Coloring Concept Detection in Video using Interest Regions

Koen Erik Adriaan van de Sande

Coloring Concept Detection in Video using Interest Regions

Doctoraal Thesis Computer Science

specialization: **Multimedia and Intelligent Systems**

Koen Erik Adriaan van de Sande

ksande@science.uva.nl

Under supervision of:

Prof. Dr. Theo Gevers

and

Dr. Cees G.M. Snoek

March 12, 2007

UNIVERSITEIT VAN AMSTERDAM

Abstract

Video concept detection aims to detect high-level semantic information present in video. State-of-the-art systems are based on visual features and use machine learning to build concept detectors from annotated examples. The choice of features and machine learning algorithms is of great influence on the accuracy of the concept detector. So far, intensity-based SIFT features based on interest regions have been applied with great success in image retrieval. Features based on interest regions, also known as local features, consist of an interest region detector and a region descriptor. In contrast to using intensity information only, we will extend both interest region detection and region description with color information in this thesis. We hypothesize that automated concept detection using interest region features benefits from the addition of color information. Our experiments, using the Mediamill Challenge benchmark, show that the combination of intensity features with color features improves significantly over intensity features alone.

Contents

1 Introduction

In the digital age we live in, more and more video data becomes available. Television channels alone create thousands of hours of new video content each day. Due to the sheer volume of data, it is impossible for humans to get a good overview of the data available. Retrieving specific video fragments inside a video archive is only possible if one already knows where to search for or if textual descriptions obtained from speech transcripts or social tagging are available. Video retrieval can be applied to video collections on the internet, such as YouTube [53] and Yahoo Video [52], to news video archives, to feature film archives, etc. Currently, the search capabilities offered by these video collections are all based on text. The addition of text by humans is very time consuming, error prone and costly and the quality of automatic speech recognition is poor. Contrary to the text modality, the visual modality of video is always available.

Unfortunately the visual interpretation skills of computers are poor when compared to humans. However, computers are experts in handling large amounts of binary data. Computers can decode video streams into individual video frames and compute statistics over them, but there is a big gap between the machine description 'this frame contains a lot of green' and the soccer match that humans see. To associate high level semantic concepts such as *soccer* to video data, we need to bridge the *semantic gap*. The semantic gap has been defined by Smeulders *et al* [38] as follows:

"The semantic gap is the lack of coincidence between the information that machines can extract from the visual data and the interpretation that the same data have for a user in a given situation."

State-of-the-art systems in both image retrieval [48, 18, 54] and video retrieval [33] use machine learning to bridge the gap between features extracted from visual data and semantic concepts. Semantic concepts are high-level labels of the video content. Systems learn semantic concept detectors from features. The accuracy of the concept detector depends on the feature used, the machine learning algorithm and the number of examples. The feature is one of the building blocks of a concept detector and its choice is very important for performance.

In concept-based image retrieval, SIFT features [21] based on interest regions are state-of-the-art [54]. These features consist of an interest region detector and a region descriptor. However, both components operate on intensity information only: they completely discard the color information present. The question arises why color information is neglected. We hypothesize:

Automated concept detection using interest region features benefits from the addition of color information.

Our goal is to extend both interest region detection and interest region description with color. We evaluate the SIFT region descriptor and color region descriptors. Criteria for selecting the descriptors include invariance against variations in light intensity, illumination color, view geometry, shadows and shading. If the descriptor used is invariant to a condition, then the concept can be detected even if the condition changes. We compute the descriptors over either entire video frames or over interest regions. The former are called global features, while the latter are local features. Depending on the region over which we compute descriptions, we can achieve invariance against scale changes, rotation or object position changes. For example, when the position of an object in the scene changes, the task of learning a concept detector is simplified if the feature vector remains similar. For local features, we have both an intensity interest region detector and a color-extended interest region detector. Furthermore, we investigate how to best aggregate the descriptors of many interest regions into a single feature vector. Finally, we investigate whether combinations of features provide benefits over individual features. We perform our large-scale evaluation of visual features using 85 hours of

video from the Mediamill Challenge [42], a benchmark for automated semantic concept detection.

The organization of this thesis is as follows. In chapter 2, we give an overview of related work. In chapter 3, we provide the necessary background for our generic concept detection framework, discussed in chapter 4. In chapter 5, we provide implementation details of our concept detection system. In chapter 6, we describe our experimental setup. In chapter 7, we show the results of our system. Finally, in chapter 8, we draw conclusions and provide directions for future research.

2 Related work

In this chapter, we will discuss research related to automatic semantic concept detection. Concept detection has already been extensively studied in the field of Content-Based Image Retrieval (CBIR), where concepts are referred to as categories. In section 2.1, we discuss the datasets and methods used in CBIR. In section 2.2, we compare this to work in the video retrieval field. Finally, in section 2.3, we conclude with the rationale for our work on concept detection in video.

2.1 Content-Based Image Retrieval

The field of Content-Based Image Retrieval studies, amongst others, the application of computer vision in the context of image retrieval. Then, the image retrieval problem corresponds to the problem of searching for digital images in datasets. Because this thesis focuses on the detection of concepts, other important issues of CBIR, such as visualization, image databases, etc. are beyond the scope of this thesis and will not be discussed here.

Content-Based Image Retrieval systems tend to follow a common scheme. First, a *description* of every image in the dataset is computed. A description is a concise representation of the image contents, typically taking the form of a numeric feature vector. Second, these descriptions are indexed to speed up the search process. Third, the description of an unseen image is computed. Unseen images can be assigned to a category depending on which categories similar descriptions come from.

We perceive the task of assigning a category to an unindexed image as a machine learning problem: Given a number of feature vectors (descriptions of images) with corresponding class labels (categories), the aim is to learn a classifier function which estimates the category of unseen feature vectors.

One of the most important properties of image descriptions for robust retrieval is invariance to changes in imaging conditions. For example, after rotation or scaling of the object in view, the image of the object should still be retrieved. Moreover, more complex imaging transformations such as object position and viewpoint changes should be handled. Other possible changes are photometric changes (light intensity, color of the light source, etc) and object shape changes. Different degrees of invariance can be achieved for variations in imaging conditions. However, increased invariance comes at the cost of reduced discriminability [12]. For example, for an image description which is invariant for the color of the light source it will be difficult to distinguish between a room with a white light source and a room with a red light source.

Because we perceive CBIR as a machine learning task, the classification scheme only applies when annotations for the original dataset are available. Methods which do not require annotations are unsupervised methods. These methods find clusters of similar items in a dataset. However, further human interpretation of these clusters is needed, because they do not assign semantic labels to the images. For example, the Query-by-Example retrieval method finds items similar to given examples, but these items need to be judged by a human. In combination with visualization tools, unsupervised methods are very useful for analysis of the nature of image descriptions. Depending on the description used, different images will be clustered together.

2.1.1 Datasets

To evaluate results in the CBIR field, many image datasets have been generated. Because the focus is on concept detection, only datasets are discussed which divide the images into different concept categories. In table 1 several datasets and their properties are listed.

Name	# images	# concepts	Properties
Xerox7 [51]	1,776	7	highly variable pose and background clutter, large intra-class variability
CalTech6 [8]	4,425	6	background set without concepts available, same pose within an object class, object prominently visible near image center
PASCAL dataset [5]	2,239	4	object location annotations available, multiple instances per image allowed
CalTech101 [7]	8,677	101	little or no clutter, objects tend to be centered and have similar pose
Corel photo stock (commercial)	16,499	89	high quality images, variable poses, low background clutter
Amsterdam Library of Object Images (ALOI) [11]	110,250	1000	objects photographed in different poses under varying lighting conditions, objects are centered, black background

Table 1: Concept-based datasets from content-based image retrieval.

The image datasets listed, except the Corel and ALOI, consist of only several thousands of images. Compared to the sizes of real-world image collections, they are relatively small. In fact, the datasets have been designed to evaluate 'desired' invariance properties of new descriptors. Because of this, some of the datasets impose certain conditions on the images (objects in similar pose, no background, object is centered, professional photography, etc). In real-world image datasets, these conditions are not always met.

Due to the high effort involved in the construction of datasets, automatic construction of datasets from image search engine results has been investigated [9, 37]. The name of a concept is used as a search query to obtain images for that concept. However, the datasets constructed using these methods contain many false positives.

2.1.2 Image indexing

Image descriptions can be created from various parts of the image. Global descriptions are created over the entire image. Partition descriptions divide the image into a (non-)overlapping grid of rectangular windows. Interest region descriptions (also known as *local* descriptions) are computed over the area surrounding salient points (such as corners). Region-based descriptions segment the image into several blobs and describes those blobs. Random window descriptions randomly select rectangular windows for description. We will discuss the various strategies in this section.

Global descriptions Examples of global descriptors of images are intensity histograms, (color) edge histograms and texture descriptors. Most global descriptions, histograms for example, do not retain positional information. This makes it hard to describe images with significant differences between the different parts of the image. There are global descriptions which do capture positional information, such as color moment descriptors [29]. This has the disadvantage that the descriptions are no longer invariant to geometrical changes.

To use descriptors for retrieval, similarity between descriptions needs to be defined. In general, the Euclidian distance is used. However this distance may not correspond to human perceptions of descriptor differences. The Mahalanobis distance takes into account the correlations within the data set, which the Euclidian distance does not. However, to be able to apply the Mahalanobis distance, the covariance matrix of the whole dataset needs to be estimated. This is expensive for high-dimensional descriptors. Another distance is the histogram intersection. Histogram intersection compares corresponding bins in two histograms and takes the minimal value of the two bins. The normalized sum over all bins forms the distance. In histogram intersection, the bins

with many samples contribute most to the distance. It is fairly stable against changes in image resolution and histogram size. However, it is limited to histograms only. Another method is the Kullback-Leibler divergence from information theory, which can be used for comparison of distributions and thus histograms. However, the Kullback-Leibler divergence is non-symmetric and is sensitive to histogram binning, limiting its practical use. The Earth Movers Distance [36] also compares distributions. The distance reflects the minimal amount of work needed to transform one distribution into the other by moving 'distribution mass' around. It is a special case of a linear optimization problem. An advantage of Earth Movers Distance is that it can compare variable-length features. However, its computation is very expensive when compared to the other distances.

We will use the Euclidian distance for descriptor comparison, because it is fast to compute and can be applied to all numerical descriptors.

Partition descriptions Partition descriptions divide the image into a (non-)overlapping grid. Every window can be described as if it is a separate image, retaining some positional information. The descriptions can be combined into one large description which is more fine-grained than the global one.

Van Gemert [48] computes Weibull-based features over every window of a grid partitioning of the image. Weibull-based features model the histogram of a Gaussian derivative filter by using a Weibull distribution. The parameters of this distribution are used as features. The histogram of a Gaussian derivative filter represents the edge statistics of an image.

Lazebnik [18] repeatedly subdivides the image into partitions and computes histograms at increasingly fine resolutions. These so-called spatial pyramids achieve performance close to the interest region methods we discuss next.

Local descriptions Most of the state-of-the-art in CBIR descriptions is based on interest regions[1] [54]. These are regions which can be detected under image changes such as scaling and rotation with high repeatability. For every interest region in an image, a description is created. Because the region is detectable after a translation, the description will be translation invariant if it does not contain position information. For scale changes, the detected region will be covariant with the scale changes, thus the same region will be described, only at a different scale. If the description is normalized for the region size and does not contain position information, then it is scale invariant. Rotation invariance can be achieved by using either a description which does not contain position information, or by using a description which is aligned to a dominant orientation of the region. For the latter, robust selection of this dominant orientation is needed. Descriptions based on interest regions are commonly referred to as *local features*, to offset them against *global features* which are computed over the entire image.

In an evaluation of interest region detectors for image matching, Mikolajczyk *et al* [24] found that the Harris-Affine detector performs best. The purpose of the evaluation is to determine which kind of interest regions are stable under viewpoint changes, scale changes, illumination changes, JPEG compression and image blur. The repeatability of interest region detection under these changing conditions is measured. The dataset used consists of 48 images, with only one condition change per image.

Zhang [54] obtains best results using the Harris-Laplace interest region detector, noting that affine invariance is often unstable in the presence of large affine or perspective distortions. The evaluation by Zhang uses several texture datasets and the Xerox7 dataset to draw this conclusion.

[1] Interest regions are also referred to as salient regions and/or scale invariant keypoints in literature.

The SIFT descriptor [21] is consistently among the best performing interest region descriptors [54, 28]. SIFT describes the local shape of the interest region using edge orientation histograms. SIFT operates solely on intensity images, ignoring color information present in the image. To address this problem, Van de Weijer [47] proposes an additional hue histogram on top of SIFT. However, no evaluation of this descriptor on similar datasets is currently available. We will include this descriptor in our experiments.

Descriptors of interest regions cannot be used as feature vectors directly, because they vary in length. Zhang [54] addresses this by using the Earth Movers Distance, discussed earlier. This distance supports comparison of variable-length descriptors, but is computationally expensive. For the CalTech101 dataset this is still feasible, but for larger datasets the computation time becomes an issue.

Most other methods are based on a visual codebook consisting of a number of representative region descriptors. A feature vector then consists of the similarity between the visual codebook and the image descriptors. The most common method is to construct a histogram where descriptors are assigned to the 'bin' of the most similar codebook descriptor. These histograms have a fixed length equal to the codebook size. The advantage of visual codebook methods is that it does not require the expensive Earth Movers Distance for comparison. A disadvantage of a visual codebook is that it is a *bag-of-features* method which does not retain any spatial relationships between the different descriptors it was constructed from.

Region-based descriptions The primary example of a region-based description is the Blobworld representation by Carson [4]. This method groups pixels with similar colors into a number of blobs. The number of blobs used depends on the image, but is constrained to lie between 2 and 5. Retrieval in Blobworld is performed using a matching strategy: the blobs in an image are compared to all blobs in the dataset using the Mahalanobis distance. The advantage of the Blobworld representation is that it works well for distinctive objects. However, there are many parameters to be tuned during blob extraction and the classification step is computationally expensive.

Random window descriptions Marée [22] questions the use of both interest regions and image partitions. Instead they propose the use of random subwindows. Descriptions of these rectangular windows together form the description of the image. While the approach is generic, the windows used are only invariant to very small rotations.

The use of a decision tree for classification of new images supports dataset sizes of up to several thousand images. However, for this method the number subwindows per image is more than two orders of magnitude higher than other approaches. It is not feasible to perform experiments using this method on a large dataset.

In conclusion, the CBIR field has extensively studied image descriptions and their properties. However, evaluation of concept-based retrieval is performed on datasets of an artificial nature or with a limited size.

2.2 Video retrieval

Where commonly accepted benchmarks are practically non-existent in content-based image retrieval, the field of video retrieval has the TREC Video Retrieval Evaluation (TRECVID) [32] benchmark. TRECVID started initially as a 'video' track inside the TREC (Text Retrieval Evaluation Conference) in 2001. Since 2003, TRECVID is an independent evaluation. The main goal of the TRECVID is to promote progress in content-based retrieval from digital video via open, metrics-based evaluation. TRECVID

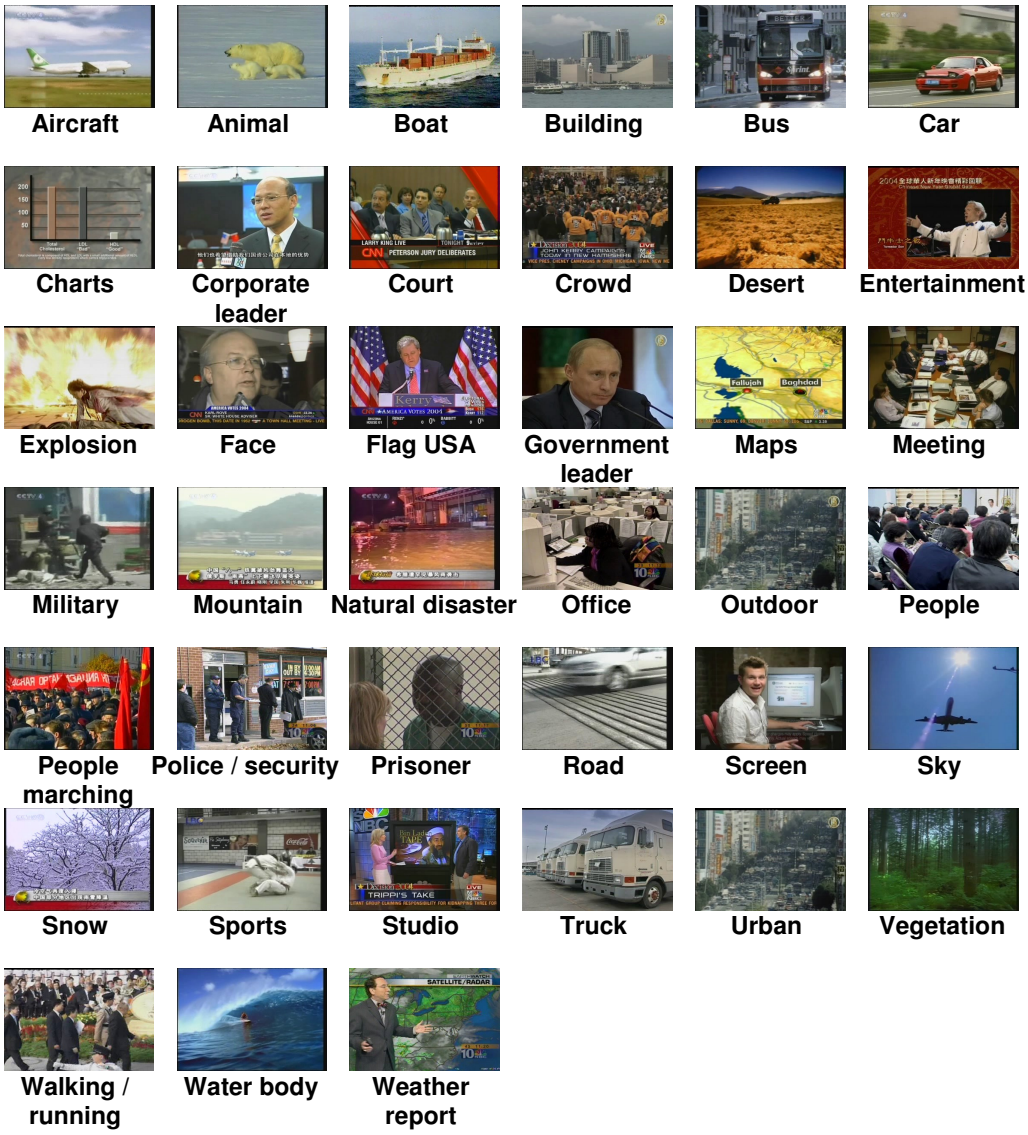


Figure 1: Semantic concepts used in this thesis. These are the same concepts as those in the TRECVID 2006 high-level feature extraction task [32].

has a high-level feature extraction task. This task benchmarks the effectiveness of automatic detection methods for *semantic concepts*. A semantic concept is a high-level label of the video content. These semantic concepts can be compared to categories in the CBIR field. The semantic concepts evaluated in TRECVID 2006 are shown in figure 1.

2.2.1 Datasets

The focus of the video retrieval field is on generic methods for concept detection and large-scale evaluation of retrieval systems. In table 2, an overview of video retrieval datasets is given. The 10 semantic concepts used in the 2005 edition of TRECVID are: sports, car, map, building, explosion, people walking, waterfront, mountain and prisoner. In 2005, the system of Snoek [39] stood out for its support of 101 semantic concepts. In 2006, the 39 semantic concepts listed in figure 1 were evaluated. Furthermore, LSCOM is developing an expanded lexicon in the order of 1000 concepts.

The TRECVID benchmark only provides a common dataset and a yearly evaluation. The evaluation is performed by humans verifying whether the top ranked results are correct. Annotating the entire test set is not feasible due to limited human resources. Performing a new experiment which ranks as of yet unseen items at the top will thus require extra annotation effort. The MediaMill Challenge problem [42] for automated detection of 101 semantic concepts has been defined over the training data from TRECVID 2005. This data has annotations for both the training and test set for 101 semantic concepts, thus providing a basis for repeatable experiments. The challenge defines unimodal experiments for the visual and text modality and several experiments for fusion of different modalities. In figure 2 an impression of the TRECVID 2005 dataset is given.

Automatic semantic concept detection experiments divide a dataset, annotated with a ground truth, into a training set and a test set. Concept detectors are built using only the training data. The detectors are applied to all video shots in the test set. Using the detector output, the shots are ordered by the likelihood a semantic concept is present. The quality of this ranking is evaluated.

2.2.2 Video indexing

For image retrieval, it is obvious to use individual images as the fundamental unit for indexing. For video retrieval, the fundamental unit to index is not immediately obvious. One could index every single video frame, but this requires huge amounts of processing and is not necessarily useful. After all, retrieving two adjacent frames of a single video is not an useful retrieval result, as these are likely to be almost the same. It would be

Name	**# shots**	**# concepts**	**Properties**
TRECVID 2002	9,739	10	from Internet Archive [16], amateur films, feature films 1939-1963
TRECVID 2003	67,385	17	CNN/ABC news broadcasts from 1998
TRECVID 2004	65,685	10	CNN/ABC news broadcasts from 1998, partial overlap with TRECVID 2003
TRECVID 2005	89,672	10	English, Chinese and Arabic news broadcasts from November 2004
Mediamill Challenge [42]	43,907	101	subset of TRECVID 2005 dataset, with extra concepts
TRECVID 2006	169,156	39	includes complete TRECVID 2005 dataset

Table 2: Overview of video retrieval datasets.

Figure 2: Impression of the TRECVID 2005 training set. It contains videos from Arabic, Chinese and English news channels. On news channels one can expect interviews, speeches, stock market information, commercials, soap operas, sports, etc.

more useful to aggregate parts of the video into larger units. This aggregation does not have to be based on entire frames per se. As in the CBIR field, it is possible to have a finer granularity than an entire frame.

In general, camera shots are currently used as the fundamental units of video for higher-level processing. In a video stream, boundaries between shots are defined at locations where the author of the video makes a cut or transition to another scene or a different camera and/or a jump in time. Fast shot boundary detectors with high accuracy are available [32].

Given a shot segmentation of a video, the subsequent problem is the description of these shots. The majority of current methods for extraction of visual descriptions uses only a single keyframe which is representative for the entire shot. However, this can be done under the assumption that other frames of the video will have similar content as the keyframe. For action-packed shots, this assumption will not hold. The description of the shot will then be incomplete. However, with the current state of technology, the reduction in computational effort is more important. With a minimum shot duration of two seconds and 25 frames per second, there is a fifty-fold reduction in processing time (at a minimum).

In TRECVID 2005, the best performance on semantic concept detection for 7 concepts has been achieved [33] using multiple global descriptions of color, shape, texture and video motion. All visual descriptions, except for video motion, were performed on the shot keyframe level. Text features based on a speech recognition transcription of the video have also been used. While the contribution of individual features is undocumented, for semantic concepts with strong visual cues, such as explosion, building and mountain, the combination of features outperformed all other systems. The combination of features is commonly referred to as fusion.

Naphade [31] shows that multi-granular description of shot keyframes improves concept detection over global description, similar to the results in Content-Based Image Retrieval. Specifically, partitioning of keyframes with and without overlap and a rudimentary form of interest regions are investigated.

In the video retrieval field, the focus lies on features which can be applied in a feasible processing time. Features from the CBIR field are applied, investigating whether the invariance properties improve retrieval results on extensive video datasets.

2.3 Conclusions

State-of-the-art in image descriptors such as the SIFT features for image retrieval do not exploit the color information in an image: they operate on intensity information only. Therefore, we will extend interest region detection and region description to include color information. We evaluate our methods on concept detection in video, exploiting the availability of large real-world datasets in the video retrieval field. To allow for experimentation on a large dataset, we need fast feature extraction methods. Finally, we will study the relative importance of using interest region detection over other image areas by offsetting interest region description against global description.

3 Coloring concept detection using interest regions

In this chapter, we present the background information necessary for understanding our concept detection system. In section 3.1, we introduce various digital image processing methods and a reflection model to explain the physics behind images and video recordings. Using this model, several color spaces and their invariance properties are discussed in section 3.2. In section 3.3, color constancy is discussed, which we use to achieve invariance to the color of the light source. In section 3.4, the use of image derivatives for detection of edges and corners in an image is discussed. In section 3.5, interest point detectors based on the Harris corner detector are used to detect scale invariant points. For description of the image area around these points, we discuss several region descriptors based on shape and color in section 3.6. In section 3.7, we discuss data clustering algorithms which we use for data reduction. Finally, in section 3.8, machine learning algorithms are discussed which we employ to learn models from annotated data.

3.1 Reflection model

It is important to understand how color arises, before we can discuss color-based features. We need to understand the process behind the image formation process. Image formation is modeled as the interaction between three elements: light, surface and observer (a human observer or image capture devices such as cameras). In figure 3 the image formation process is illustrated. Light from a light source falls upon a surface and is reflected. The reflected light falls upon the sensor of the observer (human eye or camera CCD chip) and eventually leads to a perception or measurement of color. Because we will be processing digital videos, we will focus our discussion on the creation of digital images and not on exploring human perception of color.

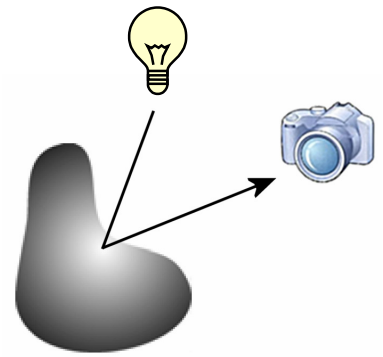


Figure 3: Image formation process. Light from a light source falls upon a surface and is reflected. Depending on the angle between the surface normal and the incident light, a different amount of light is reflected. The reflected light falls upon the sensor of an observer which eventually leads to a perception or measurement of color.

Digital cameras measure the light that falls onto their CCD chip. This chip contains many small sensors. These sensors have a sensitivity which varies depending on which part of the light spectrum they are measuring. We will refer to the sensitivity of a sensor at wavelength λ as $f(\lambda)$. As there exists no single sensor which can accurately measure the entire visible light spectrum, typical image capture devices sample the incoming light using three sensors. In general, these sensors are sensitive to red, green and blue wavelength light. The responses of these sensors are denoted by R, G and B. Together they form a triplet of numbers. Mathematically, the responses are related to light,

surface and sensor of the image formation process and are defined as follows:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} (\vec{e} \cdot \vec{n}) \int E(\lambda)\rho(\lambda)f_R(\lambda)d\lambda \\ (\vec{e} \cdot \vec{n}) \int E(\lambda)\rho(\lambda)f_G(\lambda)d\lambda \\ (\vec{e} \cdot \vec{n}) \int E(\lambda)\rho(\lambda)f_B(\lambda)d\lambda \end{pmatrix} \tag{1}$$

with $E(\lambda)$ the spectrum of the light source, $\rho(\lambda)$ the reflectance of the surface and $f_S(\lambda)$ the sensitivity of sensor S to different parts of the spectrum.

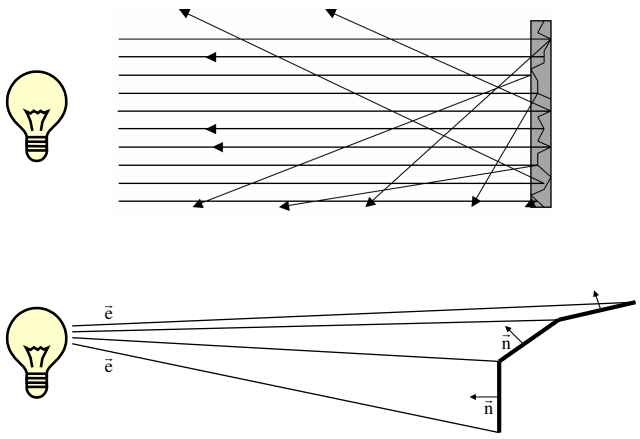


Figure 4: At the top, Lambertian reflectance of a surface is illustrated. Light penetrates a surface and leaves the surface in all directions, reflected off of micro surfaces. This is the color signal $C(\lambda)$, which depends on the incident light $E(\lambda)$ and the surface reflectance $\rho(\lambda)$. At the bottom, it is illustrated how a Lambertian surface, which reflects light with equal intensity in all directions, can still have an intensity depending on the angle between the surface normal $\vec{n}$ and the angle of incidence of the light $\vec{e}$: the light per unit area is different.

The color signal $C(\lambda)$ which is perceived or measured by a sensor is created by light $E(\lambda)$ penetrating the surface and being reflected off of micro surfaces, depending on the surface reflectance $\rho(\lambda)$. This is illustrated in figure 4 at the top.

We assume surfaces are Lambertian, i.e. they reflect light with equal intensity in all directions. This does not mean that all surfaces appear equally bright. The amount of light arriving depends on the angle of the surface with respect to the light. This is illustrated in figure 4 at the bottom. The intensity of the reflected light depends on the cosine angle between the surface normal $\vec{n}$ and the angle of incidence of the light $\vec{e}$. We can write the scale factor $(\vec{e} \cdot \vec{n})$ outside the integral as it does not depend on the wavelength of the light.

Digital images are created by taking color samples of a scene at many adjacent locations. A *pixel* is the fundamental element in a digital image. With a pixel, a single color is associated. A single pixel in this grid corresponds to a finite area of a digital camera chip. Over this finite area R, G and B sensors are sampled. These samples are combined into a RGB triplet, which is associated with a pixel. The RGB samples need to be quantified within a limited range and precision to allow for digital storage. We will assume a range of $[0, 1]$ for the RGB triplet. RGB triplets include: $(1, 0, 0)$ is red, $(0, 1, 0)$ is green, $(0, 0, 1)$ is blue, $(1, 1, 1)$ is white and $(0, 0, 0)$ is black, $(1, 1, 0)$ is yellow and $(1, \frac{3}{4}, \frac{3}{4})$ is pink.

3.1.1 Modelling geometry changes

Equation 1 shows that when the surface or lighting geometry changes, sensor responses change by a single scale factor $(\vec{e} \cdot \vec{n})$. That means that sensor responses to a surface seen under two different viewing geometries or illumination intensities are related by:

$$\frac{R_2}{R_2 + G_2 + B_2} = \frac{sR_1}{s(R_2 + G_2 + B_2)}$$

It is straightforward to make the sensor responses independent of the light intensity, by dividing by the sum of the R, G and B responses. This yields the normalized RGB color space:

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} \frac{R}{R+G+B} \\ \frac{G}{R+G+B} \\ \frac{B}{R+G+B} \end{pmatrix} \tag{2}$$

This yields the normalized RGB color model. The (r, g, b) triplet is invariant to changes in geometry and illumination intensity, as the sum of the triplet is always equal to 1.

3.1.2 Modelling illumination color changes

When the color of the light source in a scene changes, it becomes very challenging to achieve stable measures. A color measurement of the scene is needed which remains stable regardless of the illuminant spectral power distribution $E(\lambda)$. However, from equation 1, we derive that the illumination $E(\lambda)$ and the surface reflectance $\rho(\lambda)$ are heavily intertwined. Separating them is non-trivial.

However, often the relationship between sensor responses under different illuminants can be explained by a simple model. This model is called the *diagonal model* or von Kries model of illumination change [49]. In particular the responses to a single surface viewed under two different illuminants 1 and 2 is approximated as:

$$\begin{pmatrix} R_2 \\ G_2 \\ B_2 \end{pmatrix} = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{pmatrix} \begin{pmatrix} R_1 \\ G_1 \\ B_1 \end{pmatrix} \tag{3}$$

where $\alpha = \beta = \gamma$ for geometry and illumination changes only.

The diagonal model only holds when there is no ambient lighting in the scene. Also, it is assumed that a single spectral power distribution for the light source is used throughout the scene. It is possible to use multiple light sources, as long as a single spectral distribution $E(\lambda)$ is sufficient to model them.

3.2 Color spaces and invariance

In this section we will discuss various color spaces which can be used to represent a color and their invariance properties. Table 3 gives an overview of the invariance properties of the color spaces based on Gevers [13]. The listed properties will not be repeated in the text.

3.2.1 RGB

We have already encountered the RGB color space in our discussion on the image formation process. The color space consists of three channels, R, G and B. Every channel can take values in the range $[0, 1]$. In figure 5 the RGB color space is visualized as an RGB cube. All grey-values lie on the axis $(0, 0, 0)$ (black) to $(1, 1, 1)$ (white).

	I	RGB	rgb	H	$o1o2$
viewing geometry	-	-	+	+	+
surface orientation	-	-	+	+	+
shadows/shading	-	-	+	+	-
highlights	-	-	-	+	+
illumination intensity	-	-	+	+	+
illumination color	-	-	-	-	-

Table 3: Overview of color spaces and their invariance properties based on Gevers [13]. + denotes invariant and - denotes sensitivity of the color space to the imaging condition.

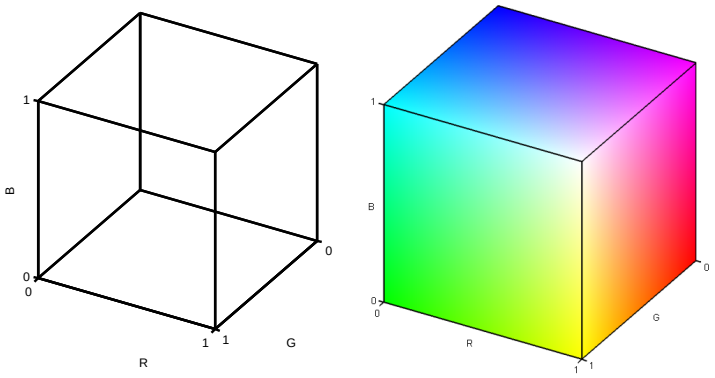


Figure 5: On the left a schematic view of the RGB color space. On the right a visualization of the RGB color space. Only three faces of the RGB cube on the left can be seen, containing red $(1, 0, 0)$, green $(0, 1, 0)$, blue $(0, 0, 1)$, yellow $(1, 1, 0)$, cyan $(0, 1, 1)$, magenta $(1, 0, 1)$ and white $(1, 1, 1)$. White is hard to discern due to the black lines of the cube.

3.2.2 Normalized RGB

The normalized RGB color space is invariant to changes in lighting geometry and overall changes in intensity. This was discussed in section 3.1.1. Equation 2 converts RGB colors to normalized RGB.

The sum of r,g and b is always equal to 1, making one of the channels redundant. It should come as no surprise then that the normalized RGB color space is equal to a plane in the RGB color space: the plane where $R+G+B=1$. In figure 6 the location of this plane inside the RGB cube is visualized, as is the plane itself. This plane takes the shape of a triangle. It is commonly referred to as the chromaticity triangle. White $(1,1,1)$ becomes $(\frac{1}{3},\frac{1}{3},\frac{1}{3})$ and is located at the center of the triangle. All values from the grey-value axis $(0,0,0)$ to $(1,1,1)$ map to this point, except black, since division by zero yields infinite values. Practical implementations of this color space do map black to the point $(\frac{1}{3},\frac{1}{3},\frac{1}{3})$.

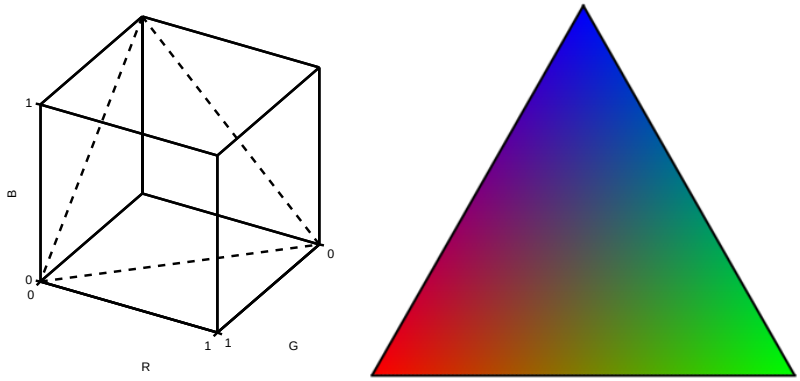


Figure 6: On the left the RGB cube. The chromaticity triangle '$R+G+B=1$' is denoted by the dashed lines. On the right the chromaticity triangle is drawn separately, visualizing the normalized RGB color space.

3.2.3 HSI

The HSI color space consists of three channels: Hue, Saturation and Intensity. In the normalized RGB color space, the light intensity $(R+G+B)$ was divided out. In the HSI color space it is a separate channel:

$$I = \frac{R+G+B}{3}$$

Hue and saturation are defined in the chromaticity triangle relative to a reference point. This reference point is a white light source, located at the center of the triangle. Saturation is defined as the radial distance of a point from the reference white point (see figure 7). Saturation denotes the relative white content of a color.

$$S = \sqrt{(r-\frac{1}{3})^2+(g-\frac{1}{3})^2+(b-\frac{1}{3})^2}$$

Hue is defined as the angle between a reference line (the horizontal axis) and the color point (see figure 7). Hue denotes the color aspect, e.g. the actual color:

$$H = \arctan\frac{r-\frac{1}{3}}{g-\frac{1}{3}}$$

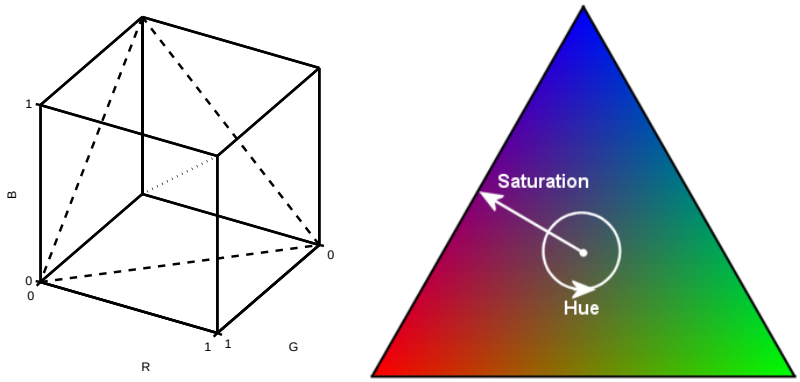


Figure 7: On the left the RGB cube. The chromaticity triangle is denoted by the dashed lines. The dotted line denotes the direction of the intensity channel: perpendicular to the chromaticity triangle. On the right the chromaticity triangle is drawn separately, denoting the meaning of hue and saturation channels. Hue and saturation are defined for all planes parallel with the chromaticity plane; the specific plane depends on the value of the intensity channel.

Figure 7 helps in understanding the HSI color space. The intensity channel is perpendicular to the chromaticity triangle. Hue and saturation are most easily defined in the chromaticity triangle. Note that the chromaticity plane's location changes depending on the intensity. This allows the HSI color model to cover the entire RGB cube. However, hue and saturation are nonlinear. Saturation becomes unstable for low intensities. Hue becomes unstable when saturation and intensity are low. This instability thus occurs for grey-values and dark colors [46].

Above we have used normalized RGB channels in our conversion to HSI. A direct conversion from RGB to HSI is as follows [19]:

$$\begin{aligned} H &= \arctan \frac{\sqrt{3}(G-B)}{(R-G)+(R-B)} \\ S &= 1 - \frac{\min(R,G,B)}{R+G+B} \\ I &= \frac{R+G+B}{3} \end{aligned}$$

3.2.4 Opponent color space

The opponent color theory states that the working of the human eye is based on three kinds of opposite colors: red-green, yellow-blue and white-black. An example of the theory is the *after-image*. Looking at a green sample for a while will leave a red after-image when looking at a white sample afterwards. The same effect occurs for yellow-blue. Both members of an opponent pair exhibit the other: adding a balanced proportion of red and green will produce a color which is neither reddish nor green. Also, no color appears to be a mixture of both members of any opponent pair.

The opponent color space is based on the opponent color theory. It is given by:

$$\begin{aligned} o1 &= \tfrac{R-G}{\sqrt{2}} \\ o2 &= \tfrac{R+G-2B}{\sqrt{6}} \\ o3 &= \tfrac{R+G+B}{\sqrt{3}} \end{aligned}$$

The third channel $o3$ is equal to the intensity channel of HSI, subject to a scaling factor. $o1$ and $o2$ contain color information. All channels of the opponent color space are decorrelated.

3.3 Color constancy

Color constancy is the ability to recognize colors of objects invariant of the color of the light source [10]. In the model of illumination color changes (section 3.1.2) corresponding pixels of two images of a scene under two different illuminants are related by three fixed scale factors. If we know the illuminant used in a scene and its relationship to a predefined illuminant like $RGB(1, 1, 1)$, then it is possible to normalize the scene and achieve color constancy. Certain assumptions need to be made in order to achieve color constancy. A number of different hypotheses are commonly used:

- Grey-World hypothesis [3]: the average reflectance in a scene is grey.
- White patch hypothesis [2]: the highest value in the image is white.
- Grey-Edge hypothesis [45]: the average edge difference in a scene is grey.

We will use the Grey-Edge hypothesis, because it outperforms the two other hypotheses [45]. With the Grey-Edge assumption the illuminant color can be computed from the average color derivative in the image. For additional details we refer to [45].

Color constancy normalization gives invariance to color of the illuminant. From equation 3 we derive that a division by the scaling factors α, β and γ achieves color constancy. These scaling factors are equal to the estimated color of the illuminant when converting to the illuminant $RGB(1, 1, 1)$.

3.4 Image derivatives

The ability to take one or more spatial derivatives of an image is one of the fundamental operations in digital image processing. However, according to the mathematical definition of derivative, this cannot be done. A digitized image is not a continuous function of spatial variables, but a discrete function of the integer spatial coordinates. As a result only approximations to the true spatial derivatives can be made.

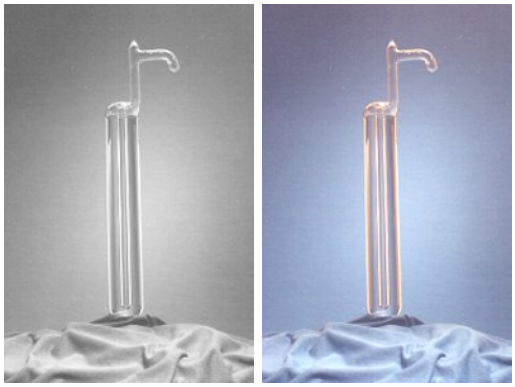


Figure 8: Example image used to illustrate digital image processing operations throughout this chapter. On the left the greyvalue version is shown. On the right the color version is shown.

The two dimensional Gauss filter is the most common filter for computing spatial derivatives [14]. The linear Gaussian filter defined with the convolution kernel g_σ is:

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

with σ the smoothing scale over which to compute the derivative. In the rest of this thesis we will adopt the notation $g(x, \sigma)$ for the Gaussian filter with x a 2D position and σ the smoothing scale.

In figure 9 the first order derivatives of figure 8 in the x and y direction are shown for different smoothing scales σ. These derivatives are referred to as L_x and L_y for the x and y directions, respectively.

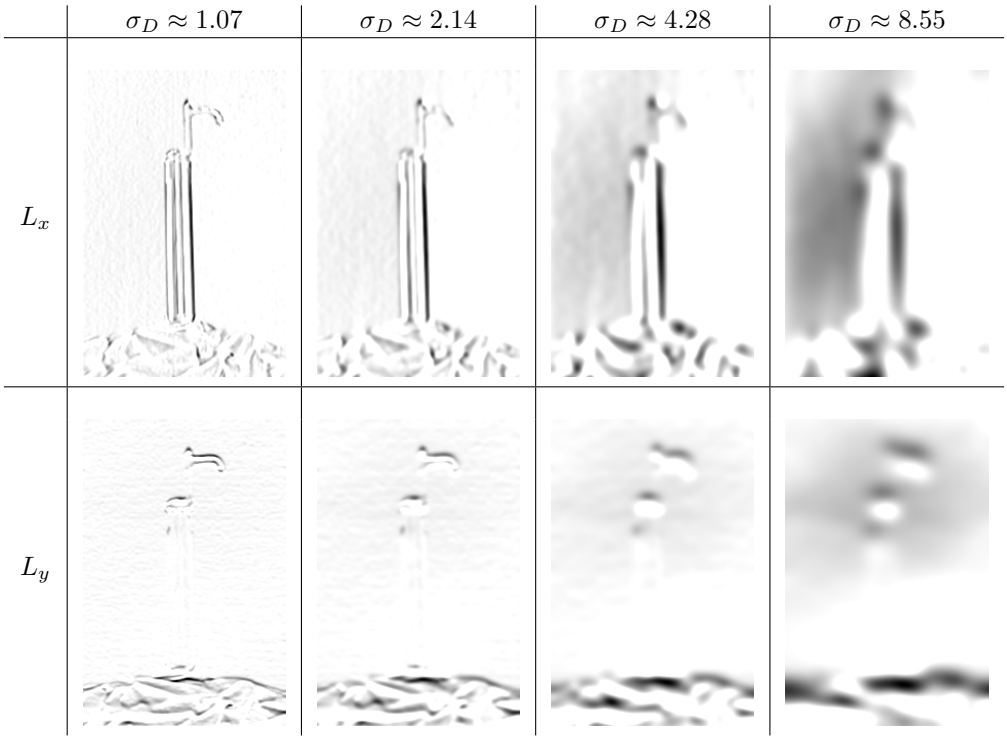


Figure 9: Image derivatives in the X direction L_x (top row) and in the Y direction L_y (bottom row) at various differentiation scales. The differentiation scale is shown above the column of the image. The derivatives have inverted and scaled for display. White signifies a derivative value near 0. Black is the maximum derivative value, with greyvalues lying in-between. Per scale the maximum derivative value is different: for higher differentiation scales, the maximum value is lower. For visualization purposes, these have been scaled to use the full grey-value range. The input image is shown on the left in figure 8.

The image gradient at a point x is defined in terms of the first order derivatives as a two-dimensional column vector:

$$\bigtriangledown f(x) = \begin{pmatrix} L_x(x, \sigma) \\ L_y(x, \sigma) \end{pmatrix}$$

The magnitude of the gradient $\bigtriangledown f$ is high at discontinuities in an image. Examples are edges, corners and T-junctions. Using local maxima of the gradient magnitude it is possible to build a very simple edge detector.

A distinction between edges and corners and T-junctions can be made using the second moment matrix (also known as auto-correlation matrix). The second moment matrix μ is defined in terms of spatial derivatives L_x and L_y:

$$\mu(x, \sigma_D) = \begin{pmatrix} L_x^2(x, \sigma_D) & L_x L_y(x, \sigma_D) \\ L_x L_y(x, \sigma_D) & L_y^2(x, \sigma_D) \end{pmatrix}$$

with σ_D is the differentiation scale. Stability of the second moment matrix can be increased by integrating over an area using the Gaussian kernel $g(\sigma_I)$ (see equation 4).

The eigenvalues of this matrix, $\lambda_1(\mu)$ and $\lambda_2(\mu)$, are proportional to the curvature of the area around point x. Using the eigenvalues point x can be categorized as follows:

- $\lambda_1(\mu) \approx 0$ and $\lambda_2(\mu) \approx 0$: There is little curvature in any direction; the point is located in an area of uniform intensity.
- $\lambda_1(\mu) \gg \lambda_2(\mu)$: Large curvature in one direction and little curvature in the other direction; the point is located on an edge.
- $\lambda_1(\mu) \approx \lambda_2(\mu) \gg 0$: Large curvature in both directions, the point is located on a corner or a junction.

In the next section interest point detectors will be built using spatial derivatives and the second moment matrix.

3.5 Interest point detectors

In this section we will discuss two scale invariant interest point detectors: Harris-Laplace and ColorHarris-Laplace. Both are based on the Harris corner detector and use Laplacian scale selection. For an overview of detectors see [24].

3.5.1 Harris corner detector

The Harris corner detector is based on the second moment matrix (see section 3.4). The second moment matrix describes local image structure. By default the matrix is not independent of the image resolution, so it needs to be adapted for scale changes.

The adapted matrix for a position x is defined as follows [27]:

$$\mu(x, \sigma_I, \sigma_D) = \sigma_D^2 g(\sigma_I) \begin{pmatrix} L_x^2(x, \sigma_D) & L_x L_y(x, \sigma_D) \\ L_x L_y(x, \sigma_D) & L_y^2(x, \sigma_D) \end{pmatrix} \quad (4)$$

with σ_I the integration scale, σ_D is the differentiation scale and $L_z(x, \sigma_D)$ the derivative computed in the z direction at point x using differentiation scale σ_D.

The matrix describes the gradient distribution in the local neighborhood of point x. We compute local derivatives with Gaussian kernels with a size suitable for the differentiation scale σ_D. The derivatives are averaged in the neighborhood of point x by smoothing with a Gaussian window suitable for the integration scale σ_I.

The eigenvalues of the matrix represent the two principal signal changes in the neighborhood of a point. This is akin to the eigenvalues of the covariance matrix in principal component analysis (PCA). The eigenvalues are paired with a corresponding eigenvector. We project the data onto the basis formed by the orthogonal eigenvectors. The eigenvalue corresponding to an eigenvector will then describe the variance in the direction of the eigenvector.

We use this to extract points for which both eigenvalues are significant: then the signal change is significant in orthogonal directions, which is true for corners, junctions,

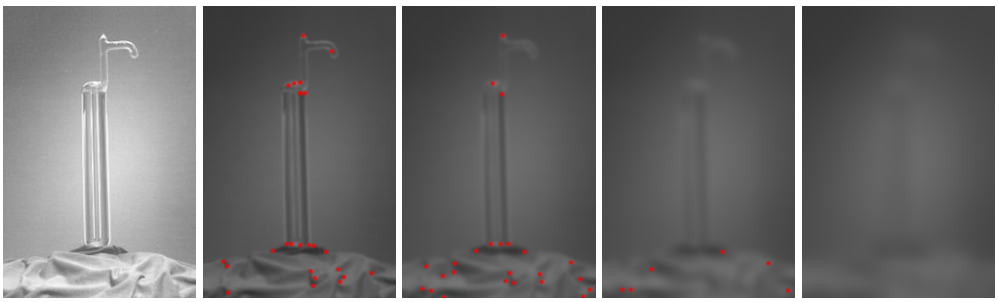
etc. Such points are stable in arbitrary lighting conditions. In literature these points are called 'interest points'.

The Harris corner detector [15], one of the most reliable interest point detectors, is based on this principle. It combines the trace and the determinant of the second moment matrix into a *cornerness measure*:

$$cornerness = \det(\mu(x, \sigma_I, \sigma_D)) - \kappa \text{trace}^2(\mu(x, \sigma_I, \sigma_D)) \tag{5}$$

with κ an empirical constant with values between 0.04 and 0.06.

Local maxima of cornerness measure determine the location of interest points. The interest points detected depend on the differentiation and integration scale. Due to the definition of *cornerness*, it is possible for the measure to become negative, yet still be a maximum. This happens if all values surrounding the negative maximum have lower values than the maximum. To counter very low *cornerness* maxima, they are only accepted if they exceed a certain threshold.



original image $\sigma_I = 1.5$ $\sigma_I = 3$ $\sigma_I = 6$ $\sigma_I = 12$

Figure 10: Corners (red dots) detected at different scales using Harris corner detector. The differentiation scale σ_D has a fixed relation to the integration scale: $\sigma_D = 0.7125\sigma_I$. The differentiation scales match those displayed in figure 9. On the left the input image is shown.

3.5.2 Scale selection

The Harris corner detector from the previous section has scale parameters. The detected points depend on this scale. It is possible for a single point to be detected at multiple (adjacent) scales.

The idea of automatic scale selection is to select the characteristic scale of the point, depending on the local structure around the point. The characteristic scale is the scale for which a given function attains a maximum over scales.

It has been shown [26] that the cornerness measure of the Harris corner detector rarely attains a maximum over scales. Thus, it is not suitable for selecting a characteristic scale. The Laplacian-of-Gauss (LoG) does attain a maximum over scales. We use it to select the characteristic scale of a point. With σ_n, the scale parameter of the LoG, it is defined for a point x as:

$$|LoG(x, \sigma_n)| = \sigma_n^2 |L_{xx}(x, \sigma_n) + L_{yy}(x, \sigma_n)| \tag{6}$$

In figure 11, the LoG kernel is visualized. The function reaches a maximum when the size of the kernel matches the size of the local structure around the point. It responds to blobs very well due to its circular symmetry, but it also responds to corners, edges, ridges and junctions.

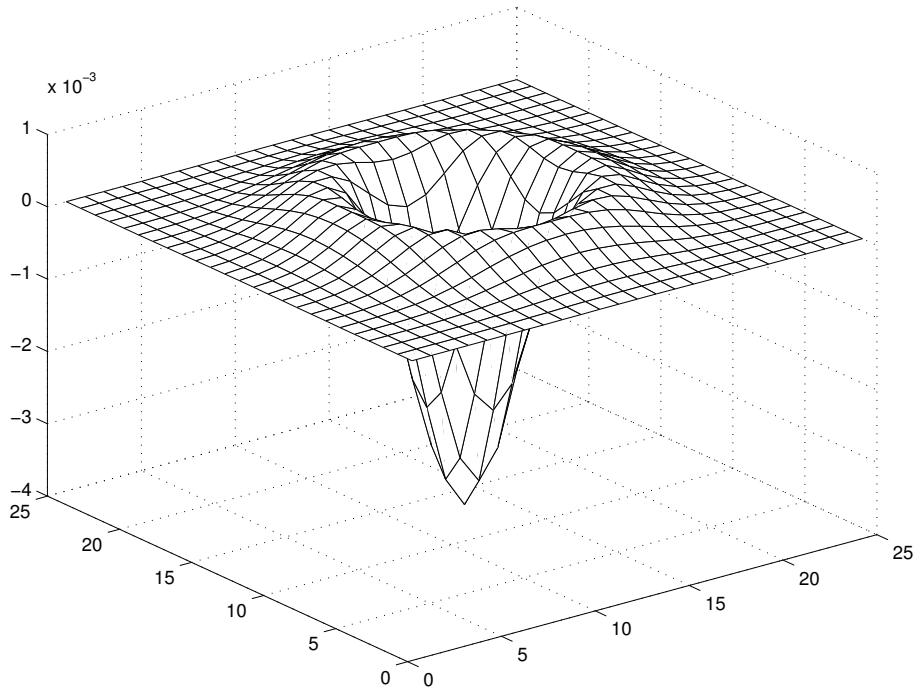


Figure 11: Laplacian-of-Gauss (LoG) kernel. The kernel shown has scale $\sigma_n = 5$.

3.5.3 Harris-Laplace detector

The Harris-Laplace detector uses the Harris corner detector to find potential scale-invariant interest points. It then selects a subset of these points for which the Laplacian-of-Gaussian reaches a maximum over scale.

Mikolajczyk [27] defines an iterative version of the Harris-Laplace detector and a 'simplified' version which does not involve iteration. The simplified version performs a more thorough search through the scale space by using smaller intervals between scales. The iterative version relies on its convergence property to obtain characteristic scales. By performing iteration, it gives more fine-grained estimations. Mikolajczyk notes that the 'simplified' version is a trade-off between accuracy and computational complexity. Due to our need for fast feature extraction (section 2.3), we will use the simplified version of the Harris-Laplace detector.

In section 5.2 we will discuss the Harris-Laplace algorithm and its implementation in more detail.

3.5.4 Color boosting

The intensity-based interest point detector in the previous section uses the derivative structure around points to measure the saliency of the point. Stable points with high saliency qualify as an interest point. In the next section interest points detectors will be extended to use all *RGB* channels.

Rare color transitions in an image are very distinctive. By adapting the saliency of an image, the focus of the detector shifts to more distinctive points. The transformation of the image to achieve this is called color saliency boosting. We use the color boosting transformation in the opponent color space [44]. This transformation is a weighing of

the individual opponent channels:

$$o1' = 0.850\ o1$$
$$o2' = 0.524\ o2$$
$$o3' = 0.065\ o3$$

where the sum of the squared weights is equal to 1. These weights are focused on the red-green and yellow-blue opponent pair, with almost no weight given to the intensity $o3$ channel. Figure 12 (on the left) shows the colorboosted version of the color image in figure 8. The difference between the glass and the background has increased significantly, increasing saliency of the edges of the glass.

3.5.5 ColorHarris-Laplace detector

So far the images consist of single channels, i.e. intensity images. Only the Harris corner detector and the LoG function operate on the image directly. In this section, we extend them to operate on color images.

The Harris corner detector uses the second moment matrix. The elements of the matrix are always the product two image derivatives L_x (with x the derivative direction). For the extension to multiple channels n, we replace L_x with a vector $\vec{f}_x = \left(L_x^{C_1}, L_x^{C_2}, \ldots, L_x^{C_n}\right)^T$ with C_i the i^{th} channel. The product between derivatives is replaced with a vector inproduct. If the vector is 1-dimensional (e.g. an intensity image), this is equivalent to the original second moment matrix.

The second moment matrix for a ColorHarris corner detector is:

$$\mu_{RGB}(x, \sigma_I, \sigma_D) = \sigma_D^2 g(\sigma_I) \begin{pmatrix} \vec{f}_x(x,\sigma_D) \cdot \vec{f}_x(x,\sigma_D) & \vec{f}_x(x,\sigma_D) \cdot \vec{f}_y(x,\sigma_D) \\ \vec{f}_x(x,\sigma_D) \cdot \vec{f}_y(x,\sigma_D) & \vec{f}_y(x,\sigma_D) \cdot \vec{f}_y(x,\sigma_D) \end{pmatrix}$$

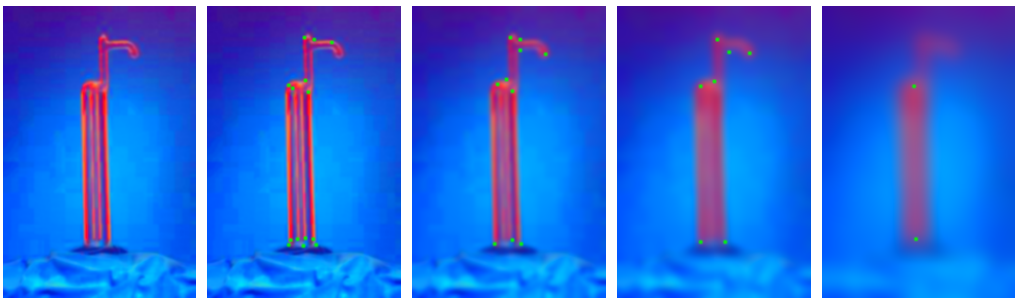
with $\vec{f}_x = \left(L_x^{C_1}, L_x^{C_2}, \ldots, L_x^{C_n}\right)^T$ for an image with channels $\{C_1, C_2 \ldots C_n\}$, with $L_x^{C_i}$ being the Gaussian derivative of the i^{th} image channel C_i in direction x. The image channels C_i can be instantiated to channels of any color model. However, it is also possible to first apply preprocessing operations (such as color boosting) to an image and then instantiate the channels. For our ColorHarris corner detector, we instantiate the channels to the R, G and B channels of the standard RGB color space. We preprocess images using color boosting. Results of this detector are shown in figure 12. Due to color boosting, the cloth underneath the glass now consists of different shades of blue only. The contribution of the R and G channels to the *cornerness* will be very low because there is almost no change in these channels in the cloth. Primarily, the *cornerness* comes from the B channel only, which by itself is not enough to exceed the threshold.

One way to extend the Laplacian-of-Gaussian kernel to multiple channels is by summing the responses of the individual channels:

$$|LoG_{RGB}(x, \sigma_n)| = |LoG_R(x, \sigma_n)| + |LoG_G(x, \sigma_n)| + |LoG_B(x, \sigma_n)|$$

It is also possible to assign different weights depending on the color channel or to use different color channels altogether.

In conclusion, we have two different scale invariant point detectors. Harris-Laplace triggers on corners in an intensity image, while Colorboosted ColorHarris-Laplace focuses on the color information in an image.



colorboosted image $\sigma_I = 1.5$ $\sigma_I = 3$ $\sigma_I = 6$ $\sigma_I = 12$

Figure 12: Corners (green dots) detected at different scales using ColorHarris corner detector on a colorboosted image (the original color image is shown in figure 8). The differentiation scale σ_D has a fixed relation to the integration scale: $\sigma_D = 0.7125\sigma_I$. The differentiation scales match those displayed in figure 9. On the left the colorboosted input image is shown.

3.6 Region descriptors

For all possible granularities, there exists an image region that needs to be described. The descriptors in this section are applicable to the entire image (global description) and to interest regions (local description). The only difference between them is that for the global description there is only a single image region to be described. For the local description, the number of regions to be described is equal to the number of interest regions.

The organization of this section is as follows. First, color histograms are discussed. Second, color histograms on transformed color spaces are discussed. Third, a histogram based on hue is discussed. Fourth, color moments and color moment invariants are discussed. Finally, the SIFT descriptor is discussed.

3.6.1 Color histograms

A histogram is the graphical version of a table which shows what proportion of cases falls into a certain data interval. In statistics, the data used for the creation of histograms is one dimensional. The number of data values that fall into each of the data intervals is counted, yielding the occurrence frequency of this interval. The data intervals are commonly referred to as histogram bins.

Image histograms can be constructed by using the color of all image pixels as data points. For intensity images, the 'color' of a pixel is equal to a single value. So, it is straight-forward to use a 1D histogram. For color images, the color of a pixel is a vector with length equal to the number of channels (typically 2 or 3). There are two ways this can be achieved:

- Create a separate 1-dimensional histogram for each of the image channels. This assume channel independence, because correlation between the channels cannot be captured. With 3 channels and 15 bins per channel, this would yield a histogram with 45 bins.
- Create an n-dimensional histogram, with each of the channels divided into a number of intervals. A pixel color falls into an interval in each dimension, together forming a bin. The number of bins is equal to the multiplication of the number of intervals in each channel. With 3 channels and 15 intervals per channel, this would yield a histogram with $15^3 = 3375$ bins.

Theoretically, a 3D histogram for *RGB* images is more attractive than three 1D histograms. However, the histogram can become sparse due to the large number of bins. Sparseness could be reduced by assigning pixels to neighboring bins of their actual bin as well. The increased amount of processing needed due to the larger number of bins is an important drawback.

Due to the need for fast feature extraction and processing (section 2.3), we will use the first method only. For a *RGB* histogram the assumption of independence between the channels obviously does not hold. However, we believe it will still contain enough useful information, albeit redundant.

The assumption does hold for a histogram based on the opponent color space (section 3.2.4). In this color space, the channels are decorrelated and hence it is theoretically sound to use 1D histograms.

3.6.2 Transformed color distribution

A *RGB* histogram is not invariant to changes in lighting conditions. If the light intensity is increased, the color distribution characterized in the histogram is shifted to the right (towards bins representing brighter colors) and possibly scaled. If the light intensity is decreased, the distribution shifts to the left. It can be derived from equation 3 that a change in illumination color scales the color distribution in every channel individually.

We achieve invariance to light intensity and illumination color by normalizing the color distribution. The shift of the distribution is addressed by subtracting the mean. The scaling of the distribution is countered by dividing through the standard deviation of the distribution:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} \frac{R-\mu_R}{\sigma_R} \\ \frac{G-\mu_G}{\sigma_G} \\ \frac{B-\mu_B}{\sigma_B} \end{pmatrix}$$

with μ_C the mean and σ_C the standard deviation of the distribution in channel C.

Overall this yields a distribution with a mean of 0 and a standard deviation of 1. A histogram is constructed just like for normal color values, though the bin intervals will need to be adapted so negative values are covered. A histogram based on the normalized distribution is invariant to changes in light intensity, illumination color, shadows, shading and viewpoint and geometry changes of the scene. A disadvantage is that distinguishing scenes from one another by these properties is no longer possible: the discriminability of the descriptor is reduced.

3.6.3 Hue histogram

The hue captures the dominant wavelength of color in the HSV color space (see section 3.2.3). To describe color information in an image area, it makes sense to construct a histogram of the hue channel. However, hue is known to be unstable around the grey axis.

Van de Weijer [47] applies an error analysis to the hue. The analysis shows that the certainty of the hue channel is inversely proportional to the saturation. The lower the saturation, the more uncertain the color information in the hue channel. As colors with low saturation are located near the grey axis, this hue unstability is expected. The hue histogram is made more robust by weighing each sample of the hue by its saturation. In a normal histogram, all data samples have an equal weight (typically 1).

The hue histogram is used as a 1D color histogram of the hue channel, where the bin of a pixel is based on the hue. The value added to the bin is weighed by the saturation. A further weighing with a Guassian mask is performed, which depends on the distance

of a pixel to the center of the interest region. This corresponds to the hue histogram of Van de Weijer [47].

3.6.4 Color-based moments and moment invariants

In mathematics, we compute moments over distributions or functions up to an arbitrary order. The first moment is equal to the mean of a distribution. The second moment is equal to the variance of the distribution. The third moment is called skewness after the skew of a distribution.

A color image corresponds to a mathematical function I defining RGB triplets for image positions (x, y): $I : (x, y) \mapsto (R(x, y), G(x, y), B(x, y))$. Regarding RGB triplets as data points coming from a distribution, it is possible to define moments. Mindru *et al* [29] have defined *generalized color moments* M_{pq}^{abc}:

$$M_{pq}^{abc} = \int \int x^p y^q [I_R(x, y)]^a [I_G(x, y)]^b [I_B(x, y)]^c dxdy \tag{7}$$

M_{pq}^{abc} is referred to as a generalized color moment of *order* $p+q$ and *degree* $a+b+c$. Note that moments of order 0 do not contain any spatial information, while moments of degree 0 do not contain any color information. Thus, moment descriptions of order 0 are rotationally invariant, while higher orders are not. A large number of moments can be created with small values for the order and degree. However, for larger values the moments are less stable. Typically generalized color moments up to the first order and the second degree are used.

Under the assumption that the image being described is planar, photometric changes in the image can be described as a linear combination of moments. By using the right combination of moments, it is possible to normalize against photometric changes. These combinations are called *color moment invariants*. Invariants involving only a single color channel (e.g. out of a,b and c two are 0) are called 1-band invariants. Similarly there are 2-band invariants involving only two out of three color bands. 3-band invariants involve all color channels, but these can always be created by using 2-band invariants for different combinations of channels.

We will use both color moments and color moment invariants for region description. The invariants have the additional benefit that they are invariant to photometric changes.

3.6.5 SIFT

SIFT as originally proposed by Lowe [20] consists of both a scale invariant point detector and a region descriptor. SIFT operates on greyvalue images only; no color information is used. The detector gives results similar to the Harris-Laplace detector: scale invariant points are detected on corners in an image. We will only look at the SIFT descriptor. SIFT describes the local shape of the interest region using edge histograms.

The descriptor consists of two steps: orientation assignment and region description.

Orientation assignment The image gradients in a region of interest are computed (e.g. a 16x16 or 32x32 area). Orientations corresponding to the dominant direction(s) of the gradients are assigned to the region. Typically there is one dominant direction, but regions with two or three dominant directions exist. Description is performed relative to the assigned orientation. The descriptor is adapted to the scale of the region by resampling the area around the region center, sized proportionally to the region scale, to a fixed-size window (see section 5.3). This provides us scale and rotation invariance.

Region description SIFT describes the local shape of the interest region using edge histograms. To make the descriptor robust to noise, while retaining some positional information, the interest region is divided into a 4x4 grid where every quadrant has its own edge direction histogram (8 bins). This edge direction histogram is constructed from the local gradient direction and weighed by the gradient magnitude. In figure 13 an example edge histogram with a 2x2 grid is shown.

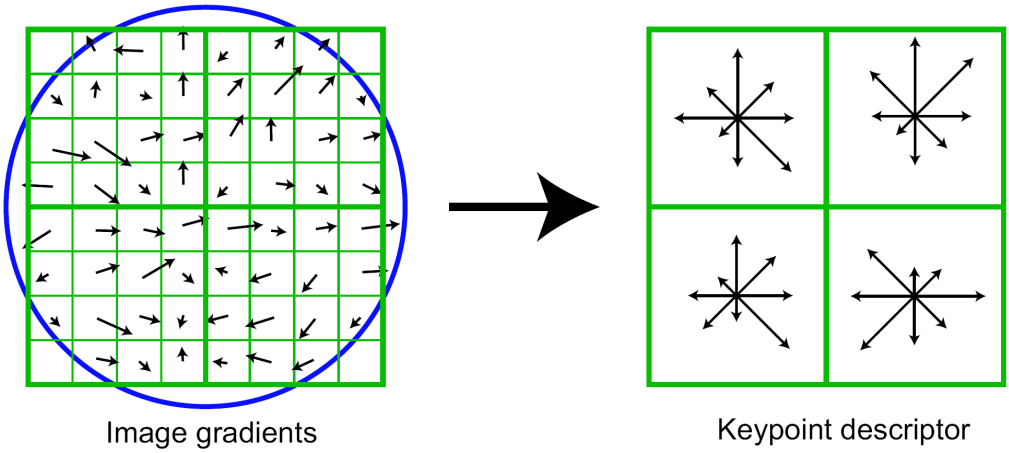


Figure 13: Schematic of the SIFT descriptor. On the left, an interest region is visualized (blue circle). The region is divided into four quadrants. Here every quadrant contains 16 samples of the image gradient. The gradient direction and magnitude samples are aggregated into an 8-bin gradient histogram. E.g. the length of the arrows in the gradient histograms on the right represent the bins of the gradient histograms. Every one of the four quadrants has its own gradient histogram. Together these form the description of the interest region. Scheme is taken from Lowe [21].

3.7 Visual vocabulary

So far, methods are discussed to find interest regions and compute descriptors for them. Every descriptor is a vector of fixed length. For every keyframe, a variable number of descriptors is available. These need to be aggregated into a fixed-length representation if we want to use it as an input for machine learning algorithms (section 3.8).

First, different ways are discussed to find clusters in visual descriptor data to construct a visual vocabulary. Second, a (dis)similarity measure is discussed to create a fixed-length representation from a variable number of descriptors using the visual vocabulary.

3.7.1 Clustering

Data clustering algorithms strive to find the most representative points in a data space such as our descriptor space. These points are typically called clusters, since we want them to be located in areas where the data is densely packed. Clusters are not required to be equal to one of the data points.

We use a data-driven clustering approach to find a visual vocabulary consisting of representative proto-concepts in descriptor space. Proto-concepts are an intermediate step towards 'semantically' modelling a scene. I.e. a specific proto-concept, which is located in descriptor space, *could* describe a region containing only blue and red. In turn this proto-concept could be used to identify a Dutch 'no parking' sign.

A popular clustering approach for finding proto-concepts is the k-means algorithm [6, 35, 43]. K-means is an unsupervised clustering algorithm based on the principle to minimize the variance between k clusters and the training data, where k is the number of clusters. The advantages of k-means are that it is simple and efficient to implement. However, the disadvantage of k-means is that the algorithm is variance-based. Thus, the algorithm will award more clusters to high-frequency areas of the data, leaving less clusters for the remaining areas. This over-sampling of dense regions is unwanted, since frequent occurring data is not so informative.

In contrast to variance-based clustering, the prototypes for a codebook model are better represented by using radius-based clustering [17]. Radius-based clustering assigns all data points within a fixed radius of similarity r to one cluster, where r is a parameter of the algorithm. This radius r, denotes the maximum similarity between data points that may be considered equal. As such, the radius determines whether two patches describe the same prototype. The radius-based clustering algorithm we use is developed by Astrahan [1] in 1970, which is a fixed radius version of the more accessible [30]. The algorithm by Jurie [17] differs from the algorithm by Astrahan that we use. We have chosen Astrahans algorithm since Jurie uses a Gaussian mean-shift kernel for finding the densest point where Astrahan does not make any assumptions about the data.

We will describe k-means and radius-based clustering algorithms in more detail in section 5.4.

3.7.2 Similarity measures

The (dis)similarity measure between the visual vocabulary and the descriptors of an image depends on the clustering algorithm used. For k-means a descriptor is assigned to the closest proto-concept. With $\vec{F}$ denoting the feature vector of length n, where n equals the number of clusters, we obtain:

$$\vec{F_i} = \frac{1}{m} \sum_{j=1}^{m} \psi(i, j) \tag{8}$$

where indicator function $\psi(i, j)$ equals 1 if the j^{th} descriptor is closer to the i^{th} cluster than to all other clusters and 0 otherwise. Closeness is computed using the Euclidian distance. Note that $\vec{F_i}$ is a similarity measure as the values of $\vec{F_i}$ will be higher for similar vectors. All elements of $\vec{F_i}$ are constrained to the range $[0, 1]$ by their definition.

It is possible to use the same similarity measure for the visual vocabulary constructed using radius-based clustering. However, it has been shown [48] that a dissimilarity measure gives significantly better results. In a dissimilarity measure, the distance of every proto-concept to a descriptor is used, instead of assigning the descriptor to the closest cluster only. The distances to descriptors for every proto-concept are averaged. Where in the similarity case $\vec{F_i}$ can be 0 if none of the descriptors are closest, in the dissimilarity case the average distance to all descriptors is kept, which is more informative.

Therefore, a dissimilarity measure is used between the descriptors of an image and the clusters to obtain a fixed-length feature vector $\vec{F}$ of length n (again n equals the number of clusters):

$$\vec{F_i} = \frac{1}{m} \sum_{j=1}^{m} d_{ij} \tag{9}$$

with d_{ij} the Euclidian distance between the i^{th} cluster and the j^{th} descriptor of the image.

3.8 Supervised machine learning

We learn semantic concept detectors from feature vectors that are extracted from video shots. The presence of a concept in a certain video shot is assumed to be binary: either the concept is present in that shot, or it is not. Assuming that a dataset annotated with concept presence is available, the dataset is divided into two classes: *concept-present* and *concept-not-present*. This is a machine learning problem: given feature vectors and corresponding class labels, learn a classifier that estimates the probability that an unseen feature vector belongs to one of the classes. We discuss two different machine learning algorithms: Support Vector Machines (SVM) and Fisher's Linear Classifier.

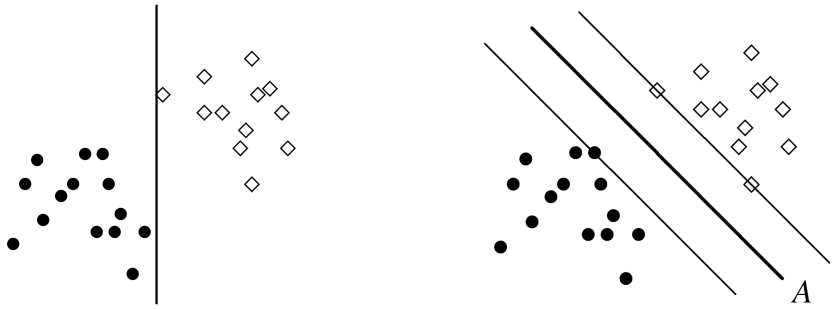


Figure 14: Two linearly separable datasets with separating hyperplane. The separating hyperplane on the right leaves the closest points at maximum distance. The thin lines on the right identify the margin. Illustration taken from Webb [50].

SVM [50] is a linear discriminant analysis. The idea is as follows: find the 'best' separating hyperplane in feature space, which has a maximal margin to the data (see figure 14). However, SVM has weight parameters for positive and negative instances of a class which need to be tuned. Every concept needs to have its parameters tuned individually.

Fisher's Linear Classifier [50] is also a linear discriminant analysis. This classifier seeks a linear combination of the feature space dimensions which separates the two classes as much as possible. That is, we are seeking a direction along which the two classes are best separated in some way. Hence, the Fisher's criterion involves maximizing the ratio of between-class variance to within-class variance. The advantage of Fisher's Linear Classifier over SVM is that it does not require parameter tuning.

4 Concept detection framework

In this chapter, we will outline our concept detection framework. This framework provides a basis for implementing a concept detection system and evaluating the different components within such a system.

The concept detection framework (see figure 15) operates on keyframe images from video shots. We extract features from these keyframes to describe the shots. Feature extraction pipelines are available for two types of features (see section 2.1.2): global features and local features. Global features are descriptions computed over the entire image area of the keyframe. Local features are an aggregation of descriptions of many interest regions in the keyframe image.

The pipeline for extraction of *global features* is shown at the top-left in figure 15. In the global feature extraction pipeline, image preprocessing steps are applied first to the keyframe image. Then, we compute descriptors over the entire image area of the (preprocessed) keyframe. Every descriptor yields a vector describing the keyframe. This vector is used as the feature vector describing the video shot. Hence, the dimensionality of global features is equal to the dimensionality of the descriptor.

The pipeline for the extraction of *local features* is shown at the top-right in figure 15. In the local feature extraction pipeline, we first detect interest regions in the (preprocessed) keyframe. Then, we compute descriptors for all detected interest regions. As in the global pipeline, optionally the keyframe images are preprocessed. We allow for different preprocessing steps for interest region detection and for region description.

With a variable number of interest regions per keyframe, the descriptors of these regions need to be aggregated in order to obtain a fixed-length feature vector. The alternative, a variable-length feature vector, only shifts the aggregation problem to a different location within the framework. We perform aggregation by computing a similarity measure between all descriptors for an image and a set of prototype descriptors. These prototypes together form a *visual vocabulary*. We select the prototype descriptors using a clustering algorithm. The vocabulary is fixed across different images. The dimensionality of the feature vector is equal to the number of prototypes in the vocabulary.

Both the global feature extraction pipeline and the interest region pipeline output fixed-length feature vectors for video shots. The concept detector pipeline in figure 15 shows how we use these feature vectors to perform experiments. The video shots are divided into a disjoint training and test set. A model for a concept ω_j is learned from the feature vectors of the training set and concept annotations of the ground truth. This concept model is applied to the feature vectors from the test set. Based on the output of the model application, all shots should be ordered by the likelihood that they contain the concept. This ordered list is the shot ranking ρ_j for the concept ω_j.

Using the ground truth for the test set and the shot ranking, we compute criteria that evaluate how well the concept detector performed. For a proper evaluation, we should use multiple concepts when comparing evaluation metrics of different instantiations of the concept detection framework. We can investigate the effect of individual components on concept detection performance by changing a single component at a time.

The applicability of our framework extends beyond the components in this thesis. New interest region detectors, region descriptors, learning algorithms, etc. can easily be plugged in. Furthermore, our framework can be extended to new types of features by adding extra feature extraction pipelines.

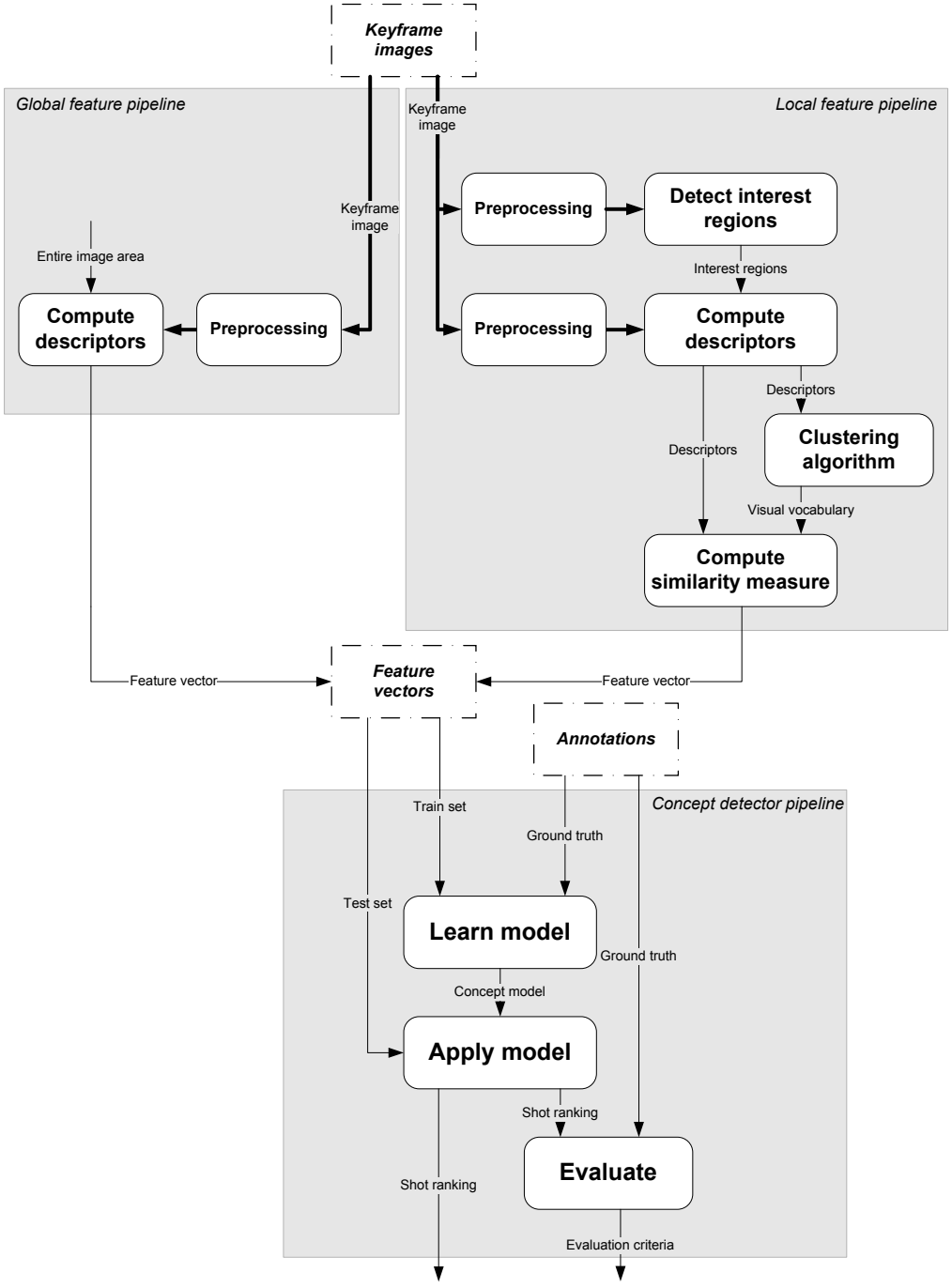


Figure 15: Concept detection framework. The framework consists of three pipeline. The global feature pipeline extracts global features. The local feature pipeline extracts features based on interest regions. The feature vectors resulting from feature extraction form the input to the concept detector pipeline. We learn a concept model from the feature vectors and a ground truth. Using the model we order unseen feature vectors by the likelihood that they contain the concept. We compute evaluation criteria over the resulting shot ranking.

5 Concept detection system

In this chapter, details are provided on how our concept detection system is implemented. In section 5.1, we discuss instantiations of the feature extraction pipelines. We then describe the implementation of components within the feature extraction pipeline in sections 5.2 through 5.4. In section 5.5, we describe the instantiation of the concept detector pipeline. Finally, in section 5.6, we discuss how to combine multiple features using feature fusion.

5.1 Feature extraction

Before feature extraction, we divide every video into camera shots. Camera shots are uninterrupted recordings of a camera. A shot boundary detection [34] is assumed to be available for all videos to be processed. Using these boundaries we can divide the video into shots. We extract features for all shots in a video. However, within a single shot there are many frames. For our features, only the most representative frame of a shot, the keyframe, is used. How to extract the most representative frame from a shot is an open issue, but often the middle frame of a shot can give reasonable results, avoiding frames with transition effects. The keyframes available to us use the MPEG keyframe closest to the middle frame of the shot [33]. Additionally, shots which are shorter than two seconds are discarded. These very short shots tend to be detected during long scene transitions and in commercials.

Having divided a video into shots and having reduced the shots to single keyframes, we instantiate the feature extraction pipelines within our concept detection framework (see figure 15 in chapter 4). For global features we instantiate the global feature extraction pipeline (see figure 16). For features based on interest regions we instantiate the local feature extraction pipeline (see figure 17).

In the global feature pipeline, shown in figure 16, keyframe images form the input

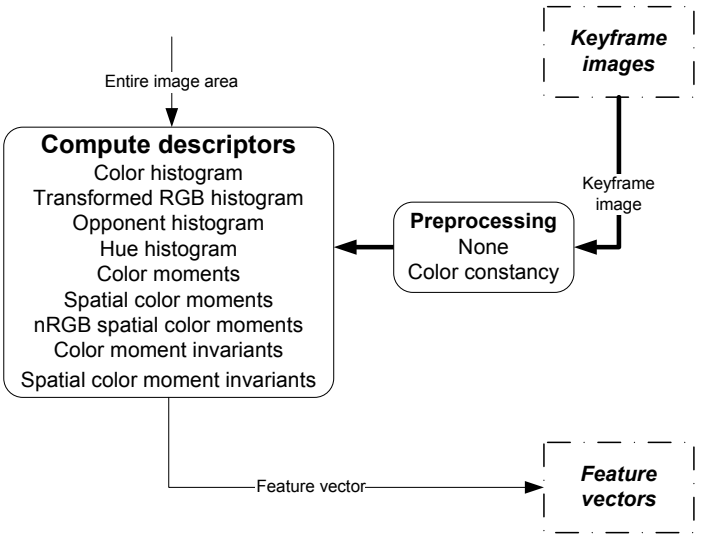


Figure 16: Pipeline for extraction of global features. A descriptor is computed over the entire keyframe image area. This descriptor can be used as a feature vector directly. Applying color constancy is an optional preprocessing step on the keyframe images. This is an instantiation of the global feature pipeline from figure 15.

to an optional preprocessing step: color constancy (section 3.3). This preprocessing step yields an image which has been normalized against the color of the light source in the image scene. Different descriptors can be computed over the entire image area of the (preprocessed) keyframe. There are 9 descriptors available in our system for global features. These descriptors will be discussed in section 5.3. Every descriptor yields a vector describing the keyframe. This vector is used as the feature vector describing the video shot. Hence, the dimensionality of global features is equal to the dimensionality of the descriptor.

In figure 17, the instantiation of the local feature pipeline is shown. Before we detect interest regions in the keyframe, we optionally preprocess the keyframe using color boosting. We have two different interest region detectors: Harris-Laplace and ColorHarris-Laplace. Harris-Laplace will be used without color boosting, while ColorHarris-Laplace will always be used with color boosting. We will refer to the latter as Colorboosted ColorHarris-Laplace. The detectors are discussed further in section 5.2. Descriptors are computed for all detected interest regions. As in the global pipeline, optionally keyframe images can be preprocessed using color constancy before descriptor computation. For interest regions, we have implemented 10 descriptors in our system. With a variable number of interest regions per keyframe, the descriptions of these regions need to be aggregated in order to obtain a fixed-length feature vector. Aggregation is performed by computing either a similarity measure or a dissimilarity measure between all descriptors of an image and the visual vocabulary. The visual vocabulary is constructed using either k-means clustering or radius-based clustering. Additional details on clustering and the dissimilarity measure are given in section 5.4.

5.2 Detectors

The Harris-Laplace scale invariant point detection algorithm, that was discussed in section 3.5.3, is given in figure 18.

The first part of the algorithm detects corners at multiple scales. The scales used have a $\sqrt{2}$ multiplication factor between them. The cornerness for a point must exceed threshold $\sigma_{threshold}$ to filter low maxima, which originate from noise or slightly curved edges. This part of the algorithm has a worst-case runtime complexity of $O(wh)$ with w the width of the image and h the height of the image, i.e. the runtime depends on the number of pixels.

The second part of the algorithm performs scale selection on all points from the first part. At scales near the scale at which they were detected, the Laplacian-of-Gaussian (equation 6) is computed. These scales are spaced $\sqrt[4]{2}$ apart. Only the range up to the previous/next possible detection scale is included. The scale at which the Laplacian-of-Gaussian attains a local maximum is the characteristic scale of the point. If the characteristic scale of the point is not in the limited range, then the point should have been detected at a different scale. If there is no local maximum, then the point is not scale invariant and hence rejected. For a point to be accepted, its maximal $|LoG|$ response should exceed threshold $t_{laplacian}$. This prevents points with a weak local structure from being accepted. The worst-case runtime complexity of this part of the algorithm is $O(n)$, with n the number of candidate points[2].

Based on the algorithm above, two scale invariant point detectors have been derived: Harris-Laplace and Colorboosted ColorHarris-Laplace.

Harris-Laplace operates on the intensity information from the image only, e.g. it does not use color information. The detector is described in section 3.5.3.

[2]We obtain this complexity under the assumption that the number of pixels that are involved in the computation of the $|LoG|$ are bounded by a constant. Since there is a maximum detection scale, the size of the Laplacian-of-Gaussian kernel, and thus the number of pixels involved, is indeed limited.

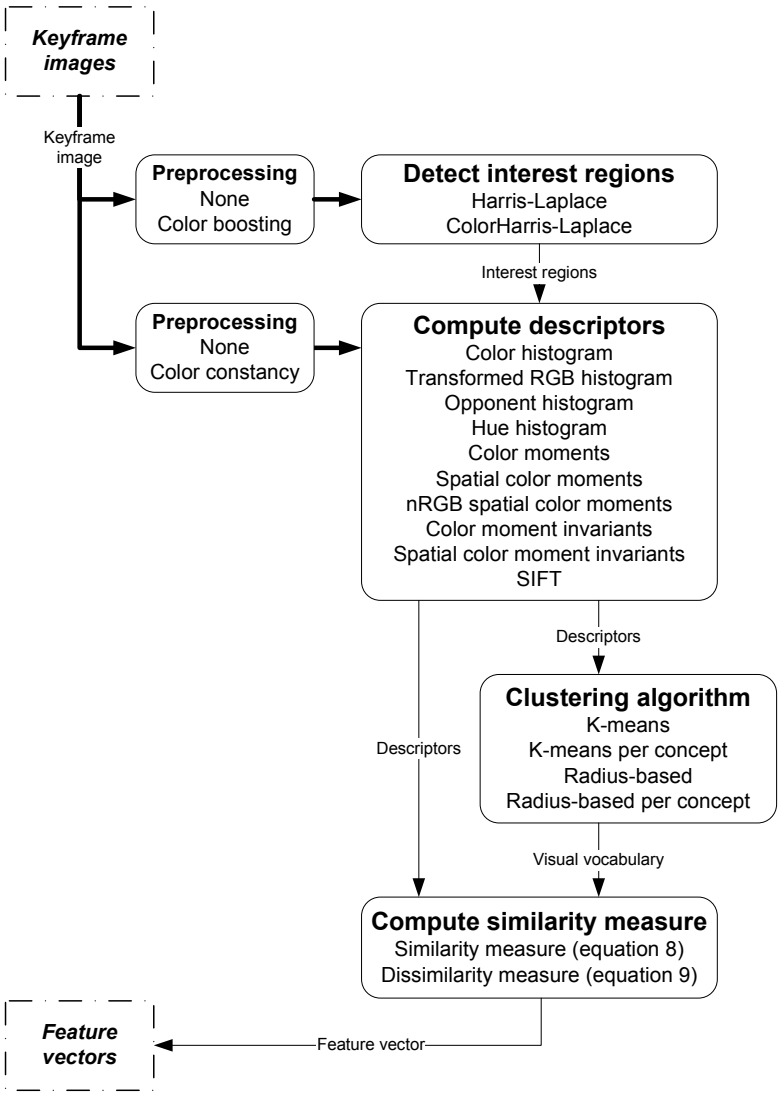


Figure 17: Pipeline for extraction of features based on interest regions. Descriptors are computed for all interest regions detected in the keyframe image. Representative descriptor prototypes are selected using a clustering algorithm. The dissimilarity between the prototypes and the descriptors forms the feature vector. Applying color constancy is an optional preprocessing step on the keyframe images. This is an instantiation of the local feature pipeline from figure 15.

function HarrisLaplaceDetector(image)
candidatePoints $\leftarrow \emptyset$
for scale **in** $\{\frac{3}{4}\sqrt{2}, 1\frac{1}{2}, 1\frac{1}{2}\sqrt{2}, 3, 3\sqrt{2}, 6, 6\sqrt{2}, 12\}$ **do**
  σ_I = scale
  σ_D = 0.7125 * scale
  cornerness $\leftarrow$ Compute *cornerness* (equation 5) **for all** image pixels
  points $\leftarrow$ FindLocal2DMaxima(cornerness)
  for all points p **do**
    if cornerness(p) $\geq t_{harris}$ **then**
      candidatePoints $\leftarrow$ candidatePoints $\cup\{(\text{p}, scale)\}$
    end if
  end for
end for
scaleInvariantPoints $\leftarrow \emptyset$
for point, detectionScale **in** candidatePoints **do**
  scalesToCheck $\leftarrow$ ScalesAround(detectionScale)
  valuesLoG $\leftarrow$ Compute $LoG(\text{point}, \sigma_n)$ (equation 6) **for all** σ_n **in** scalesToCheck
  if HasLocalMaximum(valuesLoG) **then**
    response $\leftarrow$ value $\in$ valuesLoG corresponding to local maximum
    characteristicScale $\leftarrow$ scale $\in$ scalesToCheck corresponding to local maximum
    if response $\geq t_{laplacian}$ **then**
      scaleInvariantPoints $\leftarrow$ scaleInvariantPoints $\cup$ {(point, characteristicScale)}
    end if
  end if
end for

Figure 18: Harris-Laplace scale invariant point detection algorithm.

Colorboosted ColorHarris-Laplace applies color boosting (section 3.5.4) to the keyframe image before detecting interest regions. The detector is extended to use multiple channels as described in section 3.5.5. The color channels used in the detector are the R, G and B channels of the RGB color space (section 3.2.1).

5.3 Region descriptors

The region descriptors used are described in section 3.6. In this section their implementation is described. The following descriptors are available in our system:

- RGB histogram (section 3.6.1, implementation see section 5.3.1)
- Transformed RGB histogram (section 3.6.2, implementation see section 5.3.2)
- Opponent histogram (section 3.6.1, implementation see section 5.3.3)
- Hue histogram (section 3.6.3, implementation see section 5.3.4)
- Color moments (section 3.6.4, implementation see section 5.3.5)
- Spatial color moments (section 3.6.4, implementation see section 5.3.6)
- Spatial color moments with normalized RGB (section 3.6.4, implementation see section 5.3.7)
- Color moment invariants (section 3.6.4, implementation see section 5.3.8)
- Spatial color moment invariants (section 3.6.4, implementation see section 5.3.9)
- SIFT (section 3.6.5, implementation see section 5.3.10)

Descriptors can be computed over the complete keyframe (for global features) and for interest regions (for local features). The keyframes have a size of 352 by 240 pixels. To remove black borders 15 pixels are cropped on every side of the frame, leaving an image of 322 by 210 pixels.

SIFT is the only descriptor which cannot be computed over the complete keyframe. The implementation we use [28] is limited to circular or elliptical regions of limited size. Without the source code available this descriptor could not be computed for the entire rectangular image area.

In chapter 3 it has been shown that all our interest regions are defined by their position and scale[3]. To achieve comparable descriptors for different scales, all regions are resampled to a uniform square patch of size 60 by 60 pixels. The advantage of using a uniform size is that it is not necessary to add scale invariance to the descriptors explicitly: the resampling makes regions of different size comparable. The resampling is proportional to the scale of the region: the circle with distance σ from the interest point is at the same location in the 60 by 60 patch for all scales.

Now the implementation details of all descriptors will be briefly described.

5.3.1 RGB histogram

The RGB histogram is a combination of three 1D histograms based on the R, G and B channels of the RGB color space. For each channel, 15 bins are equally spaced over the value range [0..1]. As a result, this descriptor has 45 bins in total.

[3]Besides position and scale, orientation could also be a part of the definition of an interest region. However, for rotationally invariant descriptors, the orientation of the region is not important. For SIFT, which uses the orientation, we see orientation assignment as part of the descriptor computation process. Therefore, it is not used in our definition of the interest region.

5.3.2 Transformed RGB histogram

The transformed RGB histogram is a combination of three 1D histograms based on the R, G and B channels of the RGB color space. However, the channels have been normalized as described in section 3.6.2. This normalization makes the histogram invariant to light intensity, color of the light source, shading and shadows.

Due to the normalization, the samples in each channel have a zero-mean and a standard deviation of one. If the channels are normally distributed, then 98% of the samples are expected to lie in the range $[-2\sigma, 2\sigma]$. While our data is not normally distributed, most samples do lie within this range. We have applied normalization to 1000 randomly selected images from our dataset and have found that more than 99% of the samples do lie within two standard deviations of the mean. So, as our histogram intervals we have chosen to divide the range $[-2\sigma, 2\sigma]$ into 15 equally sized bins. The samples which fall outside this range (e.g. less than 1% of the pixels) are placed in the upper or lower histogram bin, depending on the sign of the sample.

5.3.3 Opponent histogram

The opponent histogram is a combination of three 1D histograms based on the $o1$, $o2$ and $o3$ channels of the opponent color space. There are two advantages of this descriptor over RGB histograms. Firstly, the assumption of decorrelated channels (see section 3.6.1) holds. Secondly, the light intensity is isolated in channel $o3$ and color information is isolated in channels $o1$ and $o2$.

The histogram intervals for the opponent color space have ranges different from the RGB model. Channel $o1$ and $o2$ have bins spaced equally over the range $[-\frac{1}{2}, \frac{1}{2}]$. Any samples outside this range are placed in the outer bins. Channel $o3$ has its bins spaced equally over the range $[0, \sqrt{3}]$. It is not possible for samples to fall outside this range.

5.3.4 Hue histogram

The hue histogram is a 1D histogram based on the hue channel of the HSV color space. To address instability of the hue in dark and/or grey areas, the hue samples are *weighed* by the saturation. This is different from other histograms where the weight of each sample is equal to 1. The hue channel is divided into 37 equally spaced intervals.

Our implementation corresponds to the hue histogram descriptor introduced by Van de Weijer [47]. That descriptor is specifically designed for interest regions. When applied to interest regions, each hue sample is also weighed by its distance to the center of the interest region. Samples near the center are given a higher weight than those near the boundary of the region. The weighing is achieved using a Gaussian weighing mask. Since the Gaussian is rotationally symmetric, it ensures rotational invariance for the descriptor.

It does not make sense to use the Gaussian weighing mask when the hue histogram is applied to an area which is not an interest region. For example, when applied to a whole image frame, which is 322 by 210, the weighing mask will neglect the borders of the image. We leave out the Gaussian weighing step at the global level, as all other descriptors do not use this either.

5.3.5 Color moments

Generalized color moments can be used to characterize the color distribution in an image (see section 3.6.4). When using generalized color moments up to degree 2 and order 0, there are 10 color moments: M_{00}^{000}, M_{00}^{100}, M_{00}^{010}, M_{00}^{001}, M_{00}^{200}, M_{00}^{110}, M_{00}^{020}, M_{00}^{011}, M_{00}^{002} and M_{00}^{101}. The moment M_{00}^{000} is not included, because it is always equal to 1. Hence the color moment descriptor has 9 useful dimensions.

5.3.6 Spatial color moments

The spatial color moment descriptor uses all generalized color moments (section 3.6.4) up to degree 2 and order 1: it includes a spatial component over ordinary color moments. This leads to three possible combinations for the order: $M_{00}^{abc}, M_{10}^{abc}$ and M_{01}^{abc}. Combined with the 9 useful combinations for degrees, the spatial color moment descriptor has 27 dimensions.

5.3.7 Spatial color moments with normalized RGB

Spatial color moments (section 3.6.4) do not have invariance properties; they merely describe the (spatial) color distribution. Invariance to light intensity changes can be added by converting the image data to normalized RGB before calculating descriptors. As the third channel of normalized RGB is redundant, color moments over only 2 channels are needed. This leads to 15 color moments for the descriptor.

5.3.8 Color moment invariants

Color moment invariants can be constructed from generalized color moments (see section 3.6.4). We will use the 3-band invariants with order 0 from Mindru *et al* [29] as our color moment invariants. To be comparable, we also use the $\tilde{C}_{02}$ invariants. This gives a total of 9 color moment invariants.

5.3.9 Spatial color moment invariants

We will use all 3-band invariants from Mindru *et al* [29] as our color moment invariants. Over the normal color moment invariants, this yields an additional 15 invariants. This gives a total of 24 color moment invariants.

5.3.10 SIFT

To compute SIFT descriptors (see section 3.6.5), we use the implementation by Mikolajczyk [25]. This implementation was used for the evaluation of region descriptors in [28]. It corresponds to the description of the SIFT algorithm in [21].

5.4 Clustering

In section 3.7 two algorithms are given for constructing a visual vocabulary of representative descriptors: the k-means clustering algorithm and the radius-based clustering algorithm. The implementations are as follows.

5.4.1 K-means clustering

K-means is a very well-known clustering algorithm [50]. It is a simple, iterative clustering approach. The number of clusters n should be given beforehand. Initially all cluster centers are initialized by picking points randomly from the dataset. For every point from the dataset the distance to each cluster center is calculated. The point is assigned to the cluster whose center is closest. Once all points have been assigned, cluster centers are updated by taking the average of all points in the cluster. This procedure loops until the cluster centers no longer change or a fixed number of iterations has passed.

For k-means clustering we use the implementation available in Matlab [23]. The worst-case runtime complexity of a single k-means iteration is $O(kn)$ with k the number of clusters and n the number of data points.

```
clusters ← ∅
while there are points outside the radius of existing clusters do
  smallestMaxDistance ← 0
  smallestMaxDistancePoint ← 0⃗
  for all points p do
    maxDistance ← 0
    for all points q with p ≠ q and q not within radius of existing clusters do
      d ← EuclidianDistance(p, q)
      if d > maxDistance then
        maxDistance ← d
      end if
    end for
    if maxDistance < smallestMaxDistance then
      smallestMaxDistance ← maxDistance
      smallestMaxDistancePoint ← p
    end if
  end for
  clusters ← clusters ∪{smallestMaxDistancePoint}
end while
realClusters ← ∅
for all clusters c do
  if NumberOfPointsInsideRadius(points, c, radius) ≥ minimumCoverage then
    realClusters ← realClusters ∪{c}
  end if
end for
```

Figure 19: Radius-based clustering algorithm.

5.4.2 Radius-based clustering

Radius-based clustering assigns all data points within a fixed radius of similarity r to one cluster, where r is a parameter. Given an instantiation of the r parameter, the algorithm in figure 19 is run. It selects the point with the lowest maximal distance to all other points as a candidate cluster. All points within distance r of this point are then discarded in the search for more clusters, since these points fall within the selected cluster. After all data points fall within one of the candidate clusters, a pruning step is performed. Only clusters which have a minimum number of points within their radius, *minimumCoverage*, are retained. If a run of the algorithm for radius r does not yield enough clusters, the algorithm is rerun using a smaller radius.

For the radius-based clustering, a C++ implementation by Jan van Gemert (UvA) has been integrated into our system. A single run has a worst-case runtime complexity of $O(n^3)$ with n the number of data points. An example worst case is when the distance between all points is greater than r. This results in all points becoming candidate clusters. In practice, the worst case does not occur if r is large enough. An initial estimate of the radius r can be made based on the *smallestMaxDistance* from the first iteration of the while loop.

5.4.3 Visual vocabulary

For the construction of the visual vocabulary, two different methods are considered. The first method searches for clusters in the descriptor space of 1000 images from the dataset. Due to limitations of the k-means implementation, we have to constrain our search to

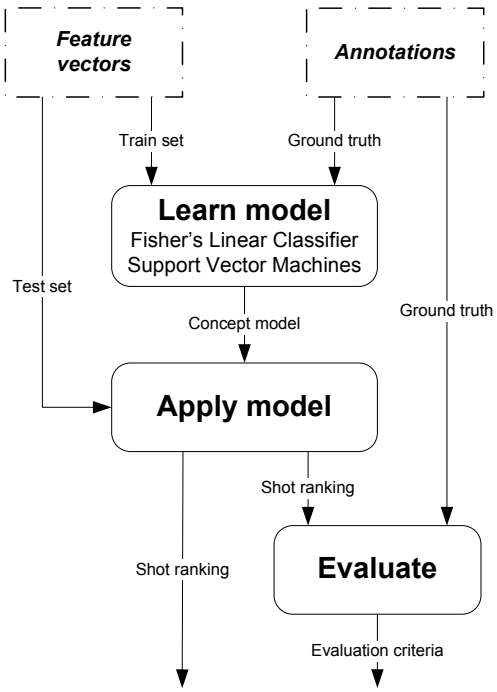


Figure 20: Pipeline for learning a concept detector from feature vectors and a ground truth for the concept. This is an instantiation of the concept detector pipeline from figure 15.

195 clusters. The second method searches for clusters on a *per-concept* basis. For every concept clustering is performed on up to 1000 images belonging to the concept. For all concepts, we search for at least 10 clusters. We will use the 39 TRECVID concepts (see section 6.2). Depending on the descriptor and the data clustered on, per-concept clustering has a visual vocabulary size of 400 to 425 clusters.

5.4.4 Similarity measures

Besides selection of the clustering algorithm (k-means versus radius-based clustering) and the clustering method (normal clustering or per-concept clustering), we can also select the similarity measure to use. In section 3.7.2, the similarity measure (equation 8) and the dissimilarity measure (equation 9) are presented. In our experiments we will compare the different options to chose from (see section 7.2).

5.5 Concept detector pipeline

The concept detector pipeline of our framework (see chapter 4) takes feature vectors together with a ground truth annotation as input. Figure 20 shows the instantiation of the concept detector pipeline.

The concept model can be learned using either SVM or Fisher's Linear Classifier (see section 3.8). Parameters of the SVM algorithm are chosen by performing 3-fold cross-validation over 126 different weight combinations. Fisher's Linear Classifier does not require parameter tuning. In terms of computation SVM takes more than a hundred times longer than Fisher's Linear Classifier.

5.6 Feature fusion

The concept detector pipeline in our concept detection framework can operate on arbitrary feature vectors. When we want to combine multiple features, it is straightforward to concatenate the feature vectors of different features into one long feature vector and feed this into the concept detector pipeline. We refer to this kind of fusion as *feature fusion*. The advantage of this fusion method is that the machine learning algorithm implicitly learns which elements of the feature vector are important. An alternative fusion method, *descriptor-level fusion*, does not have this advantage. Descriptor-level fusion concatenates multiple descriptors of the same region. When combining different region descriptors, we need to introduce weighting parameters to adjust for different scales of the descriptor spaces. Obtaining good or optimal weights is a non-trivial problem which lies beyond the scope of this thesis. Therefore, we will not use descriptor fusion. A third fusion method operates on the level of the shot ranking. *Shot ranking fusion* combines multiple ordered lists of shots, i.e. the shot rankings. The shots are reranked depending on their position in the lists. Because we focus on features and feature extraction, the fusion of shot rankings is beyond the scope of this thesis.

6 Experimental setup

In this chapter we sketch the experimental setup to evaluate the components of our concept detection framework, as detailed in chapter 4, using the implementation sketched in chapter 5.

6.1 Experiments

In experiment 1, we consider two different machine learning algorithms in the concept detector pipeline. In experiment 2, we compare methods for building and using a visual vocabulary in the feature extraction pipeline. In experiment 3, we compare local and global features using different descriptors. In experiment 4, we investigate the effect of color constancy on the performance of different descriptors. Finally, in experiment 5, we combine our features using feature fusion. In figure 21, we show an overview of the position of our experiments within the concept detection framework.

6.2 Mediamill Challenge

Repeatable experiments, using common datasets, are very important for research fields to progress further. As discussed in section 2.2.1, the Mediamill Challenge by Snoek *et al* [42] provides an annotated video dataset, based on the training set of TRECVID 2005. The annotation is referred to as a ground truth: a video shot contains a semantic concept if and only if it has been annotated as such. However, an annotation is never perfect, as mistakes are made during annotation.

Snoek *et al* have defined five repeatable multimodal experiments based on this common dataset. They decompose automatic semantic concept detection into a number of components, for which they provide a standard implementation. This provides an environment to measure which factors affect performance most. It allows for an in-depth understanding of the problem at hand. Since our framework uses visual information only, we use only the first of the five experiments:

Given a visual feature vector, $\vec{v}_i$, learn for each of the semantic concepts ω_j a ranked list ρ_j, where feature vectors are defined for shots i.

The TRECVID 2005 training set [33] of 85 hours of video is divided into a Challenge training set (70% of the data or 30993 shots) and a Challenge test set (30% of the data or 12914 shots). In figure 2 in section 2.2.1, an impression of the dataset was given. The dataset consists of television news from November 2004 broadcasted on several TV channels in different languages. In table 4 an overview of the broadcasts is given.

The Mediamill Challenge is defined for 101 concepts. However, in TRECVID 2006 a subset of 39 concepts are used. As the results from this thesis have been included in the Mediamill semantic video search engine for TRECVID 2006 (see appendix A), we will only focus on the 39 concepts that are part of a TRECVID 2006 submission.

The learning step in the above experiments uses the training set for learning. The learned concept model is used to rank shots from the test set. Thus, both the Challenge baseline and our system output a ranked list of shots ρ_j for a semantic concept ω_j.

6.3 Evaluation criteria

For the evaluation of ranked lists many metrics exist. Important notions are *precision* and *recall*. Precision is defined as the percentage of items returned that are correct:

$$precision = \frac{|R \cap \rho|}{|\rho|}$$

with R the set of relevant items, ρ the set of items returned and $|X|$ the size of set X.

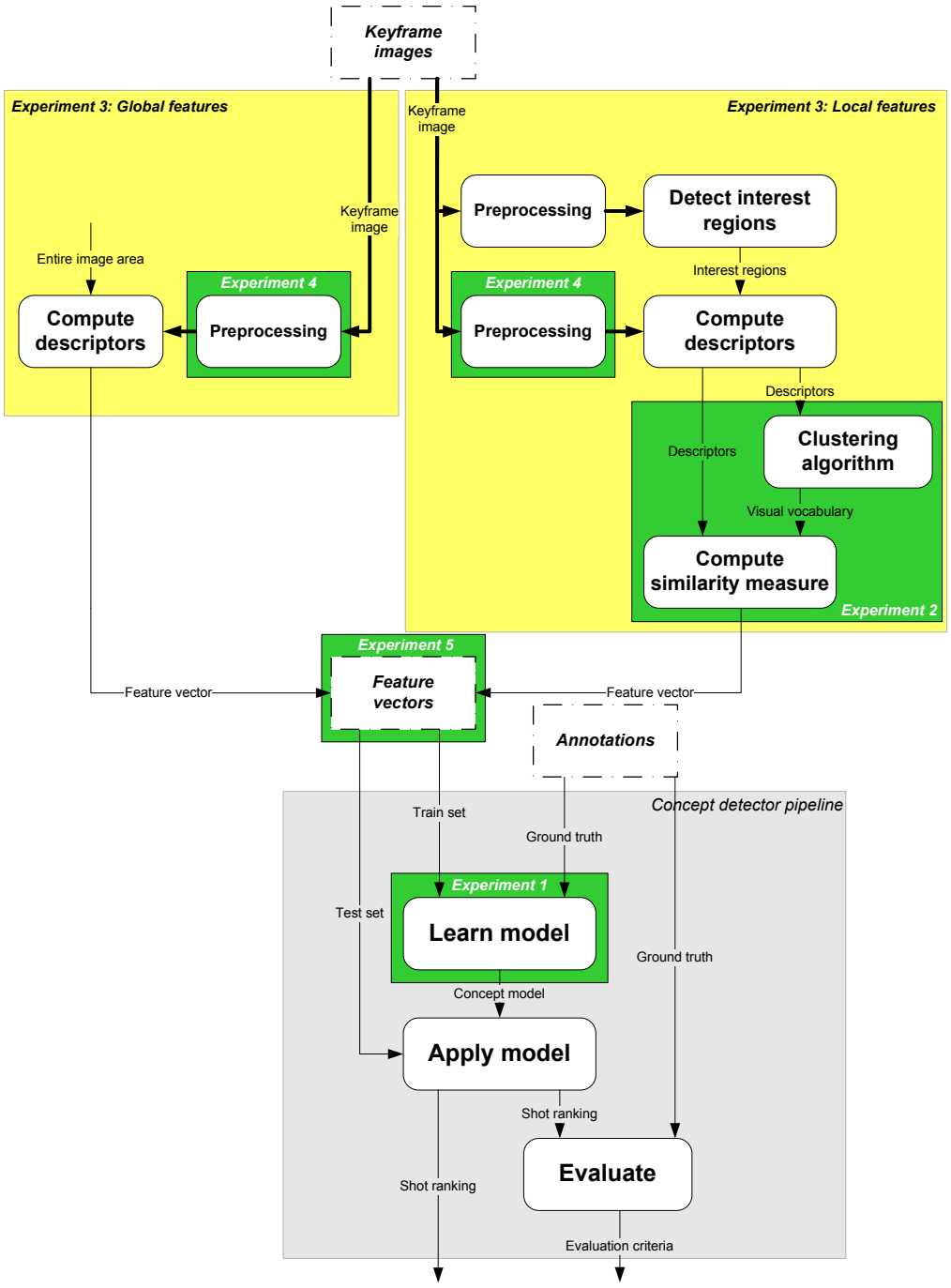


Figure 21: The concept detection framework from chapter 4 where the experiments performed in this thesis have been indicated.

Language	Episodes	Source	Program	Length
Arabic	7	LBC	LBC Nahar	6h 46min
Arabic	5	LBC	LBC News (1pm)	2h 5min
Arabic	14	LBC	LBC News (8pm)	13h 34min
Chinese	13	CCTV4	Daily News	12h 19min
Chinese	11	CCTV4	News3	5h 5min
Chinese	10	NTDTV	NTD News (12pm)	4h 42min
Chinese	9	NTDTV	NTD News (7pm)	4h 15min
English	11	CNN	Aaron Brown	10h 42min
English	9	CNN	Live From	4h 11min
English	15	NBC	NBC Philadelphia	7h 5min
English	7	NBC	Nightly News	3h 18min
English	11	MSNBC	MSNBC News (11am)	5h 12min
English	15	MSNBC	MSNBC News (1pm)	7h 3min
Total	137			**86h 17min**

Table 4: Overview of the news broadcasts present in the TRECVID 2005 training set. In literature the total size of this dataset is referred to as 'about 85 hours'.

Recall is the fraction of all relevant material that is returned:

$$recall = \frac{|R \cap \rho|}{|R|}$$

From these definitions it is obvious that one wants both high precision and recall for retrieval systems. However, there is a trade-off between precision and recall. By returning more results, one can increase the fraction of all correct material retrieved, but this will probably lower precision by adding extra incorrect results. The other way around, by returning only results which are almost certainly correct, the precision can be increased, but this will lower the fraction of all correct material retrieved. This trade-off is visualized for two systems in the precision-recall curve in figure 22. The question arises which of the two systems is better. Hence, a measure is needed that optimizes both precision and recall. One such measure is average precision.

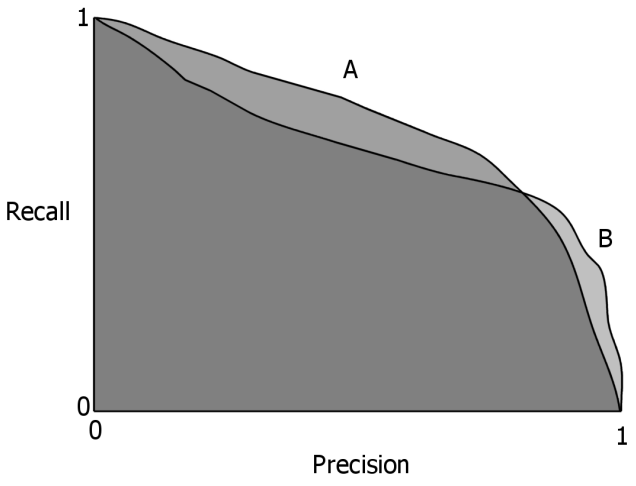


Figure 22: Precision-Recall curve sketch. A retrieval system can return multiple results with different precision/recall combinations. Together these form a precision-recall curve. Here the curves of two retrieval systems (A and B) are sketched. The average precision metric is proportional to the area under the precision-recall curve. The average precision of system A is higher than the average precision of system B.

The average precision is a single-valued measure that is proportional to the area under a precision-recall curve. This value is the average of the precision over all shots judged relevant. Let $\rho^k = \{l_1, l_2, ..., l_k\}$ be the ranked list of shots from answer set A[4]. At any given rank k let $|R \cap \rho^k|$ be the number of relevant shots in the top k of ρ, where R is the set of relevant shots and $|X|$ is the size of set X. Average precision, AP, is then defined as:

$$AP(\rho) = \frac{1}{|R|} \sum_{k=1}^{|A|} \frac{|R \cap \rho^k|}{k} \psi(l_k)$$

with indicator function $\psi(l_k) = 1$ if $l_k \in R$ and 0 otherwise. $|A|$ is the size of the answer set, e.g. the number of items present in the ranking. As can be seen from the k in the denominator, it is very important to have correct results for highly ranked shots.

From the precision-recall curves shown in figure 22, we observe that system A will have a higher average precision than system B, because the area under curve A is larger than the area under curve B.

Average precision is the standard in TRECVID evaluations for determining the accuracy of ranked concept detection results. We will also adopt this metric for our experiments. When performing experiments over multiple concepts, the average precisions of the individual concepts can be aggregated. This aggregation is called mean average precision (MAP). MAP is calculated by taking the mean of the average precisions.

[4]For our experiments, the answer set can contain shots from the test set only.

7 Results

In this chapter, we evaluate the components of our concept detection system from chapter 5. The system is an implementation of our concept detection framework from chapter 4. The experimental setup is described in chapter 6. We detect the 39 semantic concepts as listed in section 2.2. Concept detector performance is measured using average precision, which was discussed in section 6.3.

7.1 Machine learning algorithms

We plot the concept detection results using SVM and Fisher's Linear Classifier as machine learning algorithms in figure 23. For many concepts, the differences between the algorithms are small, but most often the difference is in favor of SVM. Especially the SVM algorithm performs substantially better for concepts which occur in a limited number of styles, such as computer-generated graphics (*maps*, *weather* reports, *charts*) and *sports* and *animal*. Computer-generated graphics tend to have a consistent look within a single television channel, therefore the number of styles is limited. A limited number of sports are broadcast and sport events within the same sport look similar. Many animals present in the dataset occur within commercials, which are repeated many times. For three concepts, *people*, *outdoor* and *entertainment*, the difference is clearly to the advantage of Fisher's Linear Classifier. These concepts are very open-ended: there are always new people, outdoor scenes and entertainment shows which look very different from existing occurrences. We have observed a similar pattern using a different feature (Harris-Laplace detector with hue histogram descriptor) as well.

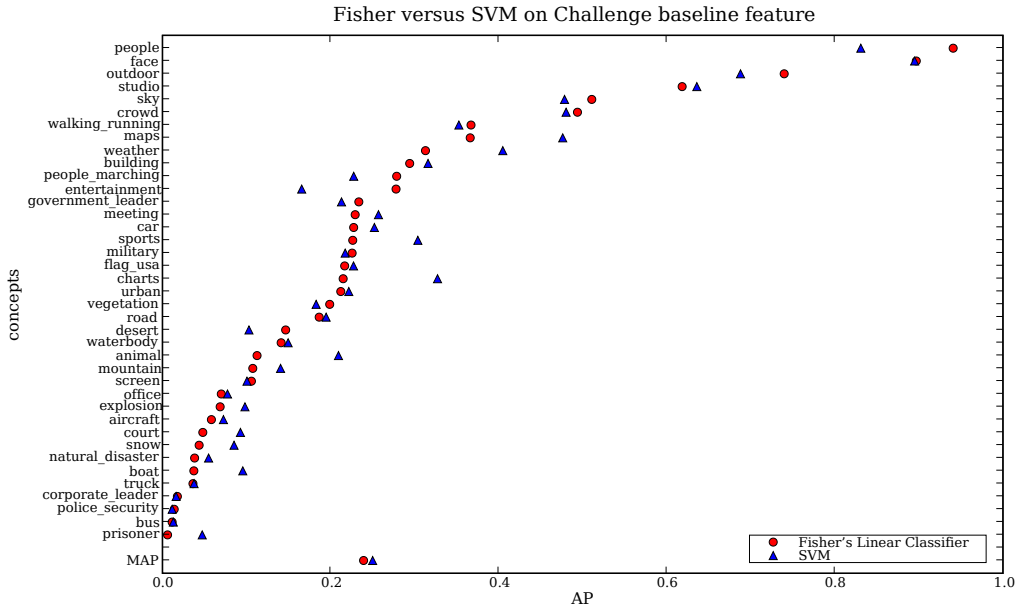


Figure 23: Performance of Fisher's Linear Classifier and SVM machine learning algorithms on visual features from the Mediamill Challenge. Concepts are ordered by their performance using Fisher's Linear Classifier. This order will be used throughout this chapter. At the bottom the MAP over all concepts is shown.

Given that the overall performance difference is less than 5%, the usage of SVM over Fisher's Linear Classifier does not warrant the hundredfold increase in computation time. Hence, in our experiments, we employ Fisher's Linear Classifier. In the remainder of this chapter, we use the performance of the visual features of the Mediamill Challenge using Fisher's Linear Classifier as the baseline performance.

7.2 Visual vocabulary

In section 5.4, methods for constructing and using a visual vocabulary for local features are discussed. There are three different choices to be made:

- Clustering algorithm (k-means clustering or radius-based clustering)
- Usage of per-concept clustering
- Similarity measure (equation 8) or dissimilarity measure (equation 9)

In figure 24, an overview of the overall performance for all possible combinations is given. The results are obtained using the Harris-Laplace interest region detector and the SIFT descriptor. We observe that the dissimilarity measure outperforms the similarity measure, independent of the other choices made. The similarity measure suffers from data sparseness: with on average 250 descriptors, there will be clusters with no descriptors assigned to them. The dissimilarity measure, which averages the distance to all descriptors, does not have this problem. Because of the clear advantage of the dissimilarity measure, we leave out the results of the similarity measure in the more detailed plot in figure 25.

Using clustering on a per-concept basis improves the performance over normal clustering. This can be partially explained by the fewer number of clusters for the normal clustering (195 versus approximately 400), but this disadvantage stems from memory limitations of the k-means clustering algorithm implementation. In the detailed plot,

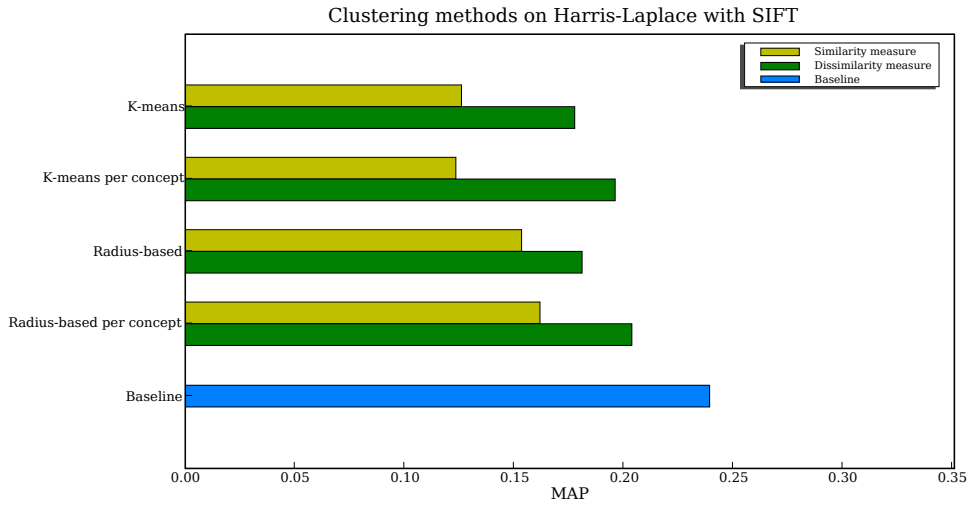


Figure 24: Performance overview of visual vocabulary construction methods (k-means versus radius-based clustering, usage of per-concept clustering) and usage methods (similarity measure versus dissimilarity measure). Results obtained using Harris-Laplace interest region detector and SIFT descriptor.

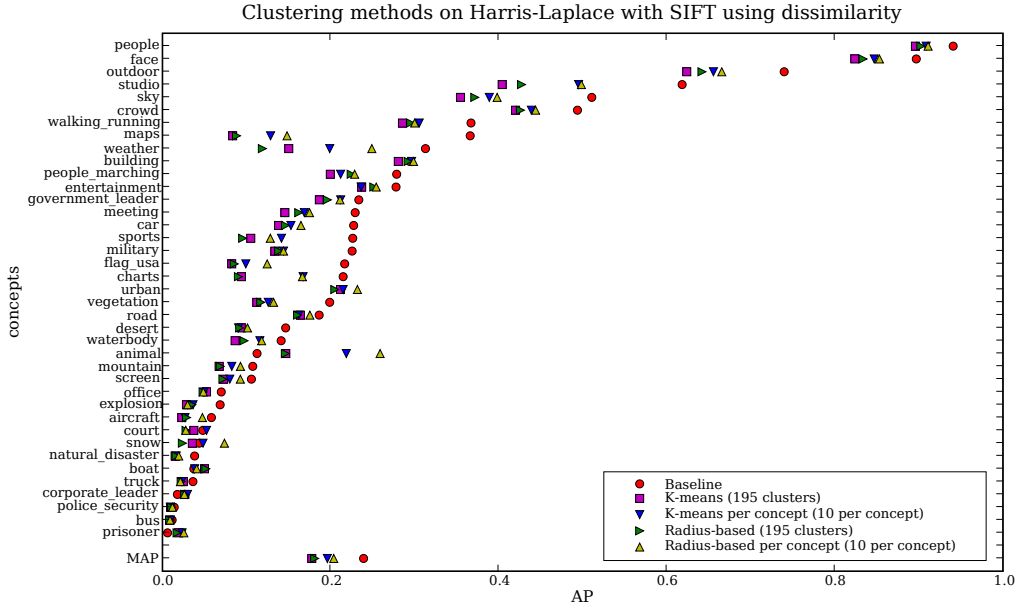


Figure 25: Detail plot of figure 24 results. Only the baseline and results which use the dissimilarity measure are shown.

it can be observed that, if the choice of clustering algorithm (k-means or radius-based) is already made, it is always advantageous to use per-concept clustering (except for an insignificant difference for the concepts *office* and *truck*). Thus, per-concept clustering gives better performance. In addition, it can also be extended to include many more concepts, where normal clustering already encounters scalability problems.

Given that the dissimilarity measure and per-concept clustering are the best choices, we only need to select the best clustering algorithm. In terms of overall performance, radius-based clustering is better than k-means clustering. Looking at individual concepts, radius-based clustering is almost always better. There a significant difference in favor of k-means for *sports* and *court* only.

In conclusion, when using visual vocabularies, radius-based clustering on a per-concept basis with a dissimilarity measure outperforms the other possible choices. Therefore, in the remainder of this chapter, we will use this method for local features.

7.3 Global and local features

In section 5.1, we distinguished two feature extraction methods: global feature extraction and local feature extraction. Global features are descriptions computed over the entire image area. Local features are an aggregation of descriptions of many interest regions. We compare two interest region detectors (see section 5.2): Harris-Laplace and Colorboosted ColorHarris-Laplace. We also compare the above when combined with different region descriptors (see section 5.3).

In the results from figure 26, we observe that local features are significantly better than global features. However, given that the dimensionality of global features is about 8 times lower, they perform quite well. Between the two types of local features (Harris-Laplace and Colorboosted ColorHarris-Laplace), there is no clear difference. On a per-concept basis, we observe that for certain concepts Harris-Laplace is better, for others Colorboosted ColorHarris-Laplace. For which concepts the detector is best also depends

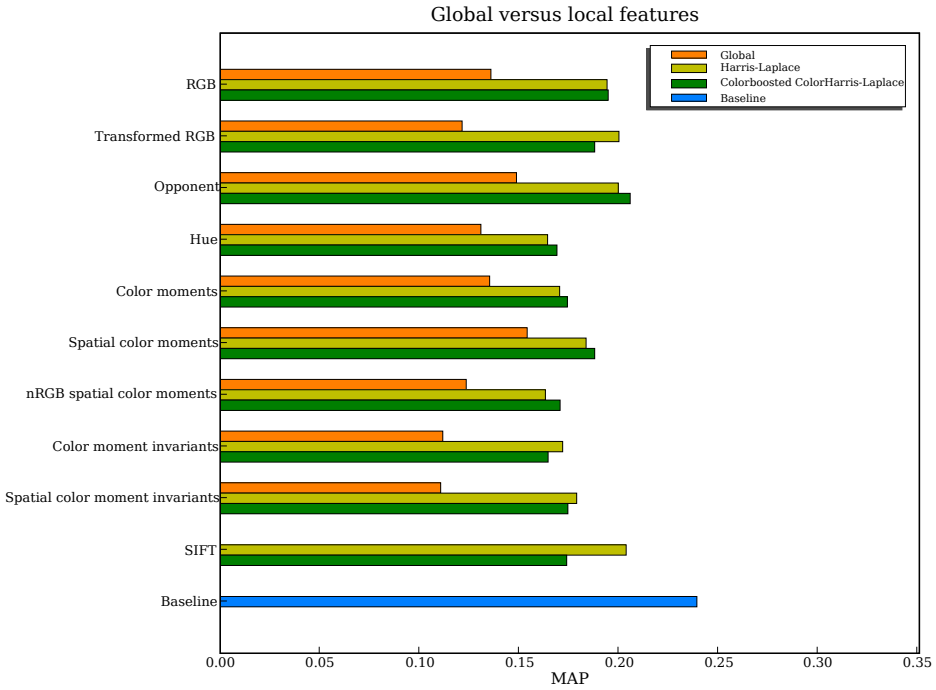


Figure 26: Performance results for global and local features paired with different descriptors. For local features there are two separate interest region detectors: Harris-Laplace and Colorboosted ColorHarris-Laplace.

on the descriptor used. As an example, for the concept *sky*, Colorboosted ColorHarris-Laplace is better in combination with the opponent histogram, the hue histogram and spatial color moments, but not in combination with the transformed RGB histogram. So, when the computational resources are available, we should use both types of local features or select the best feature on a per-concept basis.

In figure 26, we can also compare the different descriptors. For global features, spatial color moments perform best; followed by the opponent histogram and the RGB histogram. For local features, the SIFT descriptor and the opponent histogram perform best. They are closely followed by the transformed RGB histogram and the RGB histogram. Slightly farther behind are spatial color moments.

The addition of spatial information in spatial color moments improves results over color moments. However, computing spatial color moments on normalized RGB reduces overall performance. Apparently the properties against which additional invariance is achieved, i.e. light intensity, shadows, shading and viewing geometry, discriminate certain video shots better. The same argument can be made for the color moment invariants, which are invariant to light intensity and illumination color, and the hue descriptor, which does not contain any intensity information.

In conclusion, the best features to use are local features with the SIFT descriptor or the opponent histogram descriptor. However, these features do not exceed baseline performance. In section 7.5, we will combine different features, which does improve over the baseline.

7.4 Color constancy

In our feature extraction pipeline (section 5.1), we can choose to apply color constancy before descriptor computation. Results with and without color constancy are shown in figure 27.

The addition of color constancy does not change overall performance for descriptors which are already color constant: the transformed RGB histogram, color moment invariants and spatial color moment invariants are all unchanged. The SIFT descriptor does not change either. Color constancy is not expected to affect performance for a descriptor which uses intensity information only. However, for all other descriptors, the overall performance drops. The rationale for including color constancy is that it should increase performance by providing invariance to the illumination color. So, it is better not to use color constancy, because it decreases performance.

When applying color constancy, several shots from a commercial are mistakenly ranked very high. Due to the artificial colors found in the commercial, the estimation of the illumination color fails. At the top in figure 28, one such keyframe and its color constant version from the concept *maps* are shown. After normalizing with the wrong

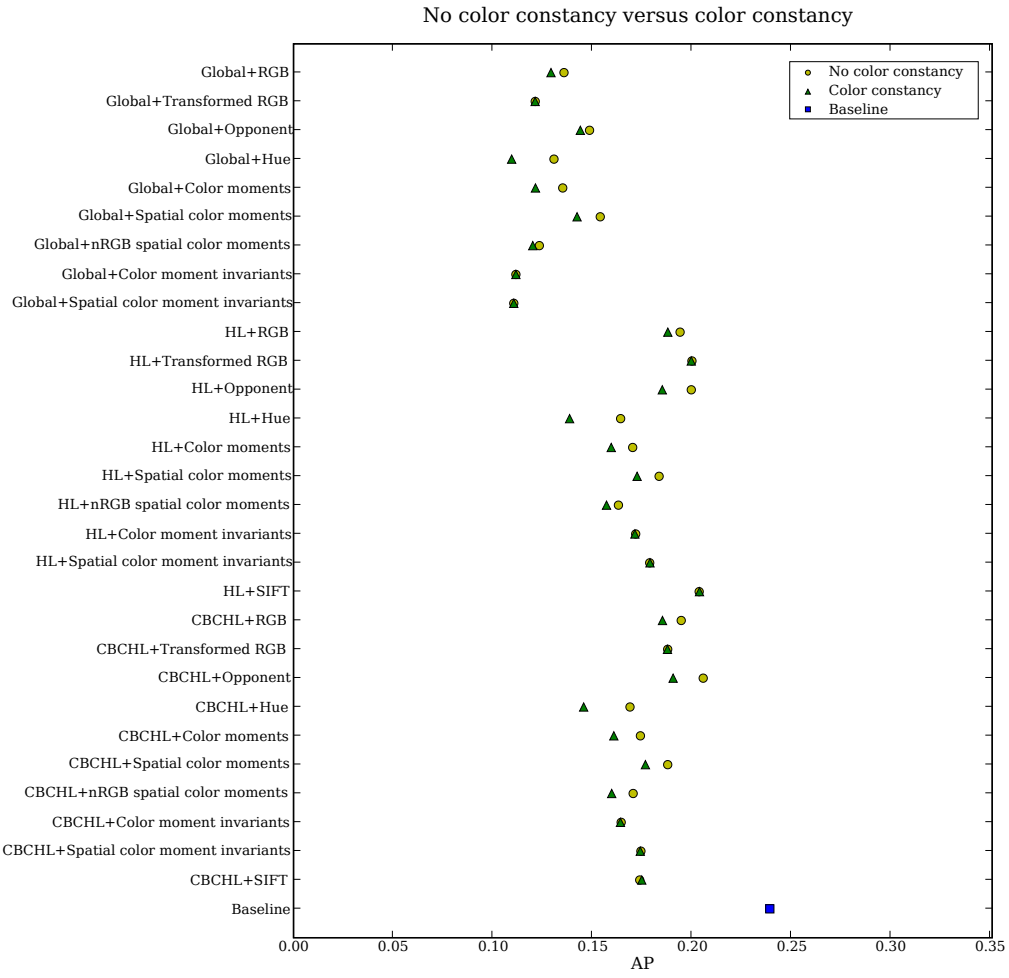


Figure 27: Performance results of features with and without color constancy. HL is shorthand for Harris-Laplace. CBCHL is shorthand for Colorboosted ColorHarris-Laplace.

Figure 28: On the left, two keyframes from shots which are highly ranked when using color constancy are displayed. On the right, color constant versions of the keyframes are displayed.

illumination color, the keyframe contains the same green typically found in maps. In other cases, the illumination has been estimated correctly, but the reduced discriminability makes the keyframe in the color constant keyframe similar to those that occur frequently for this concept. An example is shown at the bottom in figure 28: an outdoor music event is retrieved for the concept *studio*, which has a blueish background that is very similar to backgrounds that frequently occur for this concept. Especially for the concepts *studio*, *maps*, *weather* and *entertainment* there are significant decreases in performance due to color constancy. For the other concepts there are only small changes in performance, both positive and negative. We will refrain from using color constancy in the remainder of our experiments.

7.5 Feature fusion

In section 7.3, we observed that our features individually do not exceed baseline performance. However, we noted that the best feature to use differs on a per-concept basis. When we use feature fusion (see section 5.6), we can let the machine learning algorithm building the concept detector implicitly learn which elements of the feature vector to use.

We have tried all combinations of any two features from figure 26. In figure 29 results of a subset of all combinations is shown. We observe that performance of Harris-Laplace with the SIFT descriptor combined with the baseline gives the best performance. The best combinations which include only our features contain Harris-Laplace with the SIFT descriptor. Therefore, only combinations involving this feature and the baseline are shown in figure 29. The best combination which does not include the baseline is Harris-Laplace with SIFT descriptor combined with Colorboosted ColorHarris-Laplace.

When we compare the performance of individual features to fused features, we observe that good individual features fuse well. However, we note that features should be sufficiently orthogonal to provide substantial gains. As an extreme example of this, we

fuse Harris-Laplace with SIFT with itself. For this combination performance is lower than Harris-Laplace with SIFT only. Another example is Harris-Laplace with SIFT and Colorboosted ColorHarris-Laplace with SIFT: these features do not complement each other. Fusion with global spatial color moments outperforms certain local features. However, we should remember that the feature fused with already contains local information.

In conclusion, the combination of features through feature-level fusion provides significant performance improvements. Features that perform well on their own are good candidates for fusion, as long as they are orthogonal.

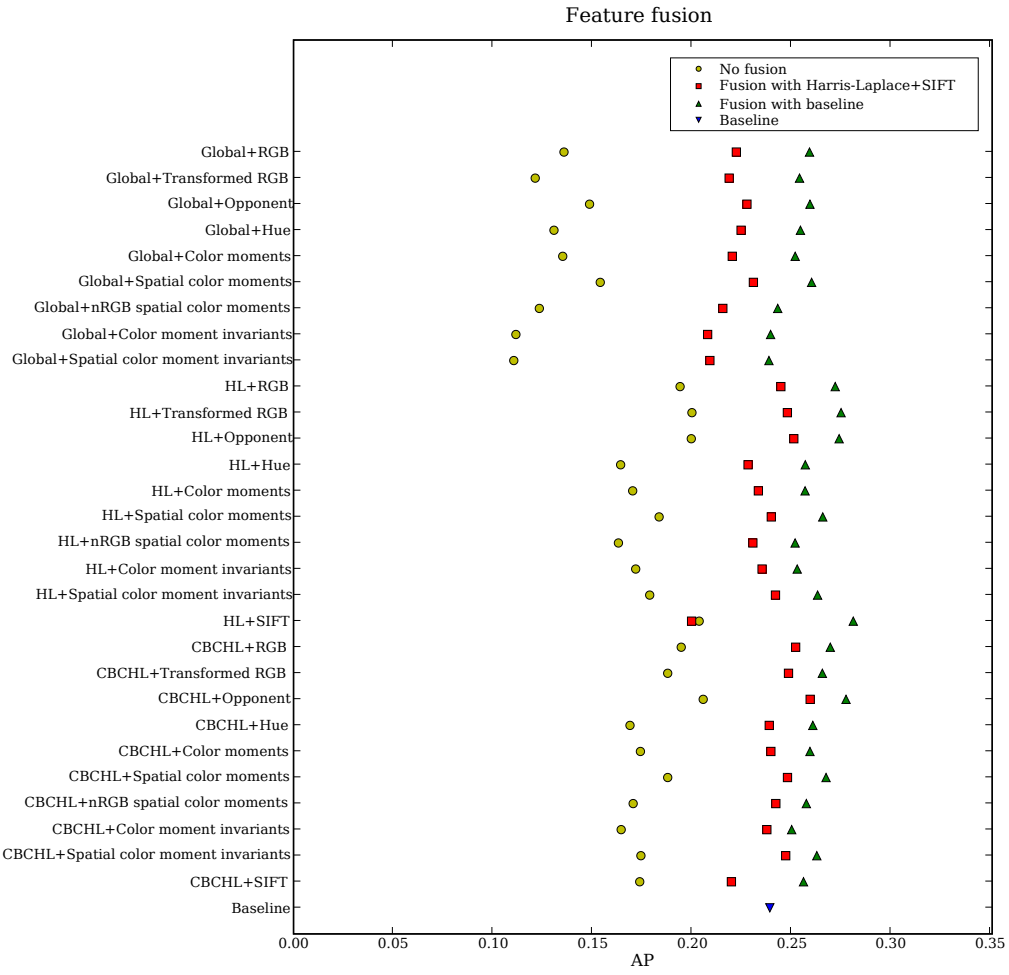


Figure 29: Performance results of feature-level fusion. Results without fusion are equal to those in figure 26. Fusion is performed with either Harris-Laplace with SIFT descriptor or with the Challenge baseline features. HL is shorthand for Harris-Laplace. CBCHL is shorthand for Colorboosted ColorHarris-Laplace.

8 Conclusions

In this thesis, we have developed a concept detection framework for large-scale evaluation of visual features. Within this framework we have provided pipelines for global features and local features based on interest regions. We have focused on including color information in interest region detection and region description. After extensive experiments on 85 hours of video data we see our hypothesis, namely: automated concept detection using interest region features benefits from the addition of color information, confirmed.

Specifically our experiments revealed that local features based on interest regions outperform global features. However, there is no single feature which is best for all concepts: the best feature depends on the concept. Overall, the intensity-based SIFT descriptor and the opponent histogram descriptor are the best local descriptors. For global features, spatial color moments perform best. Hence, when building a concept detection system, one should use multiple 'good' features. Instead of using multiple features, one could also choose to select features on a per-concept basis. This could be done using cross-validation, for example.

In our feature fusion experiment, a combination of SIFT features and our color features significantly improves over the baseline set by the Mediamill Challenge. Feature-level fusion is an attractive way of combining features which are sufficiently orthogonal. It would be interesting to see how many features can be combined before scalability issues are encountered. Another interesting approach would be to perform fusion of shot rankings, which potentially scales better than feature fusion.

For learning concept detectors, the performance increase of using SVM over Fisher's Linear Classifier is modest when compared to the hundredfold increase in computational effort. Only when the computational resources are unconstrained, SVM should be used.

The method for the construction of a visual vocabulary for aggregation of interest regions into a feature vector is of significant influence on concept detector performance. Radius-based clustering selects better clusters than k-means clustering. Clustering on a per-concept basis yields better vocabularies than one-shot clustering. When we apply a visual vocabulary, the default similarity measure from k-means suffers from data sparseness. A dissimilarity measure gives significant improvements. In the future, we aim to investigate if there is an optimal visual vocabulary size which maximizes concept detector performance.

Our color constancy method is not robust to the artificial colors found in commercials. However, given that commercials purposefully contain unnatural color distributions, the assumptions made by color constancy methods in general will not hold. This can be solved by either filtering non-natural color distributions, or by not applying color constancy in the news video domain.

Overall, we have improved automatic semantic concept detection by combining state-of-the-art intensity features with color features. The improvement is even larger when we combine our features with the baseline. Further work can be done on extending the set of well-performing, orthogonal features. These features can easily be integrated in our concept detection framework.

A TRECVID Participation

The features from this thesis have been included in the Mediamill TRECVID 2006 system. At submission time, the SIFT descriptor and the hue descriptor from this thesis were ready for use. We evaluated performance of these descriptors on the Mediamill Challenge using the Harris-Laplace detector, the Colorboosted ColorHarris-Laplace detector and a combination of the two detectors. This combination of the two detectors uses the interest regions of both. We also considered the feature fusion (section 5.6) of the hue and SIFT descriptor. Overall the 9 features listed in table 5 have been tested.

Detector	**Descriptor**	**Used in run**
Harris-Laplace	SIFT	yes
Harris-Laplace	Hue	no
Harris-Laplace	Hue+SIFT	yes
Colorboosted ColorHarris-Laplace	SIFT	yes
Colorboosted ColorHarris-Laplace	Hue	yes
Colorboosted ColorHarris-Laplace	Hue+SIFT	no
Harris-Laplace *and* Boosted ColorHarris-Laplace	SIFT	yes
Harris-Laplace *and* Boosted ColorHarris-Laplace	Hue	no
Harris-Laplace *and* Boosted ColorHarris-Laplace	Hue+SIFT	no

Table 5: Features from this thesis that have been considered for inclusion in the TRECVID participation. The right-most column lists whether the feature was used in the submission.

Based on an experiment with Challenge data, we selected the best combination of features using shot-ranking fusion. The best combination is a subset of 5 out of the 9 features. This combination has been submitted as my run. A combination of my features with Gabor features and Weibull features[5] has been submitted as a run as well.

We plot the results of all TRECVID submissions in figure 30. We achieved the best performance of all TRECVID 2006 submissions for the concepts *charts* and *flag USA* with the Mediamill run which includes my features. From all runs submitted by the Mediamill team, the run with my features only achieved the best performance for the concept *animal*.

Additional details on the shot-ranking fusion and the runs can be found in the Mediamill TRECVID paper [40]. The features from this thesis have also been used in the Mediamill semantic video search engine [41].

[5]Weibull features form the baseline in the Mediamill Challenge.

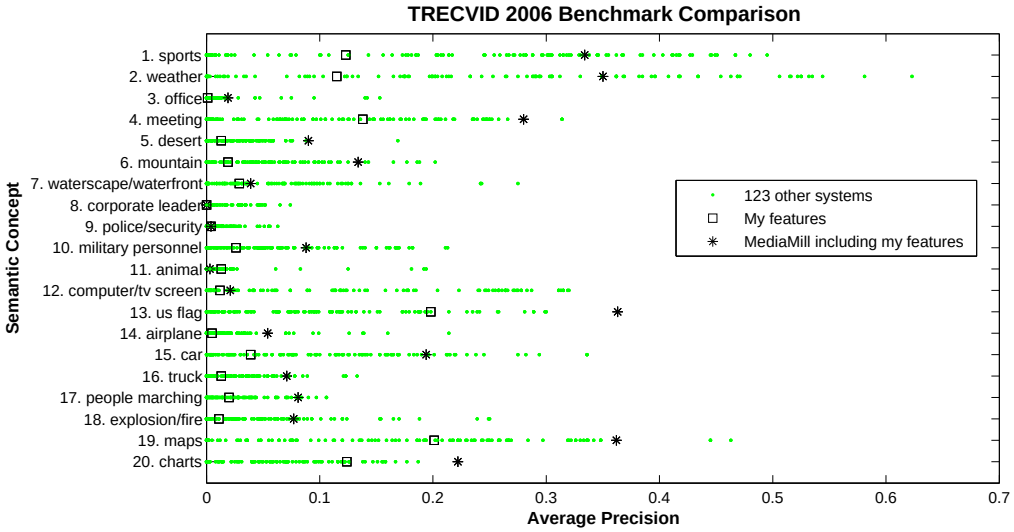


Figure 30: Results of concept detection task at TRECVID 2006. Twenty concepts have been evaluated. The run using my features and the Mediamill run which includes my features have been marked. Results of other runs have been indicated by dots.

References

[1] M. Astrahan, "Speech analysis by clustering or the hyperphoneme method," 1970, stanford A.I. Project Memo, Stanford University, CA.

[2] K. Barnard, V. C. Cardei, and B. V. Funt, "A comparison of computational color constancy algorithms. i: Methodology and experiments with synthesized data." *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 972–984, 2002.

[3] G. Buchsbaum, "A spatial processor model for object color perception," *Franklin Inst.*, vol. 310, pp. 1–26, 1980.

[4] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image segmentation using expectation-maximization and its application to image querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1026–1038, 2002.

[5] M. Everingham, A. Zisserman, C. Williams, L. V. Gool, M. Allan, C. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorkó, S. Duffner, J. Eichhorn, J. Farquhar, M. Fritz, C. Garcia, T. Griffiths, F. Jurie, D. Keysers, M. Koskela, J. Laaksonen, D. Larlus, B. Leibe, H. Meng, H. Ney, B. Schiele, C. Schmid, E. Seemann, J. Shawe-Taylor, A. Storkey, S. Szedmák, B. Triggs, I. Ulusoy, V. Viitaniemi, and J. Zhang, "The 2005 pascal visual object classes challenge." in *First PASCAL Machine Learning Challenges Workshop*, 2005, pp. 117–176.

[6] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *CVPR*, 2005.

[7] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *IEEE CVPR Workshop on Generative-Model Based Vision*, vol. 12, p. 178, 2004.

[8] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, Jun. 2003, pp. 264–271.

[9] R. Fergus, F.-F. Li, P. Perona, and A. Zisserman, "Learning object categories from google's image search." in *IEEE International Conference on Computer Vision*, 2005, pp. 1816–1823.

[10] D. A. Forsyth, "A novel algorithm for color constancy," *International Journal of Computer Vision*, vol. 5, no. 1, pp. 5–35, 1990.

[11] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The Amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005.

[12] J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts, "Color invariance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 12, pp. 1338–1350, 2001.

[13] T. Gevers and A. W. M. Smeulders, "Color-based object recognition." *Pattern Recognition*, vol. 32, no. 3, pp. 453–464, 1999.

[14] R. Gonzalez and R. Woods, *Digital Image Processing, 2nd Edition.* Prentice Hall, 2002.

[15] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of The Fourth Alvey Vision Conference, Manchester*, 1988, pp. 147–151.

[16] Internet Archive, "Internet Archive," http://www.archive.org/movies/.

[17] F. Jurie and B. Triggs, "Creating efficient codebooks for visual recognition." in *IEEE International Conference on Computer Vision*, 2005, pp. 604–610.

[18] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories." in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, June 2006, pp. 2169–2178.

[19] H. Levkowitz and G. T. Herman, "Glhs: a generalized lightness, hue, and saturation color model," *CVGIP: Graph. Models Image Process.*, vol. 55, no. 4, pp. 271–285, 1993.

[20] D. G. Lowe, "Object recognition from local scale-invariant features." in *IEEE International Conference on Computer Vision*, 1999, pp. 1150–1157.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints." *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[22] R. Marée, P. Geurts, J. Piater, and L. Wehenkel, "Random subwindows for robust image classification," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, C. Schmid, S. Soatto, and C. Tomasi, Eds., vol. 1. IEEE, June 2005, pp. 34–40.

[23] Mathworks, "Matlab," http://www.mathworks.com/.

[24] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[25] K. Mikolajczyk, "Affine Covariant Features," http://www.robots.ox.ac.uk/~vgg/research/affine/.

[26] K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *IEEE International Conference on Computer Vision*, 2001, pp. 525–531.

[27] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[28] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[29] F. Mindru, T. Moons, and L. V. Gool, "Color-based moment invariants for viewpoint and illumination independent recognition of planar color patterns," 1998.

[30] P. Mitra, C. A. Murthy, and S. K. Pal, "Density-based multiscale data condensation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 734–747, 2002.

[31] M. R. Naphade, A. Natsev, C.-Y. Lin, and J. R. Smith, "Multi-granular detection of regional semantic concepts." in *ICME*, 2004, pp. 109–112.

[32] NIST, "TREC Video Retrieval Evaluation (TRECVID)," http://trecvid.nist.gov/.

[33] P. Over, T. Ianeva, W. Kraaij, and A. Smeaton, "Trecvid 2005 - an overview," in *Proceedings of TRECVID 2005*. NIST, USA, 2005.

[34] C. Petersohn, "Fraunhofer hhi at trecvid 2004: Shot boundary detection system," in *Proceedings of TRECVID 2004*, 2004.

[35] P. Quelhas, F. Monay, J. M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. V. Gool, "Modeling scenes with local descriptors and latent aspects," in *IEEE International Conference on Computer Vision*, 2005, iDIAP-RR 04-79.

[36] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases." in *IEEE International Conference on Computer Vision*, 1998, pp. 59–66.

[37] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering objects and their localization in images." in *IEEE International Conference on Computer Vision*, 2005, pp. 370–377.

[38] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.

[39] C. G. M. Snoek, J. C. van Gemert, J.-M. Geusebroek, B. Huurnink, D. C. Koelma, G. P. Nguyen, O. de Rooij, F. J. Seinstra, A. W. M. Smeulders, C. J.Veenman, and M. Worring, "The mediamill TRECVID 2005 semantic video search engine," in *Proceedings of the 3rd TRECVID Workshop*, November 2005.

[40] C. G. M. Snoek, J. C. van Gemert, T. Gevers, B. Huurnink, D. C. Koelma, M. van Liempt, O. de Rooij, K. E. A. van de Sande, F. J. Seinstra, A. W. M. Smeulders, A. H. C. Thean, C. J. Veenman, and M. Worring, "The MediaMill TRECVID 2006 semantic video search engine," in *Proceedings of the 4th TRECVID Workshop*, Gaithersburg, USA, November 2006.

[41] C. G. M. Snoek, M. Worring, B. Huurnink, J. C. van Gemert, K. E. A. van de Sande, D. C. Koelma, and O. de Rooij, "MediaMill: Video search using a thesaurus of 500 machine learned concepts," in *Proceedings of the 1st International Conference on Semantic and Digital Media Technologies*, Athens, Greece, December 2006, pp. –.

[42] C. G. M. Snoek, M. Worring, J. C. van Gemert, J.-M. Geusebroek, and A. W. M. Smeulders, "The challenge problem for automated detection of 101 semantic concepts in multimedia," in *Proceedings of the ACM International Conference on Multimedia*, Santa Barbara, USA, October 2006, pp. 421–430.

[43] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky, "Describing visual scenes using transformed dirichlet processes," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, pp. 1299–1306.

[44] J. van de Weijer and T. Gevers, "Boosting saliency in color image features," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2005, pp. 365–372.

[45] J. van de Weijer and T. Gevers, "Color constancy based on the grey-edge hypothesis," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, oct 2005.

[46] J. van de Weijer, T. Gevers, and J.-M. Geusebroek, "Edge and corner detection by photometric quasi-invariants." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 625–630, 2005.

[47] J. van de Weijer and C. Schmid, "Coloring local feature extraction," in *European Conference on Computer Vision*, vol. Part II. Springer, 2006, pp. 334–348.

[48] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, C. G. M. Snoek, and A. W. M. Smeulders, "Robust scene categorization by learning image statistics in context," in *CVPR Workshop on Semantic Learning Applications in Multimedia (SLAM)*, 2006.

[49] J. von Kries, "Influence of adaptation on the effects produced by luminous stimuli," *In MacAdam, D.L. (Ed.), Sources of Color Vision. MIT Press, Cambridge, MS.*, 1970.

[50] A. R. Webb, *Statistical Pattern Recognition, 2nd Edition.* John Wiley & Sons, October 2002.

[51] J. Willamowski, D. Arregui, G. Csurka, C. Dance, and L. Fan, "Categorizing nine visual classes using local appearance descriptors," in *ICPR 2004 Workshop Learning for Adaptable Visual Systems Cambridge, United Kingdom 22 August*, 2004.

[52] Yahoo, "Yahoo! Video," http://video.yahoo.com/.

[53] YouTube, "YouTube," http://www.youtube.com/.

[54] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, June 2007.