computations required. LSH can reduce the number of distance computations even further because it starts the search at the query position.

Although there are differences between the two tested efficient nearest neighbour algorithms, both are useful for retrieval purposes. They both reduce the number of distance computations significantly without reducing the performance much. The effect holds both for the case without a metric as well as with a metric. And the behaviour of both algorithms

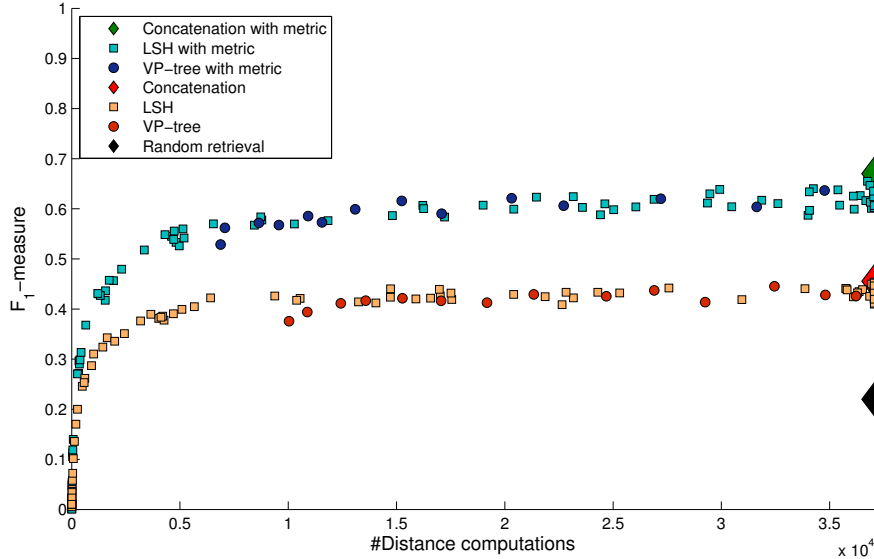


Fig. 4. Performance of LSH and VP-trees with different configurations

does not change due to the combination with a metric.

Precision at 10 and recall against the number of distance computations gives even more insight in the effects of the efficient nearest neighbour algorithms than the harmonic mean of them. The precision at 10 does not decrease almost until there are not more than 10 items returned, which is expected because only 10 items are needed if the right items are retrieved. But the recall drops slowly while reducing the number of distance computations, which is to be expected because to keep recall high most instances of the given class need to be retrieved.

The performance of both algorithms is similar in terms of the number of distance computations. But there are differences between the algorithms that are relevant when they are used in a practical application. LSH requires more memory because it has to store the hash functions for every table, while there is almost no overhead over the items themselves for VP-trees. Another problem with LSH is the complex parameter configuration. 2 or 3 parameter that depend on each other need to be set, while the VP-trees algorithm in its basic form has only one parameter, which is actually not a parameter of the structure but only used for a search through the structure.

For both algorithms it is possible to update the structure for insertion or deletion of individual items. In a LSH structure the item can simply be added or removed from a bucket, though this also needs to be done for the second level hash if it is the only item in the bucket. Tree structures are in general easy to change by removing an item and connecting the parent and the child, or by adding an item between an existing pair of parent and child.

VI. CONCLUSION

In this paper, we presented a system for video retrieval with content-based features. The videos we want to retrieve are those that share the same genre label as the query video. We

focus on two aspects of such a system, the fusion of features and an efficient retrieval algorithm.

We showed that using learned metrics for the fusion of the features from the individual feature sets results in a significant improvement over simple feature vector concatenation. Both in the early fusion setting as well as in a late fusion setting the metric gave an improvement in terms of the harmonic mean of precision at 10 and recall. But more improvement was possible in the early fusion setting than in the late fusion setting with 46,9% and 19,9% respectively. With early fusion the performance of the whole system is above that of the individual features, while late fusion cannot improve upon the best individual feature set.

However, learning a metric has a time complexity that is quadratic in the number of feature dimensions. For a reasonable number of feature dimensions (1-5000) this is not a problem on modern computers, but in the domain of multimedia retrieval the number of features can be even higher. In that case late fusion might become preferable depending on the hardware available. Or a different metric learner that learns in linear or close to linear time with respect to the dimensionality.

The use of a more efficient nearest neighbour search algorithm gives an improvement for a multimedia retrieval system compared to an exhaustive search in terms of the number of exact distance computations. We tested two options for an efficient nearest neighbour algorithm, Vantage Point trees and Locality Sensitive Hashing. Both reduce the number of exact distance computations per query significantly, but the way they do this is completely different. However the reduction in performance in terms of less distance computations is the same for both techniques.

A. Further research

One direction that can be researched further is the combination with unsupervised dimension reduction methods. In the proposed system we transform the feature space before or in combination with efficient retrieval. But initial tests suggest that the dimensionality of the feature space could be reduced significantly without reducing performance. This depends of course on the chosen features. Besides the standard unsupervised dimension reduction methods available, a different metric learning algorithm that simultaneously reduces the number of dimensions might also be an option.

Another direction for further research could be the effect of larger databases. Here we used a relatively small database compared to real systems. It is worth to investigate the behaviour of the efficient nearest neighbour algorithms with different amounts of videos. It might also be more difficult to find the right parameter configuration when it takes more time to test different settings.

For practical applications of this approach the aspect that needs attention is the choice of features. Here we focused on the fusion of features and efficient retrieval, but for practical applications there exist better features. A lot of research is done on semantic concept detectors for images and videos. These kinds of features might perform better because they close the semantic gap [31].

REFERENCES

- [1] A. Andoni. *Nearest neighbor search: the old, the new, and the impossible*. PhD thesis, 2009.
- [2] A. Andoni and P. Indyk. 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. IEEE, 2006.
- [3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.
- [4] David M. Blei. Implementations for topic modeling.
- [5] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.
- [6] S.S. Cheung and A. Zakhor. Efficient video similarity measurement with video signature. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(1):59–74, 2003.
- [7] CMU. Cmu sphinx at sourceforge.
- [8] CMU. Sphinx-4.
- [9] J.V. Davis, B. Kulis, P. Jain, S. Sra, and I.S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.
- [10] H.K. Ekenel and T. Semela. Multimodal genre classification of tv programs and youtube videos. *Multimedia Tools and Applications*, pages 1–21, 2011.
- [11] Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer. Performance characterization of video-shot-change detection methods. *Circuits and Systems for Video Technology, IEEE Transactions on*, 10(1):1–13, 2000.
- [12] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. 2004.
- [13] ALAN Hanjalic, RAINER Lienhart, W-Y Ma, and JOHN R Smith. The holy grail of multimedia information retrieval: So close or yet so far away? *Proceedings of the IEEE*, 96(4):541–547, 2008.
- [14] Alexander Hauptmann, Rong Jin, and Tobun D Ng. Video retrieval using speech and image information. SPIE.
- [15] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.
- [16] P. Jain, B. Kulis, I.S. Dhillon, and K. Grauman. Online metric learning and fast similarity search. *Advances in neural information processing systems*, 22, 2008.
- [17] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- [18] L. Kennedy, S.F. Chang, and A. Natsev. Query-adaptive fusion for multimodal search. *Proceedings of the IEEE*, 96(4):567–588, 2008.
- [19] J. Kludas, E. Bruno, and S. Marchand-Maillet. Information fusion in multimedia information retrieval. *Adaptive Multimedial Retrieval: Retrieval, User, and Semantics*, pages 147–159, 2008.
- [20] Neeraj Kumar. What is a good nearest neighbors algorithm for finding similar patches in images?
- [21] Z. Lan, L. Bao, S.I. Yu, W. Liu, and A. Hauptmann. Double fusion for multimedia event detection. *Advances in Multimedia Modeling*, pages 173–185, 2012.
- [22] Hong Lu, Beng Chin Ooi, Heng Tao Shen, and Xiangyang Xue. Hierarchical indexing structure for efficient similarity search in video retrieval. *Knowledge and Data Engineering, IEEE Transactions on*, 18(11):1544–1559, 2006.
- [23] B. McFee and G. Lanckriet. Learning multi-modal similarity. *The Journal of Machine Learning Research*, 12:491–523, 2011.
- [24] S. Shalev-Shwartz, Y. Singer, and A.Y. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the twenty-first international conference on Machine learning*, page 94. ACM, 2008.
- [25] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. *Computer Vision/ECCV 2002*, pages 181–185, 2006.
- [26] A.F. Smeaton. Techniques used and open challenges to the analysis, indexing and retrieval of digital video. *Information Systems*, 32(4):545–559, 2007.
- [27] C.G.M. Snoek and M. Worring. Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications*, 25(1):5–35, 2005.
- [28] C.G.M. Snoek, M. Worring, and A.W.M. Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402. ACM, 2001.
- [29] Jan C van Gemert, Cees GM Snoek, Cor J Veenman, Arnold WM Smeulders, and Jan-Mark Geusebroek. Comparing compact codebooks for visual categorization. *Computer vision and image understanding*, 114(4):450–462, 2010.
- [30] Keith Vertanen. English gigaword language model.
- [31] Machiel Visser. Working document. 2013.
- [32] K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *In NIPS*. Citeseer, 2006.
- [33] Peter Willett. Combination of similarity rankings using data fusion. *Journal of Chemical Information and Modeling*, 53(1):1–10, 2013.
- [34] S. Xiang, F. Nie, and C. Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600–3612, 2008.
- [35] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2002.
- [36] R. Yan and A.G. Hauptmann. Probabilistic latent query analysis for combining multiple retrieval sources. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 324–331. ACM, 2006.
- [37] Peter N 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, 1993.