TU Kaiserslautern & DFKI
Image Understanding and Pattern Recognition
Prof. Dr. Thomas Breuel

Diplomarbeit

Topic Models for Content-Based Video Retrieval

Jörn Wanke

Juni 2009

Betreuer:
Adrian Ulges
&
Prof. Dr. Thomas Breuel

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Textauszüge und Grafiken, die sinngemäß oder wörtlich aus veröffentlichten Schriften entnommen wurden, sind durch Referenzen gekennzeichnet.

Kaiserslautern, im Juni 2009

Jörn Wanke

Abstract

Content-based video retrieval (CBVR) tasks such as autoannotation or clustering are based on low-level descriptors of video content, which should be compact in order to optimize storage requirements and efficiency. In this thesis, the semantic compression *of video is addressed, i.e. the reduction of low-level descriptors to a few semantically expressive dimensions. To achieve this,* topic models *have been proposed, which cluster visual content into a low number of latent aspects and have successfully been applied to still images before. This thesis analyzes the use of topic models for semantic compression in a variety of CBVR tasks such as autoannotation, clustering, and shot retrieval.*

In quantitative experiments on web video content, it is demonstrated that topic models outperform other dimensionality reduction techniques such as Principal Component Analysis or Restricted Boltzmann Machines, reaching a performance comparable to higher-dimensional bag-of-visual-words descriptors at a compression rate of $1/20$*. It will be shown that this result can be improved further by using multimodal features, while maintaining the same resulting compression rate. Another approach that will be exploited for substantial performance enhancement is the use of large but noisy data, as opposed to a small, manually selected set of training samples.*

Finally, the temporal structure offered by videos, which has thus far been neglected by topic models, is employed, using a combination with Hidden Markov Models as recently proposed in the text domain as 'Hidden Topic Markov Model' (HTMM). An analogy between sentences in the text and shots in the video is drawn, such that changes of semantics are enforced to coincide with shot transitions. Compared to a plain topic model, this adaption leads to a further performance gain in a clustering scenario.

Contents

Chapter 1

Introduction

Assume a video collection of your family which was build throughout the years. Would it not be great if it was possible to easily find all those video shots that contained your grandfather? Imagine if this could be done by just showing one example picture of your grandfather. There are a lot of these scenarios where content-based video retrieval can benefit humans (see [1, 26, 66]): a journalist might be looking for appropriate footage for an article about a particular soccer player – CBVR can help by sorting all video shots of that player according to their content (such as goals or interviews), like a big catalog in which one page contains all videos of a particular topic. A system that learns what programs an avid TV fan likes could learn to recommend new programs, reducing the risk of watching unknown programs that turn out to be unsatisfying. Copyright violations are also an issue where CBVR is useful, as it can provide a system that helps users find videos that are alike. Automatic annotation of important events within a soccer video such as goals or red cards, would allow someone who missed part of the game to gain a quick summary – just like a personal highlight reel. As a final example, consider looking for a particular video, and being aided automatically during that search. Imagine starting out with a particular soccer team, then focusing on a certain player of that team, before finally yielding all videos of that player in a specific season with just a few mouse clicks.

The demand for such applications is already there, as a wide variety of video collections exist. These range from small home video collections to huge archives of TV stations. The web also contains a wide variety of media: roughly 60,000 new videos are uploaded to YouTube.com per day, and in the image domain, several thousand images per minute are added to Flickr.com. This leads to the question on how to extract the information from within media files and how to make this wealth of information easily accessible to the user. In order to achieve this goal, it is necessary to be able to capture the semantics automatically and describe it with as few bytes as possible. Ideally, envision a video about politicians which has embedded information in the sense that it 'knows' that *'at seconds 43-55, the german chancellor is speaking, followed by the french president'*. Unfortunately, it is not feasible to extract this type of

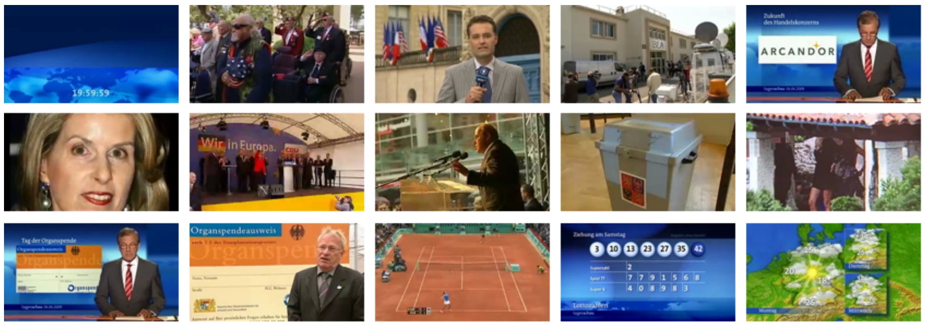

Figure 1.1: An example video of a newscast. Each shot of the newscast belongs to a particular category, like politics, sports, the talking news anchor or the weather. Capturing the semantics of the individual categories is an important step for content-based video retrieval. *Topic models*, the focus of this work, can capture these semantics by associating image parts (like a soccer player, a ball, the green lawn) with specific topics (like soccer).

detailed information manually for videos on a large scale.

CBVR works by extracting interesting properties, so-called features of a video (*'This soccer video has a lot of green areas'*) and then using these features from multiple videos to derive a statistical model (*'Most soccer videos have green areas, thus it is likely that a video with green areas is about soccer'*).

While there are already some automated approaches in the image domain (e.g. face recognition, classification of indoor against outdoor scenes, etc. [4, 14, 21, 73, 76, 77]), these cannot be easily applied to the video domain for several reasons: first of all, video-specific features like motion are not considered. Second, and more importantly, the amount of data is magnitudes higher for videos than it is for images (as an example, a 60-second video with 25 frames per second consists of 1500 images). One strategy to resolve this problem is to employ dimensionality reduction techniques, which reduce the amount of data to describe an image. As the semantics of the video should not change due to dimensionality reduction, this process can be referred to as *semantic compression*. Dimensionality reduction techniques have to fulfill several conditions: they have to be adaptable to video content, scale and generalize well, and be reasonably fast while achieving good compression results.

One solution to achieve this sort of dimensionality reduction are topic models, which will be the main focus of this thesis. Topic models have originally been developed in the text domain and have already been successfully applied to the image domain [3, 21, 58, 63, 73]. The underlying idea is to decompose a document (or image) collection such that each document is described by a mixture of latent associated topics. This approach also makes sense in the video domain: a newscast video might generally consist of topics such as *'sports', 'interview', 'weather'* and so on (c.f. Figure 1.1). The representation of a *video as a topic mixture* can be used for semantic video compression, which is the description of semantics of a video with only a few bytes. A big advantage of topic models is that they work unsupervised, i.e. they estimate what topics exist within videos without additional manual help. Being able to perform unsupervised learning is a key requirement, as manually annotated training data is hard to obtain at a large scale.

This work will investigate the use of topic models for semantic compression in the video domain. In particular, several appearance and motion features will be evaluated. Their dimensionality will be reduced using different techniques, and the tradeoff between a low target dimensionality and a high retrieval performance is explored. The combination of different features to enhance performance will be investigated. Finally, the impact of shot information as well as the temporal structure of shots will be utilized for topic models to further improve performance.

This thesis is structured as follows: in Chapter 2, the general pipeline for content-based video retrieval will be outlined and the underlying concepts of each step will be explained. Chapter 3 will discuss feature extraction and dimensionality reduction techniques in more detail, as they are closely related in the task of producing low-dimensional video descriptors. Chapter 4 will then explain different topic models and how they can be applied to video content. Chapter 5 describes the experiments conducted in this work and concludes that topic models are indeed a good choice for semantic compression. Finally, Chapter 6 will summarize this work and discuss potential further research areas.

Chapter 2

System Overview

Several processing steps (c.f. Figure 2.1) are required in order to achieve the main goal of content-based video retrieval (CBVR): aid the user in searching for videos by analyzing their contents. The first section of this chapter will outline the general steps required for CBVR. The rest of this chapter will then explain the several processing steps in more detail.

2.1 Architecture of Content-Based Video Retrieval Systems

In general, CBVR consists of multiple steps, which will be briefly outlined here. A video will first be preprocessed (e.g. adjusting contrast), followed by the feature extraction step, which will emit a video description. In the case of a soccer video, a discriminative property that might be captured is the fact that a lot of green (the grass) occurs. As these feature descriptors are usually high-dimensional, a dimensionality reduction step may be applied to enable further processing, resulting in a low-dimensional feature vector. Finally, this feature vector can be used as input to statistical analysis. As an example, a soccer video might be used as a query video to find similar videos – in that case, more soccer videos should be returned.

Preprocessing Before actually trying to identify key properties of a video, it can be useful to gain additional meta information about it, which is done in a preprocessing step. An example would be the estimation of occurrences of transitions of one video shot to the next, as the shots defined by these transitions might have varying semantic meanings (e.g. in a newscast, an anchorman might first talk about a war, before some footage about a soccer game is shown). This task is called *shot boundary extraction* and will be detailed in Section 2.2. Another example is the thought that it might be interesting to know which frames are representative for a video. This process called *keyframe extraction* will be discussed in Section 2.3.

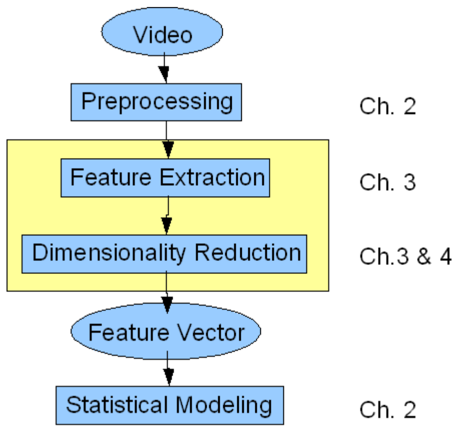

Figure 2.1: A generic pipeline for content-based video retrieval as outlined in Section 2.1. As producing low-dimensional features is the key part of this thesis and feature extraction and dimensionality reduction are both required for this, both subjects will be discussed in Chapter 3. Chapter 4 will then deal with topic models, which are specific models that can be used for dimensionality reduction and are the main focus of this thesis.

Other possibilities of preprocessing include rescaling the videos to have equal size, noise reduction or contrast enhancement.

Feature Extraction In general, a video is a temporally ordered sequence of images where all images share the same height and width. However, this does not imply any information about the similarity of any two given images. While it is possible to compare them on a per-pixel basis, this is a bad solution in practice for a variety of reasons. Some examples include computation time as well as issues with translated, scaled, or rotated images, variance in brightness etc. This makes it necessary to extract features for each frame, which should be low-dimensional (so comparison of features of different frames becomes feasible) and invariant to variations (like different scales) of the image. Properties which have to be fulfilled by features will be discussed in Section 2.4.

Dimensionality Reduction As videos contain a very large amount of frames, the computed resulting features would accordingly require a lot of storage space, which makes further processing very slow or even impossible. While this is not as bad in the case of single still images, it makes it absolulety necessary in the video domain to reduce the amount of data while preserving as much information as possible. One simple way to achieve less data is to only extract features for keyframes, as those are likely to contain the most discriminative information about a video. As less data also implies less information, the goal of dimensionality reduction is to minimize information loss while maximizing the ’compression’.

The tasks of feature extraction and dimensionality reduction are highly intertwined – extracted features can have varying dimensions, and different dimensionality reduction methods can be employed depending on the properties of features. As the goal of this thesis is to enable CBVR, and this absolutely requires low-dimensional feature vectors, feature extraction and dimensionality reduction are the key parts of this thesis. Thus, these topics will be discussed together in detail in Chapter 3. Topic models, which are a specific technique that can be used for dimensionality reduction and which are the main focus of this thesis, will be explained in Chapter 4.

Statistical Modeling Given the extracted, low-dimensional features for all videos, it is now possible to apply pattern recognition methods to allow CBVR. The actual techniques that are employed highly depend on the task that should be achieved. Three possible applications will be briefly outlined here.

- Given a set of tuples (videos, category label) that serve as training examples, the system learns to associate previously unseen videos with a category label. This task is called *classification*. An example for this lies in the protection of minors from sexually offensive material: given some sexually offensive videos and some counter-examples (each being marked as either offensive or non-offensive), the system needs to derive a model from analyzing these videos. Using this model, the system then should be able to decide for new, previously unseen videos if this new video is sexually offensive.
- The task called *clustering* aims at identifying groups of videos. One example application where clustering is necessary is browsing: if the system is able to arrange all videos in the dataset according to the property desired by the user, this helps him to quickly access those videos he is looking for. As an example, given a big archive of soccer games, a user might be looking for all those games with a particular player – if the system is able to cluster all videos according to the players contained within, this will greatly ease the search process for the user. Note that clustering for browsing can also be useful on a shot level: here, particular game events like goals, free kicks, red cards etc. might be desirable clusters.
- *K-similarity search* tries to identify the k videos which are most similar to a given query video and return them. Note that 'being similar' can have several semantic meanings: given a video of a golf course, 'good' matches might either be about golf tournaments or about green grass. Ideally, handing a video of a soccer player scoring a goal would return videos with more goals scored by the same player. Although it is possible to compare the query video with each video in the database, this approach would be highly inefficient as it requires linear time. Instead, faster data structures are desired.

These exemplary tasks and one method to solve each problem will be explained in more detail in Section 2.5.

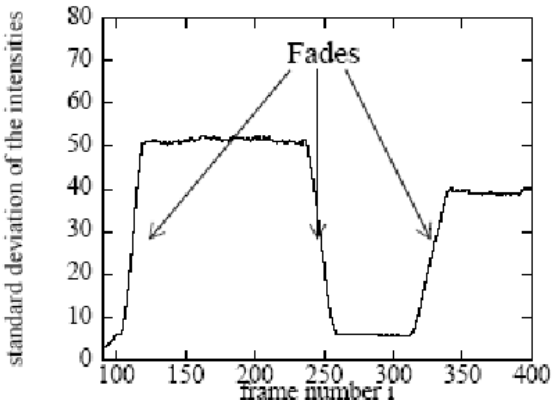

Figure 2.2: Standard deviation of pixel intensities. As a rapid change occurs during fades, the standard deviation can be used for fade detection. The picture was taken from [42], p. 7.

2.2 Shot Boundary Detection

Shots are the smallest semantic units within a video and are comprised of an ordered set of frames. Two shots are separated by a transition, like a fade-over or simply a hard cut. As the goal of this thesis is to gain semantic information about a video, information about shots might be interesting, either as information that can be directly conveyed to the user (e.g. in a browsing task) or intermediate results *('there are 6 shots in this video and 3 of these shots are very similar, so there are likely different 4 concepts in the video').* This is were the task of shot boundary detection comes in, which is still a heavily researched topic. Some basic ideas for detecting the most frequently used transition types as outlined in [42] are discussed.

Hard Cut Detection Hard cuts are two directly concatenated shots without any sort of transition in-between. As this is a very rapid change assuming appropriate features (e.g. color distributions), they are rather easy to identify. One example for such features would be simple color histograms, which are usually sufficient for a detection of hard cuts according to Lienhart [43].

Fade Detection Fades are gradual changes to/from a scene from/to a monotone (e.g. completely black) image. If the scene is disappearing, this is called a fade out, otherwise the transition is called a fade in. One way to detect this is by analyzing the standard deviation of the pixels colors (c.f. Figure 2.2): Given a normal image, the standard deviation should be fairly high, while in a very monotonic image the pixel deviation will be very low. Thus, a fade can be detected whenever there is a rapid change between temporally close pixel standard deviations of frames.

Another possible approach is edge detection - in a fade-in, edges will slowly build up as more details of the image become available. The same is true for fade-outs, where edges will gradually disappear. Thus, quick changes of the amount of quick edges can be an indicator for fades. However, it is reported that approaches based on the standard deviation outperform edge-based fade detection in practice [42].

Dissolve Detection A dissolve is defined as blending over from one shot to another. Thus, while the first shot slowly fades out, the second one is fading in - this is why a dissolve is also called a cross-fade. Previously discussed detectors (color histograms, edge-based) can detect dissolves in several scenarios:

- If two shots have different color distributions, it is possible for a hard cut detector to identify this, as the distributions will quickly change during overblending (assuming the blendover is not very slow).
- Assuming two shots have similar color distributions, they can still be identified given object shapes between frames. In this case, edge-based detectors will detect rapid changes.
- There are also cases where there might be spatial as well as color similarities. However, this is often argued to just be a technical, but not a semantic transition and thus discarded. An example for this would be the moon turning into the face of a person.

More refined variants of these algorithms and various other approaches towards shot boundary detection are discussed by Lienhart [42, 43].

2.3 Keyframe Extraction

One merit of keyframe extraction is to only process keyframes instead of all frames, while not losing too much discriminative information. On a shot level, it has been shown that using keyframes instead of either regularly sampled frames or the first frame of a shot improves performance [12].

The approach to extract keyframes described in [12] is pretty simple: Assuming that shot boundaries have already been estimated, all frames within a shot are clustered (using e.g. k-Means, c.f. Section 2.5.2). The frame that is closest to each cluster center is then treated as a keyframe.

Since keyframes are extracted within a shot, a possible problem is that they might repeat themselves in different shots. A simple example might be an interview, in which each interview partner is shown in turn – as these turns are treated as different shots, there will be multiple keyframes of the same interview partners. One way to solve this is to cluster all extracted keyframes on a video level.

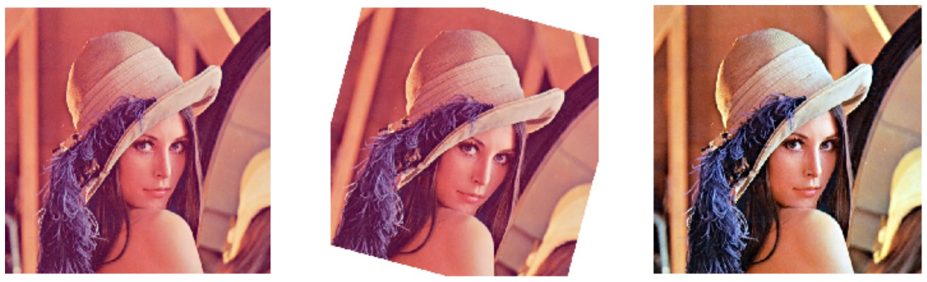

Figure 2.3: *Left:* An example image. *Middle:* A rotated version. *Right:* A differently illuminated variant. As all three images share the same semantics, a feature descriptor should not be affected by the variations.

2.4 Feature Extraction & Dimensionality Reduction

While feature extraction is the process of extracting useful features for a video (or image), dimensionality reduction aims to create low-dimensional representations of these feature descriptors.

There are a variety of properties a feature descriptor has to fulfill to be of use. Some of these properties include (c.f. Figure 2.3):

- **Rotation, Scaling, Translation Invariance** Descriptors should be invariant to rotation, scaling and translation of an image. As an example, an image which has been rotated by 15 degrees is likely to still express the same semantics as the original image and correspondingly, the feature should not be changed by the transformation.

- **Illumination Changes** Assuming a picture from an object has been taken under varying lightning conditions or with a different camera, it is likely that brightness, contrast etc. differ between images. Still, it is desirable to capture the similarity, which requires invariance to change in lightning conditions.

- **Perspective Change** Pictures that are taken from different angles should still be considered similar as they share semantics.

While these invariances are required to create descriptive features, 'soft' requirements like computational speed and storage requirements are also of particular importance. Given that a single video can consist of hundreds of thousands of frames, feature extraction must be very fast – if the user has to wait for several minutes for his retrieval results in a search application, productivity is severely affected. Similarly, the description of these hundreds of thousands of frames have to be stored and processed – thus, feature descriptors can not be high-dimensional. While the choice of proper feature descriptors is one way to achieve these requirements, it is useful to also employ dimensionality reduction techniques. These techniques reduce high-dimensional data into fewer dimensions, while maintaining as much of the original information as possible.

As feature extraction and dimensionality reduction are highly intertwined and in the focus of this thesis, they will be explained in Chapter 3. Topic models, which are a specific technique for dimensionality reduction and the key part of this thesis, will be thoroughly discussed in Chapter 4.

2.5 Statistical Modeling

To achieve the goals of CBVR, the extracted features are normally used to derive a statistical model. For example, to find videos that are similar to a query video, the system needs to build a model containing information about all videos in the database. This section will introduce some possible target applications of statistical modeling and present one way to solve each problem in more detail.

2.5.1 Classification

One scenario in supervised machine learning is the task of classification. Given a training set with associated category labels, the system is supposed to derive a model which will be able to infer labels for new data.

A typical example are spam filters – in a training step, the system is shown some emails, and each of the emails is either marked as 'spam' or 'no spam'. After learning to separate these emails, the system should predict for new, unseen emails whether or not they are spam.

A state-of-the-art method to tackle the problem of classification is called Support Vector Machine (SVM), which will be explained in the following paragraph.

Support Vector Machines When formalizing the binary classification task with features x_i and associated labels $y_i \pm 1$, a linear discriminative function can be defined as

$$f(x) = sign(w * x + b) \tag{2.1}$$

The classification will then be correct if

$$\forall i[y_i(w * x_i + b) > 0]$$

Accordingly, Support Vector Machines separate the input space with a hyperplane, so that geometrically, all data entries from one category are on one side of the hyperplane and the data from the other category on the other side. A SVM constructs the hyperplane that maximizes the distance to the closest data points of the two datasets, and is thus called a *maximum-margin classifier.*

Hyperplanes are linear functions, which can be a problem if the dataset is not linearly dividable. This problem can be circumvented by employing a kernel

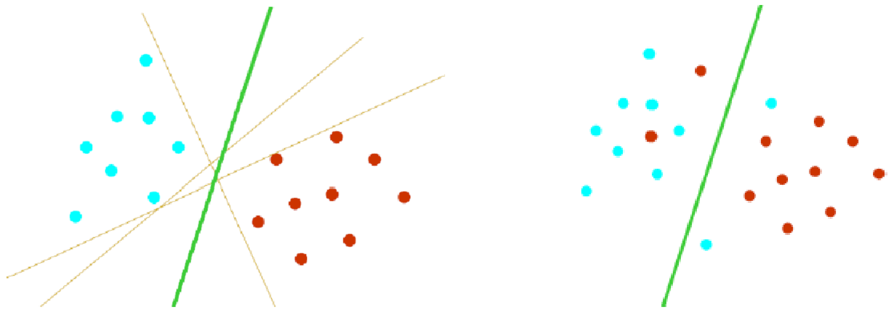

Figure 2.4: Features of two classes in a 2-dimensional space. *Left:* the brown lines represent possible SVM hyperplanes, while the green line is the optimal hyperplane given a max-margin criterion. *Right:* the categories are not linearly dividable. The green line represents the optimal hyperplane if no kernel function is used. The figures were taken from [25], p. 5 & 11.

function, which projects the input into a higher-dimensional space. An example of such a Kernel function is the Radial Base Function or RBF-Kernel:

$$k(x, y) = exp(-\frac{||x - y||^2}{2\sigma^2})$$

In this case, it will become possible to divide the resulting higher-dimensional space by a linear function (c.f. Equation 2.1).

It has to be noted that training with Support Vector Machines implies a certain instability in the results, depending on the distribution of training and test sets. One way to counter this instability is called cross validation: here, the training set is split up into an actual training and a validation set – this is repeated several times, before the best trained model (averaged over all validation sets) is taken. While crossvalidation can improve results, it has the drawback of increased computation time.

2.5.2 Clustering

In the context of machine learning, clustering is the task of grouping items according to some criterion (e.g. a similarity measure). This is often done by finding group centers (called cluster centers) and then associating each item to its nearest cluster center. A key difference to classification is that no label information is available to help generate a good model, which is why clustering is also called unsupervised learning.

Considering a mixed set of soccer and swimming videos, a clustering algorithm should split the data in two groups (one for each respective category), although no class labels for the videos are available. This is in contrast to classification, where videos within one region of the feature space would be considered as soccer, while another region would be considered as swimming.

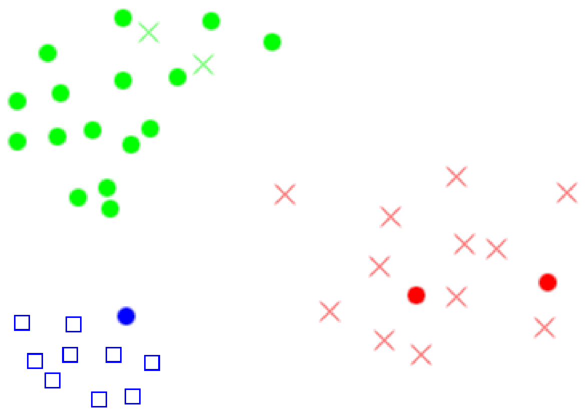

Figure 2.5: A simple clustering example with 3 categories (cross, square, circle) and 3 clusters (red, green, blue). While the clustering performance is good in general, some misclassifications occur, like the blue circle being associated with the squares.

A difficult problem in the context of clustering is the choice of the number of clusters – this decision can either be left to the user (if he wants to cluster all images of members of his family, he can easily specify how many family members exist) or be estimated using cluster validity measures such as the Bayes Information Criterion [61].

A simple algorithm that yields a local optimum as a clustering result is k-Means, which will now be explained in more detail.

K-Means K-Means is an algorithm introduced by Lloyd [45] in 1982 that solves the problem of finding cluster centers (or means) $m_1, .., m_k$, so that the overall distance (e.g. L^2) of all points $p_1, .., p_n$ to their closest mean m_j is minimal ($NN(j)$ finds the cluster center of j):

$$\underset{m_1,..,m_k}{arg} \quad min \sum_{j}^{P} \|p_j - m_{NN(j)}\| \tag{2.2}$$

K-Means consists of several steps:

1. Pick k cluster centers (e.g. randomly)
2. Map each point p_j to its closest cluster m_i (resulting in cluster sets $C_1, .., C_k$)
3. For each cluster set C_i calculate the mean and choose it as new m_i

4. Go to step (2) as long as at least one p_j changes its cluster in step (2)

There are several problems with this algorithm. First of all, it will only find a local maximum instead of a global optimum of Equation 2.2. Due to this, it is common practice to restart k-Means several times with different initial clusters and take the best overall result. Another problem lies in the unknown number of required iterations until k-Means converges to a solution. These reasons make it infeasible to predict the required computation time, although k-Means performs quite fast in practice.

While the presented basic k-Means algorithm performs well, several modifications exist. An example for this is the so-called k-Means++ [2], which attempts to pick the initial cluster centers (step 1) in a more intelligent matter: after the first center is randomly chosen, the distance of each point to the center is calculated and proportionally used as probability to choose this point as next cluster center, encouraging initial cluster centers to be spread out over the dataset. Then, distances to their next cluster are recalculated for all points and the process iterated k times. This is supposed to help avoid local maxima and lead to better minimum errors, but takes additional time for the selection of initial clusters. Another approach named MPI-k-Means [19] uses triangle inequalities to speed up the individual iterations, resulting in lower computation times.

2.5.3 Similarity Search

Similarity search is the process of finding those videos that are similar to a given query item. In practice, it depends on the application what items are considered similar – as examples, they might have a similar location in feature space, they might belong to the same object, or they might be modified versions of the same video. Ideally, when queried with a video of a soccer team, the system would return more videos of that soccer team as a result. While it is possible to compare the query video with each other video within the data set and return those with minimum distance, this would have a complexity in $O(n)$ and hence be slow with bigger corpuses. Thus, one problem of similarity search is finding a good data structure which minimizes comparisons and accordingly, computational speed.

One frequently used data structure to minimize the required amount of comparisons are kd-trees, which will be explained in more detail in the next paragraph.

Kd-Trees Kd-trees [8] represent a k-dimensional hyperrectangle that includes the dataset, which is iteratively subdivided in two subparts. Searching for the nearest neighbor within the hyperrectangle is then simply a matter of comparing the query point with the splits (similar to a binary tree) and thus an operation in $O(log(n))$. It is important to note that the point lying in the subspace of the query point does not necessarily have the smallest distance. To ascertain this, a hypersphere with the minimum found distance is placed around the search node.

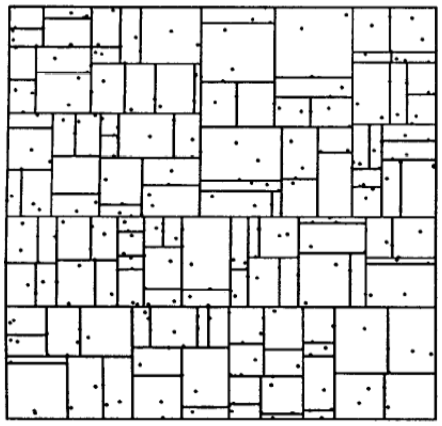

Figure 2.6: An example of an 2-dimensional Kd-tree with 300 points. The figure was taken from [9], p. 2.

All subspaces that are intersected by this hypersphere have to be evaluated to make sure no point in any of the other subparts has a smaller distance. As a lot of subspaces will be intersected by the hypersphere in high-dimensional spaces, kd-trees are not very efficient in those cases. This requirement can be relaxed if an approximate solution is sufficient. For example, Paredes et al. [56] employ a parameter ϵ, so that point p is an $1 + \epsilon$-neighbor of q if (with $NN(p)$ being the nearest neighbor of p):

$$dist(p, q) < (1 + \epsilon) * dist(p, NN(p))$$

Such approximations can significantly reduce necessary computation time if exact solutions are not necessary.

Chapter 3

Feature Extraction & Dimensionality Reduction

Feature extraction and dimensionality reduction are highly related to each other, as the combined goal of the two processing steps is to generate a compact representation of the content of an image. Dimensionality reduction can be unnecessary if extracted features are designed to represent an image in a very low-dimensional vector.

It is important to note that videos might need to be described by a lot of features, of which in turn each needs a lot of information to be represented. Thus, if either the number of features or the number of dimensions of feature is too high, dimensionality reduction can be helpful.

The first part of this chapter deals with feature extraction and discusses texture, shape, color and motion as different basic modalities to achieve good descriptors. Patch-based image description and feature bags as advanced concepts are then explained. Specific algorithms for image descriptors named Color Layout Descriptors, Motion Window Histograms, Moments and Speeded Up Robust Features are also explained.

The second part of this chapter will introduce various dimensionality reduction techniques. In particular, the aggregation of feature histograms as well as Principal Component Analysis and Restricted Boltzmann Machines are introduced. A specific dimensionality reduction technique called topic models will then be thoroughly explained in Chapter 4, as they are the main focus of this thesis.

3.1 Feature Extraction

Features have to describe images/videos with as few dimensions as possible, while still preserving properties of interest. Different modalities to satisfy these requirements will be explained in this section.

3.1.1 Texture

According to Tamura et al. [70], textures can be described by six basic features, namely *coarseness, contrast, directionality, line-likeness, regularity* and *roughness.* As the first three are highly correlated with the latter, only the former will be explained:

- **Coarseness** Coarseness describes the scale at which elements within the texture are repeated. One way to calculate Coarseness is to compute intensity averages around each pixel for different window sizes. As it is intended to capture coarseness in both horizontal and vertical directions, the averages on opposite sides of the pixel are then subtracted from each other for the various scales. The idea is that fine texture is very 'noisy' and thus the averages will be alike across the image, resulting in low average differences. On the other hand, the average of coarse textures should vary more across the image. Thus, the value of the scale with the largest variation is chosen for each pixel and this coarseness indicator then summed and normalized over all pixels.

- **Contrast** Contrast is an indicator for the variation of pixel intensities and can also be calculated using the variance of a pixel intensity histogram: if half of the pixels in an image are white and the other half are black, the variance over all pixels (and thus the contrast) will be very high. On the contrary, if half the pixels are a light gray and the other half a dark gray, the variance over all pixels will be smaller. The variance is divided by the kurtosis of the histogram, which is an indicator for the 'peakedness' of the image, as more peaks also imply more contrast (e.g. a gaussian distribution will have less contrast than a histogram with two peaks).

- **Directionality** Directionality expresses the orientation of elements within a texture. It can be computed by calculating the orientation and strength of local edges (using e.g. Canny/Sobel [13]). A histogram over all edge orientations will have a relatively uniform distribution if there is no particular direction, but will have a large peak in directed images.

For more information on texture descriptors, see e.g. Manjunath et al. [49].

3.1.2 Shape

In general, descriptors that try to catch similar shapes of objects can be classified in two categories (c.f. Figure 3.1). Region-based descriptors consider the spatial distribution of pixels within an image, while contour-based descriptors are concerned with the outline of image regions.

Accordingly, either descriptor type might be more suitable depending on the task: Region-based descriptors are more robust to noise and can capture properties of disjunct regions, such as the letter 'i', which would be treated as a single shape - on the opposite, a contour-based descriptor would describe the base and the dot of the letter as two separate objects.

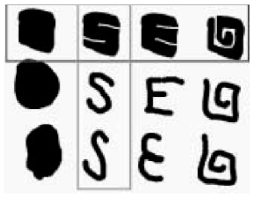

Figure 3.1: Shapes which are similar based on regions (rows) and contours (columns). The picture was taken from [11], p.1.

Some basic properties for shape descriptors are (for a more detailed survey of shape analysis, c.f. Lonaric [46]):

- **Area** The number of pixels within the object.
- **Axes of Orientation** The major axis is the line that connects the two points within the object that are the most far away from each other. The minor axis is perpendicular to the major axis.
- **Perimeter** The number of pixels that outline the object (i.e. the boundary).
- **Compactness** Indicates the ratio over which the object spreads out, defined as $\frac{perimeter^2}{area}$.
- **Moments** Weighted averages over pixel intensities and (for grayscale images) defined as:

  $$M_{ij} = \sum_{x,y} x^i y^j p(x, y)$$

  Two examples are M_{00}, which is the sum of all pixel values within the image and $(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}})$ as the centroids $(\overline{x}, \overline{y})$. A set of 7 moments that are invariant to translation, rotation and scaling were proposed in 1962 by Hu [36].

3.1.3 Color

Color histograms are a simple and frequently used image descriptor. The color space (e.g. RGB) is divided into several bins. Assuming 3 bins per color channel, this would result in a total of $3 * 3 * 3 = 27$ bins. Then, a loop over each pixel within the image finds the corresponding bin per pixel and increases its counter by one. This histogram can either be treated as a descriptor or processed further (e.g. to find the dominant color of an image).

While color histograms are very simple and fast to calculate, they are still pretty high-dimensional even with a very coarse quantization. Also, color similarity across bins can not be captured – this means that very similar colors might still end up in completely different bins.

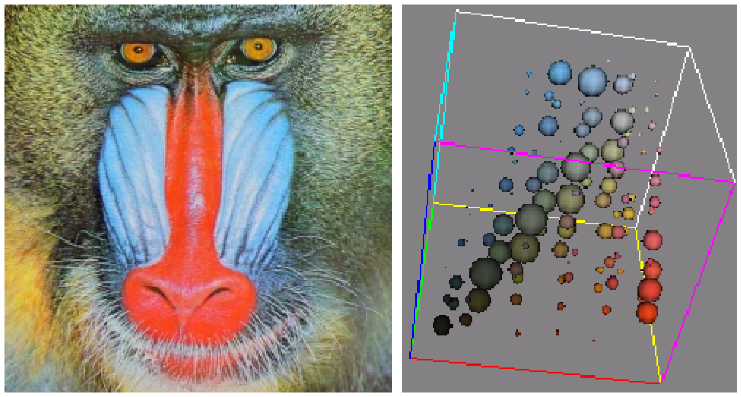

Figure 3.2: A picture and its corresponding color histogram. The visualization was generated using Color Inspector 3D, available at http://rsb.info.nih.gov/ij/plugins/color-inspector.html (05/05/09).

3.1.3.1 Color Layout Descriptor

Color Layout Descriptors (CLD) were introduced in 2001 as part of MPEG-7 [39]. CLD achieves the description of images with only 12 bytes through 4 steps:

1. The image is subdivided in 8x8 blocks of size $W/8 * H/8$.
2. A single dominant color for each block is selected by averaging colors within the block.
3. The 8x8 colors are converted in the YC_bC_r color space and a discrete cosine transform (DCT) is performed for each component.
4. The first 6 DCT coefficients for luminance as well as 3 coefficients per chrominance channel are taken to form the final 12-dimensional image descriptor. The 6-3-3-distribution was empirically evaluated [39] as delivering the best tradeoff of descriptor size vs. accuracy.

3.1.4 Motion

Motion is a feature exclusively available to videos. Called *'optical flow'* [6], it is described as the difference of a pixel position from one frame to the next – if $p_t(x, y) = p_{t+1}(x + \delta x, y + \delta y)$, then the motion vector is described as $(\delta x, \delta y)$ (c.f. Figure 3.3). This raw motion information can then be processed in several ways.

Global Motion Global Motion is caused by camera movement and can be used in several scenarios. For instance, if a camera zooms in on an object, this might indicate an importance of that object.

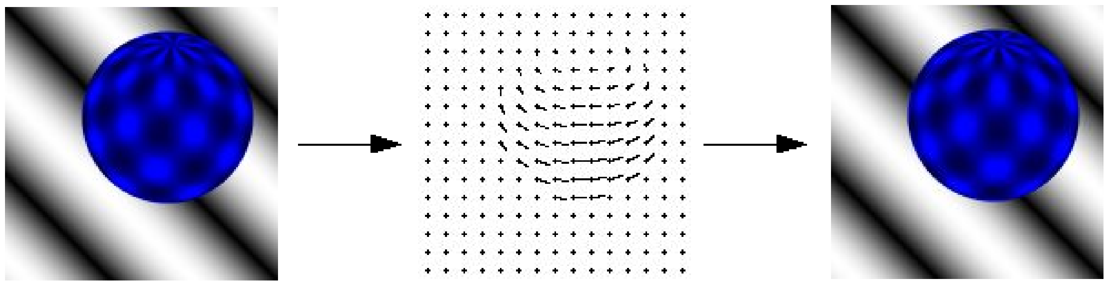

Figure 3.3: The middle picture shows the optical flow that leads from the left to the right picture. The optical flow was generated by the algorithm described in McCane et al. [51], available at http://of-eval.sourceforge.net/ (05/04/09).

One way to estimate global motion are affine motion models, where the movement $(\delta x, \delta y)$ of a pixel (x, y) is described by (c.f. [37]):

$$\delta x = a_1 + a_2 * x + a_3 * y$$
$$\delta y = a_4 + a_5 * x + a_6 * y$$

Assuming constant brightness of pixels and the unknown parameters $A = (a_1, ..., a_6)$ are global, six constraints introduced by different pixels would be sufficient to calculate A. In practice, all pixel constraints for an image are combined to minimize the error. The parameters can then be estimated using e.g. least square estimation.

Local Motion Assuming that global motion has been estimated, it is easy to recover the motion of local objects. Given the original motion vector V and the global motion G, the local motion L simply is

$$L = V - G$$

Local motion can be used for a variety of applications, an obvious one being object tracking.

3.1.4.1 Motion Window Histograms

Motion window histograms intend to capture general frame motion in frames. To achieve this, the frame is split into $4 * 3$ subregions, and for each subregion all motion blocks over all frames analyzed. As motion is quantized into one of seven bins (representing some negative motion, a lot of positive motion etc.) in both X- and Y-direction, each motion block will be assigned to one of 49 bins. The results of each motion block are aggregated into a histogram, thus describing each subregion by a 49-dimensional descriptor. To describe the whole video, all subregions are then concatenated, resulting in a total video descriptor of 588 dimensions.

To be more precise, the algorithm for describing a video looks like this:

1. For each $4 * 3$-subregion:
   (a) For each frame and each motion block within that subregion:
      i. Quantize the block into one of $7*7$ bins (given (d_x, d_y)), resulting in a 49-dimensional vector with one non-zero entry
   (b) Sum up these entries in a subregion-wide histogram
2. Concatenate the 12 subregion histograms, resulting in a video 588-dimensional vector

3.1.4.2 Moments

Moments express general information about how much motion is happening in frame regions over the whole video. To do this, several steps are required:

1. For each motion block B of the video:
   (a) For each frame F of the video:
      i. Calculate the amount of motion difference in that block:

$$D_{B_F} = \sqrt{\delta x(B_F)^2 + \delta y(B_F)^2}$$

   (b) Calculate the mean over all motion differences of that frame:

$$Mean(B) = \frac{1}{|Frames|} \sum_F D_{B_F}$$

   (c) Calculate the variance over all motion differences of that frame:

$$Var(B) = \frac{1}{|Frames|} \sum_F (D_{B_F} - Mean(B))^2$$

   (d) Quantize both mean and variance into one of 6 bins. The ranges for each bin are

Bin	Means Range	Variance Range
0	0	0
1	0.5 - 1	0.5 - 1
2	1 - 2	1 - 5
3	2 - 3	5 - 10
4	3 - 4	10 - 20
5	4 - 5	20 - 40
6	5 +	40 +

   (e) Add one to the corresponding entry of a video histogram with $6 * 6$ dimensions.

This results in a 36-dimensional video motion descriptor. In the case of YouTube videos with a resolution of $320 * 240$ and thus $20 * 15$ motion blocks, the descriptor has a total of 300 entries in the histogram.

3.1.5 Patch-Based Image Description

Global image constructors (like color histograms) can be computed quickly and describe the whole image in very few dimensions. The problem is that they are potentially unable to describe various changes of an image. As an example, an image of the eiffel tower might be partially occluded or cluttered with humans, but the observer will still regard it as a picture of the eiffel tower. Similarly, scale changes of objects within an image are not captured by global image descriptors: a miniature model of the eiffel tower standing on a table will not be regarded as being the actual eiffel tower. For these reasons, it is desirable to describe specific image regions in detail.

This is where patchbased image description fits in [64]. The idea is to find discriminative regions (like corners or blobs) within an image and describe them in great detail (c.f. Figure 3.4 for some examples of detected interest points within images). Thus, patch-based image description consists of two components: a *feature detector* that finds interesting regions and a *feature descriptor* which captures the properties of these regions.

Depending on the detector, up to thousands of interest points within an image can be found. The descriptor might then describe each of them in hundreds of dimensions, resulting in a very high-dimensional overall representation. Thus, a third processing step called 'bag of visual words' is used to reduce the number of dimensions, which will be thoroughly explained in Section 3.1.6.

In the first subsection, several interest point detectors will be explained. The next subsection will then introduce different interest point descriptors, before the last subsection will discuss a combination of feature detector & descriptor called Speeded Up Robust Features in detail.

3.1.5.1 Interest Point Detectors

Interest point detectors need to find 'interesting' regions of an image – as an example, regions with rapid changes in intensity might be interesting and thus should be described in more detail by a feature descriptor. A good feature detector is able to find the most interesting regions of an image while being fast to compute. Three different interest point detectors called Dense Grid Over Several Scales, Laplacian of Gaussian (LoG) and an approximation of the latter called difference of gaussians (DoG) will now be introduced in more detail.

Dense Grid Over Several Scales A very simple approach to find interest points is to regularly sample over the image. With a sampling step of K pixels and an image of size $M * N$, this would result in $\frac{M}{K} * \frac{N}{K}$ interest points. Note that the size of the described region of interest can differ from the sampling step. Since the size of possible regions of interest is unknown, it might thus be preferable to do the sampling on different scales.

Regular sampling can give good results, as a lot of interest points can be generated. In [31], regular sampling outperforms other feature detectors including *Difference of Gaussians* (DoG), which will be explained later. The

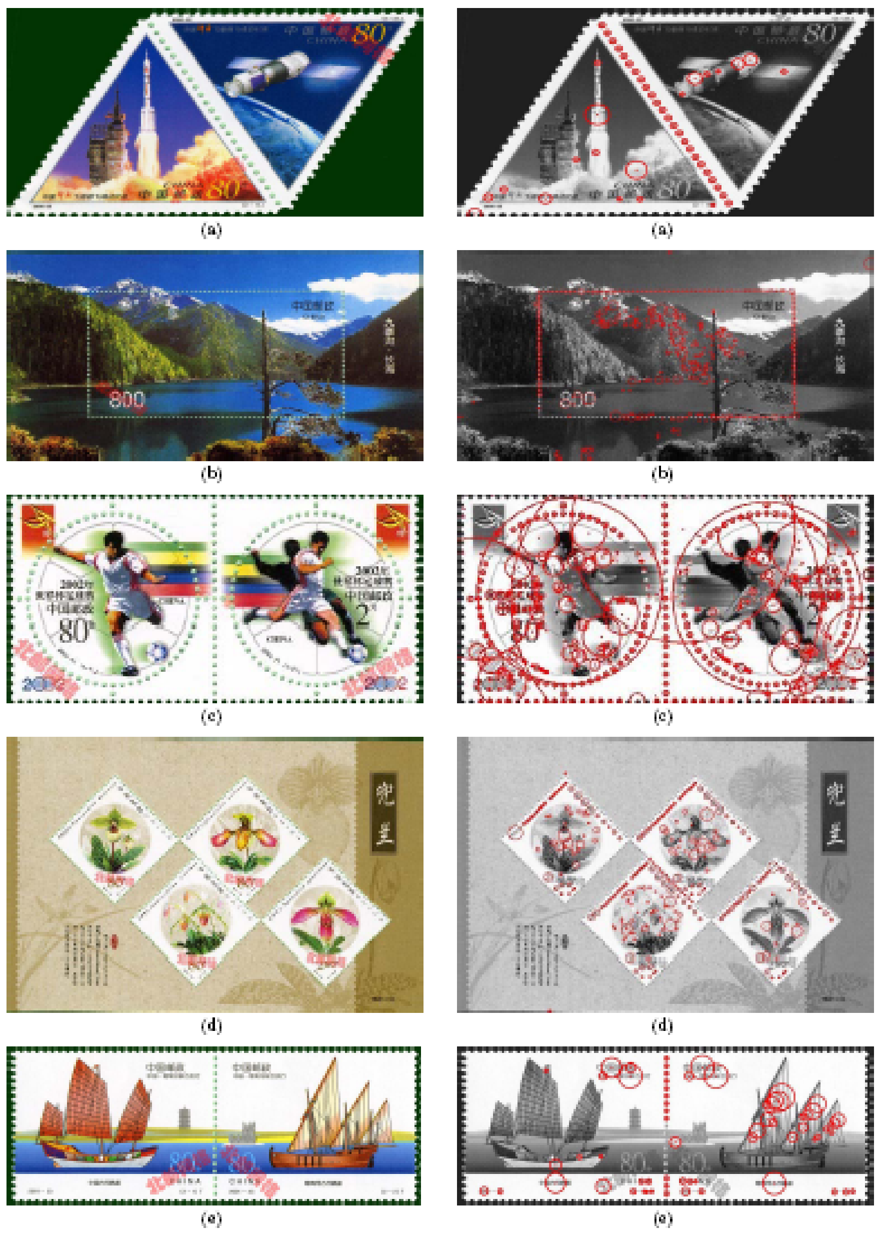

Figure 3.4: Laplacian of Gaussian (LoG) used for interest point detection. *Left:* The original colored images. *Right:* The detected interest points, indicated by red circles. The figure was taken from [53], p.6-7.

reason for this is that regular sampling generates a lot more interest points than other detectors (in the case of [31], regular sampling yielded 5250 interest points per image, whereas DoG averaged 550 points). The obvious drawbacks of this high-dimensional image description are increased computation time and memory requirements in the feature description step.

Laplacian of Gaussian The Laplacian of Gaussian (LoG), also known as Marr-Hildreth-Operator after its inventors Marr and Hildreth [50], aims at finding image blobs that contain rapid intensity changes. With $p(x, y)$ being the pixel value, the Laplacian is defined as:

$$L(x,y) = \frac{\delta^2 p(x,y)}{\delta x^2} + \frac{\delta^2 p(x,y)}{\delta y^2}$$

and can be approximated using a convolution filter, e.g.

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

As this is very sensitive to noise, the image is usually first smoothed with a Gaussian filter:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

By combining the two filters, the LoG has the form

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

This will yield large positive/negative values for dark/bright blobs. These extrema can then be treated as interest point centers. As σ defines the filter size and thus the scale of the investigated potential interest region, using different σ ensures scale invariance. Some examples can be found in Figure 3.4.

Difference of Gaussians The LoG filter can be approximated using a Difference of Gaussians (DoG). This is done by using two different sized gaussian filters on an image and subtracting them. The filter then looks as follows:

$$DoG(x,y) \equiv G_{\sigma_1} - G_{\sigma_2} = \frac{1}{\sqrt{2\pi}}\left(\frac{1}{\sigma_1} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{\sigma_2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}\right)$$

3.1.5.2 Interest Point Descriptors

Feature descriptors need to fulfill the properties described in Section 2.4. While a variety of feature descriptors exist, Mikolajczyk and Schmid [52] identify them into three main categories:

Distribution-Based Descriptors These descriptors categorize an interest region by a histograms. An example of this is the Scale Invariant Feature Transform (SIFT) [47], that describes a region by a 3D histogram of gradient locations and orientations over $4 * 4$ subregions. A variant of this is PCA-SIFT [40], which samples over $39*39$ locations and reduces the resulting feature vector using PCA. Another example is Shape Context [7], which is based on edges extracted by Canny [13] and creates a 3D histogram of edge point locations and orientations.

Frequency-Based Descriptors Describing an image in the frequency domain using a fourier transform is also a viable technique. The main problem is that the spatial relation between pixels is not converted into the frequency space. This limitation can be addressed with the Gabor transform [23]. However, a large amount of Gabor filters is needed to capture frequency and orientation changes, resulting in a high-dimensional descriptor.

Differential Techniques These methods rely on derivatives of image regions, up to a particular order. Steerable filters [22] are one example of these methods: based on the properties of local derivatives, they steer derivatives in a specific direction to achieve rotation invariance. The derivation is approximated by convolving the image region with gaussian derivatives.

These are just some examples how image descriptors can be created. Mikolajczyk and Schmid [52] compared several descriptors and found that SIFT and its variations performed best. The following section will now thoroughly explain Speeded Up Robust Features (SURF), which are a more recently developed descriptor that is said to perform similar to SIFT while being less computationally expensive [5].

3.1.5.3 Speeded Up Robust Features

SURF was originally introduced in 2006 by Bay et al. [5] and consist of both a feature detector and descriptor, which can share intermediate results to increase speed. Both heavily use *integral images* (introduced by Viola and Jones [75]), which represent the sum of all pixel values from the origin to the point (x, y):

$$I_\Sigma(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} p(i,j)$$

Feature Detector SURF tries to detects blob-like structures within an image and expands on the concepts of DoG (c.f. Section 3.1.5.1). It uses the determinant of the Hessian Matrix as a blob detector, with the Hessian Matrix defined as

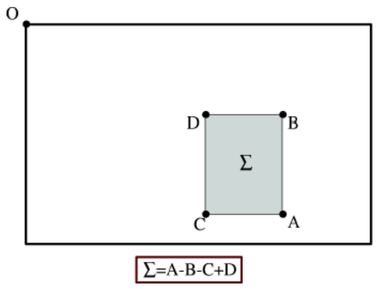

Figure 3.5: With integral images, only three additions are needed to calculate the sum of intensities within an image region. The figure was taken from [5], p. 3

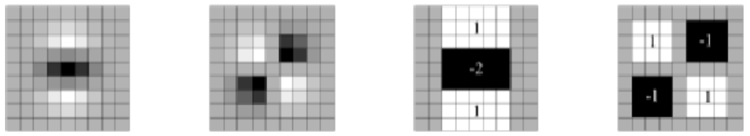

Figure 3.6: SURF interest point detection: the first two images show second order derivatives, while the latter are the approximations used by SURF. The figure was taken from [5], p. 3.

$$Hessian(x,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix}$$

L_{ab} is the convolution of the Gaussian second order derivative of scale σ with the image at point (a,b). Unlike LoG, which actually calculates those Gaussian second order derivatives, SURF approximates them with the help of box filters (c.f. Figure 3.6). These can be calculated very quickly (and on different scales) trough integral images, as the sum of intensities within a box can be calculated with only 3 additions (c.f. Figure 3.5).

The approximation of the determinant yields

$$det(Hessian_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2$$

with D_{ab} being the respective Gaussian approximations in x- and y-directions and w a fixed constant (roughly 0.9). Interest points are then detected by simply searching for local maxima of the approximated determinant within the image.

Note that it is important to search for interest points at different scales. Detectors like LoG do this by iteratively applying a gaussian filter to the original image and performing interest point detection at every 'level of blur'. With the help of the integral images, it is possible to perform interest point detection at different scales without this smoothing step.

Feature Descriptor In the case of SURF, feature description consists of two steps. The first step is optional, but can be used to achieve rotation invariance and involves finding the main patch orientation. The description of this step will be omitted as Bay et al. [5] report that a rotation invariance of about +/-15° can be achieved without applying it.

The second step consists of the actual description of the interest region. To do this, the region is first subdivided into $4 * 4$ smaller sub-regions. For each subregion, the Haar wavelets in both horizontal and vertical direction are calculated. Note that a simple Haar wavelet can be defined as the difference of pixel sums within a rectangle, and can thus be calculated with the help of integral images. The responses in both directions as well as their absolute values are summed up, resulting in a 4-dimensional descriptor $(d_x, d_y, |d_x|, |d_y|)$ of the subregion. Concatenating all subregion descriptors leads to a patch description vector of 64 dimensions. An alternative version of SURF splits d_x and $|d_x|$ in separate values for $d_y < 0$ and $d_y \geq 0$. This holds true for d_y and $|d_y|$ as well, generating 8 values per region and thus 128 dimensions in total. By normalizing this vector, it becomes robust to both illumination differences and contrast.

3.1.6 Feature Bags

As indicated, feature detectors can find up to thousands of interest points per video frame, which in turn can be represented by hundreds of dimensions. As an example, the SURF detector detected 150-350 interest points per frame for each video of size $320 * 240$, with each interest point being described by 128 dimensions.

To circumvent this very high-dimensional image description, an approach called *bag of visual words* has been introduced [64]. This approach will first be explained for patches. Afterwards, it will be adapted to motion features and frames.

Bag Of Visual Words While hundreds of interesting patches are given per image, a lot of them are very similar in nature. Figure 3.8 is one example for this: while the 2 objects are very different, they are in fact made up of the same two patches (indicated by the red respectively blue color). Another example is a rectangle, which (assuming rotation invariance) can either be described by 4 simple lines ('|'), or by 4 patches describing the corners ('L').

Thus, one way to achieve a discrete, low-dimensional representation of an image is to cluster similar patches and then store the cluster sizes in a histogram, instead of the original patches.

To elaborate on this, imagine a '*codebook*' which contains K typical representatives of interest points. Each interest point will then be mapped to its nearest representative and the total number of matches for this representative will be counted in a K-dimensional histogram (c.f. Figure 3.9). One way to generate this codebook is by formulating the problem of choosing good representatives as a clustering problem over interest points in a training set. This clustering problem can be solved by algorithms like k-Means (c.f. Section 2.5.2), where

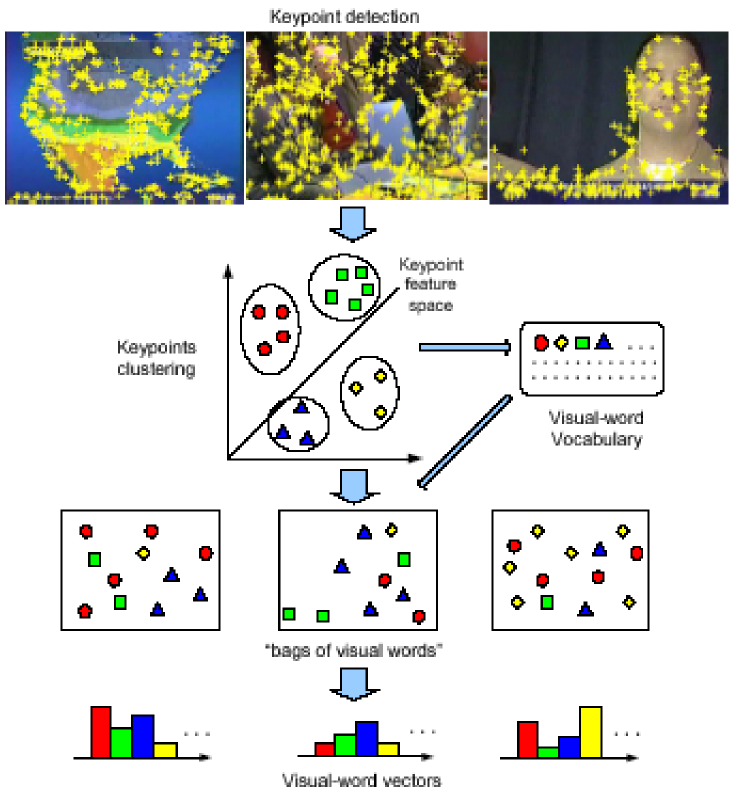

Figure 3.7: The process of creating *bag of visual words* histograms. First, interest points within an image are detected. Second, a codebook of visual words is created using a clustering algorithm. Given the codebook, each interest point is assigned to its closest mean. Finally, the number of times a mean was chosen is counted in a histogram. The figure was taken from [78], p. 2

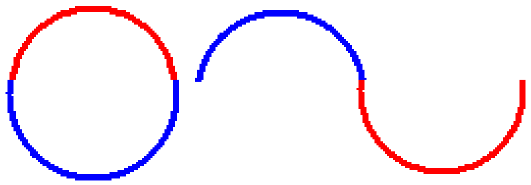

Figure 3.8: Figures of a wave and a circle, which are made up of the same (red respectively blue) patches. This is an example for the similarity of image patches.

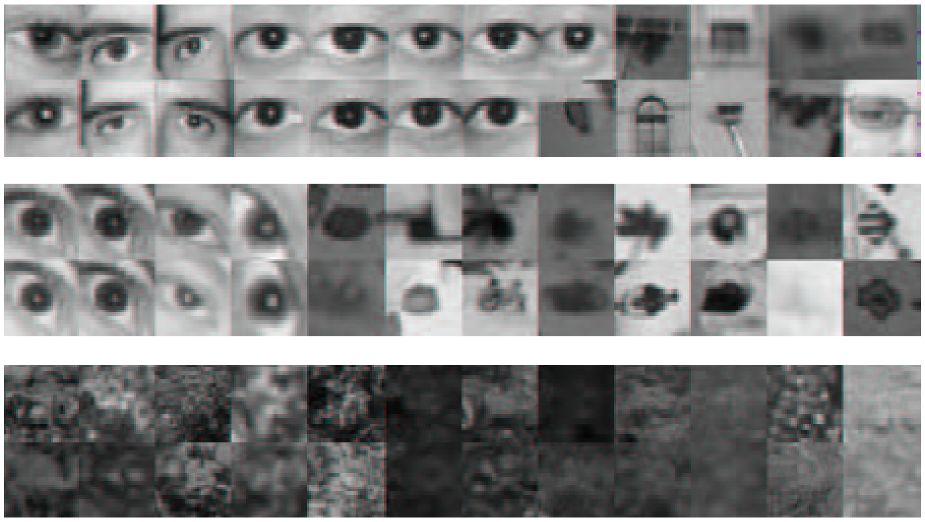

Figure 3.9: Three different examples for patches that were matched to the same codebook entry. The figure was taken from [58], p. 3

the resulting means will be considered representatives of the training set for the codebook. For a visualization of the process, see Figure 3.7.

The codebook can be seen as a vocabulary, where each mean represents a (visual) word. The term 'bag of visual words' (BoVW) stems from this analogy, with the 'bag' relating to the fact that spatial information between patches is discarded (e.g. the fact where a patch is located within an image is neglected).

The result is a K-dimensional histogram which describes the whole image, instead of hundreds of single patches which are each described by multiple dimensions. Note that K has to be sufficiently high to achieve a discriminative representation of the image, and may thus still have thousands of dimensions.

Bag Of Frames The *bag of frames*-descriptor (BoF) extends the BoVW-analogy to a video level: unlike before, whole frames are treated as words. Thus, a video is described as a histogram over representative frames.

In particular, BoF uses all BoVW-histograms for each frame as input to train a codebook with K frame means. Then, all BoVW-histograms are quantized over their respective nearest frame means. This results in a global video descriptor of size K.

Bag Of Motion The *bag of motion*-approach is similar to the bag of visual words and bag of frames in the sense that all methods quantize occurrences of features into a histogram using a codebook. The difference is that, in the context of bag of motion, the input features are patches of motion differences (d_x, d_y). Moreover, the number of words within each frame is fixed, as a sliding window of $4 * 3$ blocks is used. This results in 221 features per frame of size

$320 * 240$, where each feature has 24 dimensions. Once again, these features are then used to train a K-sized codebook, before word mappings are counted into a K-dimensional histogram.

3.2 Dimensionality Reduction

Obtaining low-dimensional image descriptors is a prerequisite to enable further video processing on a large scale. Thus, dimensionality reduction is a key step in the CBVR pipeline and a key part of this thesis.

There are two main ways to achieve dimensionality reduction without losing too much information: the first is to find those dimensions that are noisy and thus do not add any discriminative power to the feature descriptor. The second way is to identify highly correlated dimensions, as they are providing redundant information. Those dimensions that provide noisy or redundant information can then be ommited without suffering a performance loss regarding descriptive power.

This section will introduce three different means of dimensionality reduction, namely Principal Component Analysis, Restricted Boltzmann machines and the aggregation of feature histograms. Another technique which can be used for dimensionality reduction named topic models will be discussed in more detail in Chapter 4.

3.2.1 Aggregating Feature Histograms

One simple way to reduce the dimensionality of descriptors is to aggregate their subparts. As an example, consider a video with 100 frames, in which each frame gets described by a 2000-dimensional histogram. In total, the whole video will then be described by $200,000$ values. For a feature representation of the whole video, it might be beneficial to sum up all individual frame histograms into one global video histogram. This can decrease the video descriptor size dramatically, as the whole video can be described by a 2000-dimensional vector. This can also improve results, as noise will be averaged out over several histograms. Note that this fusion of vectors is not the same as the bag of frames approach discussed in Section 3.1.6: there, individual frames are treated as words and thus clustered and treated as histogram bins. Here, individual patches are still treated as words – each patch in each frame is treated as one entry in the global histogram, which will thus yield in a lot more histogram entries than in a BoF approach.

3.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a method whose underlying statistical principles were invented in 1901 by Pearson [57]. It transforms an N-dimensional feature vector into an K-dimensional vector (K « N) by calculating the covariance matrix C on a training set. Then, the K largest eigenvalues of C are

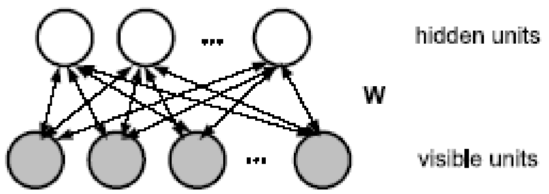

Figure 3.10: Layout of an Restricted Boltzmann Machine. The figure was taken from [33], p. 2.

identified and their corresponding (pairwise orthogonal) eigenvectors used to project the original data. Basically, PCA can be subsumed in five steps:

1. Calculate the means M for all data dimensions and subtract them from each data point. This effectively creates a data set D which has a mean of zero along every axis.

$$D_{centered} = D - M$$

2. Calculate the $N * N$-covariance matrix C of $D_{centered}$.

3. Calculate the $N * N$ eigenvectors and N eigenvalues of C.

4. Choose the K most significant eigenvalues and put their corresponding eigenvectors in a feature matrix F.

5. Multiply the transposed of the feature matrix F with the dataset to gain the compressed data P:

$$P = F^T * D_{centered}$$

6. To retrieve the original data set (aside from the loss of information due to the dimensionality reduction), simply multiply the transformed data P with F^T and add the originally subtracted means back to the data:

$$D' = (F^T * P) + M$$

As F is only a subset of all eigenvectors, an information loss of the $(N-K)$ least significant dimensions occurs, which means that $D' \neq D$. This information loss is minimized in terms of mean squared error with PCA. For more information on PCA, see [18, 62].

3.2.3 Restricted Boltzmann Machines

Restricted Boltzmann Machines were introduced in 2006 by Hinton et al. [27, 28, 60]. They are stochastic networks with neuron-like units and consist of one layer of hidden units and one layer of visible units. Units are not connected within each layer, but there is a symmetric connection to all units of the other layer (c.f. Figure 3.10). As the output for each unit is binary, the probability that a particular unit s_i is on is given by (with w_{ij} being the connected weights, b_i the bias for unit i):

$$p(s_i = 1) = \frac{1}{1 + e^{-b_i + \sum_j s_j w_{ij}}}$$

The key idea is that the network will eventually reach a Boltzmann distribution if all units are updated sequentially in any order. This Boltzmann distribution describes the probability of an output vector as its *'energy'* relative to all other possible energies:

$$p(v) = \sum_h \frac{e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}$$

Here, the energy of one state vector which is the joint configuration of visible units v and hidden units h is given by:

$$E(v,h) = -\sum_{i \in V} b_i v_i - \sum_{j \in H} b_j h_j - \sum_{i,j} v_i h_i w_{ij}$$

Training of the weights w_{ij} and biases b_i is done by a process called contrastive divergence [28], which aims to maximize likelihood. It is also possible to create a deep network to capture higher-order correlations between input units. This is done by stacking several RBMs on top of each other, where the visible output of one layer is treated as input for the next hidden layer [27]. Given a particular input in the context of dimensionality reduction, the probabilities for each output node to be active (i.e. $p(s_i = 1)$) are stored.

Chapter 4

Topic Models

In the previous chapter, different techniques for dimensionality reduction like Principal Component Analysis or Restricted Boltzmann Machines have been discussed.

This chapter will focus on a specific technique that can also be used for dimensionality reduction, and a family of different methods build on this technique. Thus, the model and training of three variants called Probabilistic Latent Semantic Analysis (PLSA), Latent Dirichlet Allocation (LDA) and Hidden Topic Markov Models (HTMM) will be explained. The adaption of these methods to video content and other work in the field will also be discussed.

Topic models originate from the textual domain. They work on a document collection D, in which each document D_i contains N_i words out of an alphabet W. Their goal is to capture the semantic relations between documents and words. This is achieved by introducing intermediate latent variables called 'topics'. A document is described as a mixture of topics, which in turn are mixtures of words. As an example, a newscast transcript might relate to topics like '*financial world crisis*', '*soccer game*' and '*weather*'. In turn, the topic '*soccer*' would be strongly associated with words like '*goal*', '*ball*' and '*referee*'.

4.1 Probabilistic Latent Semantic Analysis

PLSA was originally introduced in 1999 by Hofmann [29]. It defines a generative model for sampling the words $W_1, .., W_n$ of a document D given a multinomial topic mixture $P(Z|D)$ and topics $P(W|Z)$ (c.f. Figure 4.1):

1. For $i = 1, .., n$:
    (a) Sample a topic $Z_i \sim P(Z|D)$
    (b) Sample a word $W_i \sim P(W|Z = Z_i)$

Thus, the distribution of a word within a document is approximated by a mixture of latent aspects:

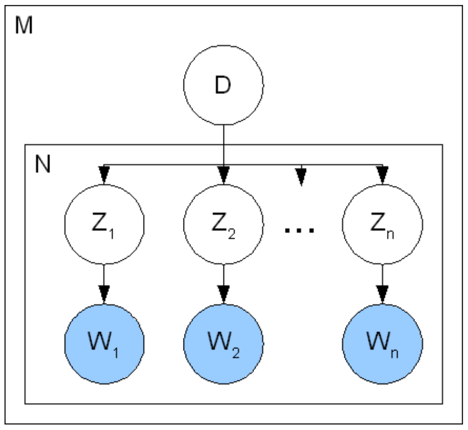

Figure 4.1: Generative model of PLSA: given a topic mixture of a document D_i, pick a topic Z_k and then a word W_j from that corresponding topic distribution.

$$P(W_j|D_i) = \sum_k P(W_j|Z_k)P(Z_k|D_i) \tag{4.1}$$

The model is usually fitted to a training corpus using Expectation Maximization (EM), which will be explained in Section 4.1.1.

Note that PLSA has several interesting properties. Topics help capture synonymy and polysemy within documents. Synonymy means that different words imply the same meaning, like the words '*buy*' and '*purchase*'. These words would have a high probability of being associated with the same topic, as they tend to be used in the same context. Similarly, polysemy implies that a word might have several meanings, like the word '*football*' implying either american football or european soccer. Polysemious words have a good chance of being associated with multiple topics. '*Football*' could have a high probability for a topic which also includes words like '*quarterback*', '*receiver*' and '*touchdown*' (american football), as well as be made up of words such as '*goalkeeper*', '*striker*' and '*corner kick*' (soccer).

Another interesting remark about PLSA is that it assumes a bag of word model: essentially, this means that syntactical information between words is discarded. It is only important how often a word occurs within a document, not the contextual information. As an example, the probability to generate the sentence *'the quick brown fox jumps over the lazy dog'* in a PLSA model would be as high as generating any random permutation of it, like *'brown dog fox jumps lazy over quick the the'*.

4.1.1 Model Fitting with Expectation Maximization

To fit a PLSA model to training data, it is desired to approximate the joint probability of documents and words. Weighted with the document's length, this is done by maximizing the log likelihood

$$L = \sum_{i=1}^{N} \sum_{j=1}^{M} n(D_i, W_j)\ logP(D_i, W_j) \tag{4.2}$$

As it is infeasible to find a closed form solution for the maximization of L, a different approach called Expectation Maximization (EM) is often used. Dempster et al. [17] formalized EM in its generalized form in 1977, which basically consists of two alternating steps:

- In the expectation step (**E-Step**), posteriors for the latent variables Z based on the current parameter estimates are calculated. Using Bayes rule and Equation 4.1, this results in:

$$P(Z_k|D_i, W_j) = \frac{P(W_j|Z_k)P(Z_k|D_i)}{\sum_{l=1}^{K} P(W_j|Z_l)P(Z_l|D_i)} \tag{4.3}$$

- In the maximization step (**M-Step**), parameters are updated given the expected complete log-likelihood, which is a lower bound to the log-likelihood defined in Equation 4.2 and depends on the latent posterior estimates from Equation 4.3:

$$E[L^C] = \sum_{i=1}^{N} \sum_{j=1}^{M} n(D_i, W_j) \sum_{k=1}^{K} P(Z_k|D_i, W_j)\ log\left[P(W_j|Z_k)P(Z_k|D_i)\right]$$

  Using Lagrange multipliers, this leads to the following simplified equations for estimating the parameters:

$$P(W_j|Z_k) = \frac{\sum_{i=1}^{N} n(D_i, W_j)P(Z_k|D_i, W_j)}{\sum_{m=1}^{M} \sum_{i=1}^{N} n(D_i, W_m)P(Z_k|D_i, W_m)}$$

$$P(Z_k|D_i) = \frac{\sum_{j=1}^{M} n(D_i, W_j)P(Z_k|D_i, W_j)}{n(D_i)}$$

Both steps are alternated until either the algorithm converges or an alternative termination condition like a maximum number of iterations is met. The latter is often used to prevent overfitting. For more details, see Hofmann [29]. Note that EM does not guarantee to find the global optimum, but will converge to a local maximum as L^C only gives a lower bound of L.

To increase the chance of escaping local maxima, a method called Tempered Expectation Maximization (TEM) can be employed [29]. This annealing method, which is also likely to reach good results with less iterations than normal EM, introduces a parameter β in the E-Step:

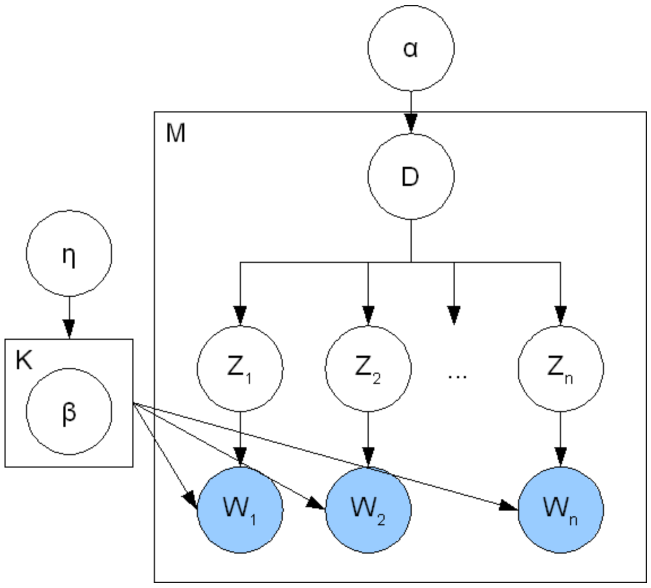

Figure 4.2: The generative model of LDA: given a topic mixture for a document D_i, draw a topic Z_k and then a corresponding word W_j. The word mixtures for a topic Z_k are given by β_{Z_k} and determined by a common prior η.

$$P(Z_k|D_i, W_j) = \frac{[P(W_j|Z_k)P(Z_k|D_i)]^\beta}{\sum_{l=1}^{K} [P(W_j|Z_l)P(Z_l|D_i)]^\beta}$$

The algorithm is then run with $\beta = 1$ for T iterations (early stopping). Afterwards, the performance in each iteration is compared with the previous iteration (on held-out data) and β decreased if performance did not improve. This process continues until decreasing β does not improve results on the held-out data any further. Aside from helping to escape local maxima and significantly speeding up the training process, this will also gradually smooth the posterior probabilities with decreasing β.

4.2 Latent Dirichlet Allocation

A topic model similar to PLSA is called Latent Dirichlet Allocation (LDA). Introduced in 2003 by Blei et al. [10], LDA models the document distribution using a Dirichlet prior. Thus, assuming α as parameter for the Dirichlet distribution and given a prior η that estimates the word-topic-probabilities β (for each Z_j), the generative model for a document is as follows:

1. Sample N $\sim$ Poisson(ξ) // document length
2. Sample $P(Z|D) \sim$ Dirichlet(α)

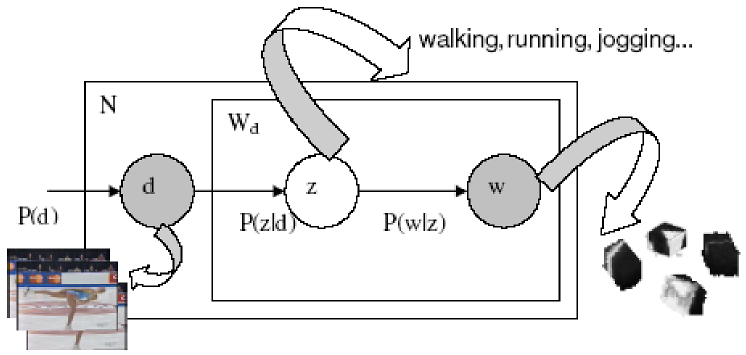

Figure 4.3: The PLSA model applied to video. Videos correspond to documents, words to frame patches. In this example, different movement types correspond to topics. The Figure was taken from [55], p. 5.

3. For word $i = 1,..,N$:
   (a) Sample a topic $Z_i \sim P(Z|D)$
   (b) Sample a word $W_i \sim P(W|Z = Z_i, \beta_{Z_i})$

Modeling documents with a Dirichlet prior is meant to overcome two flaws of PLSA: First, the number of parameters to estimate does not increase linearly with the number of documents in the training set. This is supposed to help prevent overfitting. Second, it is clear how to assign probabilities to documents that are not in the training set - in that sense, PLSA is not a generative model at the document level.

4.3 Topic Models for Videos

Applying PLSA to images is pretty intuitive assuming a bag of visual words-approach as explained in Section 3.1.6. In this case, local patch descriptors are quantized over a given codebook, resulting in a histogram with K bins. The codebook corresponds to the vocabulary W of the textual domain with a dictionary of size K. Each histogram bin represents a word W_j and thus, the histograms represent the number of occurences of each word within a document (the document corresponds to the image). This is possible because spatial information that gets discarded when patches are put into bins is not important for the 'bag of words'-model used by PLSA.

Intuitively, a topic is expected to correspond to a semantic meaning. An example is shown in Figure 4.3, where different topics are supposed to correspond to different types of human movement. Another example is the field of sports broadcast – here, one topic could correspond to soccer, while another one might be equivalent to basketball content. Note that the granularity of the semantics can vary: given the field of soccer, one topic could correspond to the lawn, another one to the ball, others to players from different angles etc. This representation can be extended to videos in several ways. First, the whole video can be treated as a document, where all patches from all frames are fused into one global bag of visual words. The problem with this approach is that temporal structure is lost and no information about specific video parts is maintained.

Another approach is to model each frame as a document with a separate frame-wise histogram. This results in a large feature of size $|frames| * |visual\ words|$, and an additional fusion of frame information is required for video representation.

Finally, a video can also be treated as a collection of shot-level documents, i.e. all frames within a shot are assumed to be semantically connected. Using such a representation, finer notions (e.g., certain events in the video), which cannot be captured by the global video representation, could be preserved while maintaining a more compact representation than there would be on only the frame level.

It is important to note that the different representations also have consequences for the creation of the topic model. In the example of a soccer game with different shots of the field and the audience, an analogy between shots and documents is likely to lead to two topics – one for each the field and the audience. However, if one document corresponds to one video, a single topic will be mixed and contain both patches of the field and the audience.

4.4 Hidden Topic Markov Model

While topic models like PLSA and LDA neglect temporal order within a video stream, a more appropriate view might be to understand shots as the semantic units of video. Usually, the content within a shot is heavily semantically related, while switches of semantics occur at shot boundaries. The Hidden Topic Markov Model (HTMM) [24] described in the following makes use of this temporal structure. It models documents as sequences of sentences (which will become *shots* in the video scenario). Words within a sentence are assumed to be derived from the same topic, while topic transitions occur only between sentences. Developed in 2008 by Gruber et al. [24], the HTMM is based upon LDA.

Each document D consists of sentences $S_1, .., S_m$, and each sentence S_j again consists of words $W_1^j, .., W_{|s_j|}^j$. A topic distribution $P(Z|D)$ is assumed to be given (which is drawn from a Dirichlet prior with parameter α as in LDA [10]). Topic transitions are only allowed between sentences, and at the beginning of each sentence it is decided whether a transition occurs (which happens with probability ε). Either the topic from the previous sentence is used or a new one is chosen according to $P(Z|D)$. Each word within the sentence is then sampled from the chosen topic.

1. Draw $P(Z|D) \sim Dirichlet(\alpha)$

2. For $j = 1, .., m$: *// sample sentences*

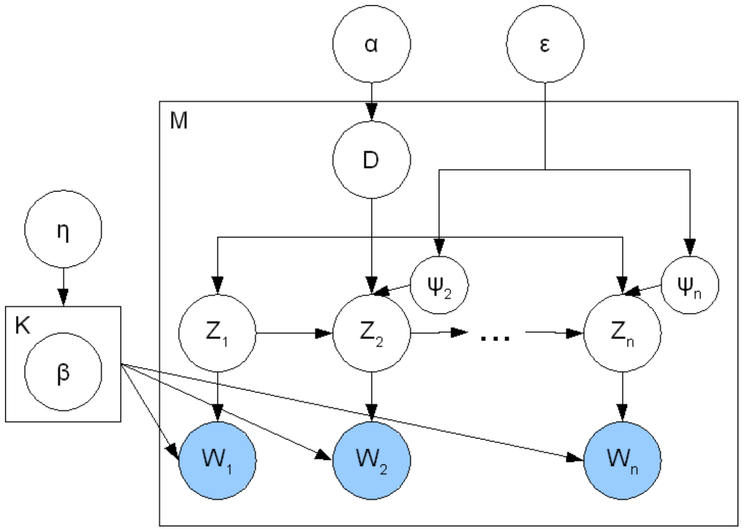

Figure 4.4: The generative model of HTMM: given a topic mixture for a document D_l (drawn from a Dirichlet prior α), consider each word in the document. If the word does not start a new sentence (Ψ_i, depending on the probability ϵ), the topic equals the topic of the previous sentence. Otherwise, the topic is freshly chosen according to the documents topic mixture.

(a) Sample $\Psi_j \sim Binom(\varepsilon)$

(b) If ($\Psi_j == 0$) set $Z_j = Z_{j-1}$
else $Z_j \sim P(Z|D)$

(c) For $i = 1, .., |s_j|$:

i. Draw $W_i^j \sim P(W|Z_j)$

The idea is that words within a sentence are semantically connected, so that enforcing a single topic per sentence leads to a more stable document structure. The connection of the sentences is modeled using a Hidden Markov Model. Training of the HTMM is done by Expectation Maximization (c.f. Section 4.1.1) and the forward-backward algorithm [59].

In the case of HTMM, the latent variables are twofold: aside from the topics Z_k, the decision whether a topic transition occurs at a sentence transition Ψ_j is also a latent variable. Thus, Expectation Maximization is slightly modified:

- The E-Step is adapted to calculate $P(Z_m, \Psi_m|D, W_1, \ldots, W_M; \theta, \beta, \epsilon)$ for each sentence using the forward-backward algorithm.

- In the M-Step, the parameters $P(Z|D)$ and $P(W|Z)$ (which corresponds to β in the LDA/HTMM notation) are calculated, as explained in Section 4.1.1. Additionally, the parameter ϵ also has to be estimated, which serves

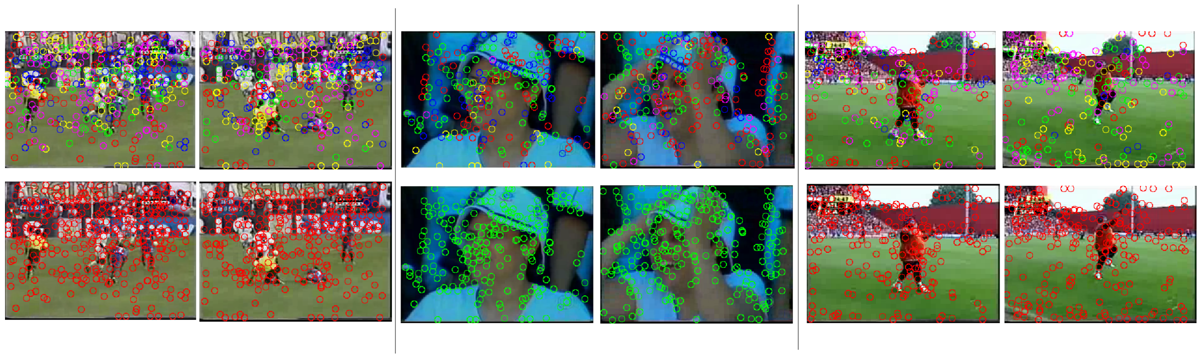

Figure 4.5: A soccer video consisting of three shots, which was analyzed using PLSA (top) respectively HTMM (bottom). While PLSA yields different topic associations for different patches within the same shot (indicated by the circle color of the patch), the HTMM associates a shot (and frames within a shot) with a single topic and allows topic transitions only at shot transitions.

as prior for the probability of topic transitions and can be calculated using Lagrange Multipliers(c.f. [24]).

For more details of the training, see Gruber et al. [24].

4.5 Related Work

Topic models have been employed in various works in the image domain. As an example, Barnard et al. [3] use a multimodal LDA for automatic annotation. The analogy between the 'bag of words' in the textual domain and 'bag of visual words' in the image domain has been explored by Sivic et al. [63]. They employed LDA and PLSA for object discovery in images and reported that both outperformed previous approaches [21]. Similarly, Quelhas et al. [58] investigated the use of PLSA for scene classification and compared the dimensionality reduced features to the original BOVW-representation on a small dataset of 9457 images. Both approaches outperformed standard methods like binary hierarchical bayesian classifiers [73].

Topic models for dimensionality reduction in the image context on larger data sets has been investigated by Hoerster et al. [33]. They compared PLSA and LDA with Restricted Boltzmann Machines [60], which can also be employed for dimensionality reduction. While all three approaches perform similarly, PLSA seems to have a slight edge over both LDA and RBM.

Several extensions to topic models have been proposed to tailor them more specifically to the image domain. One direction is the attempt to integrate the spatial information of patches within an images. For example, Tirilly et al. [72] propose to capture the spatial information of patches by projecting them to a main axis (computed by PCA). This is supposed to introduce structural order, shaping the analogy of 'visual sentences'. Liu and Chen [44] also try to use spatial information by integrating correspondence (shape and location of patches). To be more precise, they give 'rewards' R for corresponding patches and learn those depending on the topics ($P(R|Z)$). While this improves performance, computation time is also increased. Fergus et al. [20] learn object categories by retrieving images from Google Image Search and use PLSA to filter noisy content. Moreover, in order to better capture the position of the objects, they introduce a latent variable into the PLSA model that describes the centroid, resulting in invariance to translation and scaling. Hohl et al. [30] enhanced LSA in the video domain to capture spatial information by modifying the visual codebook: instead of treating each codebook mean C_i as a visual word, they use tuples (C_i, C_j) as visual words, hoping to capture the co-occurrence of regions this way.

Monay and Garcia-Perez [54] argue that, in word-image associations, the semantic information gained by words is much higher than for images. Thus, instead of concatenated BOVW+word-features, they propose to first train a PLSA model on the captions and then use the previously computed $P(Z|D)$ to train a PLSA on the viswords for $P(W|Z)$. This way they enforce the topic distributions implied by the caption model on the BOVW model.

Hoerster et al. [35] also explored an interesting extension to PLSA where they model visual words as continuous distributions rather than quantized high-dimensional descriptors. They argue that this is more natural in the image domain and show a performance gain of 4% over the normal discretized PLSA model.

There are only few works that utilize topic models in the video domain for semantic meaning. Souvannavong et al. [67, 68, 69] explored Latent Semantic Analysis (LSA) with region-based descriptors in the video domain for tasks like object retrieval and scene classification. LSA was based on linear algebra and heavily inspired PLSA. One of its shortcomings was the lack to capture polysemy, i.e. the possibility of a word to have several meanings. Niebles et al. [55] used PLSA to model human action categories like walking, running etc. In their approach, they use spatial-temporal visual words as vocabulary.

Chapter 5

Experiments

This chapter will describe the experiments that took place within the scope of this thesis. This includes the setup, results and discussion for each experiment. There are two main goals for these experiments: the first goal is to decide whether topic models are a good tool for dimensionality reduction in the context of video content. The second goal is to explore whether exploiting structural video information on a shot level benefits results.

To reflect these goals, the first section will discuss the general test setup, including the datasets, evaluation measures and third-party software used. The following section will then aim to answer the question of the first goal, namely if topic models are a good fit to aid in content-based video retrieval. The last section of this chapter will then incorporate shot structure to decide if structural information benefits system performance.

5.1 Experimental Setup

This section will briefly discuss the tools used in each step of the CBVR pipeline (c.f. Figure 5.1).

Preprocessing All experiments are based on the same preprocessing steps: for each video, the extended SURF (Section 3.1.5.3) detector and descriptor with 128 dimensions was run on each 10th frame. These features were then aggregated into a bag-of-viswords histogram (Section 3.1.6) with 2000 entries. The visual codebook was pre-generated from a bigger, generic dataset consisting of Youtube videos and was not changed throughout all experiments. Thus, as a start point, each video was described by $\frac{|Frames|}{10} * 2000$ (integer) values.

Shot Boundary Detection As cuts and fades make up the majority of transitions, it seemed to be sufficient to focus on these transition types for shot boundary detection (Section 2.2). For this task, two programs developed at the university of Mannheim, Germany by Lienhart [41] were used, which focus on

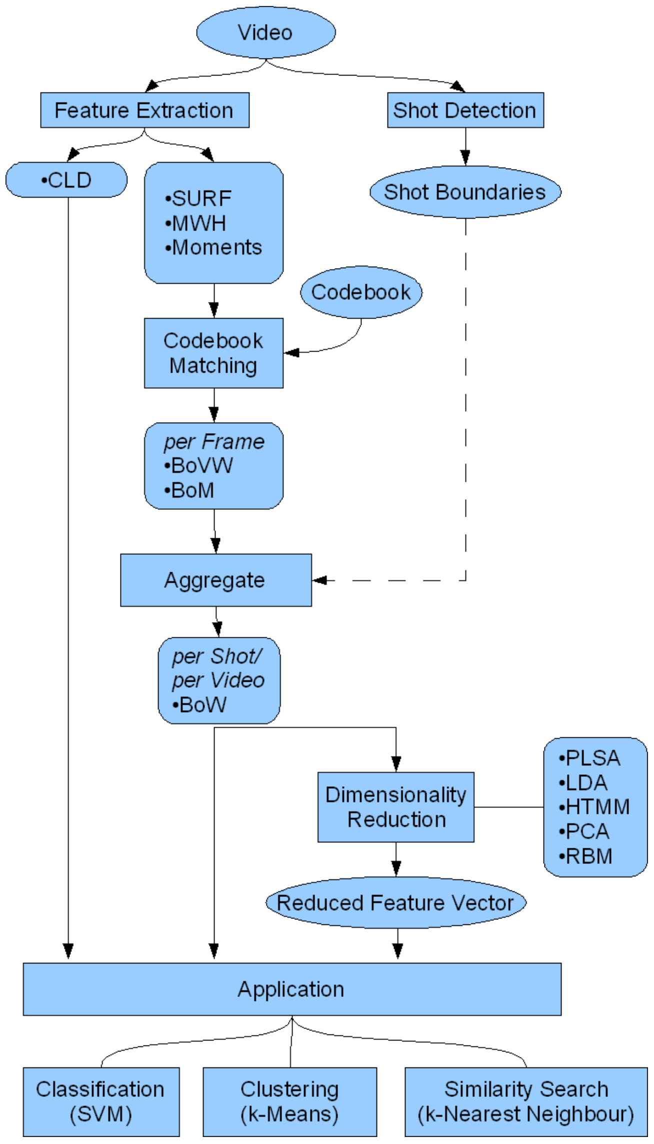

Figure 5.1: The general experimental processing pipeline. The first step is to extract features (like SURF) for each video. Patchbased features are then matched to a codebook, resulting in bag-of-word histograms. These can then be aggregated to shot-level histograms (if shot boundary information is available) or video-level histograms. The histograms can then either be directly used as (potentially) high-dimensional feature vectors or be reduced to a lower dimension using dimensionality reduction techniques like topic models. Either way, the resulting features are then used as input for various applications like classification or similarity search.

detecting cuts respectively fades. To the best of our knowledge, these are the only open source implementations available for shot boundary detection.

Feature Extraction Motion window histograms (Section 3.1.4.1), Moments (Section 3.1.4.2) and Color Layout Descriptors (Section 3.1.3.1) were extracted using in-house implementations. While there are several variants for SURF available, the implementation provided by Bay [5] was employed throughout this thesis.

Codebook Generation & Matching All codebooks for the varying feature bags were generated using a k-Means variant (Section 2.5.2). While the original k-Means algorithm is pretty straightforward, it can be speeded up using triangle inequalities. This variant of k-Means was developed at the Max Planck Institute in Tübingen, Germany by Gehler [19].

Codebook matching was performed using approximate kd-trees (Section 2.5.3), using an implementation developed at the University of Valencia, Spain by Paredes et al. [56].

Dimensionality Reduction A variety of dimensionality reduction techniques were tested. While PLSA(based on Expectation Maximization, Section 4.1) and PCA (Section 3.2.2) were provided in-house, the Hidden Topic Markov Model, which was originally developed during an internship with Google, USA by Gruber [24], was adapted from the textual to the video domain. Restricted Boltzmann Machines (Section 3.2.3) were tested with the help of the modular toolkit of data processing, a python toolkit that was originally build by Zito et al. [80] at the Bernstein Center for Computational Neuroscience, Berlin, Germany.

Statistical Modeling Classification experiments were performed using Support Vector Machines (Section 2.5.1). In particular, the library libsvm, created and maintained by Chang et Lin[15] at the National Taiwan University in Taipei, Taiwan, was used. Unless stated otherwise, an RBF kernel and 5-fold crossvalidation were employed throughout this work.

For the clustering tasks, the same faster k-Means (Section 2.5.2) variant from the Max Planck Institute [19] that was also employed for codebook generation was used.

Finally, the similarity search was implemented by a nearest-neighbor search in an approximate kd-tree structure (Section 2.5.3), which was the same as used for the codebook matching [56].

5.1.1 Datasets

In this work, three different video datasets were employed. The first dataset *(DS-1)* consists of 900 YouTube[1] videos, which belong to one of the five categories *hiking, ice-skating, library, soccer, talkshow.* Each category is made up

[1]http://www.youtube.com

out of 150-200 videos, where 50 were put into the test set and the rest used as training set. This results in a training set size of 650 videos and a test set size of 250 videos. The videos have an average length of 4 minutes. They were downloaded from YouTube by searching for the respective category and similar words. For example, soccer videos were obtained by querying YouTube with words like '*soccer*' or '*soccer world championship*'. Furthermore, to increase variety within videos, only one video per author was retrieved. Note that the videos have enormous intra-class variability, since videos are manually tagged by the author when uploaded. As an example, one video in the *hiking* category consisted of a presentation about hiking boots.

The second dataset *(DS-2)* is also made up of YouTube videos, but with ten, more difficult categories (*basketball, cats, desert, eiffeltower, helicopter, riot, sailing, soccer, swimming, tank*). In total, it is comprised of 3618 videos, where 250 videos of each category are in the training set (amounting to a total training set size of 2500) and the rest in the test set (1118). Note that the amount of test samples for each category varies, for example there are only 57 videos in the *eiffeltower* category, whereas the categories *riot* and *sailing* have more than twice as much(c.f. Table 5.1).

To create ground truth on a shot level, DS-2 was also annotated manually on a keyframe basis, with the keyframes being extracted by the method proposed by Borth et al. [12]. This means that the human evaluator decided for each keyframe whether or not the associated concept was actually present in the video. To generalize this keyframe annotation to shot level, shots were only accepted as 'containing the concept' if all keyframes within that shot were marked as containing the concepts. If that was not the case, they were discarded.

This resulted in a third dataset (DS-3) where videos were annotated on a shot level. The training set DS-3B was comprised of 12900 manually annotated shots (once again, differing largely in number of shots per category) and the test set of 5830 shots. Finally, the training set DS-3A contains all shots extracted from a video (not only the manually annotated ones). The differentiation was introduced to be able to analyze the impact of noisy data (c.f. Experiment 5.3.4).

5.1.2 Performance Criteria

This section introduces criteria that can measure the quality of CBVR applications. In particular, precision and recall will be explained as performance criterion for classification and similarity search, and purity will be introduced as a way to measure performance of clustering algorithms.

Classification

Common measures to evaluate the performance of classification are *precision* and *recall* [48]. Precision is a measure of how many retrieved documents of a fixed number of documents are relevant (which means that they belong to the same category as the query document), and thus defined as

	DS-1 (video)		DS-2 (video)		DS-3 (shots)		
	Train	Test	Train	Test	Train (A)	Train (B)	Test
hiking	150	50					
ice-skating	100	50					
library	100	50					
talkshow	150	50					
soccer	150	50	250	123	14762	2324	1187
basketball			250	113	11211	1901	827
cats			250	109	8935	1884	925
desert			250	100	6662	378	148
eiffeltower			250	57	5751	682	129
helicopter			250	126	4957	546	321
riot			250	131	12576	2140	778
sailing			250	134	8063	647	389
swimming			250	112	11985	1951	905
tank			250	113	5682	447	221
Total	650	250	2500	1118	90854	12900	5830

Table 5.1: The amount of training and test data for the different datasets.

$$Precision\ P := \frac{|relevant\ documents \cap retrieved\ documents|}{|retrieved\ documents|} = \frac{TP}{TP + FP}$$

Similarly, recall is a ratio for how many documents are needed to retrieve a fixed number of document of the relevant class:

$$Recall\ R := \frac{|relevant\ documents \cap retrieved\ documents|}{|relevant\ documents|} = \frac{TP}{TP + FN}$$

True positives (TP) are the number of items that were correctly classified as the relevant label, whereas *false positives* (FP) are items that, while actually belonging to a different class, were classified as belonging to the evaluation-relevant class. Lastly, *false negatives* (FN) are the number of items that actually belong to the relevant class, but were associated with a different one.

As precision and recall are highly related to each other (higher recall will lead to lower precision and vice versa, as it is common to have a lot of good retrieval results in the beginning, while they are more sparse later; c.f. Figure 5.2), an evaluation measure called *average precision* (AP) is often used, which is the integral over the function of precision against recall ($\delta R(i)$ is the recall change from $i-1$ to i):

$$Average\ Precision\ AP := \int_0^1 P(R) dR = \sum_{i=1}^{|Docs.|} P(i)\delta R(i)$$

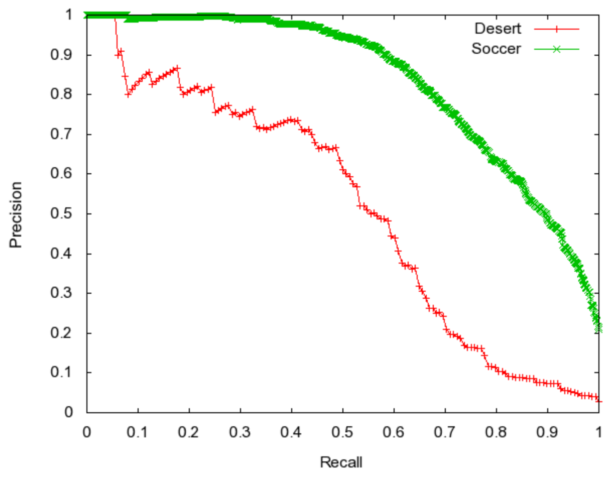

Figure 5.2: Two functions of precision (Y) given different target recalls (X) in a classification task. Average precision (AP) is the integral of the function and commonly used as evaluation criterion. While the soccer function appears to be nearly optimal (retrieving only correct results in the beginning), a curve like the red one is more often encountered in practice: here, the first results are also correct, but degrade in a more unstable way.

Clustering

Evaluating clustering performance can be approached from either an information theoretical (with measures such as *mutual information* or *conditional entropy*) or a statistical point of view. In this work, the latter approach is employed, represented by the *purity* criterion.

Purity [79] is an evaluation criterion which counts the number of samples per category k in each cluster C_i. The category $C_{i,k}$ with the maximum samples is divided by the total number of samples in a cluster and (with adjusted weights) summed up over all clusters C_i:

$$purity(C_i) = \frac{1}{|C_i|} \max_k (|C_{i,k}|)$$

$$purity(C) = \sum_j \frac{|C_j|}{|D|} purity(C_j)$$

Note that purity will get higher with more clusters and peak at 1.0 in case every shot gets it own cluster assigned. Thus, it is not a fair measure to evaluate the best number of clusters. However, it is an intuitive approach that is fair for comparing different methods on the same amount of clusters. Keep in mind that the clustering process itself does not utilize any labeling information, but purity as an performance criterion then uses these labels to evaluate cluster quality.

Similarity Search

A simple way to evaluate performance in the context of similarity search is to pick a fixed number K, and retrieve the first K results of the evaluation engine. The performance then equals the precision of classification tasks:

$$P(K) = \frac{TP}{TP + FP}$$

Both recall and average precision (up to a fixed number of retrieved results) can be specified similarly.

5.2 Video Classification

This series of experiments will deal with the task of video classification [65]: the system learns to recognize unseen videos and gives information about the content. As an example, it should be able to differentiate between soccer and swimming videos.

The first step to achieve this is the choice of good feature descriptors, which will be investigated in the first experiment. Visual descriptors, which have been widely investigated for the image domain, will be included, but the experiment will focus more on the video-specific motion features. As good feature descriptors are often high-dimensional, different dimensionality reduction techniques like PCA and topic models will then be investigated. As it has been previously illustrated that PLSA and LDA perform comparably [33, 63], this thesis will focus on PLSA as representative for topic models in all experiments. The third experiment will then explore the incurred performance loss given different target compression rates. The last experiment in that series analyzes the performance gain induced by combining different features (like BoVW and MWH), when maintaining the same overall target compression ratio.

5.2.1 Comparing Features

The first series of experiments was concerned with choosing good initial features for video retrieval. An overview of features has been given in Section 3.1. As the emphasis is on video content, only one descriptor that is also applicable to images was analyzed. Apart from that, several motion-related features were evaluated.

In particular, the focus lay on SURF as a patch-based image descriptor and it was quantized over a codebook with 2000 entries into a 'bag of visual words'-histogram. This was compared with the 'bag of frames'-approach, which quantizes over those BoVW-histograms of all video frames using a codebook with 500 entries. These 500-dimensional histograms represents the number of times a frame occurs within a video and is only available on a video level. A motion-based approach called 'bag of motion' works in a similar fashion: here, motion descriptors within each frame are quantized over a codebook of size 100. Motion window histograms put single block windows into fixed bins and sums

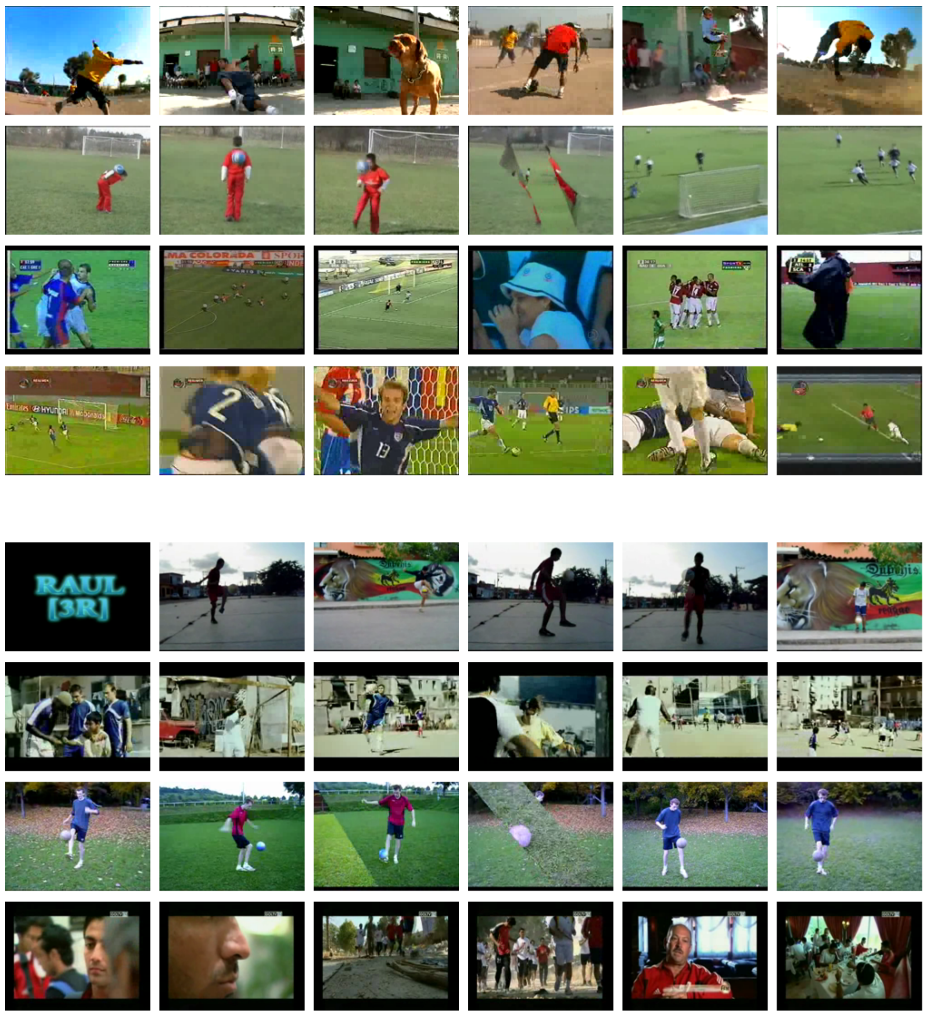

Figure 5.3: Examples of videos that get classified correctly as soccer (upper 4 rows) or misclassified (lower 4 rows) using pure BOVW descriptors. Each video is represented by a row of 6 regularly sampled frames. It is interesting to note that the wrongly classified videos seem to not belong as strictly to the concept of soccer as the correctly classified: the first and third wrong video show a person doing tricks with a ball, the second one is starring soccer players in an advertisement, and the last one is a video of an interview about training methods. Thus, the system seems to have learned a very strict, but nevertheless correct definition of soccer.

Feature	# Dim.	MAP
Bag of visual words (BoVW)	2000	**0.911173**
Bag of frames (BoF)	500	0.703738
Bag of motion (BoM)	100	0.502281
Moments	36	0.375507
Motion window histograms (MWH)	588	0.668701
BoVW + MWH (early fusion)	2588	0.872751
BoVW + MWH (late fusion)	2588	**0.926052**
BoVW + BoF(late fusion)	2100	0.918476
BoVW + MWH + BoF (late fusion)	2688	**0.932854**

Table 5.2: Mean average precision for different single features and their fused derivations.

those bins over the whole image. Finally, moments capture the general motion activity in each block over the whole video and quantize the results for each block into a 36-dimensional descriptor.

The experiments were concluded on DS-1 on a per-video level. The benefits of early and late fusion were also investigated, which is the combination of different features. In this context, early fusion means that several features are concatenated and the (larger) total feature vector then used for machine learning. A potential drawback for early fusion is that features can not be as easily normalized, e.g. features with different machine learning performances are weighted equally. Late fusion implies running machine learning techniques on each feature separately and fusing the outputs afterwards. The drawback is that several models for the different features have to be kept and that fusion of the scores of each model is a nontrivial problem.

Results for this experiment can be found in Table 5.2. Some examples for correctly and wrongly misclassified samples can be found in Figure 5.3. It turns out that the initial BoVW approach with SURF features already provides an excellent performance for this dataset, whereas the more time-consuming BoF approach lacks expressive power. Regarding motion features, the motion window histograms performs best.

Performing an early fusion of BoVW and MWH (87.2%) leads to worse results than plain BoVW (91.1%). This can be explained by the fact that features were just concatenated, but not weighted in the process. However, a late weighted fusion works well and improves overall performance. It is interesting to note that, while the plain BoF model (70.3%) performs better than plain MWH (66.8%), fusing both with BoVW works better for MWH (92.6%) than BoF (91.8%). This can be explained by considering the properties both descriptors are based on: BoF and BoVW both utilize visual information, which probably makes the combination partially redundant. On the other hand, MWH is a motion feature, which is not exploited by BoVW.

As expected, fusing BoVW with both MHW and BoF into one model leads

Category	BoVW 2000	PCA	PLSA	RBM
basketball	0.812526	0.787871	0.767004	0.726583
cats	0.728590	0.673048	0.707340	0.543335
desert	0.626539	0.402013	0.494892	0.457253
eiffeltower	0.771503	0.455239	0.659240	0.529383
helicopter	0.546355	0.331466	0.384607	0.328899
riot	0.708391	0.619775	0.716111	0.556713
sailing	0.810469	0.632837	0.745042	0.598603
soccer	0.837547	0.771523	0.831563	0.768688
swimming	0.748331	0.671698	0.706466	0.546396
tank	0.555097	0.343956	0.470825	0.341197
Overall	*0.714535*	*0.568943*	*0.648309*	*0.539705*

Table 5.3: The average precision (AP) of the different categories of DS-2 for different dimensionality reduction techniques (PCA, PLSA, RBM) and a 2000-dimensional control run.

to the best results. This indicates that, in general, more and diverse features are always better. The drawback is that having more features automatically leads to larger input data. Thus, there is a trade-off between complexity and performance. This makes it desirable to either exploit low-dimensional feature descriptors or analyze dimensionality reduction techniques which reduce feature descriptors to fewer dimensions without losing much information. This way, it would be possible to fuse a variety of different features (resulting in better machine learning performance) while maintaining compact descriptors.

5.2.2 Comparing Dimensionality Reduction Techniques

As indicated by the previous experiment, it is desirable to include a variety of features while maintaining a low-dimensional, compact representation.

To figure out which DR technique works best the original 2000-dimensional BoVW histograms was reduced to 50 values by various means. The commonly known Principal Component Analysis (reduced to the first 50 principal components), PLSA as representative for topic models (reduced to 50 latent topics) and Restricted Boltzmann Machines (with 50 output nodes) were tested. The 50-dimensional features were then used for a classification task on the dataset DS-2.

As illustrated in Table 5.3, PLSA comes closest to the non-reduced performance with a mean average precision (MAP) of 64.8%. PCA has a MAP of 56.8%, with RBM trailing at 53.9%. This is a bigger difference than reported in the experiments performed in [34], where PLSA only slightly outperformed RBM. Nevertheless, both experiments are good indicators that topic models are indeed a good approach to effectively compress semantic information of videos. Therefore, the performance of topic models will be further investigated in the following experiments.

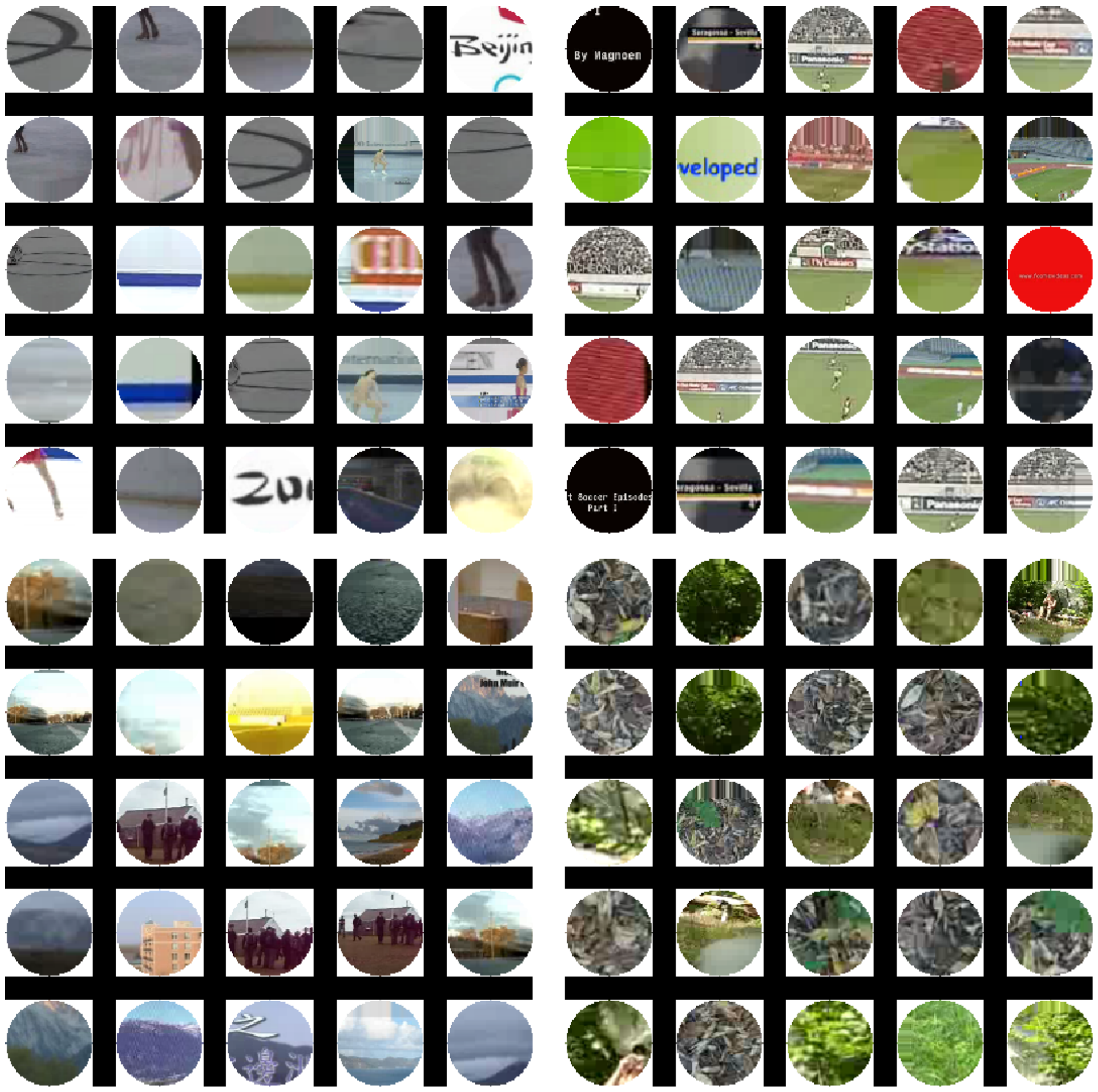

Figure 5.4: Some example topics that developed while training a PLSA on DS-1. It can be seen that the topics capture the properties of particular categories: the uppper left topic captures an *ice-skating* rink, the upper right topic the side fence of a *soccer* field. The lower left topic consists mostly of sky, and the lower right topic corresponds to brushwood, which can both be attributed to the *hiking* category.

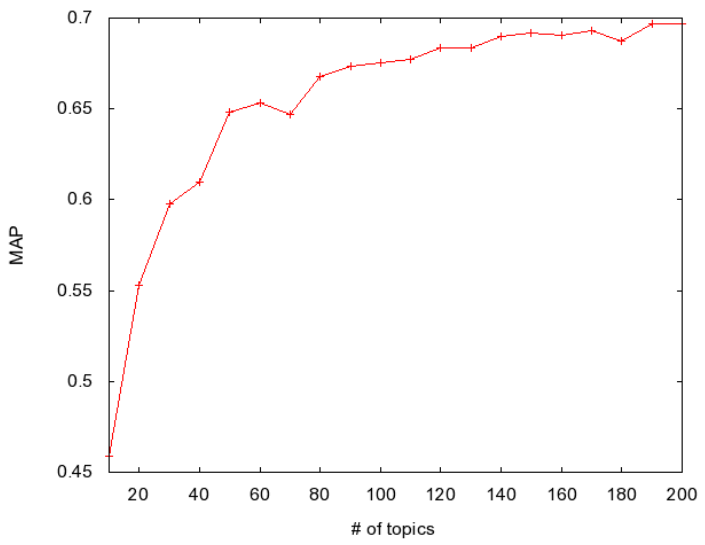

Figure 5.5: The mean average precision given different topic amounts for a PLSA model.

To get an idea of the visual nature of topics, Figure 5.4 shows some patches from sample topics that developed while training a PLSA on DS-1.

5.2.3 Different Topic Sizes

As indicated by the previous experiment, topic models seem to be very suitable for the task of semantic video compression. As the goal in semantic compression is to make the descriptor dimension as small as possible without losing too much information, it is worth to investigate the performance behaviour of topic models with varying number of topics. Thus, PLSA models were trained for DS-2 with different topic values, ranging from $10, 20, 30, ..., 200$. The resulting model was then applied to both training and test sets, and the resulting video descriptors were used as input for a classification task.

As can be seen in Figure 5.5, it is important to provide enough latent topics for the data set. If that is not the case, performance will suffer, which explains the steep curve until around 90 topics. From there on, the performance is increasing more and more slowly, with the best value in the test set being *69.7%* for 190 topics. This is less than 2% worse than the classification performance on the original, 2000-dimensional BoVW-histograms (*71.4%*) and thus very impressive considering that storage requirements are smaller by a factor of 10.5. Given target reduction factors of 20 and 40 (and thus 100 respectively 50 topics), still respectable performances of *67.5%* and *64.8%* are achieved.

These results indicate the trade-off between performance and storage requirements and hint at the minimum amount of required topics for this dataset.

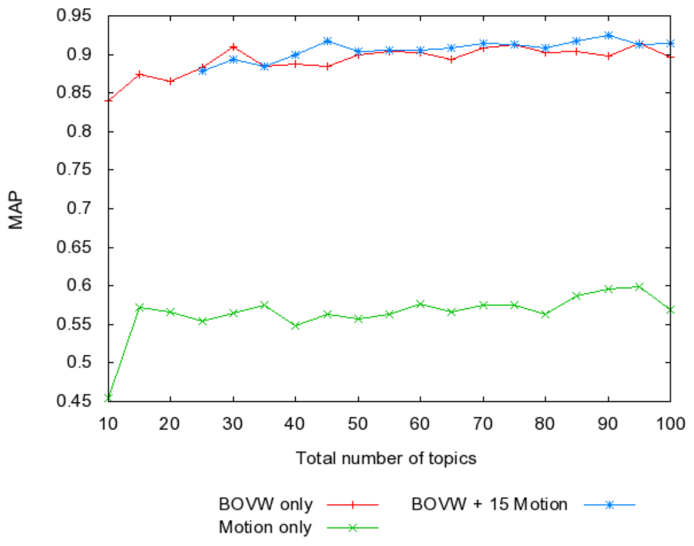

Figure 5.6: Mean average precision given different topic sizes for various features. Compared are plain bovw features, plain motion features and a fusion of both (with a fixed amount of 15 topics for the motion feature).

5.2.4 Fusing Dimensionality Reduced Features

This experiment investigates the combination of several feature descriptors in the context of topic models. As seen in Experiment 5.2.1, it is beneficial to combine different features for a performance increase. However, the problem of that approach is that the resulting feature dimensionalities will become larger and larger the more features are added. As topic models are a good way to limit dimensionality size, it is interesting to find out how features behave if they are reduced to small dimensions and then fused together, to reach a fixed feature-wide dimensionality size. Hoerster and Lienhart [32] also report that performing a late fusion on topic models instead of an early fusion gives better results, but did not investigate fixed-size target dimensionalities with different numbers of topics for each feature type.

Thus, the behaviour of both BOVW-histograms and MWH if they are reduced to $K = 10, 15, 20, \ldots, 100$ topics using PLSA is analyzed. The results from all classifiers are then merged pair-wise using late fusion, reaching up to a total of 200 topics in case of 100 topics for both PLSA-BOVW and PLSA-MWH.

An exemplary performance for a fixed number of 15 topics for the PLSA-MWH plus x topics for PLSA-BOVW can be seen in Figure 5.6. The resulting total amount of topics $(15 + x)$ is compared to pure PLSA-BOVW and PLSA-MWH with the respective number of topics. As expected from the results in Section 5.2.1, the pure PLSA-MWH performs worst. However, it can be seen that the fused model regularly outperforms the pure PLSA-BOVW, albeit only

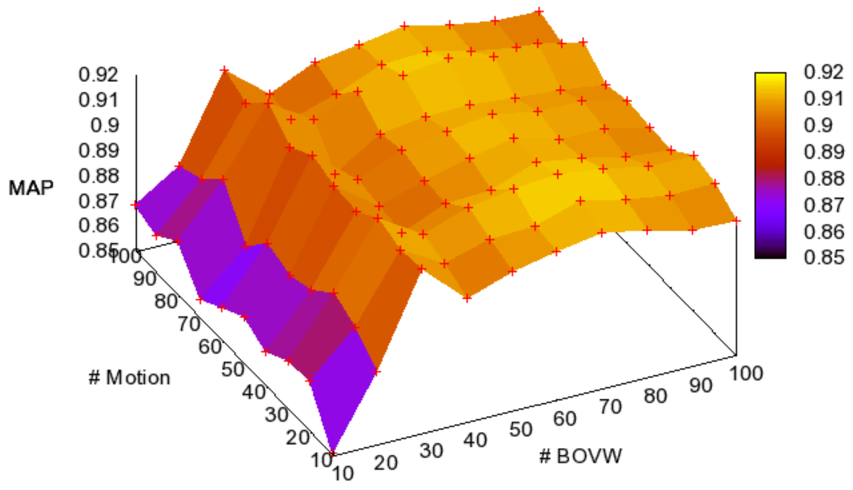

Figure 5.7: Mean average precision (Y) against the fusion of PLSA-BOVW features (X) and PLSA-MWH features (Z). The feature axes indicate the amount of topics used for a single feature.

by a small margin (the maximum absolute difference being 3.3% at a total 45 topics and 2.7% for 90 topics respectively). This is especially remarkable considering that the PLSA-MWH is not a very strong feature on its own, which leads to the conclusion that fusing two stronger independent features will give even stronger performance increases.

The complete results of the different fusions can be found in Figure 5.7. They indicate that it is generally a good idea to give more topics to the 'better' features. The purple region is a good example of this: here, only 10 topics for PLSA-BOVW are used, while the rest is added by PLSA-MWH. Note that, at this high level of performance for classification, it does not help in general to allow a lot of topics for all features – using all topics (100 for each PLSA-BOVW and PLSA-MWH) yields a performance of 90.5%, which is worse than e.g. a total of 105 (75 PLSA-BOVW and 30 PLSA-MWH) or 160 (75 + 85) topics with performances of 92.6% and 92.7% respectively.

These results indicate that assigning different topic sizes to different features is a good measure to reach a target dimensionality. It is preferable to train better features on bigger topic models, as they are likely to contribute more in the resulting fusion. Finally, given an appropriate topic distribution over the various features, the fused features are able to outperform single feature descriptors.

5.3 Using Shot Structure

The series of experiments in Section 5.2 dealt with video classification and the application of topic models to videos. This means that each video was treated as one document in the context of topic models. This section will focus on videos on a shot level, i.e. one shot represents one document. As shots are semantic units within a video, this will capture a finer content granularity.

The first experiment of this section will compare different dimensionality reduction techniques (like Experiment 5.2.2 for videos) on a shot level and include Color Layout Descriptors as an additional evaluation method. The next experiment will then discuss the impact of training on either whole videos or shots for testing on shots respectively videos. This will indicate whether it is worthwhile to spend additional computation time for training on a shot level (which is a much bigger dataset, e.g. 2500 (DS-2) against 90854 (DS-3A) features). Moreover, the benefit of using all shot information (as opposed to a single frame as shot representative) will be explored. The impact of noisy against manually annotated data is also investigated. Finally, different applications besides classification are considered: in a clustering scenario, the performance gain by utilizing the temporal structure of shots (with the Hidden Topic Markov Model) against a plain bag-of-words model will be evaluated. Also, the impact of topic models in a retrieval scenario is analyzed.

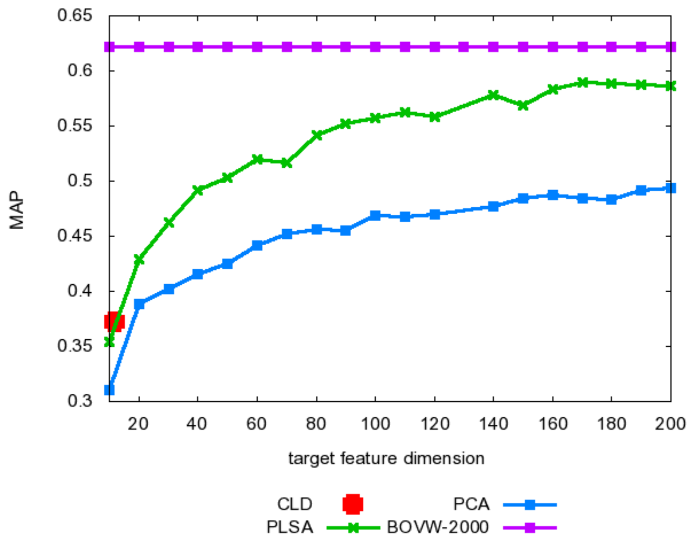

Figure 5.8: The mean average precision (MAP) given different feature dimensionalities for the control run (with fixed 2000 dimensions), principal component analysis, probabilistic latent semantic analysis and color layout descriptors.

5.3.1 Comparing Dimensionality Reduction Techniques

In a first experiment, the results from Experiment 5.2.2, where different dimensionality reduction techniques were investigated, were validated on a shot level. Aside from PCA, Color Layout Descriptors (CLD) were added as control method.

Thus, the 2000-dimensional shot vectors were reduced to $k = 10, 20, ..., 200$ dimensions using PLSA. The resulting video descriptors were then used as input for an SVM classifier, which performed concept detection on the shot test set.

Results are illustrated in Figure 5.8, where the mean average precision (MAP) is plotted against the feature dimensionality. It can be seen that PLSA comes close to the performance of the full descriptor (58.9% for 170 topics as opposed to 62.1% for the full 2000-dimensional feature). Even for 100 topics, performance is comparable to the full bag-of-visual-words representation (a relative performance loss of 10.3% occurs) at a compression rate of 1/20. While PCA is outperformed by PLSA by up to 10.0%, CLD (37.2%) seems to be roughly on par with PLSA for 12 topics and can be calculated a lot faster. However, CLD is restricted to 12 components. Also, the performance of topic models could easily be further improved by using features that also incorporate color information. For example, van de Sande et al. [74] showed that there is a significant improvement between normal SIFT and versions of SIFT where SIFT is performed seperately for each color channel or concatenated with a color histogram.

		Training	
		Video	Shots
Testing	Video	**0.648309**	0.645237
	Shots	0.625874	**0.638786**

Table 5.4: Mean average precision achieved when training on videos or shots (X-axis) and testing accordingly (Y-axis). Each result is based on PLSA-models with 50 topics.

	Shots	Frames
MAP	0.8246	0.7949

Table 5.5: Results for Experiment 5.3.3, in which single frames as representatives for shots are compared to the aggregation of shot information. The MAP of the aggregated shots is 3.0% higher than for single frames.

This experiment confirms that topic models are indeed a promising approach to effectively compress semantic information of videos.

5.3.2 Shot vs. Video Information

This experiment investigates if it is sensible to actually train on a set of shots for shot classification and respectively, on a set of videos for video classification. While a larger dataset (shots) should generally increase performance, it will also increase computation time. Moreover, it is usually sensible to train on the same kind of data for which the model will be used later.

The results in Table 5.4 support these assumptions: for shot classification, training on a shot set delivers the best results. The same holds true for videos. However, it is interesting that performance differences are so small. Thus, it might be worthwhile for shot classification to train on a video level, as it will be trained much faster but will only lead to a small performance penalty (1.3%).

5.3.3 Shots vs. Single Frames

In this experiment, the impact of using shot information (aggregating all frame histograms) of a shot against using only a single frame of a shot is analyzed. In particular, it is interesting to see if the additional processing time spent on extracting features for all frames transfers into a performance gain.

To test this, a shot boundary detection was run on the DS-1 dataset, resulting in a training set of roughly 13200 shots, and a test set of 5800 shots. For the classification using shot information, all frame histograms within that shot were summed up. For the classification using frames, only the first extracted frame of that shot was used.

$\|Shots\|$ annotated	12900 manually	90854 automatically
MAP	0.512401	0.638786

Table 5.6: Results for Experiment 5.6, in which the impact of a lot of noisy data against less, but manually annotated data is compared. The MAP of the noisy data is 12.6% higher than for the manually annotated data.

As can be seen in Table 5.5, using the whole shot information results in a performance gain of 3.0%. Thus, it is actually beneficial to use all available information for a performance gain, although this leads to longer computation time as features for all frames have to be extracted. Another interesting result in this context comes by Borth et al. [12], who compared keyframes against the first frame of a shot and reports performance gains of 2-7%.

5.3.4 Impact of Noise

The question whether manually annotated data is improving results is discussed in this experiment. To reflect this, we used the DS-3B dataset, consisting of 12900 manually annotated shots, to train a PLSA and the SVM. This is compared with the full 90854 shots of DS-3A, which are the shot boundaries that were automatically detected. Each of these shots is automatically annotated with the tag of the video, resulting in data with a lot more intra-class variance. Accordingly, the PLSA and SVM were trained on the full 90854 shots. Both trained models were then investigated on the same test set of 5830 manually annotated shots.

Thus, this experiment is supposed to show if it is worth investing additional manual labor to tag shots, or if spending this time in obtaining sheer amounts of data (roughly 7 times as much data) is preferable.

The results (c.f. Table 5.6) clearly favoring the whole, automatically annotated dataset, which has a performance gain of 12.7%. This implies that it is not worthwhile to invest in manual annotation of data. Moreover, adding more and more data is beneficial for model training, even when it might be very noisy.

5.3.5 Shot Clustering

This experiment investigates whether utilizing the temporal structure of shots as proposed in Section 4.4 gives further improvements. Both PLSA and HTMM were applied to the training set of 12,900 shots, and the reduced topic features were used to assign shots to clusters. Thereby, each topic corresponds to one cluster, and each shot D is assigned to the topic (or cluster) with maximum posterior:

$$C(D) = \underset{k}{arg}\ max\ P(Z_k|D)$$

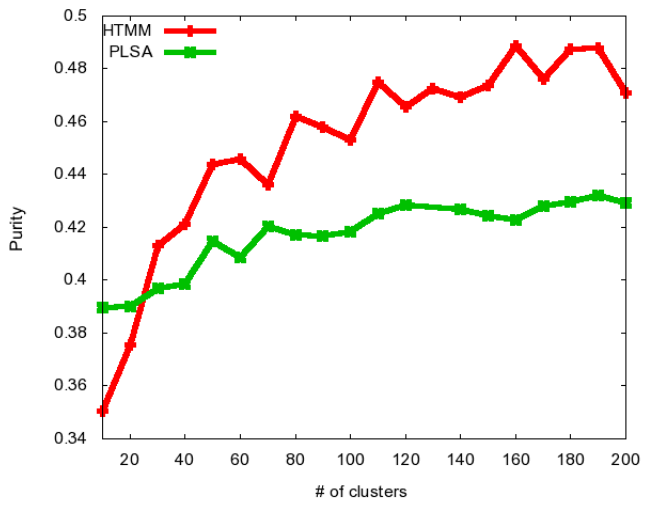

Figure 5.9: Purity for PLSA and HTMM plotted against the number of topics, which correspond to clusters.

A good clustering should assign shots from the same semantic category to the same cluster. *Purity* (c.f. Section 5.1.2) was employed as an evaluation criterion.

The performance of both HTMM and PLSA is plotted against the number of topics in Figure 5.9. It can be seen that the HTMM outperforms PLSA, which indicates that the temporal structure enforced by the HTMM indeed helps for the task of shot clustering.

5.3.6 Shot Retrieval

This experiment assesses the use of topic models in the context of shot retrieval. This is especially important for large video collections, as retrieval speed is of major importance and multiple videos have to be returned. As a dataset, the manually annotated shots (DS-3B) were used.

The results (c.f. Figure 5.10) show that results perform as expected: having only few topics (20-40) decreases performance significantly (i.e., of the first 5 results for 20 topics, only 38.8% were of the same category, opposed to 44.7% for 200 topics), while results tend to vary only slightly once enough topics are reached (80-200).

This experiment also indicates the usefulness of topic models for content-based video retrieval, and validates the previous observation that a minimum lower bound for topics has to be reached, since performance will otherwise suffer.

Image Retrieval Note that there is a fundamental flaw with the ground truth for the shot retrieval experiment: while each shot was manually annotated to

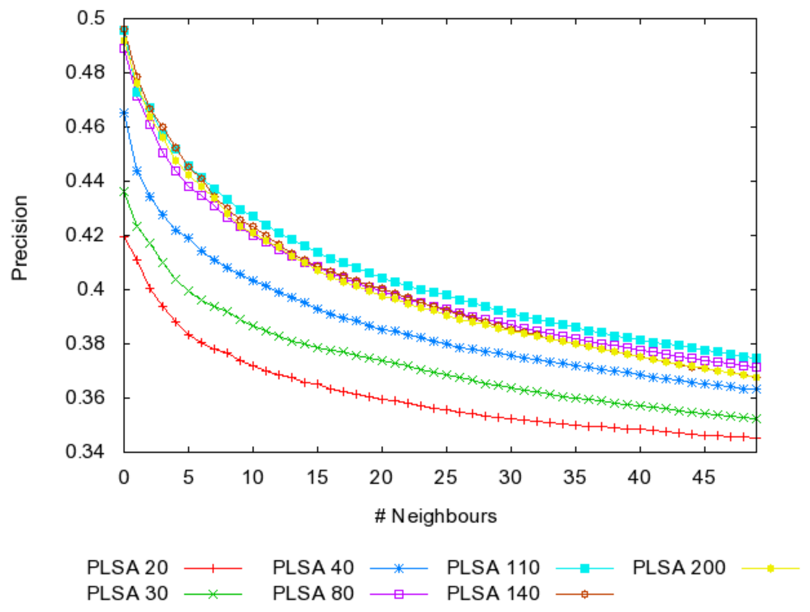

Figure 5.10: Achieved precision given different amounts of retrieved results. It is clear that a minimum number of topics for a PLSA model is necessary, as performance will suffer otherwise. In this case, using less than 80 topics has bad effects on performance.

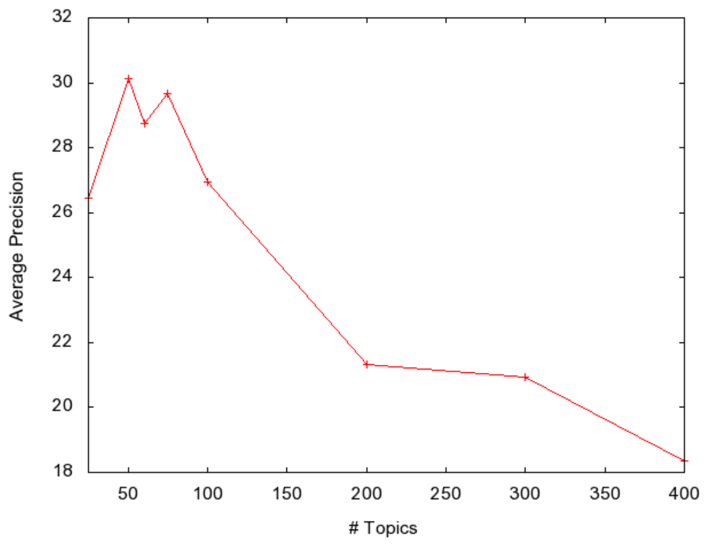

Figure 5.11: Average precision given different topic values in an image retrieval task (c.f. Experiment 5.3.6). The performance peaks at roughly 50 topics, with a large drop around 200 topics, likely due to overfitting.

belong to a particular category, this does not provide any further insight into how close those shots are related. For example, both frames of a ball and a referee can be associated with the category soccer, even though these two are completely unrelated – however, the annotation is performed on the concept level of 'soccer'.

Thus, an additional retrieval experiment was performed in the image domain, where better ground truth was available. Specifically, the INRIA holidays dataset [38] with 1488 images were used. This dataset specifies in detail how many and which documents should be returned for which query images in which order and can thus be employed for a far more detailed evaluation.

For the experiment, SIFT features [47] were extracted for all images, and a codebook with 10000 entries was generated using the extracted patches. 500 of the images where then used for evaluation purposes (using the L_1 distance), and measured using average precision.

The results (c.f. Figure 5.11) indicate that a best performance is reached at roughly 50 topics, whereas performance seems to degrade at around 200 topics. A possible explanation for this is the issue of overfitting, as both the number of documents (1488 images) are very large and the number of words per document is very sparse (on average, 3000 patches occur in an image, which are mapped to a codebook with 10000 entries). Note that a control run with the full, 10000-dimensional bag-of-visual-words histogram yields an average precision of 40.66%. Thus, using topic models incurs a performance loss of roughly 10.5% in this case.

Chapter 6

Conclusion

6.1 Summary of Methods and Results

In this work, topic models were used for semantic compression of videos and applied to the tasks of video and shot classification and clustering.

Generally, topic models are a way of decomposing a set of documents with K latent variables called topics, resulting in a topic mixture for each document and a word mixture for each topic. This means that a document can simply be described through K topic coefficients – resulting in a reduced description of the document if the number of topics is limited enough.

The experiments conducted in this work lead to several conclusions:

- Different methods for feature dimensionality reduction like PCA and RBM were analyzed and it was shown that topic models (represented by PLSA) are the best choice for this task.

- Fusion of different visual and motion-related features was explored in the context of target compression ratios, and found to be beneficial for overall performance.

- More data, even though it might be more noisy, is beneficial during training of a topic model.

- Representing shots through the aggregation of all frames of that shot is better than the representation by a single frame.

- In addition, the temporal structure between shots was exploited with the HTMM and showed superior performance in a clustering scenario compared to plain PLSA.

To summarize these results, it can be stated that topic models are indeed a good approach for semantic video compression.

6.2 Future Work

There are several areas of interest that might be worth to pursue further related to this thesis. For instance, it might be worth to adapt the Hidden Topic Markov Model to allow it to model each sentence as a topic mixture instead of a single topic, as this would greatly enhance its usability for tasks like classification or similarity search. Another interesting path would be to adapt more topic models, such as Hierarchical Dirichlet Processes [71], to video content, and investigate their performance. Analyzing video-specific features and their performance with topic models also seems like an interesting task – the spatial-temporal features introduced by Niebles et al. [55] are a first step in this direction. Lastly, online learning in the scope of topic models is a task which seems worth investigation: websites like YouTube with roughly 60,000 new videos per day are not capable of training completely new topic models with an ever increasing amount of data – aside from the storage requirements, training of a model would only be complete when the model has already become outdated again. Thus, adaption of existing models to incorporate new data seems like an interesting prospective research area.

Bibliography

[1] P. Aigrain, H. Zhang, and D. Petkovic. Content-based representation and retrieval of visual media: A state-of-the-art review, 1996.

[2] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[3] K. Barnard, P. Duygulu, D. Forsyth, N. D. Freitas, D. M. Blei, J. K, T. Hofmann, T. Poggio, and J. Shawe-taylor. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

[4] K. Barnard and D. A. Forsyth. Learning the Semantics of Words and Pictures. In *Proc. Int. Conf. Computer Vision*, pages 408–415, July 2001.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[6] S. Beauchemin and J. Barron. The Computation of Optical Flow. *ACM Computing Surveys*, 27(3):433–467, 1996.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, April 2002.

[8] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[9] J. L. Bentley. K-d trees for semidynamic point sets. In *SCG '90: Proceedings of the sixth annual symposium on Computational geometry*, pages 187–197, New York, NY, USA, 1990. ACM.

[10] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[11] M. Bober. Mpeg-7 visual shape descriptors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):716–719, Jun 2001.

[12] D. Borth, A. Ulges, C. Schulze, and T. Breuel. Keyframe extraction for video tagging & summarization. In *Informatiktage*, volume S-6 of *LNI*, pages 45–48. GI, 2008.

[13] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, November 1986.

[14] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):394–410, 2007.

[15] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[16] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60, 2008.

[17] A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[18] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[19] C. Elkan. Using the Triangle Inequality to Accelerate KMeans. In *Proc. Int. Conf. Machine Learning*, pages 147–153, August 2003.

[20] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google"s image search. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1816–1823, Washington, DC, USA, 2005. IEEE Computer Society.

[21] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *In CVPR*, pages 264–271, 2003.

[22] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[23] D. Gabor. Theory of communication. *JIEE*, 93(3):429–459, 1946.

[24] A. Gruber, M. Rosen-Zvi, and Y. Weiss. Hidden Topic Markov Models. In *Artificial Intelligence and Statistics (AISTATS)*, March 2007.

[25] S. R. Gunn. Support vector machines for classification and regression. Technical report, Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, May 1998.

[26] A. Hanjalic, R. Lienhart, W. Y. Ma, and J. 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.

[27] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[28] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006.

[29] T. Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, 42:177–196, 2001.

[30] L. Hohl, F. Souvannavong, B. Mérialdo, and B. Huet. Enhancing latent semantic analysis video object retrieval with structural information. In *ICIP*, pages 1609–1612, 2004.

[31] E. Hörster, T. Greif, R. Lienhart, and M. Slaney. Comparing local feature descriptors in plsa-based image models. In *Proceedings of the 30th DAGM symposium on Pattern Recognition*, pages 446–455, Berlin, Heidelberg, 2008. Springer-Verlag.

[32] E. Horster and R. Lienhart. Fusing local image descriptors for large-scale image retrieval. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[33] E. Hörster and R. Lienhart. Deep networks for image retrieval on large-scale databases. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 643–646, New York, NY, USA, 2008. ACM.

[34] E. Hörster, R. Lienhart, and M. Slaney. Image retrieval on large-scale image databases. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 17–24, New York, NY, USA, 2007. ACM.

[35] E. Hörster, R. Lienhart, and M. Slaney. Continuous visual vocabulary modelsfor plsa-based scene recognition. In *CIVR '08: Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 319–328, New York, NY, USA, 2008. ACM.

[36] M.-K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IEEE Transactions on*, 8(2):179–187, 1962.

[37] M. Irani and P. Anandan. About direct methods. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 267–277, London, UK, 2000. Springer-Verlag.

[38] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In A. Z. David Forsyth, Philip Torr, editor, *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317. Springer, oct 2008.

[39] E. Kasutani and A. Yamada. The mpeg-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 674–677 vol.1, 2001.

[40] Y. Ke and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:506–513, 2004.

[41] R. Lienhart. Comparison of automatic shot boundary detection algorithms. In *Storage and Retrieval for Image and Video Databases*, number SPIE 3656, pages 290–301, January 1999.

[42] R. Lienhart. Reliable transition detection in videos: A survey and practitioner's guide. *International Journal of Image and Graphics (IJIG)*, pages 469–486, 2001.

[43] R. W. Lienhart. Comparison of automatic shot boundary detection algorithms. In M. M. Yeung, B.-L. Yeo, and C. A. Bouman, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 3656 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 290–301, Dec. 1998.

[44] D. Liu and T. Chen. Unsupervised image categorization and object localization using topic models and correspondences between images. *Computer Vision, IEEE International Conference on*, 0:1–7, 2007.

[45] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982.

[46] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31:983–1001, 1998.

[47] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, 2004.

[48] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction, 1999.

[49] B. S. Manjunath, J. rainer Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11:703–715, 2001.

[50] D. Marr and E. Hildreth. Theory of Edge Detection. *Royal Society of London Proceedings Series B*, 207:187–217, Feb. 1980.

[51] B. McCane, K. Novins, D. Crannitch, and B. Galvin. On benchmarking optical flow. *Comput. Vis. Image Underst.*, 84(1):126–143, 2001.

[52] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.

[53] A. Ming and H. Ma. A blob detector in color images. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 364–370, New York, NY, USA, 2007. ACM.

[54] F. Monay and D. Gatica-Perez. Plsa-based image auto-annotation: constraining the latent space. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 348–351, New York, NY, USA, 2004. ACM.

[55] J. C. Niebles, H. Wang, and L. Fei-fei. Unsupervised learning of human action categories using spatial-temporal words. In *In Proc. BMVC*, 2006.

[56] R. Paredes, J. C. Pérez, A. Juan, and E. Vidal. Local representations and a direct voting scheme for face recognition. In *In Workshop on Pattern Recognition in Information Systems*, pages 71–79, 2001.

[57] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[58] P. Quelhas, F. Monay, J. Odobez, D. Gatica-Perez, and T. Tuytelaars. A Thousand Words in a Scene. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(9):1575–1589, 2007.

[59] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[60] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *In Machine Learning, Proceedings of the Twenty-fourth International Conference (ICML 2004). ACM*, pages 21–24. AAAI Press, 2007.

[61] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[62] J. Shlens. A tutorial on principal component analysis, 2005.

[63] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2005.

[64] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proc. Int. Conf. Computer Vision*, pages 1470–1477, Oct. 2003.

[65] C. Snoek, M. Worring, O. de Rooij, K. van de Sande, R. Yan, and A. Hauptmann. VideOlympics: Real-Time Evaluation of Multimedia Retrieval Systems. *IEEE MultiMedia*, 15(1), 2008.

[66] C. G. M. Snoek, M. Worring, J. C. V. Gemert, J. mark Geusebroek, and A. W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *In Proceedings of the ACM International Conference on Multimedia*, pages 421–430. ACM Press, 2006.

[67] F. Souvannavong, L. Hohl, B. Merialdo, and B. Huet. Structurally Enhanced Latent Semantic Analysis for video object retrieval. *IEE Proceedings on Vision, Image and Signal Processing, Volume 152, N°6, 9 December 2005*, 2005.

[68] F. Souvannavong, B. Mérialdo, and B. Huet. Latent semantic analysis for an effective region-based video shot retrieval system. In *Multimedia Information Retrieval*, pages 243–250, 2004.

[69] F. Souvannavong, B. Mérialdo, and B. Huet. Latent semantic indexing for semantic content detection of video shots. In *ICME*, pages 1783–1786, 2004.

[70] H. Tamura, S. Mori, and T. Yamawaki. Textural Features Corresponding to Visual Perception. *IEEE Trans. System, Man, Cybernetics*, 8(6):460–472, 1978.

[71] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[72] P. Tirilly, V. Claveau, and P. Gros. Language modeling for bag-of-visual words image categorization. In *CIVR '08: Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 249–258, New York, NY, USA, 2008. ACM.

[73] A. Vailaya, M. Figueiredo, A. Jain, and H.-J. Zhang. Image classification for content-based indexing. *Image Processing, IEEE Transactions on*, 10(1):117–130, Jan 2001.

[74] K. E. van de Sande, T. Gevers, and C. G. Snoek. A comparison of color features for visual concept classification. In *CIVR '08: Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 141–150, New York, NY, USA, 2008. ACM.

[75] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision - to appear*, # 2002.

[76] J. Vogel and B. Schiele. Natural scene retrieval based on a semantic modeling step. In *In CIVR*. Springer Verlag, 2004.

[77] J. Willamowski, D. Arregui, G. Csurka, C. R. Dance, and L. Fan. Categorizing nine visual classes using local appearance descriptors. In *In ICPR Workshop on Learning for Adaptable Visual Systems*, 2004.

[78] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *MIR '07: Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 197–206, New York, NY, USA, 2007. ACM.

[79] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis, 2001.

[80] T. Zito, N. Wilbert, L. Wiskott, and P. Berkes. Modular toolkit for data processing (mdp): A python data processing framework. *Frontiers in neuroinformatics*, 2, 2008.