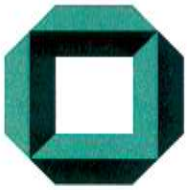


UNIVERSITÄT KARLSRUHE (TH)
FAKULTÄT FÜR INFORMATIK
INTERACTIVE SYSTEMS LABS
Prof. Dr. A. Waibel

DIPLOMA THESIS

Automatic identification of persons in TV series

SUBMITTED BY

Mika Fischer

MAY 2008

ADVISORS

M.Sc. Hazım Kemal Ekenel
Dr.-Ing. Rainer Stiefelhagen

Interactive Systems Labs
Institut für Theoretische Informatik
Universität Karlsruhe (TH)
Title: Automatic identification of persons in TV series
Author: Mika Fischer

Mika Fischer
Kronprinzenstr. 62
76177 Pforzheim
email: mika.fischer@ira.uka.de

Statement of authorship

I hereby declare that this thesis is my own original work which I created without illegitimate help by others, that I have not used any other sources or resources than the ones indicated and that due acknowledgement is given where reference is made to the work of others.

Karlsruhe, 30. May 2008

(Mika Fischer)

Abstract

In recent years, more and more video content in digital form is becoming available, for example through websites like YouTube. For effective search in these large amounts of video data, content-based approaches are necessary. An important sub-task of search in videos is the search for specific persons.

The goal of this work is to develop a baseline system for person identification in videos, more specifically in TV series. The system has to be built from the ground up, starting with shot boundary detection, which segments the video into separate shots and is an important first step for all work in the area of video analysis.

Furthermore, persons have to be detected and tracked in the video. For this purpose, a face tracker is developed, which is based on the particle filter approach and uses skin color segmentation and Haar cascade-based face detectors as features. The Haar cascades are extended to supply confidence values, and not only binary decisions, which has significant advantages for the tracking algorithm.

From the extracted face images, features are extracted using a successful local appearance-based approach, and three application scenarios are explored: closed-set identification, automatic retrieval and interactive retrieval. Experiments are performed with eye location-based alignment for the first two scenarios and without alignment for the last scenario.

The system is evaluated using episodes of TV series on DVD, as well as a standard evaluation for shot boundary detection. The shot boundary detector achieves very good results for the TV series, as well as for the shorter shot transitions in the standard evaluation. There is still room for improvement, especially for longer shot transitions.

The face tracker is also shown to have good performance. It finds a large percentage of the persons in a video and can successfully track them. It is also demonstrated that the tracks are very reliable and that they switch persons only very rarely.

The results of the application scenarios show the viability of the local appearance-based face recognition approach to the video retrieval domain and also point to possible enahncements of the system, especially in the area of alignment.

Contents

List of Figures

List of Tables

List of Abbreviations

AAM	Active appearance model
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
EHMM	Embedded hidden Markov model
FOI	Fade out / fade in shot boundary transition
FSM	Finite state machine
GMM	Gaussian mixture model
HSV	Hue-saturation-value color space
i.i.d.	independent and identically distributed
MOT	Multiple object tracking
PCA	Principal component analysis
PDF	Probability density function
RGB	Red-green-blue color space
SBD	Shot boundary detection
SIFT	Scale-invariant feature transform
SVM	Support vector machine
WAFP	Weighted average frame precision
WAFR	Weighted average frame recall

1 Introduction

The last few years have seen an unprecedented explosion in the amount of video content that is readily available over the internet. Only on YouTube over 150,000 video clips are uploaded every single day. Additionally broadcasters and other media companies possess a huge amount of video data in digital form. Sometimes even older analog tapes are being digitalized, further enlarging the dataset.

These huge amounts of video data, that are already in a form that can be conveniently processed, transmitted and presented by computers, call for effective and efficient automatic methods to find interesting content, in order to increase the usefulness of the data. However, in almost all cases today, search is performed using textual context, like the title and description of the video, tags that the users may attach to the content, or also social recommendation-based systems. Searching videos by using the actual video data is still very much an area of active research. To a smaller degree this is even true for images.

It is clear, however, that content-based approaches are needed because the approaches in use now have obvious disadvantages that are significantly decreasing the usefulness of having large amounts of video data available.

For one, textual information can only help in finding videos, but it cannot be used to structure the video or help the user find interesting parts of the video. Also, user-provided tags might me misleading or even wrong. And in the worst case, there might just be no textual data available for the video.

Some things that would help solve the problem of content-based search in videos are: automatic structuring or segmentation of videos, automatic speech recognition, object detection, concept detection, genre classification, person detection and identification, etc.

In many cases, persons that are depicted in the videos are of special interest to the users. Therefore it seems a worthwhile goal to enable users to search video databases or even single videos for occurrences of specific persons. The usefulness of finding videos that contain a specific person of interest is obvious. But the case of searching for a person within a video is equally important, e.g. for long videos with only a few scenes that contain the person of interest, or also for automatic structuring of videos.

The *Quaero* project [Qua], a large franco-german research project, has as its goal to enable research in the area of multimedia analysis, in order to build commercial applications in this area. A large part of Quaero concerns the automatic

analysis of videos, exactly for the use in video retrieval, as described above. A part of this task is the identification of persons in images and videos. This work presents an approach at building a baseline system that can segment the video into shots, find and track people in the video as well as extract features that can later be used for the retrieval of scenes where specific persons occur.

The rest of this work is structured as follows: in the rest of this chapter, the motivations and goals of this work will be detailed, previous work will be discussed and an overview of the system will be given. In Chapter 2, the major techniques that are used in this work will be introduced. In Chapters 3 and 4, the system components for shot boundary detection and tracking will be described in detail. In Chapter 5, the classification component will be detailed. In Chapter 6, the experimental results will be presented. Finally, conclusions and ideas for future work will be given.

1.1 Motivation

The *Interactive Systems Labs* (ISL) are involved in the *Quaero* project in various tasks, among which there are several tasks related to video analysis, in particular the task of identification of persons in videos. So the goal of ISL is to develop a system that can perform fully automatic video-analysis, especially person-identification. Since there are no systems that can perform this task at ISL, yet, the goal of this work is to build a baseline system, that can perform all the necessary steps to identify persons in videos. In particular, the following tasks have to be successfully addressed by the system:

- Segmenting the video into shots
- Detecting and tracking persons in each shot
- Extracting features for each person, which can be used for
  - Classification
  - Automatic and semi-automatic retrieval

The first step, called *shot boundary detection* (SBD), is a necessary step for every video analysis system, so this component is not only important for this work, but also for possible future works in other areas of video analysis, like genre classification, concept detection, etc. No work has been done at ISL in this area, so this component has to be built from the ground up. Since the SBD component will be used for multiple systems in the future, it is an important design goal, that the architecture be extendible and maintainable.

The second step, detecting and tracking faces in videos, has been approached before at the ISL, but in very different scenarios. Either the system was deployed at a fixed location and with a fixed view-angle [Sta06], or the system was expecting implicit cooperation of the user, by interacting with him [ST07]. Other works in the area of tracking were mostly done in the area of 3D tracking [NGSM05]. The problem posed for this work is much harder, since the data contains a lot more variations than in the previous works. In particular, the following properties make this task especially challenging: Large variations of face size and pose, changing illumination and background within shots and across shots, as well as camera movement. Example frames showing the difficult conditions can be seen in Figure 1.1.

Thus, many features that were used in these previous works are either unavailable (3D information, background segmentation), or are very difficult to use with this type of data (motion, color segmentation). So it is not surprising that the systems developed earlier did not work well on this data. Thus a large part of this work is to develop a robust face tracker, that can find most of the persons in a video and reliably track them through a shot, while keeping the false detections minimal.

For the feature extraction step, a very successful approach has been developed at ISL [ES05], so no effort has gone into developing new approaches to feature extraction.

Extensive previous work has been done at the ISL on the topic of closed-set and open-set identification, but this was not yet tested in a retrieval scenario. So another goal of this work is to test the feasibility of our general face-recognition methodology in this new scenario. To this end, three application scenarios are explored in this work:

Closed-set identification In this scenario, only the main characters of an episode of a TV series are considered. A part of the episode will be used to train a classifier, while the rest of the episode is used to test the classifier.

This scenario is useful, because it will allow the results to be compared with previous work in closed-set identification scenarios.

Automatic retrieval In this scenario, all persons of the episode will be considered, and again a part of the episode will be used to build query sets for the main characters, while the rest will be used as a database from which images, that depict the queried person, have to be retrieved.

Interactive retrieval This scenario is similar to the scenario above, with the difference, that here it will be tried to work on all the extracted data, even the part where facial feature localization and subsequently alignment fail to work. Since this is much harder, it cannot yet be done fully automatic

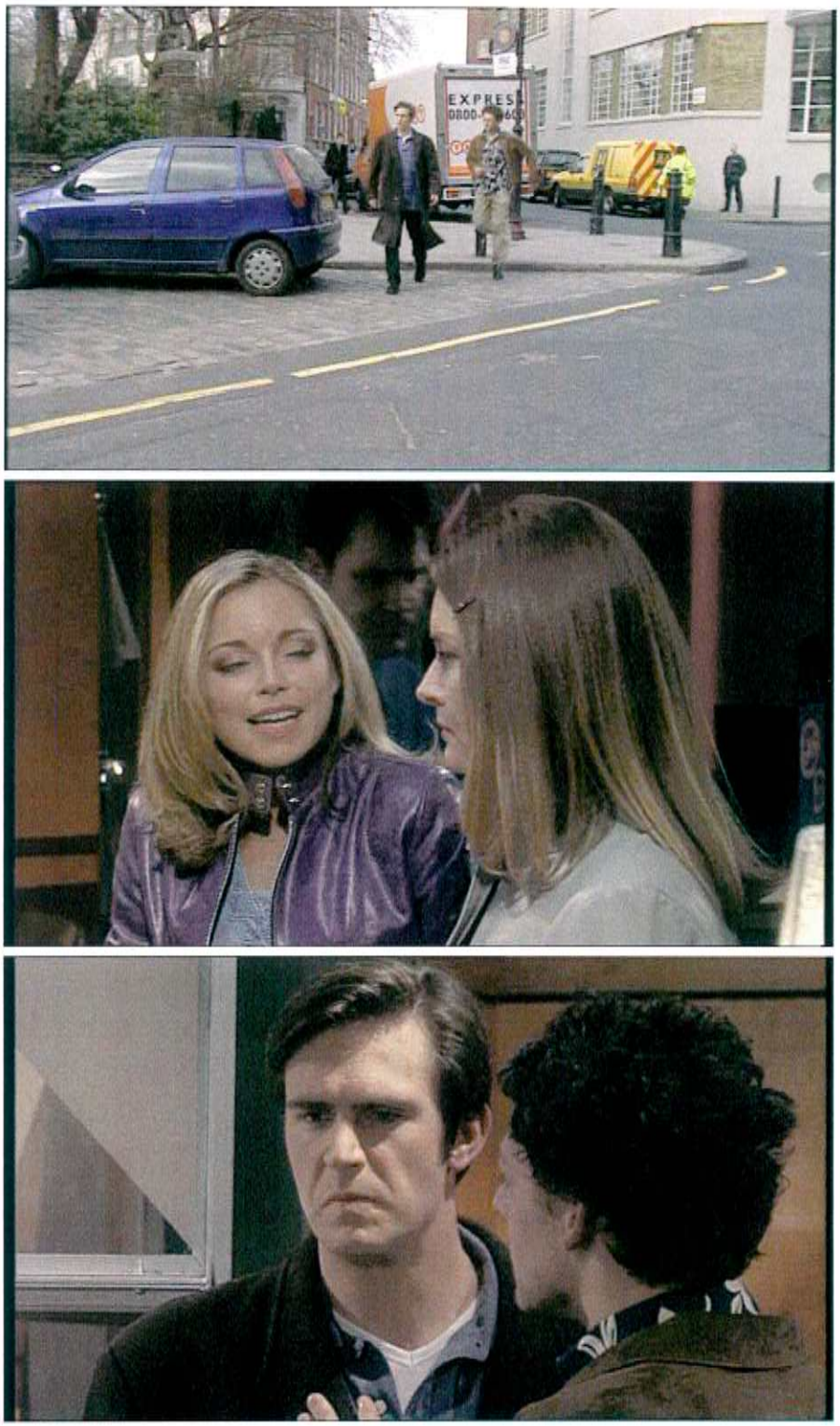


Figure 1.1: Examples of difficult image data

and thus an interactive search application is created, so that the user can assist in the retrieval process.

1.2 Previous work

In this section, previous work in the areas of this work is summarized. The analysis is separated into shot boundary detection and video retrieval systems using person identification.

1.2.1 Shot boundary detection

Shot boundary detection has been an area of research for a long time. In recent years the TRECVID evaluation workshop has been a driving force in the research on shot boundary detection. State of the art systems exist that reach precision and recall rates above 90%. Examples of such systems include [LGZ+06] and [YWX+05]. The different approaches are too numerous to summarize them all, so only the system developed at AT&T [LGZ+06], which was one of the best systems of the TRECVID 2006 evaluation workshop, is described shortly.

The system uses several detector, each for a specific type of shot boundary. It has detectors for: cuts, short dissolves, fade ins, fade outs, dissolves and wipes. The detectors are modeled as finite state machines (FSM), that perform state transitions based on features extracted from the video. The amount of features extracted is extensive and includes histogram- and edge-based features for the current frame as well as histogram difference-based features for adjacent frames and frames that are 6 frames apart. Motion-based features are also used, which are computed by matching edge-maps. FSMs are created for each detector, that use either a model-based approach (for fades, short dissolves and wipes), a thresholding approach (for cuts), or an SVM-based classifier (for dissolves and optionally for cuts) to detect the respective shot boundaries. The results achieved by this system are very good, and approach 90% precision and recall for the difficult TRECVID 2006 dataset.

1.2.2 Video retrieval systems using person identification

The three works that most closely resemble a system like the one envisioned for this work, are described in some detail below.

In [SEZ05], the authors consider the task of person retrieval for movies. The system first segments the video into shots and runs a frontal face-detector on the whole video. The face detections are associated by using a general-purpose region tracker that can track deforming objects, such as turning faces. The face detections within each shot are then associated using a clustering approach that

uses the number of region tracks connecting two face detections to determine, whether they belong to the same person. On the detected faces, a constellation- and appearance-based facial feature localizer is run that determines the position of eyes, nose and mount. It does this by considering the constellation of the features (e.g. the nose should be somewhere between the point between the eyes and the mouth, etc.) and the appearances of the features, which are modeled as a mixture of Gaussians in a PCA subspace for each feature. The facial features are then used to perform an affine warp of the face images, that moves the features into canonical locations. The feature extraction consists of applying the SIFT [Low04] feature extraction algorithm on each facial feature location and also on the point between the eyes. To represent face tracks compactly, the authors perform vector quantization in feature space (separately for each feature) and then assign each image to the closest representative vector. Multiple faces are then represented as a histogram of representatives that got assigned to the face images. This histogram is the final representation of the face tracks in this work. For matching, the authors use the χ^2 metric, which is commonly used for histogram comparison. The authors report some example retrievals, which exhibit partly good performance but partly only medium performance.

In [AZ05], the authors consider the same problem of person identification in movies. In contrast to the work described above, they just use the results of the face detector and don't try to associate the faces with a tracker. They localize facial features (eyes and mouth) using an SVM-based approach that uses image patches and gradients as features. The facial features are then used to align the image using an affine warp, so that the facial features are moved to canonical positions. The authors then perform several preprocessing steps, including segmentation of the image into face and background, band-pass filtering to reduce illumination effects and an alignment refinement procedure. For matching, the squared distance of the preprocessed images is used, but the pixels where the probability for an occlusion is high, are ignored in the distance computation. The occlusion probability is determined by building a statistical model of the image differences for each pixel using a corpus of unoccluded images and then checking each pixel against this model. For matching using multiple query images, the authors propose a method using subspace matching, where essentially, for every query set, a PCA subspace that retains 95% of the variance of the set, is constructed. Then the probe image is projected into this space and reconstructed. Finally the reconstructed image is compared with the original image using the same distance metric as for single images. The authors report very good results on an episode of a TV series. Especially the background removal seems to have a large effect on the system performance.

In [RBK07], the authors describe a semi-automatic approach for labeling of persons in very large datasets. As a first step, a face detector is run on the whole video. Then temporally close detections are associated using histograms

of the hair, face and torso, after which a tracker is employed to obtain more face images around the detected ones. The tracker uses color models, again for the head, hair and torso. The tracks are finally represented as a set of histograms for the hair, face and torso, as well as eigenfaces-based representations of the face images, that are clustered using k-means. Using these representations of the tracks, the authors try to cluster temporally close tracks using a weighted sum of the histogram differences (computed using the χ^2 metric) and the smallest differences of the cluster means for the face representation for the two tracks. For labeling, the authors first manually labeled the clusters of tracks extracted from one episode of the TV series Friends, after which the clustering approach described above was applied across episodes, but with different weights. The face and torso histograms are ignored across episodes because they have been found not to be reliable in that case. A dataset consisting of 22 episodes of Friends was used to test the approach and the authors report good results with as little as two labeled episodes (70% precision and recall).

1.3 System overview

Since this work is concerned with offline-processing of video data, the system components can be conveniently isolated from each other. Each component saves its results to disk and later components read the results of previous components to perform their respective tasks. This approach has the additional advantage, that multiple calculations can be avoided.

The system has three main components: the shot boundary detection module, the face tracker and the application component, which is different for each application scenario.

The shot boundary detection module outputs a list of shot boundaries, or equivalently a list of scenes with start and end frame numbers. This list is then used by the face tracker, to analyze the shots separately. The results of the tracker are similarly stored as a list of tracks, with start and end frame and the face position and size for each frame, as well as some pose information.

The three application modules then extract features from the face tracks and use those features for their tasks: closed-set identification, automatic retrieval and interactive retrieval.

2 Basic principles

This chapter describes several techniques that are used in this thesis as they appear in the literature. Deviations from the standard form of the algorithms as well as implementation details are described in later chapters.

2.1 AdaBoost

Boosting is a well known method of improving the accuracy of any given classification algorithm. The basic principle is to use many *weak classifiers*, that only have to be slightly better than chance, and advantageously combine them to form an ensemble classifier, that has a high accuracy [DHS00].

There are several variants of boosting algorithms, the most popular being *AdaBoost* [FS97]. In AdaBoost, weak classifiers are iteratively added to the ensemble classifier until some condition is satisfied, for example the training error falls below a threshold or a predefined number of iterations has been reached. The main idea of AdaBoost is to assign weights to the training samples, which determine the probability of the sample being used to train the next weak classifier. Depending on the current classification of a training sample, its weight is increased if it is incorrectly classified and decreased otherwise. This way, the AdaBoost algorithm is focusing on the difficult training samples.

The AdaBoost algorithm is shown in Figure 2.1. It takes as input a training set, consisting of samples $\mathbf{x}_i$ and labels y_i, in the binary classification case $+1$ or -1, as well as the number of iterations that the algorithm should run, $k_{\max}$. The training samples are first weighted equally with $1/n$. Then a weak classifier is trained on a random sample drawn from the training set according to the weights $W_k(i)$ and the training error rate E_k is measured on the same random sample. The training error rate is used to assign a weight α_k to the trained weak classifier. This weight increases the influence of classifiers with low training error rates, and decreases the influence of those that have a training error rate close to 0.5. If a classifier has an error rate greater than 0.5, its weight gets negative, effectively reversing its decision so that the error rate is $1 - E_k$ and thus smaller than 0.5. In the next step, the weights of all training samples are updated, depending on whether they are classified correctly or incorrectly by the weak classifier. The weights of correctly classified training samples decrease, while the weights of incorrectly classified samples increase. The final result of

input : $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subseteq \mathcal{X} \times \{+1, -1\}$, $k_{\max} \in \mathbb{N}$
output: $H : \mathcal{X} \to \{+1, -1\}$

begin
  Initialize $W_1(i) = \frac{1}{n}$
  for $k \leftarrow 1$ **to** $k_{\max}$ **do**
    get weak hypothesis h_k by training weak classifier C_k using $\mathcal{D}$ sampled according to $W_k(i)$
    $E_k \leftarrow$ training error rate of C_k measured on $\mathcal{D}$ using $W_k(i)$
    $\alpha_k \leftarrow \frac{1}{2} \ln\left(\frac{1-E_k}{E_k}\right)$
    $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{if } h_k(\mathbf{x_i}) = y_i \quad \text{(correctly classified)} \\ e^{\alpha_k} & \text{if } h_k(\mathbf{x_i}) \neq y_i \quad \text{(incorrectly classified)} \end{cases}$
  end
  return $H(\mathbf{x}) = \text{sign}\left(\sum_{k=1}^{k_{\max}} \alpha_t h_t(\mathbf{x})\right)$
end

Figure 2.1: AdaBoost algorithm

the algorithm is the hypothesis H, that is the sign of the weighted sum of the weak hypotheses h_i.

It can be shown that the training error drops exponentially fast in AdaBoost. It has also been shown that AdaBoost aggressively enlarges the margins of the training set examples, where the margin is defined as

$$m(\mathbf{x_i}) = \frac{y_i \sum_k \alpha_k h_k(\mathbf{x_i})}{\sum_k \alpha_k} \qquad y_i \in \{+1, -1\} \tag{2.1}$$

The margin can be interpreted as a confidence in the prediction. Also, large margins are related to low generalization errors, which explains that AdaBoost is quite robust against *overfitting*. An important conclusion of this is, that it is often beneficial to keep adding weak classifiers even if the training error is already zero, because AdaBoost tries to maximize the margin, leading to lower generalization error. Of course, if the number of weak classifiers gets excessively large compared to the size of the training set, even AdaBoost suffers from overfitting, however this happens much later than it does for other learning algorithms [FS99].

2.2 Haar-feature based object detectors

In [VJ01] the authors describe a general purpose object detection system that can reach very high detection rates while keeping the computational effort com-

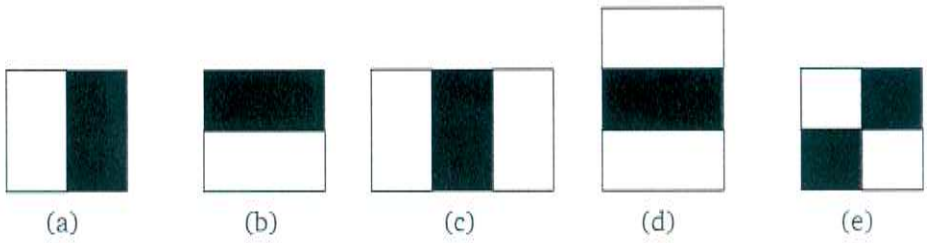


Figure 2.2: Original Haar-like features used in [VJ01]

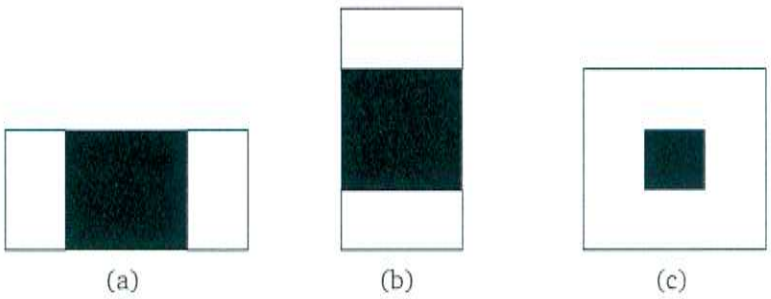


Figure 2.3: Additional Haar-like features introduced in [LKP03]

paratively low, so that detection can be performed faster than real-time.

These so called Haar-cascades have been very successfully employed for several object-detection problems, an important problem of which is face detection.

2.2.1 Haar-like features

Viola and Jones' system uses so called Haar-like features as the basis for classification. These features are differences between the sums of the image intensities in adjacent rectangular areas of the image. More specifically they use three different types of features, which can be seen in Figure 2.2. These features can be thought of as simple detectors for horizontal and vertical edges, horizontal and vertical lines, as well as diagonal lines.

Later, Lienhart et al. extended the feature set to also include features that are rotated by 45°, a center-surround feature that can match quadratic regions, and three-rectangle features with a larger middle part [LKP03]. The additional features are shown in Figure 2.3 and the rotated features are shown in Figure 2.4.

2.2.2 Integral Image

The main advantage of using the simple features described in the previous section, instead of using more sophisticated features, like for instance Gabor-

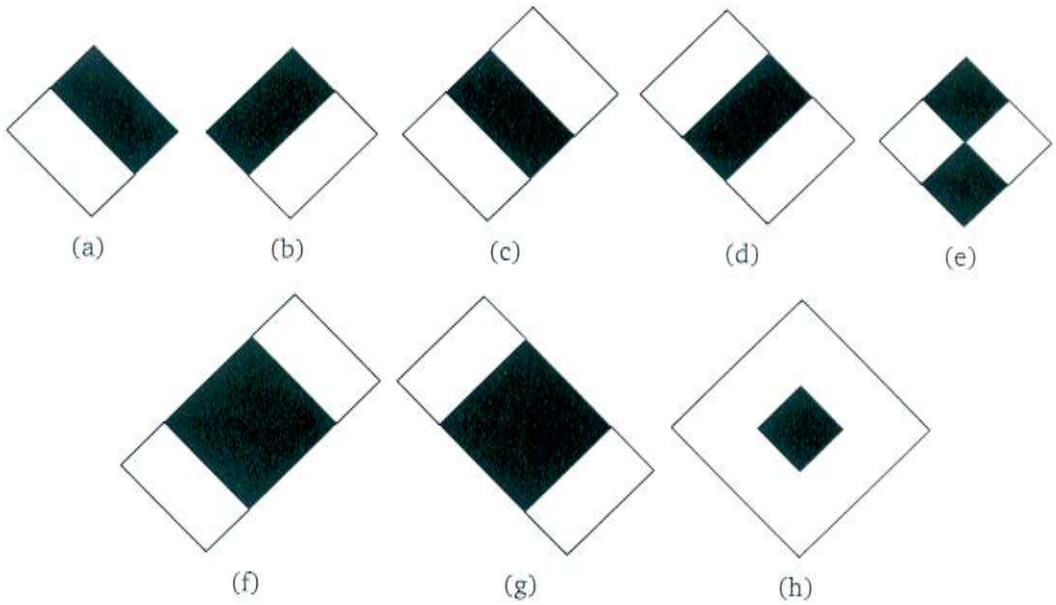


Figure 2.4: Rotated Haar-like features introduced in [LKP03]

wavelets or steerable filters lies in the ability to compute them very rapidly using an intermediate representation of the input image, called the integral image. Thus more features can be used, at least partly compensating the simplicity of the features.

The integral image contains at location (x, y) the sum of the image intensities of the pixels to the right of and above the pixel (x, y), including the pixel itself:

$$ii(x, y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} i(x', y'), \tag{2.2}$$

where $ii(x, y)$ is the integral image and $i(x, y)$ is the input image.

The integral image can be computed efficiently in one pass over the input image using the recurrences

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{2.3}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{2.4}$$

where $s(x, y)$ is the sum of image intensities in rows 0 to y of column x. $s(x, y)$ and $ii(x, y)$ are considered to have the value 0 for negative parameters.

With the integral image, the sum of pixel intensities inside a rectangular area can be computed very rapidly using only four table lookups. Considering the rectangular area defined by the two points (x_1, y_1) and (x_2, y_2), then the sum of the pixel intensities in this area can be computed as:

$$S = ii(x_2, y_2) - ii(x_1, y_2) - ii(x_2, y_1) + ii(x_1, x_1) \tag{2.5}$$

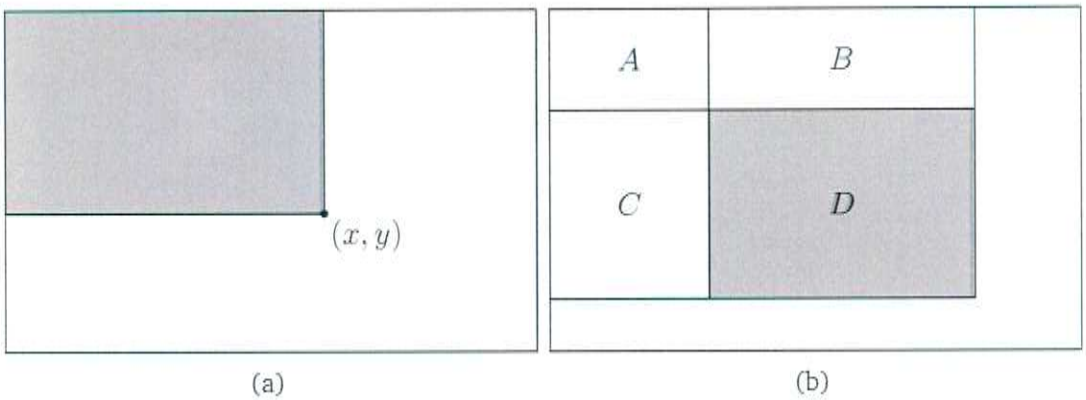


Figure 2.5: Integral image: (a) The value of the integral image at (x, y) is the sum of the pixels to the right and above (x, y) in the original image. (b) The area of the rectangle D can be computed as area$(A + B + C + D)$ − area$(A + C)$ − area$(A + B)$ + area(A). Each area term is just the value of the integral image at a specific location, so the area of D can be computed with just four table lookups.

See Figure Figure 2.5 for a graphical explanation.

Since the rectangles of the features described in the previous section are adjacent, they can be computed using six, eight or nine table lookups, for the two-rectangle, three-rectangle and four-rectangle features respectively.

The additional rotated features introduced by Lienhart can also be efficiently computed using an integral image for rotated rectangles [LKP03].

2.2.3 Classifier Learning

Since for a 24×24 pixel image, there are already over $180,000$ features, considering only the ones originally proposed by Viola and Jones, they can not all be evaluated, even if the evaluation is very fast. The authors empirically confirmed that an effective classifier can be built using only a small number of the features and proposed a learning algorithm to select those features.

They chose to use a variant of AdaBoost (cf. Section 2.1), both to select the features and to train the classifier that uses those features. The proposed variant uses a greedy feature selection strategy: for each feature, a classifier is trained that only uses this feature. The classifier with the lowest classification error is selected and put into the standard AdaBoost algorithm as the next weak classifier. As explained in Section 2.1, the AdaBoost algorithm outputs a weighted sum of the weak classifiers.

The first two features that are selected for the task of face detection are shown in Figure Figure 2.6. The first feature exploits the fact that the eye-region is

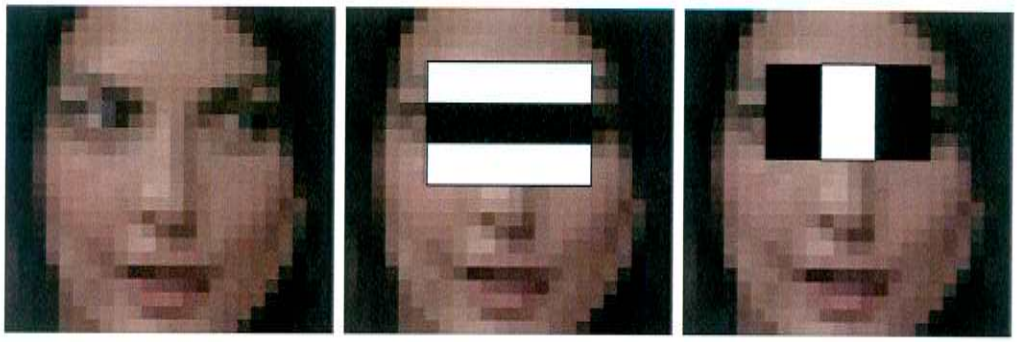


Figure 2.6: Haar features used in first stage of face detector

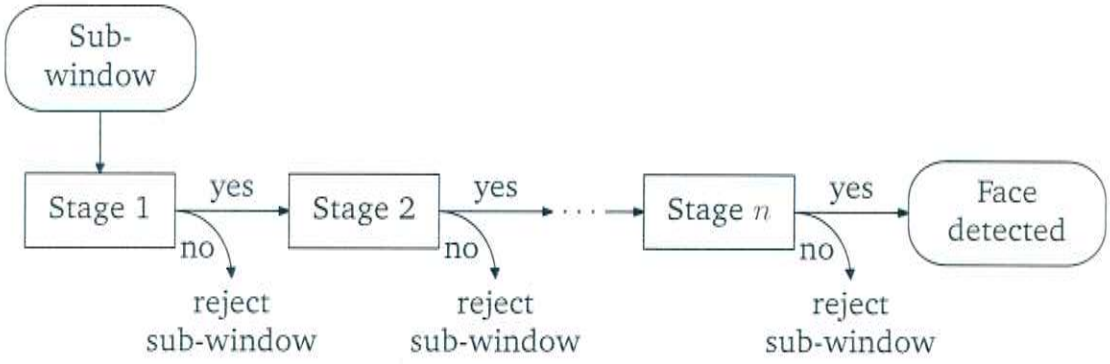


Figure 2.7: Cascade structure

usually darker than the cheek-region below it while the second feature uses the fact that the bridge of the nose is generally brighter than the eyes.

With this classifier training approach, very good results can be obtained. However, the number of features needed to reach high detection rates is still too large to achieve reasonable system speed, considering that all the features have to be evaluated for a sliding window over the whole image in different scales.

2.2.4 Classifier Cascades

To further reduce the processing time drastically the authors introduce a so-called classifier cascade, where an input image has to pass through stages to be classified as positive and if one stage rejects the image, the whole process is aborted (see Figure 2.7). The idea behind this approach is that it is very easy to build a simple classifier using a small number of features that rejects most of the negative sub-windows while detecting almost all positive sub-windows. The later stages are then mainly concerned with achieving a low false positive rate,

and thus use a larger number of features and are more expensive to compute. By employing this cascade structure, the number of sub-windows, for which the later stages have to be evaluated, is reduced dramatically. The large majority of the sub-windows are rejected in the early stages, which are evaluated very quickly.

The stages are trained by using the modified AdaBoost approach described above and then lowering the threshold, so that the false negative rate is minimized. For each stage, features are added until a target detection rate and a target false positive rate have been met. More stages are added until a global target detection rate and false positive rate are met.

The global detection rate D and the global false positive rate F can be computed from the rates for the stages d_i and f_i as:

$$D = \prod_{i=0}^{N} d_i \qquad F = \prod_{i=0}^{N} f_i \tag{2.6}$$

where N is the number of stages.

Using these formulas it is possible to train the classifiers and stages in such a way that they reach a predefined detection and false positive rate (if that is at all possible).

So for instance ten stages with detection rates of 0.999 and false positive rates of 0.3 would yield a global detection rate of about 0.99 with a very low false positive rate of about 10^{-5}.

2.3 Particle filter

The general problem of tracking objects can be stated as a problem of dynamic state estimation. That is, trying to estimate the state of a system, given only indirect and noisy observations. Several algorithms have been proposed to solve this task, most notably the Kalman filter [Kal60]. It assumes Gaussian probability distributions for the state of the system and the various noise terms, as well as a linear motion model. If these assumptions are fulfilled, the Kalman filter can be shown to be the optimal algorithm to solve this problem [Kal60]. In many practical problems, however, the probability distribution of the system state is multimodal and cannot be adequately modeled as a Gaussian distribution. A class of algorithms that can work in cases of arbitrary probability distributions are so-called *particle filters*, which will be described below.

2.3.1 Bayesian tracking

In the following, we consider a system that has an internal (i.e. unobservable) state $\mathbf{x}_t \in \mathbb{R}^{n_x}$, that progresses over time, yielding a state sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$.

The state propagation can be generally modeled as:

$$\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) \tag{2.7}$$

where $\mathbf{v}_{t-1}$ is an independent and identically distributed (i.i.d.) process noise sequence and $\mathbf{f}_t$ is a possibly non-linear and temporally varying function of the previous state and the process noise term.

Since the state is hidden, we can only measure it indirectly. This is expressed in the following definition of the measurement $\mathbf{z}_t$:

$$\mathbf{z}_t = \mathbf{h}_t(\mathbf{x}_t, \mathbf{n}_t) \tag{2.8}$$

where $\mathbf{n}_t$ is an i.i.d. measurement noise sequence and $\mathbf{h}_t$ is a possibly non-linear and temporally varying function of the current internal state and the measurement noise term.

The goal of tracking algorithms is now to estimate the internal state at time t, $\mathbf{x}_t$, given all the measurements up to time t, $\mathbf{z}_{1:t}$. Since the confidence of the result is also useful, a natural way to express this estimate is the probability density function (PDF) $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, which, from a Bayesian perspective, can be seen as a degree-of-belief for possible states $\mathbf{x}_t$, given the measurements $\mathbf{z}_{1:t}$.

In principle, the PDF can be obtained recursively in two stages, that are at the core of most tracking algorithms: prediction and update.

First note that the PDF can be rewritten using Bayes' theorem as:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{\int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})\, d\mathbf{x}_t} \tag{2.9}$$

The prediction stage takes the estimate from time step $t-1$, $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$ and uses the system model (2.7) to compute the prior PDF at time step t:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})\, d\mathbf{x}_{t-1} \tag{2.10}$$

The measurement $\mathbf{z}_t$, that becomes available at time step t, is then used in the update step to modify the prior density from the prediction step using the likelihood function $p(\mathbf{z}_k|\mathbf{x}_k)$, to finally get the wanted posterior density $p(\mathbf{x}_t|\mathbf{z}_{1:t})$.

If the initial PDF $p(x_0)$, before any measurements are made, is known, then the recurrence relations (2.9) and (2.10) can in principle be used to derive the optimal Bayesian solution. However, the solution can not be analytically solved, except for very specific cases. The Kalman filter is the optimal solution for the case of linear functions $\mathbf{f}$ and $\mathbf{h}$ and Gaussian noise terms $\mathbf{v}$ and $\mathbf{n}$.

For other cases, the exact solution has to be approximated. An example of an approximating algorithm is the extended Kalman filter, which can work with non-linear functions $\mathbf{f}_t$ and $\mathbf{h}_t$, by linearizing them at each time step. The

extended Kalman filter has however the same underlying assumption as the Kalman filter, that the probability density $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ is Gaussian. In cases where this is not true, for example when the PDF is multi-modal, the (extended) Kalman filter can never describe the real PDF well. In this case, other models are preferred, one of which is the particle filter.

2.3.2 Particle Filter

The term particle filters describes a class of algorithms, the most basic of which was described by Isard and Blake in [IB98a], and termed CONDENSATION. It will be described in detail in this section.

The basic idea of particle filter algorithms is to represent the probability density function $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ as a finite set of weighted samples (or particles):

$$\{(\mathbf{s}_t^{(1)}, \pi_t^{(1)}), \ldots, (\mathbf{s}_t^{(N)}, \pi_t^{(N)})\}$$

where $s_t^{(n)}$ is a sample and $\pi_t^{(n)}$ is its associated weight. This technique is called factored sampling.

The CONDENSATION algorithm is depicted in Figure 2.8 and works iteratively as follows: first, a new set of samples $\{\mathbf{s}_t^{(n)}\}$ is built by sampling (with replacement) from the old set $\{\mathbf{s}_{t-1}^{(n)}\}$, choosing an element with probability $\pi_{t-1}^{(n)}$. Elements with high weights can be selected multiple times, while elements with low weights might not be selected at all.

The second step corresponds to the prediction step in the Bayesian algorithm. Here it is divided into two sub-steps. First, a deterministic motion model is applied to each element. Afterwards a noise term is added to each element, separating elements that were selected multiple times from the old set. The resulting elements form the new sample set, which can be seen as an approximation of a random sample from the prior density $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$.

The final step corresponds to the update step in the Bayesian algorithm. It uses an observation model, that should ideally resemble the observation density $p(\mathbf{z}_t|\mathbf{x}_t)$, to assign a weight to each element in the new sample set. The final result then, is the new sample-set representation of the state at time t: $\{(\mathbf{s}_t^{(1)}, \pi_t^{(1)}), \ldots, (\mathbf{s}_t^{(N)}, \pi_t^{(N)})\}$.

The CONDENSATION algorithm is relatively simple, compared to the Kalman filter for example, despite the fact that it can handle more general cases.

The design parameters include the number of particles N, the motion model, the observation model and the noise terms. The way in which the observation model is used is very convenient, because in practice all that is needed is to assign weights to a finite set of states, which can afterwards be normalized to sum up to one. The computational complexity can also be conveniently adjusted

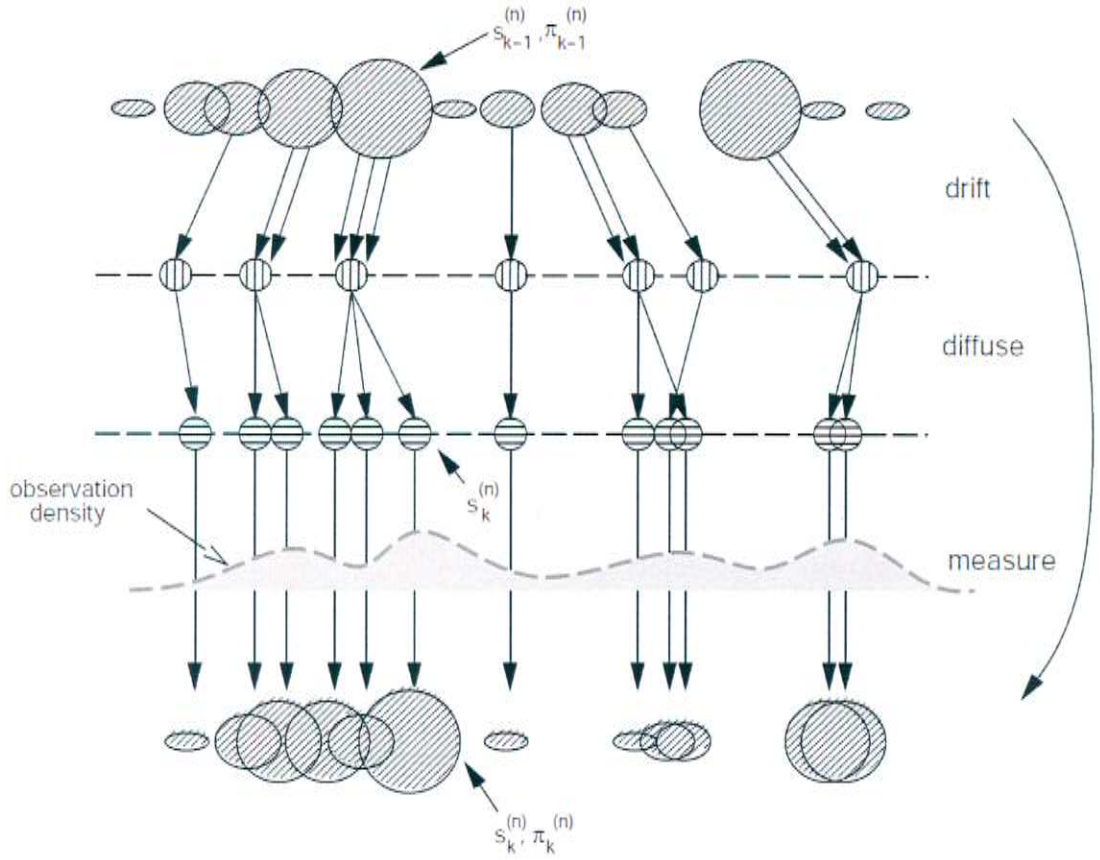


Figure 2.8: Overview of the CONDENSATION algorithm [IB98a]

using the number of particles N. Of course, the higher the number of particles is, the better the weighted samples can represent the true posterior density $p(\mathbf{x}_t|\mathbf{z}_{1:t})$. The number of particles, which can represent the posterior density well, of course, also depends on the dimensionality of the state vector.

The CONDENSATION algorithm has been successfully used in many practical systems, e.g. [NGSM05]. There are also improvements to the algorithm [IB98b, AMGC02].

2.4 Dimensionality reduction using DCT

A problem often encountered in computer vision is that of working in high-dimensional vector spaces. The *curse of dimensionality* [DHS00] is a term coined for the (often counter-intuitive) problems one runs into when applying algorithms in high-dimensional spaces.

Because of these problems, the dimensionality of the vector space needs to be reduced, while on the other hand keeping the important information. For this task, many algorithms have been proposed, the most important of which are the principal component analysis (PCA) and the discrete cosine transform (DCT). The goal of both algorithms is to transform the input vector into a different coordinate system, where most of the dimensions are expected to contain very little information. In other words, the goal is to find a coordinate system, where the information (that is the variation between different vectors) is contained in very few dimensions, while the others carry very little information. The idea then, is to remove the unimportant dimensions by projecting on the important ones, thereby reducing the dimensionality without losing much information.

The discrete cosine transform (DCT) is related to the Fourier transform, and very similar to the discrete Fourier transform (DFT). The difference is that the DCT uses only real numbers and cosine basis functions while the latter uses sines and cosines through the use of complex exponentials.

The formal definition of the DCT is as follows:

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \quad u = 0, 1, 2, \ldots, N-1 \tag{2.11}$$

where

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } u = 0 \\ \sqrt{\frac{2}{N}} & \text{otherwise} \end{cases} \tag{2.12}$$

The inverse DCT is given by:

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) C(u) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \quad x = 0, 1, 2, \ldots, N-1 \tag{2.13}$$

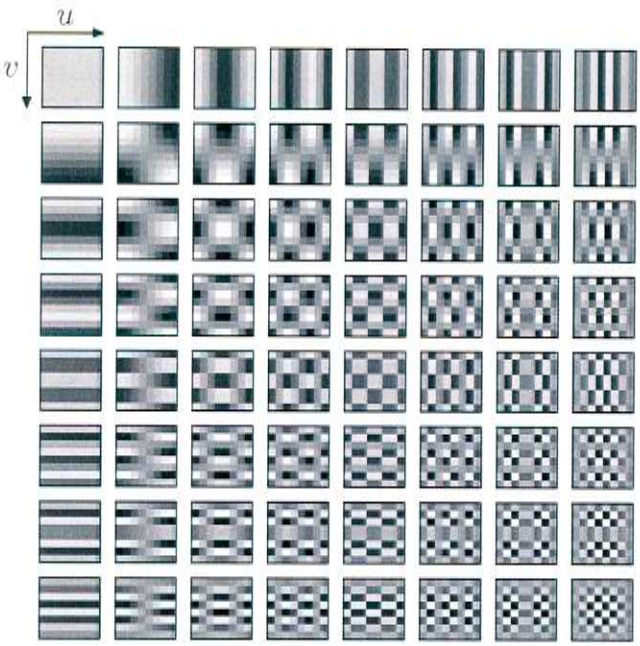


Figure 2.9: DCT basis functions for 8×8 pixel images

The DCT can be easily extended to 2D data, giving the 2D DCT:

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \cos\left(\frac{\pi(2y+1)v}{2N}\right) \quad (2.14)$$

for

$$u = 0, \ldots N-1 \quad \text{and} \quad v = 0, \ldots, N-1 \quad (2.15)$$

and the inverse 2D DCT:

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \cos\left(\frac{\pi(2y+1)v}{2N}\right) \quad (2.16)$$

for

$$x = 0, \ldots N-1 \quad \text{and} \quad y = 0, \ldots, N-1 \quad (2.17)$$

In (2.16) the coefficients $C(u,v)$ can be interpreted as weights for a number of basis functions, which are shown in Figure 2.9 for 8×8 pixel images. From (2.14) it can be seen that $C(0,0)$ just computes the mean of the data (multiplied by the constant N). This can also be seen from Figure 2.9 where the base for $u = 0$ and $v = 0$ is a constant function. It can also be seen from this figure that the spatial frequencies increase from left to right for the horizontal frequencies and from top to bottom for the vertical frequencies. So the lowest frequencies are on the top left, while the highest frequencies are on the lower right.

The DCT has many useful properties, that make it a convenient transformation for many tasks.

Energy compaction The PCA is the optimal transform in terms of energy compaction, but it has been shown that the DCT approaches the energy compaction of the PCA for typical correlated images. In general, the DCT yields high values for low-frequency coefficients while the high-frequency coefficients generally have very small values. It is thus very easy to reduce the dimensionality without losing much information by just removing a number of the high-frequency components. This property is exploited by the JPEG standard which has the DCT at its core.

Separability The DCT can be computed in two steps by performing the 1D DCT on the rows and column of an image. This has computational advantages over other transforms like the PCA, which are in general not separable.

Orthonormality The DCT is an orthonormal transform, which means that it is invertible and therefore lossless. So one retains full control over which information is discarded and which is kept. No information is lost in the transformation itself.

Data-independence The basis functions of the DCT are independent of the data to which they are applied. This has the advantage that the transform itself always stays the same. In contrast, when using PCA, if new data arrives, that is significantly different from the old data, the learned basis functions of the PCA might not be able to represent the new data well. In that case one would have to choose between the sub-optimal representation or computing new basis functions. But in the latter case all old data would have to be transformed again, since the transform itself has changed. These problems are completely avoided by using the DCT.

2.5 Histogram backprojection

In many tasks, one is interested in finding regions in an image that have some specific color, modeled for example with a histogram or a Gaussian mixture model (GMM) in some color-space. An approach that is often used in this case is histogram backprojection. The idea is to model the interesting colors as a histogram and then to replace the pixel colors by the value of the model histogram for each pixel's color. The result is a gray-value image that can be interpreted as a probability map of the presence of the color(s) that are modeled by the histogram.

A modification, that is often applied to the above standard form of backprojection, is called histogram division. The idea is not using the model histogram directly, but computing the histogram of the current image and divide the model histogram by it:

$$\mathbf{H}_r(\mathbf{c}) = \min\left(\frac{\mathbf{H}_m(\mathbf{c})}{\mathbf{H}_i(\mathbf{c})}, 1\right) \tag{2.18}$$

where $\mathbf{c}$ is a vector in some color-space, $\mathbf{H}_m$ is the model histogram and $\mathbf{H}_i$ is the image histogram.

Using histogram division has the advantage that colors ,that are part of the model, but are also very common in the background, get a lower score, while colors, that are in the model and are very rare in the image, get a higher score.

The amount of model color an image may contain before the probability is reduced can be tuned by adjusting the sum of the model histogram, which can be seen as the number of pixels, that are expected to have interesting colors.

3 Shot Boundary Detection

Virtually all video content produced today for commercial or private purposes is a sequence of so called shots. A shot is just a temporal sequence of images, usually taken with a video camera. At a boundary between two shots, the sequence of images of the first shot stops, being followed by the images of the second shot. Often some kind of visual effect like a fade to black is applied to the frames close to the shot boundary to make the transition more visually pleasing.

Usually the time, location and/or camera angle change between different shots. Therefore it is very important that low-level video content analysis like tracking is performed only within shot boundaries and not across them. Consider for example tracking a person's face through a video. Then a shot boundary occurs and in the new scene there's the face of a different person at the same location in the image as in the preceding shot. The tracker would continue tracking and not notice the change in identity!

So accurate shot boundary detection is very important for this work, but it is equally important for all the tasks in video content analysis.

3.1 Types of shot boundaries

There are many transition effects that can be applied to shot boundaries. Some of the widely used ones will be described in the following subsections.

3.1.1 Cuts

In the simplest case of a shot boundary, the frames of the first shot are immediately followed by the frames of the second shot, without addition of intermediate frames or alterations of existing ones. This type of shot boundary is called *cut*.

3.1.2 Fades

Fades are generated by linear scaling of the image intensity over time. Usually the intensity of a number of frames preceding the shot boundary is scaled linearly in time from the original intensity to zero (i.e. a black frame). Then optionally more black frames are inserted, after which the first frames of the

second shot follow, with their intensities scaled linearly from zero to the original image intensity. Visually this gives the impression of the first shot fading away to black frames, followed by the second shot fading in from the black frames. Sometimes one of the fades is also replaced by a *cut*.

An example of a *fade* can be seen in Figure 3.1. Note that the lengths of the fade-out and the fade-in are significantly different. In this case the fade-in consists of only one intermediate frame between zero intensity in frame 2176 and full intensity in frame 2178. In other cases even this single intermediate frame is left out and the fade-in effectively becomes a cut.

3.1.3 Dissolves

Dissolves are similar to fades but instead of fading to intermediate black frames, the shots fade into each other. So the ending frames of the first shot and the starting frames of the second shot overlap and are combined by computing a weighted sum of the frames. This is done in a way so that in the beginning, the combination only consists of the frame from the first shot. Then the influence of the second shot is increased linearly until in the end the combination consists solely of the frame from the second shot. In practice, dissolves are further categorized into *short* or *fast dissolves*, that last up to five frames, and *long dissolves* or just *dissolves*, that last longer. Although there is no difference in the generation of the two types of dissolves, the visual appearance is very different. Also, because of the limited amount of image changes that can occur within the duration of a *short dissolve*, different algorithms can be used to detect them.

An example of a fast dissolve can be seen in Figure 3.2 and an example of a (slow) dissolve is shown in Figure 3.3. Note that in the longer dissolves, significant image changes can be part of the transition, while for the shorter dissolves the images stay approximately constant during the transition.

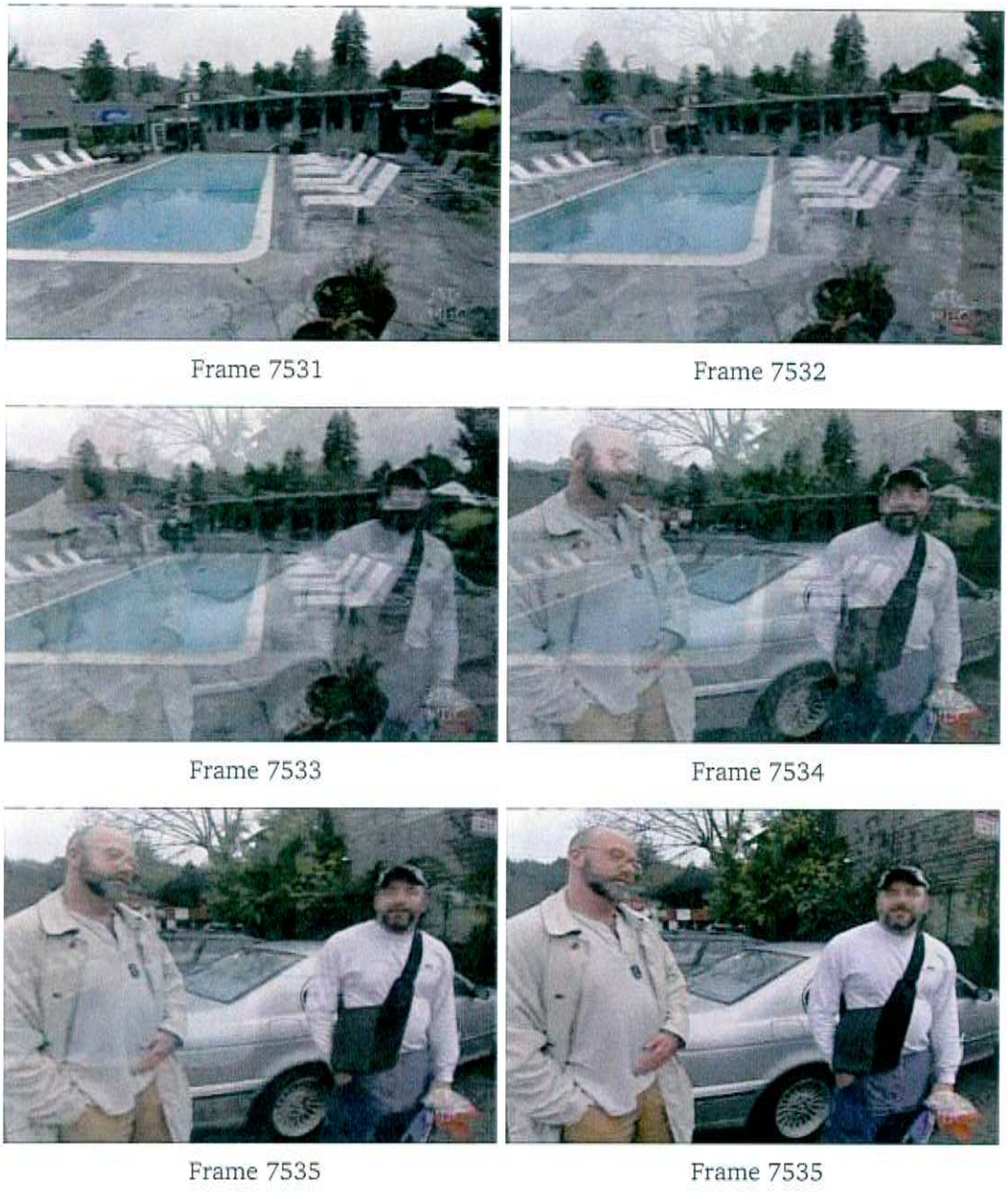
3.2 Shot boundary detection module

The goal of this system component is to segment a given video file into a number of shots, labeling each shot by the starting and ending frames.

Note that since the effects applied to the video data to generate shot boundaries other than cuts can affect the video analysis algorithms negatively, and since the transitions are generally not very long and/or contain very little useful information, these altered frames are not considered to be part of either shot.

The shot boundary detection system is built in a modular way and consists of an MPEG decoder [FFM], several feature extractors, several detectors for different types of shot boundaries, and a fusion module, that fuses the results of the different detection modules.

Figure 3.1: Example of a FOI transition



Frame 7531 Frame 7532

Frame 7533 Frame 7534

Frame 7535 Frame 7535

Figure 3.2: Example of a fast dissolve transition

Figure 3.3: Example of a (slow) dissolve transition

Each detection module declares which features it requires, and how many frames of history should be kept. All the requests are consolidated, so that only required features are computed, and features that are used by multiple detection modules are only computed once. Because of this flexible design, it is also very easy to experiment with different inputs to the detection modules.

The detection modules are specialized for a specific type of shot boundary and are described in detail below. New modules can be easily integrated into this framework.

The fusion module has the purpose of resolving any conflicts that may arise, like overlap of shot boundaries detected by different modules.

After fusion, the system outputs an XML file containing the detected shot boundaries, or equivalently a list of of shots with start and end frame numbers.

3.2.1 Cut detector

Good results have previously been achieved with using color histogram differences to detect cuts, which should in almost all cases cause a peak in the color histogram difference between adjacent frames.

In [Lie99], the author uses the L_1-distance:

$$d_t = \frac{1}{N} \sum_{r=0}^{R} \sum_{g=0}^{G} \sum_{b=0}^{B} |\mathbf{H}_t(r,g,b) - \mathbf{H}_{t-1}(r,g,b)| \tag{3.1}$$

However, it was determined experimentally that a quadratic difference as used in [LGZ+06], results in much stronger peaks, making this distance measure more appropriate for the task at hand. In contrast to [LGZ+06], in this work, the RGB color space was used instead of HSV. The histogram extracted from each frame has 8 bins per color.

The histogram difference is then computed as follows: first the 3D histograms are vectorized, yielding histogram vectors $\mathbf{x}$ and $\mathbf{y}$. The difference is then given by:

$$d(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T \mathbf{A} (\mathbf{x}-\mathbf{y}) \tag{3.2}$$

where a_{ij} is the similarity of the colors corresponding to the centers of the histogram bins i and j:

$$a_{ij} = 1 - \frac{\sqrt{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}}{N} \tag{3.3}$$

and where N is a normalization constant, so that $0 \leq a_{ij} \leq 1$.

The cut detector uses this RGB histogram difference feature to find cut candidates. For this, it considers a window of a fixed size around the current frame. In this work, the size was set to $w_c = 7$. The cut detector searches for peaks,

so if the current frame is not the largest value in the window, the candidate is immediately discarded. Otherwise the following values are computed:

- the kurtosis of the values in the window,
- the ratio of the current value and the sum of the other values in the window, and
- the ratio of the current value and the second largest value in the window.

If the current candidate is at frame number t_0, then the largest value within the window is by definition at frame t_0. Let the second largest value in the window be at frame t_1, and let $f(t)$ be the value of the histogram difference feature at frame t, then the three values can be computed as follows:

$$k_c(t_0) = \frac{\frac{1}{w_c}\sum_{i=1}^{w_c}(x_i - \bar{x})^4}{\left(\frac{1}{w_c}\sum_{i=1}^{w_c}(x_i - \bar{x})^2\right)^2} - 3 \tag{3.4}$$

$$s_c(t_0) = \frac{f(t_0) + c}{c + \sum_{i=1}^{w_c/2} f(t_0 + i) + f(t_0 - i)} \tag{3.5}$$

$$r_c(t_0) = \frac{f(t_0)}{\max\{f(t_1), c\}} \tag{3.6}$$

where $x_i = f(t_0 - \frac{w_c}{2} + i)$, $\bar{x} = \frac{1}{w_c}\sum_{i=1}^{w_c} x_i$ and c is a small positive constant that prevents problems when the feature values are close to zero.

The histogram difference $f(t_0)$ and the computed values $k(t_0)$, $s(t_0)$ and $r(t_0)$ are then checked against thresholds. If any threshold is not reached, the candidate is discarded as a cut candidate but passed to the fast dissolve detector.

Normally, this would be the time to declare a detected cut. It has however been observed, that sudden flashes of light, like a flash from a camera or also flashlights of a police car can lead to false detections, because they cause large histogram differences, while the image structure stays constant. To prevent those false detections, each cut candidate that has passed the thresholds is subjected to a flash detector.

The flash detector [LGZ+06] works on edge maps, because they stay relatively constant during flashes in contrast to the large changes in the color histograms. The edge maps are generated using the Sobel filter [GW01]. The inner part of one of the edge maps is divided into 48×48 pixel blocks and each block is used as a template $\mathbf{T}$, the best match is found in the other edge map $\mathbf{E}$, in a region around the block's position. As matching error, the normalized squared difference is used:

$$d(x, y) = \frac{\sum_{x',y'}\left(\mathbf{T}(x', y') - \mathbf{E}(x + x', y + y')\right)^2}{\sqrt{\sum_{x',y'}\mathbf{T}(x', y')^2}\sqrt{\sum_{x',y'}\mathbf{E}(x', y')^2}} \tag{3.7}$$

where (x, y) is the position of the template in the edge map and x' and y' cover the size of the template. So in this case x' and y' are in the range $[0, 47]$.

The matching errors of all blocks are summed up and if the sum fails to exceed a threshold, a flash is detected. In that case the candidate is discarded as a cut and passed to the fast dissolve detector. Otherwise, the candidate is declared a cut.

3.2.2 Fast dissolve detector

The fast dissolve detector is employed on cut candidates that fail to pass the strict requirements for a cut. It considers the histogram differences within a window w_{fd} of nine frames around the current candidate and computes the ratio of the three center frames and the sum of the rest of the frames in the window:

$$s_{fd}(t_0) = \frac{c + \sum_{i=-1}^{1} f(t_0 + i)}{c + \sum_{i=2}^{w_{fd}/2} f(t_0 + i) + f(t_0 - i)} \tag{3.8}$$

where c is again a small positive constant to prevent problems when the features are close to zero.

If this value does not exceed a threshold, the candidate is discarded. Otherwise a model-based approach is used to detect a fast dissolve [LGZ+06]. As explained in Subsection 3.1.3, a fast dissolve can be modeled as a linear combination of two shots. Formally is can be specified as:

$$FD(t) = \left(1 - \frac{t}{t_{\max}}\right) \cdot S_1(t_0 + t) + \frac{t}{t_{\max}} \cdot S_2(t) \qquad t = 0, \ldots, t_{\max} \tag{3.9}$$

where $S_1(t)$ and $S_2(t)$ are the t-th frames of the fist and second shot respectively, t_0 is the frame in the first shot, where the fast dissolve starts, and $t_{\max}$ is the length of the dissolve.

Since for fast dissolves, there are at most five frames in a transition, $S_1(t)$ and $S_2(t)$ can be approximated for the duration of the transition as the first frame before the transition, and the last frame after the transition: $S_1(t_0+t) \approx S_1(t_0) = X$ and $S_2(t) \approx S_2(t_{\max}) = Y$.

If we now let $\alpha(t) = \frac{t}{t_{\max}}$, then (3.9) becomes:

$$FD(t) = (1 - \alpha(t)) \cdot X + \alpha(t) \cdot Y \qquad t = 0, \ldots, t_{\max} \tag{3.10}$$

This model of a fast dissolve is used by the detector to check whether a candidate really is a fast dissolve. All frame sequences of lengths between three and seven frames (the two frames for $\alpha(t) = 0$ and $\alpha(t) = 1$, and are not part of the transition) that contain the two candidate frames (note that a candidate is a histogram difference, so it concerns two frames) are considered. It is assumed

that the sequence is generated according to (3.10), that is the interior frames are linear combinations of the start frame X and the end frame Y, while $\alpha(t)$ increases linearly from 0 to 1. The $\alpha(t)$ can be computed from the boundary frames X and Y and the interior frame using a least-squares approach. It gives the optimal $\alpha(t)$ and also the residual error. The $\alpha(t)$ are checked for consistency, by ensuring that they increase with t and that they are in $[0; 1]$. If they are inconsistent, the candidate sequence is discarded. Otherwise, the average reconstruction error is checked against a threshold, and if it is too high, the candidate sequence is discarded as well. From the remaining sequences the one with the lowest average reconstruction error is selected as the final result. The frames before and after this candidate transition are then checked for a flash using the flash detector described above. If no flash is detected, the candidate transition is declared a fast dissolve, otherwise it is discarded.

3.2.3 Fade out / fade in detector

As described in Subsection 3.1.2, a FOI transition is generated by linearly scaling the pixel intensities. It can be easily shown that the same scaling is in effect applied to the standard deviation of the pixel intensities. Using this idea, the standard deviation of the pixel intensities has been successfully used as a feature [Lie01] for FOI detection.

The basic idea is to first detect frames in the video with very low standard deviation. Although it is possible for several visually distinct colors to have the same intensities, in practice these frames are almost always completely black or contain exactly one color. Therefore they are also called *monochrome frames*. In this approach, contrary to [Lie99], these monochrome frames are regarded as a (possibly partial) shot boundary in themselves. The reason is that sometimes, a FOI is not finished by a fade in but by a cut. In this case the approach described in [Lie99] would break down because it only models fade-out / fade-in transitions. Care has to be taken though, that the threshold for the detection of monochrome frames is set to a conservatively low value, because otherwise dark scenes might be erroneously classified as monochrome frames. Also, as already mentioned, monochrome frames can also result from colored frames. Since they rarely constitute a shot boundary, a color detector is used that uses the same RGB histogram that is extracted for the cut detector and determines the percentage of gray pixels in the image. If the ratio is smaller than a threshold, the frame is not marked as monochrome.

Consecutive monochrome frames are fused into a block and this block is then declared as a monochrome block. Additionally it is checked for a fade-in at the start of the block and a fade-out at the end of the block. This is done by computing a line of regression through the points of the standard deviation time

series [Lie99]. If the correlation is high enough and the slope of the line is steep enough, a fade-in or fade-out is detected, respectively.

The fade-in, fade-out and the monochrome block are later merged in the fusion module.

3.2.4 Dissolve detector

For the dissolve detector a heuristic approache similar to the one used in the fast dissolve detector was implemented.

Dissolve candidates are detected similarly to [LGZ+06]. The smoothed intensity standard deviation is checked for a decrease followed by an increase. Each of these cases is a dissolve candidate.

For each candidate the point of minimal standard deviation is found and it is checked that the start or the end of the dissolve has a significantly larger standard deviation than the minimum. If this is the case, the sequence of frames with a length of nine frames around the frame with minimum standard deviation is checked for a dissolve using the same technique as described in section 3.2.2. If a dissolve is detected, the complete candidate is declared a dissolve.

3.2.5 Fusion module

The fusion module is responsible for fusing the monochrome frames returned by the FOI detector with corresponding fades, cuts or dissolves. The other function is to resolve overlapping transitions. Especially the dissolve detector produces many false positives, so if a dissolve is overlapping with any other transition, the dissolve is usually discarded.

4 Face tracking

Tracking faces in TV series is a difficult task. This comes from the fact that many of the features generally used for tracking are not available or unreliable in TV series types of video. For instance, 3D information is very useful for tracking, but obviously not available in TV series. Foreground-background segmentation is also very difficult to use in TV series, since the background changes from shot to shot and often also within a shot due to camera movement. Movement information is also difficult to extract from these videos, since there may be any number of objects that move through the image (cars, animals, etc.) and further camera movements can make the movement information close to useless. Color information is also often used for tracking, but due to the large variations in the environmental conditions, using color robustly is much more difficult than in a fixed environment.

The tracking approach proposed in [DDCF01] and the CamShift algorithm [Bra98] have been tried without success. In the end the particle filter approach was used for the tracker in this work. It has been shown to yield good results for various tasks and has been successfully used for several works in our institute. The particle filter tracker is explained in detail in this chapter.

4.1 Particle filter parameters

Since the goal is to track faces, the location and the size of the face have to be part of the state. Because the face detector, which plays a significant role in the tracker, considers only square regions of the image, the face was modelled as a square region as well.

Additionally, depending on the motion model in use it might be necessary to model the velocities of the location and size. It was experimentally determined, that the location parameter benefits significantly from a linear motion model, whereas for the size parameter simple Gaussian noise suffices.

So in total the state vector looks as follows:

$$\mathbf{s} = (x, y, v_x, v_y, s) \tag{4.1}$$

where (x, y) is the location of the center of the face box, (v_x, v_y) is the current velocity of the face box and s is its size.

Each track consists of 500 particles. The motion model the the location is linear with Gaussian noise. The size does not use a motion model but just Gaussian noise.

4.2 Features

As mentioned already, many features generally used for tracking are not available or difficult to use robustly in the videos this work is analyzing.

The two features, that could be used to build a robust tracker, are skin color segmentation and results from several face detectors. The features will now be explained in detail.

4.2.1 Color segmentation

Since this work is concerned with tracking faces, an obvious idea is to find regions of the images that have skin color. Several approaches have been proposed for this task [PBC05], using different methods of modelling skin and non-skin color and also various ways of classifying pixels as skin or non-skin.

In this study, the color model consists of a color histogram and classification is performed by histogram backprojection, as described in Section 2.5.

However, the use of histogram division is problematic because it is impossible to know how many faces are expected in the image and because of pose variation the amount of skin color can also vary drastically even if no persons appear or disappear. Because of this, histogram division was not used in this work.

Color model

The model histogram was built fully automated by running a frontal face-detector on one of the videos and extracting a patch from the center of the face detection with height and width one third of the height and width of the face detection. Of course the face detector produces some false detections, but because they are relatively rare compared to the large number of correct detections, their influence on the color model is negligible. Some samples of the detected faces and the regions used to generate the model histogram are shown in Figure 4.1.

Several color spaces and histogram sizes were tested. While the differences between them were small, in the end the HS (HSV without intensity) color space with 128×128 bins was used. Additionally, bins corresponding to small saturations were set to zero, because the colors they represent are mostly white and have very little real color information.

Obviously this single model is not very well suited to cover all possible environmental conditions, so it is a good idea to adapt the model to the current

Figure 4.1: Examples of the data used to build the color model: (a)-(f) Successful extractions of skin colored pixels. (g)-(i) Non skin-colored pixels are included because of large pose variations or false face detections.

scene. In this work, the initially trained color model is only used at the beginning of the video. Afterwards, the model is continuously adapted when the tracker has a very high confidence of tracking a frontal face. The adaptation is performed similarly to the initial model building, in that the inner part of the tracker's current face box is used to extract a histogram, which is then used to adapt the model histogram using a given learning rate α:

$$\mathbf{H}_{\text{new}} = \mathbf{H}_{\text{old}} + \alpha \mathbf{H}_{\text{face}} \tag{4.2}$$

Afterwards, the histogram is normalized.

Histogram backprojection

On each frame that was analyzed, the backprojection algorithm was applied, yielding a gray-value image that can be interpreted as a probability map for the presence of skin. This map is then thresholded with a fixed threshold, to extract skin-color regions.

In other works several more or less sophisticated post-processing steps are used to reduce the amount of clutter from skin-colored regions in the background. Examples of this include blob analysis, where certain conditions on the size and shapes of the connected components of the skin regions are used to remove some of the clutter. Constraints might include a certain roundness, or a minimal and maximal aspect ratio of the principal components. This was however very difficult to do in this work, mainly due to the fact that the varying environmental conditions lead to significant background clutter, which is in many cases directly connected to the real skin region. Also, because of the large pose variations and different hairstyles and other occlusions, the shape of the face blobs varies dramatically, even if they can be extracted reliably. So in the end connected-component analysis was not performed for this work.

Another post-processing step that is sometimes performed is hysteresis thresholding. Here, the regions that resulted from the thresholding, are grown as long as the new pixels lie above a second threshold, lower than the first. This is generally used to get larger regions of skin color without having to use a lower initial threshold, which would lead to more background regions being falsely detected as skin colored. However, in this case the problem was that the threshold had to be set quite conservative already in order to cover all the lighting variations inherent in the data. Additionally, using hysteresis thresholding did not add much for the correct face blobs but significantly added to the amount of background clutter. Because of this, hysteresis thresholding was also not performed in this work.

In general, the skin color segmentation works reasonably well, considering the difficult input data, but there is still significant background clutter, which in

some scenes with difficult illumination, can make the skin-color segmentation close to useless. Therefore it is very difficult to perform tracking based on the color segmentation alone.

Some examples of the skin-color segmentation can be seen in Figure 4.2.

4.2.2 Confidence based Haar detectors

The second feature that is used in this work to track faces is the output of a Haar-cascade based face detector. As described in Section 2.2, Haar-cascades have been successfully used for face detection [VJ04]. For this work the implementation contained in OpenCV [Ope] was used.

A simple idea would be to just run such a face detector on each frame and associate close detections in adjacent frames. This is also known as tracking-by-detection. However, it turned out that the face detector, while having good performance, is not quite good enough to do tracking-by-detection. There are too many misses.

So the next idea is to use the output of the face detector as a feature for the particle filter tracker. However, the problem with this is, that the face detector gives a binary result: face / non-face. The particle filter however works with confidences, which are impossible to derive from a binary output. This means, for instance, that there is no way to distinguish a close miss from a correct negative result, so this feature would bring us no further than tracking-by-detection; and as explained in Subsection 4.2.1, the skin color feature is not robust enough to be used for tracking on its own in the case the face detector misses a face.

So it would be highly desirable if the face detector could be extended to output confidences and not only binary decisions.

One way to do this would be to use the number of the last stage, the candidate image passed, as a confidence. This is somewhat crude, but has been successfully used in practice.

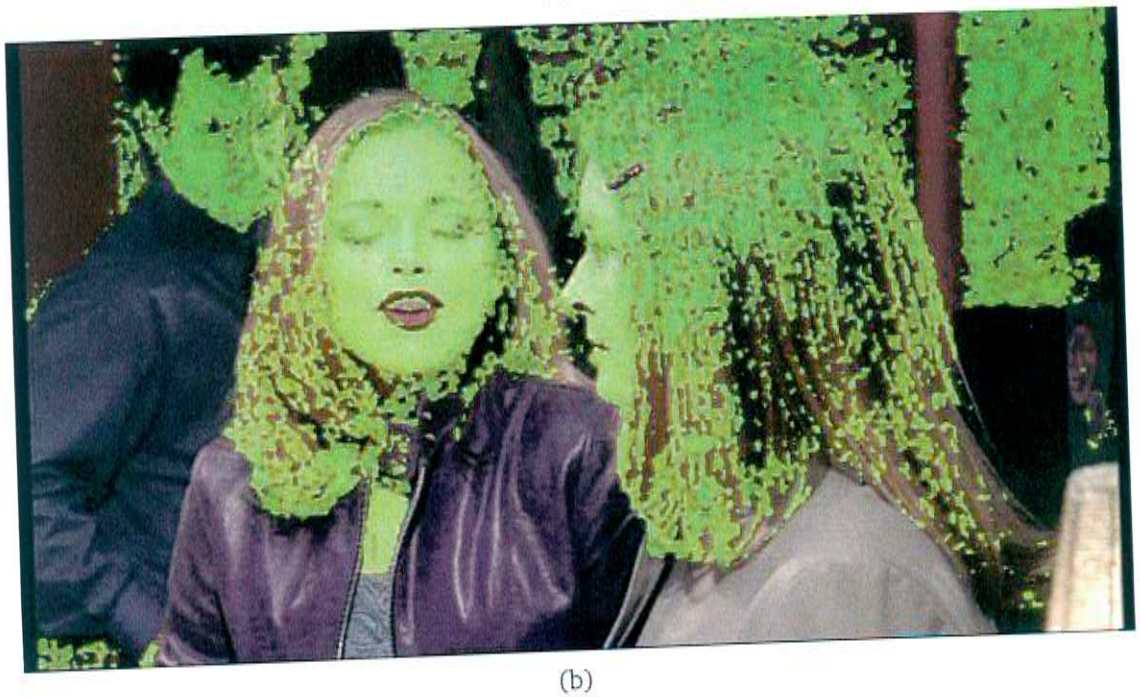
A better approach exploits a property of the AdaBoost algorithm that is used in the face detector. As shown in Section 2.1, AdaBoost outputs weights α_k for the so called weak classifiers $h_k(\alpha)$ and the final classification function is:

$$H(\mathbf{x}) = \text{sign}\left(\sum_k \alpha_k h_k(\mathbf{x})\right) \tag{4.3}$$

It would now be an obvious idea to just use the sum $\sum_k \alpha_k h_k(\mathbf{x})$ as a measure of confidence. And indeed it can be shown that if one wraps a sigmoid function around this sum, the result converges to the posterior probability of an object (a face in our case) existing, with increasing training data [MDWW04]:

$$P(\text{face}|\mathbf{x}) = \frac{1}{1 + 2\exp\left(\sum_k \alpha_k h_k(\mathbf{x})\right)} \tag{4.4}$$

(a)



(b)

Figure 4.2: Sample skin color segmentation results: (a) Good segmentation. (b) Significant background clutter.

However, the matter is complicated by the use of separate stages in the Haar-cascade algorithm (cf. Section 2.2), since the computation of the weak classifiers is completely aborted if a stage in the cascade is not passed. The result of the weak classifiers contained in later stages cannot be incorporated without giving up the efficiency that the stage-structure provides. This means that if we only include the results of the weak classifiers from the stages that are passed, the final result could be higher or lower than the real result obtained when evaluating all stages. In practice, the results will be too high for candidates discarded in the first stages, since those are almost certainly not faces and it can be expected that they would accumulate further negative scores in subsequent stages. On the other hand, it can be expected that correctly detected faces accumulate a substantial positive score through all the stages, since each stage has to be passed for a detection. So the fact that the confidences assigned to the non-faces are too high is not a big problem, since the correct detections (and also the close misses, that fail in one of the final stages) are expected to have a much higher score, since they have more stages where they can accumulate positive scores.

One could also think to include a penalty term for missed stages, however in experiments it was determined that this is not necessary and would be difficult to tune.

Additionally it was found that including the sigmoid function compressed the confidences too much and just using the sum directly yielded more stable results.

An example of a confidence map for an image with a face in it can be seen in Figure 4.3. This confidence map was computed for the size of the face of the person in the foreground. It shows at each position the confidence that there is a face in the image at that position.

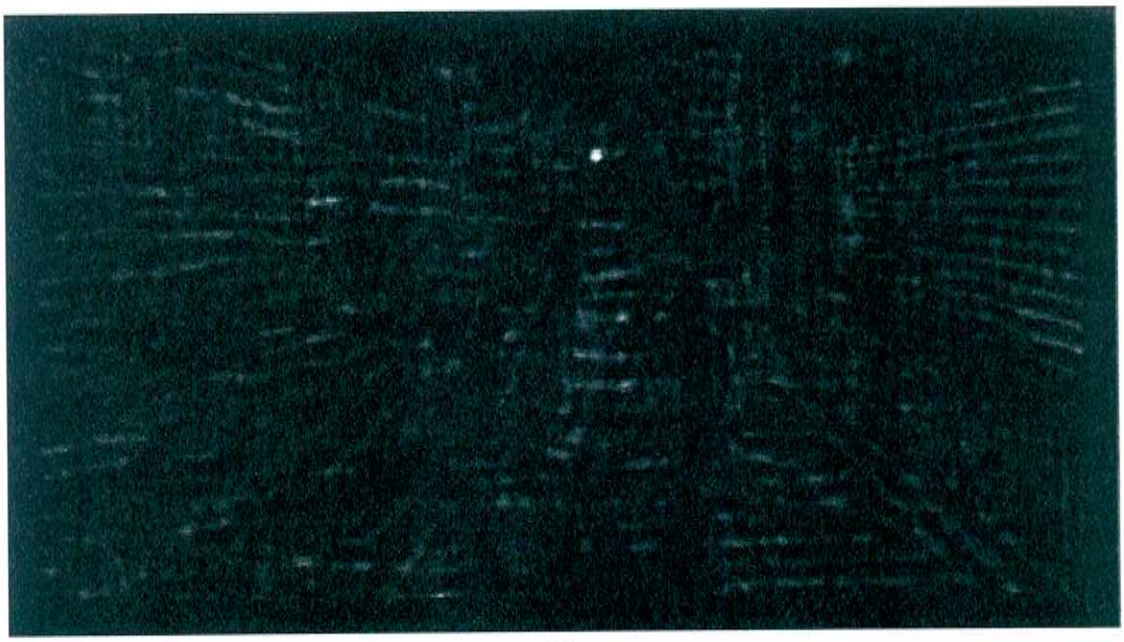
Computing this confidence map is quite slow even for just one size. However, the particle filter works in such a way, that the confidences are only needed for a fixed number of locations and sizes. This is very convenient, since computing these specific confidences is both simple and fast.

In total, three face detectors were used, one for frontal faces and one for left- and right-profile faces respectively.

4.3 Observation model

As described in Section 2.3, the observation model of the particle filter is used to assign a confidence to each of the current particles (states), based on the current observation. The two features described above are the sources from which the confidences are derived in this work. Since, as already hinted, the skin color segmentation is not robust, it cannot be used for tracking alone. Thus in this work, the skin color map is used as a binary confidence: if the percentage of skin-pixels in the current face box exceeds a certain threshold c_{skin} is set to

(a)



(b)

Figure 4.3: Example of the confidence map generated by the face detectors: (a) Input image. (b) Confidence map for a fixed face size.

one, otherwise it is set to zero. This way, obvious false detections of the face detectors are discarded with help of the skin color segmentation, but the skin color segmentation is not used for deriving the confidences of the other particles.

The confidences of the three face detectors are used directly as described above. Since the three detectors detect different views, the maximum of the confidences of the three detectors was used to give the final confidence. The confidences of the face detectors are not normalized, and, since the frontal face detector has more stages and more weak classifiers, the scores for the frontal face detector are generally higher than for the profile detectors. This is convenient, since the profile face detectors are not as robust as the frontal face detector. Therefore a higher confidence for the more robust frontal face detector is warranted. Because of this inherent asymmetry the confidences of the face detectors did not need to be weighted to incorporate the higher robustness of the frontal face detector.

So the final confidence is computed as follows:

$$c(\mathbf{s}) = \begin{cases} \max(c_f, c_{lp}, c_{rp}) & \text{if } S(\mathbf{s}, \mathbf{I}) > \theta_s \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

where c_f, c_{lp} and c_{rp} are the confidences from the three face detectors, $S(\mathbf{s}, \mathbf{I})$ is the percentage of skin pixels within the face box represented by the state $\mathbf{s}$ in the current image $\mathbf{I}$, and θ_s is a threshold.

4.4 Initialization

To find new tracks and initialize the particles, the face detectors are run in regular detection mode in regions with significant amounts of skin-color. For each detected face a new track is started.

4.5 Detection of lost tracks

A track is deleted if the maximum confidence of all its particles falls below a threshold. If the complete track contains no more than 10 frames, which corresponds to 0.4 seconds, the whole track is discarded.

4.6 Overlap handling

A big issue in tracking of multiple objects is the handling of overlaps between several objects. The general difficulty of this problem is only alleviated in this

work due to the difficult data and the limited amount of information that can be extracted reliably from it.

It can, however, be noted, that when two faces overlap, one face will always be occluded during the overlap, while the other will not. The idea now is that the occluded face should generally have a lower confidence than the non-occluded face. So the overlap handling in this work was done by detecting that two tracks overlap, and then deleting the track with lower confidence. This turned out to be a very effective solution.

5 Classification

The tracker described in the last chapter provides a number of tracks for each shot in a video, where a track is a sequence of face boxes in subsequent frames. The next step, in order to classify or cluster the face images, is feature extraction.

5.1 Alignment

To account for rotations and different scalings of the faces that are gathered by the tracker, it is a good idea to normalize the view of the face, a step usually called registration or alignment.

There are several approaches on how to do this, the simplest of which is to detect the locations of the eyes in the face and then perform a Euclidean transform to move the eyes to predefined coordinates. This way, the in-plane rotations and the different scalings can be normalized. The issue of out-of-plane rotations is however not addressed by this alignment approach.

5.1.1 Eye detection

To detect the eyes in a face image, Haar-cascade based eye detectors are employed, one for each eye. They output a number of detections, from which the most likely ones have to be selected.

It has to be noted, that the eye-detectors were trained for a fixed environment and do not nearly work as reliable as the face detectors. There are many false detections as well as a considerable number of misses.

Because of this, it does not make much sense to apply the eye detectors to face images that are not close to frontal. In profile or half-profile images, eyes will be detected but the locations will most certainly be wrong.

So in this work, the eye detector was only applied to face images with a high confidence for a frontal face.

To determine the best candidates out of the detected eye locations, the mean eye locations of face images were computed by applying the frontal face detector to a database of images with known eye locations. The eye locations nearest to the means were selected as the final detections.

5.1.2 Experiments without alignment

Normally, alignment is a vital step in a face-recognition system. There are however systems, that could avoid the step by providing a lot more data to the system [RBK07]. The additional data is expected to cover at least partly the variations that are normalized out by the alignment step. So in this work, experiments were also done without the alignment step. The idea is, that the tracker provides a sequence of images that are known to belong to the same person and possibly contain variations in pose, illumination and/or scale.

5.2 Feature extraction

The feature extraction method used in this work is based on local appearanches and works as follows [ES05]: first, it is assumed that the input images are divisible into 8×8 pixel blocks. This can always be accomplished by scaling the image to an appropriate size, when no alignment is used, or by cropping the aligned face image to an appropriate size if alignment is used. The image is then split into non-overlapping 8×8 pixel blocks and the DCT is applied to each of the blocks. Then local feature vectors are built using zig-zag scanning, as shown in Figure 5.1. The first coefficient is omitted since it represents the mean gray value of the local block and is mainly influenced by illumination. From the remaining coefficients, the first N are selected, where N is usually 5 or 10. As a final step, the local feature vector is normalized to a unit-norm vector, in order to further reduce illumination effects [ES06]. The global feature vector is then created by concatenating the local feature vectors.

Because of the properties of the DCT (cf. Section 2.4), the resulting feature vector retains much of the original information in the image, while discarding information that is mostly influenced by unwanted effects like illumination and considerably reducing the dimensionality of the data.

5.3 Classification

The details of classification differ for the three application scenarios, but in general, distances between extracted features are used to infer the identity to associate with the images.

For the closed-set identification task, only frontal faces with successful eye detection are considered. The face images are aligned, so that the eyes are at predefined coordinates and then cropped to 56×64 pixels, so that they can be divided into 8 pixel blocks. Then the DCT feature extraction is applied and five or ten coefficients are retained (DCT-5 & DCT-10). The classification then works

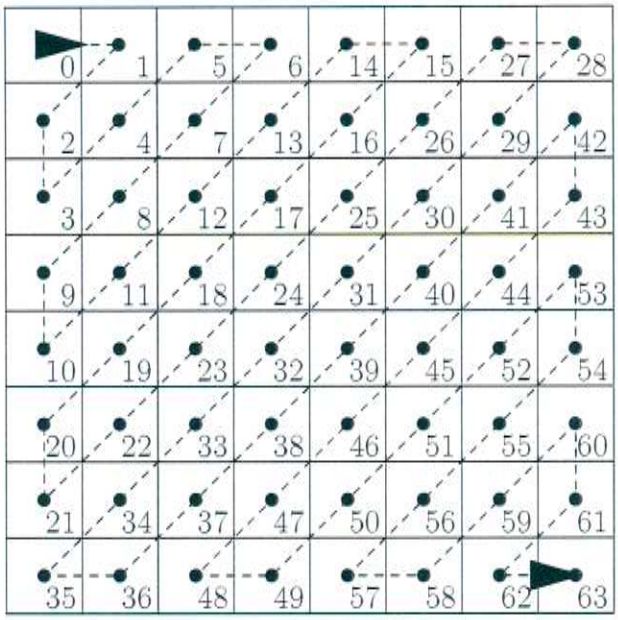


Figure 5.1: Zig-zag scanning of DCT coefficients

as follows: DCT features are extracted from a training set of face images with labeled identities. Then classification is performed by a nearest-neighbor classifier. That is, each testing feature vector is compared with all training feature vectors, and the identity of the nearest training vector is assigned to the testing vector. Very good results have previously been achieved using this approach [ES05].

For automatic retrieval, the process is similar. In this case, one provides a query set, consisting of face images that are aligned as described for the closed-set identification scenario. DCT features are then extracted from each image in the query set. The retrieval then extracts DCT features from each face image to be searched, and retrieves the face images, whose features have a distance to the features one of the query images, that is lower than some threshold.

For interactive retrieval, the face images were not aligned, but just cropped to 48×64 pixels, to reduce the amount of background information in profile face images. Then DCT features are extracted as described above for each face image, that is detected in the video data. Since classification with a large number of feature vectors is slow, and since only tracks are compared with each other in this scenario, as a preprocessing step, the closest distances between face images of two tracks are computed and saved for later use. The retrieval process then starts with a single face that the user selects. The system then enlarges the query set by including face images, that lie on the same track as the selected one. Then the query set is automatically further enlarged by searching for tracks with

distance to one of the tracks in the query set, that is lower than some threshold. This is done iteratively until no more tracks can be found. At that point, the user is presented with a list of the 20 closest matching face images and is asked to select the ones that belong to the person queried for. The tracks containing the selected images are then added to the query set and the automatic query set enlargement procedure is run again. The images selected as not belonging to the queried person are stored as well, in order to not present them to the user again. This process is repeated until no more tracks can be found automatically and the 20 closest matches all belong to different persons than the one queried for.

6 Experiments

Several experiments were performed to assess the performance of the parts of the system, as well as the performance of the whole system in some simple application scenarios. In this section the experimental data will be described and the results of the experiments will be presented.

6.1 Experimental data

Since this work is on TV series, several DVDs of two TV series were used as data. The TV series used are the british series *Coupling* and the german series *Stromberg*. Each episode is about 30 minutes in length and contains around 40,000 frames. The resolution of the video data is 720×405 pixels.

Three episodes of *Coupling* and one episode of *Stromberg* were labeled with shot boundary information, and one episode of *Coupling* was labeled with track and identity information that consists for each frame of the location of the centers of the faces and the identity of the person.

Additionally, the shot boundary detector was run on the TRECVID 2007 data for shot boundary detection, which consists of general TV recordings of various genres, including documentaries, movies, news and commercials. The data consists of 17 video files with 637,805 frames in total and contains 2,463 shot boundaries of various types.

Example frames from the video data can be seen in Figure 6.1.

6.2 Recall / precision metric

In the retrieval literature, precision and recall metrics are commonly used to measure the performance of a system. These metrics have been extensively used in this work as well and are now explained shortly.

The precision and recall metrics suppose a task of retrieval of documents. Here, documents should be understood as an abstract term, which can mean literal documents in a newspaper database for example, but also shot boundaries or tracks. The task is now to retrieve *interesting* documents from some corpus of data.

(a)

(b)

(c) (d)

Figure 6.1: Example frames from the video data: (a) Coupling. (b) Stromberg. (c)-(d) TRECVID 2007 shot boundary data set

Episode	Recall	Precision	F_1-Measure
Coupling S1E1	98.3%	98.3%	98.3%
Coupling S1E2	96.3%	98,0%	97.1%
Coupling S1E6	98.3%	98,1%	98.2%
Stromberg S1E1	97.0%	98.5%	97.7%

Table 6.1: Results of shot boundary detection on TV series data

The precision and recall metrics are then defined as:

$$\text{precision} = \frac{\#\text{ relevant documents retrieved}}{\#\text{ documents retrieved}} \tag{6.1}$$

$$\text{recall} = \frac{\#\text{ relevant documents retrieved}}{\#\text{ relevant documents in corpus}} \tag{6.2}$$

$$\tag{6.3}$$

The precision measures the reliability of the results, or the probability that a retrieved document is really interesting, while the recall measures the completeness of the results, or the probability that a given interesting document is found by the system.

Additionally, sometimes the two metrics are combined yielding the so-called F_1-measure, which is computed as

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6.4}$$

6.3 Shot boundary detection

The shot boundary detector was run on four episodes of the TV series, as indicated in Section 6.1. The results are shown in Table 6.1. As can be observed, the results are very good. Because over 95% of the shot transitions in the TV series are cuts, the results show that the shot boundary detection system handles cuts very well. Indeed most of the errors on this type of data come from the few more difficult transitions, like dissolves or wipes, the latter of which are not handled at all by our system.

Because the TV series data can be considered simple in terms of the shot transitions, the system was also evaluated on the TRECVID 2007 shot boundary detection data, which is also described in Section 6.1 and which contains much more diverse types of video and also much more non-cut shot transitions.

Four different metrics are applied in the TRECVID evaluations

Total precision and recall This metric takes into account all transitions and measures the precision and recall of the system.

Cut precision and recall This metric only takes into account the transitions that consist of five frames or less.

Gradual precision and recall This metric only takes into account the transitions that consist of more than five frames.

Frame precision and recall This metric takes into account the gradual transitions (i.e. transitions consisting of more than five frames), that were correctly detected by the system and measures the recall and precision of the frames in each transition.

Ten runs with different parameters were submitted. The runs are described in Table 6.2. The results for shot boundary detection on the TRECVID 2007 data are shown in Tables 6.3-6.6. The F_1-measure for each metric has been added. Graphs of the results including the results from the other groups participating in the TRECVID 2007 evaluations are shown in Figures 6.2 and 6.3.

Again, it can be observed, that the system handles short transitions like cuts and fast dissolves very well. However, the results for longer transitions are not quite as high. This mainly comes from the fact, that the dissolve detector failed to work reliably. From Figure 6.3, it can be inferred, that the candidate selection works, because the frame precision and recall are reasonably high, but the detection of dissolves did not work well. This means that the heuristic modelling that was used successfully for the short dissolves, can not be employed for longer dissolve transitions, because the model can not cover image changes, that occur within the dissolve transition. This problem has to be addressed in the future to make the shot boundary system more robust.

However, since the video data used for the rest of this thesis contains mostly cuts, the shot boundary detector in its current state is completely adequate.

6.4 Face tracking

To evaluate the performance of the face tracker, first a useful performance metric has to be established. In the literature, the *CLEAR Multi-Object Tracking (MOT)* metrics can be found [BS08]. They measure a tracker's ability to accurately localize tracked objects and consistently track objects through time, so that ideally only one trajectory per object exists, and each trajectory follows exactly one object.

There are two problems in using this approach. First, because in this work, only the face center is labeled and the MOT metrics expect face bounding boxes,

Run	Description
base	Baseline
mod1	Monochrome frames with with color are discarded
mod2	Lower thresholds for cuts
mod3	Higher thresholds for cuts
mod4	Lower thresholds for fast dissolves
mod5	Higher thresholds for fast dissolves
mod6	Dissolves have precedence over other transitions
mod7	Lower thresholds for dissolves
mod8	Higher thresholds for dissolves
mod9	No dissolve detection

Table 6.2: Descriptions of the ten runs submitted for TRECVID 2007

Run	Comment	Recall	Precision	F_1-Measure
base	Baseline	91,3	59,8	72,2
mod1	Discard FOIs with color	91,4	60,1	72,5
mod2	lower thr. for cuts	93,1	58,8	72,1
mod3	higher thr. for cuts	90,0	59,8	71,9
mod4	lower thr. for fast diss.	92,7	51,3	66,0
mod5	higher thr. for fast diss.	90,9	60,1	72,3
mod6	dissolves have precedence	81,3	53,4	64,5
mod7	lower thr. for dissolves	91,8	40,1	55,8
mod8	higher thr. for dissolves	90,4	81,7	85,8
mod9	no dissolve detection	87,6	92,0	89,8

Table 6.3: Total precision and recall on TRECVID 2007 data

Run	Comment	Recall	Precision	F_1-Measure
base	Baseline	93,6	94,1	93,9
mod1	Discard FOIs with color	93,9	93,9	93,9
mod2	lower thr. for cuts	95,6	89,8	92,6
mod3	higher thr. for cuts	92,3	95,1	93,9
mod4	lower thr. for fast diss.	95,5	70,2	80,9
mod5	higher thr. for fast diss.	93,2	95,7	94,4
mod6	dissolves have precedence	82,6	94,4	88,1
mod7	lower thr. for dissolves	93,5	94,1	93,8
mod8	higher thr. for dissolves	93,7	94,1	93,9
mod9	no dissolve detection	93,8	94,0	93,9

Table 6.4: Cut precision and recall on TRECVID 2007 data

Run	Comment	Recall	Precision	F_1-Measure
base	Baseline	65,5	8,9	15,7
mod1	Discard FOIs with color	63,1	8,7	15,4
mod2	lower thr. for cuts	65,5	9,0	15,9
mod3	higher thr. for cuts	65,5	8,9	15,7
mod4	lower thr. for fast diss.	61,7	9,2	16,0
mod5	higher thr. for fast diss.	65,5	8,9	15,6
mod6	dissolves have precedence	67,0	7,8	13,9
mod7	lower thr. for dissolves	72,8	4,4	8,3
mod8	higher thr. for dissolves	54,4	23,3	32,6
mod9	no dissolve detection	20,4	44,2	27,9

Table 6.5: Gradual precision and recall on TRECVID 2007 data

Run	Comment	Recall	Precision	F_1-Measure
base	Baseline	74,8	71,2	72,9
mod1	Discard FOIs with color	74,6	71,3	72,9
mod2	lower thr. for cuts	74,8	71,2	72,9
mod3	higher thr. for cuts	74,9	71,3	73,0
mod4	lower thr. for fast diss.	73,5	72,0	72,8
mod5	higher thr. for fast diss.	74,6	71,2	72,8
mod6	dissolves have precedence	76,2	69,9	72,9
mod7	lower thr. for dissolves	73,8	70,5	72,1
mod8	higher thr. for dissolves	74,5	73,4	73,9
mod9	no dissolve detection	77,6	87,2	82,1

Table 6.6: Gradual frame precision and recall on TRECVID 2007 data

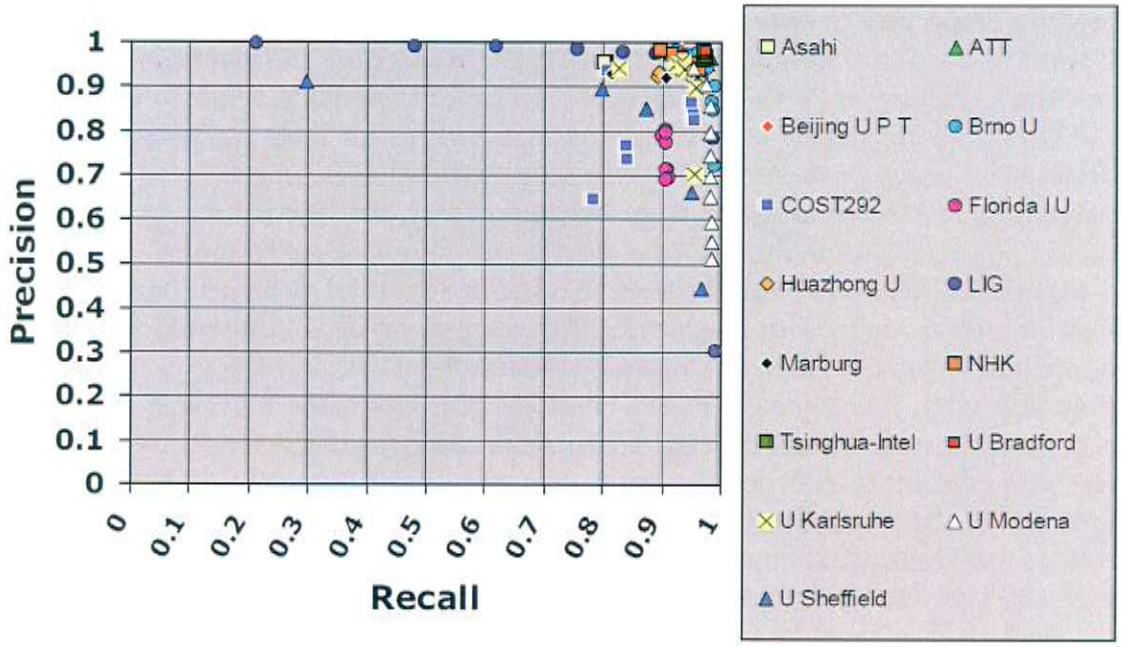


Figure 6.2: Cut precision and recall on the TRECVID 2007 data

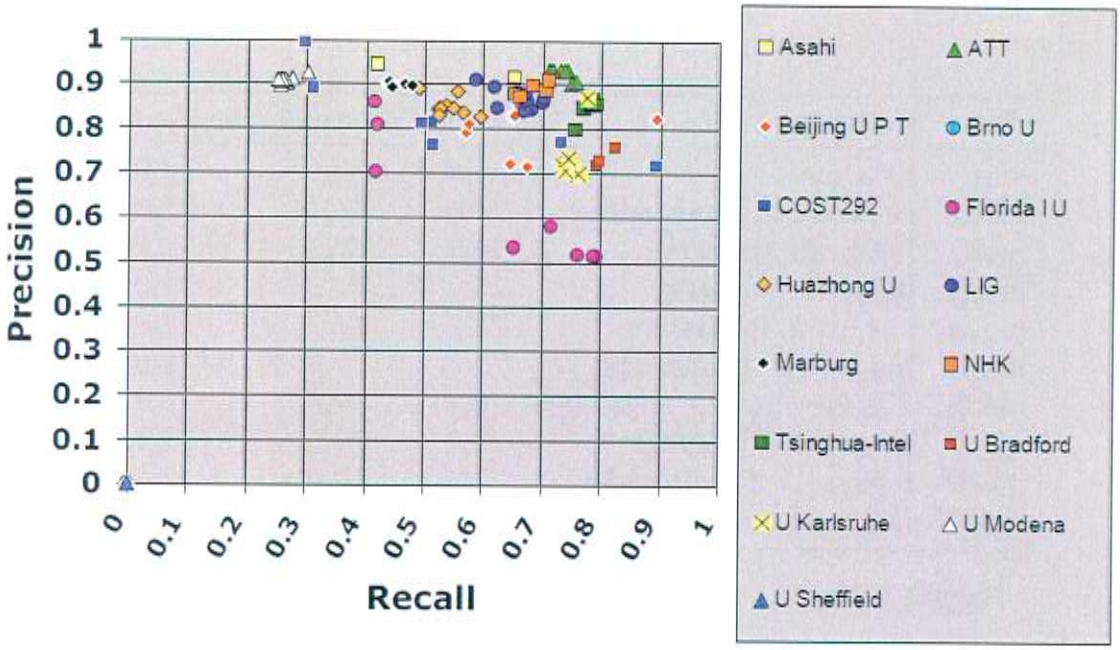


Figure 6.3: Gradual frame precision and recall on the TRECVID 2007 data

the metric would have to be changed, so that the results could not be meaningfully compared to other systems. And second, the MOT metrics expect the tracker to do identification while tracking, i.e. associating the two parts of a broken track. Failure to do this is penalized in the MOT metrics. Since in this work, the face tracking step is used to extract face images for later classification, this association is not performed while tracking. Instead, it is implicitly or explicitly performed in the classification step. Because of this, the face tracker of this work would be penalized and the results, again, would not be representative.

So instead, the *idea* behind the development of the MOT metrics was used to build a metric suitable for this work. First, it needs to be established, how many of the persons in the video are found by the tracker. Of course it has to be defined precisely, what *found* means in this case. For this work, the following approach was used: in each frame, the face labels were assigned to the tracks by choosing for each face label that nearest track, but only if the face label lies within the face box of the track (i.e. the track covers the face label). The identities for the frames in a track are then used to assign an identity to the whole track by simple majority vote. So in the end each track has one identity assigned to it, or it has no identity, if none of its face boxes are near a label. This can happen for false detections but also for background characters, that are not labeled. In the latter case, the tracks were not considered, since tracking a background character is not an error.

Using the established track correspondences, the *track precision* and *recall* are

defined as follows:

$$\text{track precision} = \frac{\#\text{ tracks that have an identity assigned}}{\#\text{ of tracks found}} \tag{6.5}$$

$$\text{track recall} = \frac{\#\text{ of tracks that have an identity assigned}}{\#\text{ tracks that are in the manual labels}} \tag{6.6}$$

where in the case of track recall, multiple detected tracks for one person in a shot are counted only once in the numerator, because otherwise the results would overestimate the actual performance, since broken tracks would be counted multiple times.

This metric gives an idea what percentage of the persons occurring in the video are correctly detected (at least for a short time) and how many completely false detections there are. It says however nothing about how good the tracking is really working. Therefore another metric is needed.

The second metric concerns the tracking quality. What we want to know is, for each track that is found, how many of the found face boxes actually cover the correct face and how many of the faces in the actual labeled track are covered by the detected track. To this end, exactly those two things are computed for each track, and then a weighted average is taken with weights proportional to the length of the track. The idea is that a badly tracked trajectory with a high length is worse than a badly tracked trajectory that is very short. So the metric is termed *weighted average frame precision* and *recall* (WAFP & WAFR), and is computed as follows:

$$\text{WAFP} = \sum_{t=1}^{N} \frac{\text{frames}(t)}{C} \frac{\#\text{ face boxes in track } t\text{, that cover correct face}}{\#\text{ face boxes in track } t} \tag{6.7}$$

$$\text{WAFR} = \sum_{t=1}^{N} \frac{\text{frames}(t)}{C} \frac{\#\text{ face boxes in track } t\text{, that cover correct face}}{\#\text{ frames in labeled track corresponding to track } t} \tag{6.8}$$

where $\text{frames}(t)$ is the number of frames in track t, N is the number of tracks detected, and $C = \sum_t \text{frames}(t)$ is a normalization constant.

This measure gives an idea of the quality of the tracks. If the tracks cover a majority of the faces actually occurring in the shot, then WAFR will be close to 100%. And if the tracks contain no false detections, i.e. they do not lose the tracked face, or even worse, switch from one person to another, then WAFP will be close to 100%. Obviously only the tracks, where a correspondence to a person could be established, are considered in this metric, since for the others, the frame precision and recall have no meaning.

The results for the labeled episode of Coupling are shown in Table 6.7. As can be seen, the track recall is quite high. If only tracks that contain frontal faces are considered, the track recall is even higher with 96.2%. This means

Metric	Result
Track recall	86.2%
Track recall (frontal)	96.2%
Track precision	77.1%
WAFR	92.2%
WAFP	98.5%

Table 6.7: Tracking results on labeled Coupling episode

that almost all labeled faces are detected and tracked by the tracker. The track precision is a bit lower at 77.1%. This mainly comes from false detections of the face detectors, that are not transient but consistent, and lead to a strong enough confidence in the face detector, that the tracker continues tracking it. Of course this also means that a significant amount of skin color is present at the location. Sometimes this is true, for example necks and ears are sometimes consistently detected as faces by the face detector. But on other occasions the problem is clearly that the color model is not good enough to discard the false detections. So to solve this problem, further work has to go into the color model and maybe some better means of initializing the tracker or some other additional feature to be used for tracking has to be found, in order not to depend solely on the face detectors, which make errors from time to time. On the other hand, the false detections do not play an important role in the classification step, because they have a significantly different appearance compared to regular face images.

The WAFR and WAFP metrics are also very high, indicating that the tracks that follow a real person have a high quality, i.e. they track most of the faces that it should track (WAFR) and very little else (WAFP). A histogram of the frame precisions and recalls for the tracks can be seen in Figure 6.4.

Additionally, it was checked, whether track switches occurred, i.e. whether one track covers faces of two different persons in its course. Of the 975 tracks that were found by the face tracker, only eight contained a track switch. Of those eight cases, six were of the kind that an untracked person occluded a tracked person. In that case there is very little the tracker can do and the real solution would be to track the other person in the first place. The other two cases are real failures of the overlap handling procedure. However two tracks is a very low number, corresponding to only 0.2% of the tracks and is acceptable.

From the test video with about 44,000 frames, about 66,000 face images were extracted by the face tracker.

So it can be concluded that the face tracker works very reliably, extracting most of the faces of the tested video.

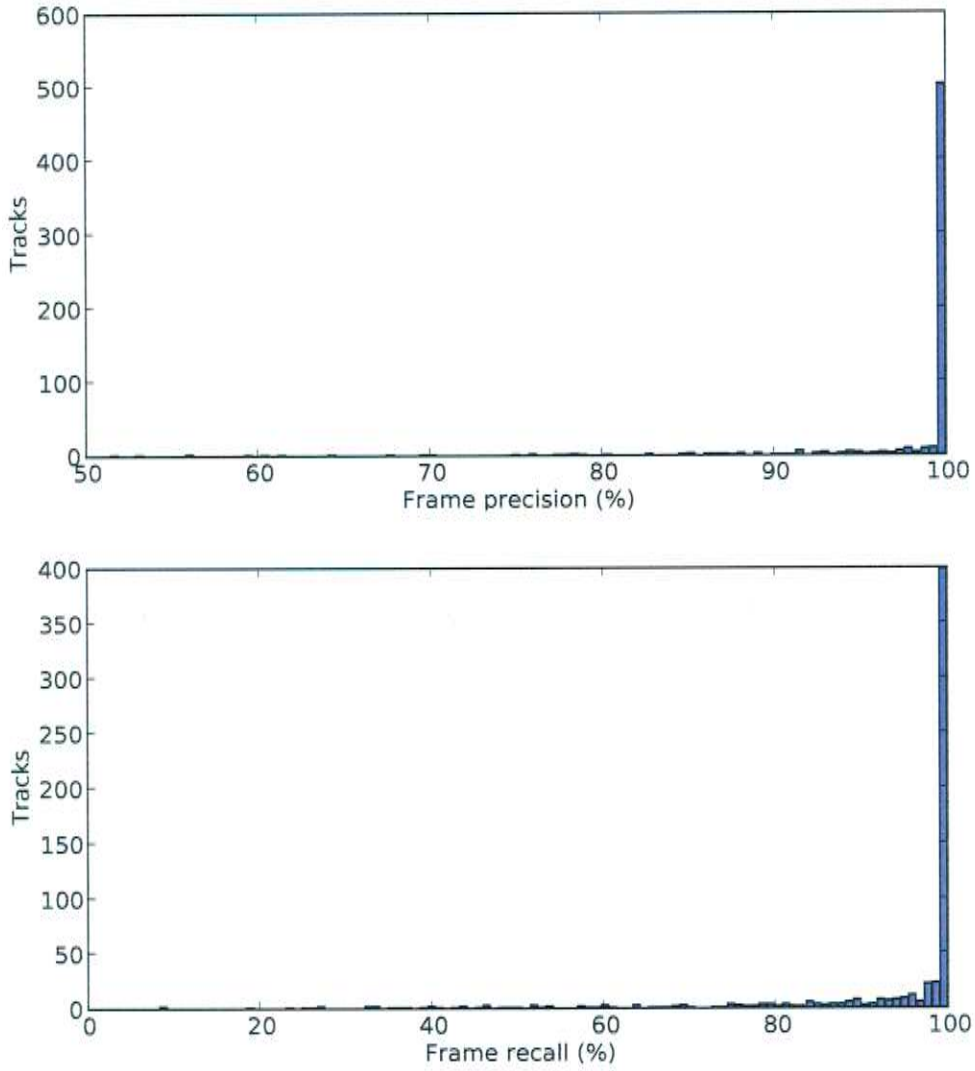


Figure 6.4: Histogram of frame precision and recall of the face tracks

6.5 Application scenarios

Apart from testing the shot boundary detector and the tracker, three simple application scenarios were explored in this work, as described in Section 5.3. They are now explained in more detail. All application scenarios work on the labeled *Coupling* episode.

6.5.1 Closed-set identification

This use-case works on the aligned frontal images provided by the face tracker after applying eye-location based alignment. The face images that do not belong to one of the six main characters, shown in Figure 6.5, are removed to create a closed-set identification scenario. Then the face images extracted from the first quarter of the video were used as training images and the rest were used for testing. The feature extraction algorithm of this system was compared with several other well-known face recognition algorithms. This task also has the advantage that its results can be compared with previous results on frontal face databases, giving an indication of how difficult the data used in this work is relative to well-known frontal face databases often used for face recognition benchmarks.

The results for the closed-set identification scenario are shown in Table 6.8. It can be seen that our DCT algorithm performs significantly better than all the other algorithms. However, the result of about 70% correct recognition rate is also lower than the results obtained on well-known face databases such as FERET [PWHR98], CMU PIE [SBB02], YALE [BHK97] or FRGC [PFS+05], even though the number of subjects in this case is quite small. The main reason for this is that the face databases come with hand-labeled eye locations. In this work, the eye locations are determined by eye detectors and this step does not work reliably in all cases. Examples of misalignments can be seen in Figure 6.6. This also explains why the Bayesian intra-/extra-personal face recognizer has a very low correct classification rate. It works with difference images between images of the same person and different persons and obviously difference images are very badly affected by misalignment.

If the alignment step fails, the matching performed by our algorithm is not as meaningful anymore, because the local blocks will correspond to different face areas than in the case of correct alignment. Considering this, the results are acceptable but clearly demand for a more robust alignment solution.

Another reason for the low results is that even though only faces, where the confidence for a frontal view is very high, were selected, still a significant number of faces are non-frontal. Since our Euclidean alignment can not compensate out-of-plane rotations, this also affects the performance negatively. To overcome this problem, even more sophisticated alignment approaches have to be

Figure 6.5: The six main characters from Coupling

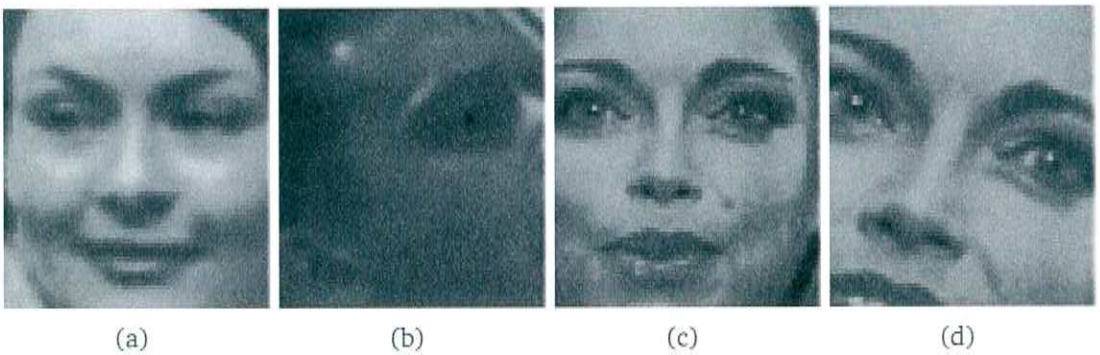


Figure 6.6: Examples of misaligned images: (a) Correctly aligned image. (b)-(d) Misaligned images.

Algorithm	Correct classification rate
DCT (10)	70.5%
DCT (5)	69.4%
EHMM [Nef99]	67.9%
Fisherfaces [BHK97]	63.2%
Eigenfaces [TP91b]	50.4%
Bayesian [MJP00]	27.7%

Table 6.8: Results of closed-set identification

Character	Training images	Testing images
Steve	316	1292
Jane	118	673
Susan	198	1255
Sally	772	353
Jeff	501	2224
Patrick	6	423

Table 6.9: Occurrences of the main characters in the training and testing sets

employed that can at least partly compensate different poses.

And finally, inspection of the training set turned out that the training set is heavily imbalanced, which can be seen from Table 6.9. Especially the character Patrick has a lot less training data, compared to the other characters. This additionally makes this problem more challenging.

6.5.2 Automatic retrieval

In this task, the persons different from the six main characters were not removed from the dataset and the problem setting is different: given a set of query images, find as many images of the same person as possible. Similarly to the closed-set identification task described above, the faces extracted from the first quarter of the video were used to build six query-sets, one for each person. These query-sets were then used to retrieve images of the queried person from the rest of the extracted faces.

The threshold of the distance between a probe image and the query images was varied to give a plot of precision and recall for the automatic retrieval, which is shown in Figure 6.7. Here it becomes obvious that the training/query set for the character Patrick is not representative. For the other characters the results are much better, giving recall rates between 35% and 58% at 90% precision.

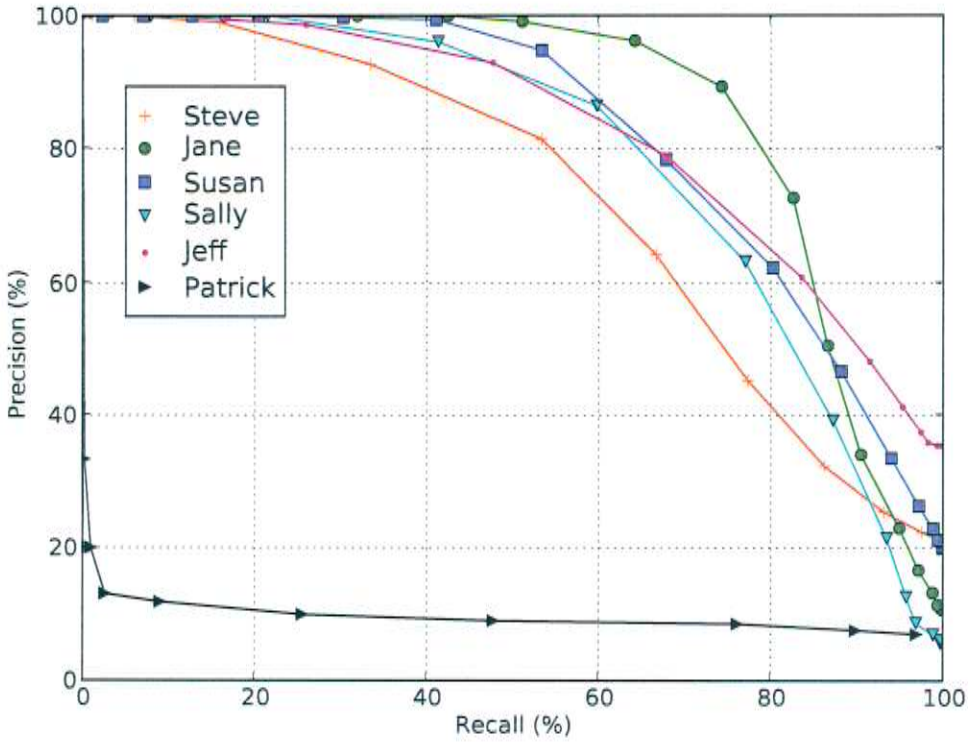


Figure 6.7: Results of automatic retrieval

The problems in this scenario are exactly the same as the ones mentioned in the closed-set scenario. So in order to improve the results further, the same problems have to be solved. However, the results are encouraging because the performance can still yield a useful system, the reasons for the non-optimal performance are known and can be solved in further works.

6.5.3 Interactive retrieval

In this task, the goal was to test, how far one can get with unaligned face images. The idea, as already mentioned, is that the tracker provides sets of face images, that are known to belong to the same person and possibly cover some of the variations that are normalized out by the alignment procedure.

However, the amount of variation in different tracks varies greatly. In some tracks, the person sits still and the face barely moves at all, while in others, the person is moving and looking in different directions, covering large variations in pose. Obviously the latter type of track is more useful for this task, since with it, face images with different poses can be retrieved, whereas with a track of the

Figure 6.8: Interactive selection of matching faces: The top row shows face images from the query set, the bottom row shows the closest matches from the corpus. The user can select matching face images to assist in the retrieval process.

former type, only face images with a similar pose can be retrieved.

Thus, the next idea was to develop an automatic clustering approach that successively enlarges the query set with found tracks in order to retrieve more data. This approach works as follows: first, the user interactively selects a face to be queried from the video. The system then enlarges the query set by including all images from the track that includes the selected face. Then the query set is iteratively enlarged by comparing all images in the query set with all other images and finding close matches. If the distance is lower than a threshold, the found image and the images in the track that contains it are added to the query set.

If no more matches can be found, the user is queried by providing the 20 closest matching faces and letting the user select the correct matches. The correct matches, as well as the faces from the tracks containing them, are then added to the query set. The images and tracks not selected by the user are also stored, in order not to present them again. Afterwards the automatic matching is attempted again and the process repeats until no more matches can be found automatically and none of the 20 closest matches is a correct match.

An example image of the system can be seen in Figure 6.8. In the top row query images are displayed, with the closest matches in the bottom row. The color of the distance denotes the choice of the user whether the images depict the same person.

For this approach to be successful, it is important that the precision of retrieval stays very high. If face images of other persons than the original queried person get in the query set, the approach obviously breaks down because it will automatically extend the query set with more and more images of this wrong person. So the approach critically hinges on the accuracy of the tracker. It should ideally never switch from one person to another in one track. And also the threshold for automatic matching has to be set rather conservative, since there is no recovery from wrong matches.

The results of a representative query for different numbers of interactive selections and different thresholds for automatic query set enlargement are shown in

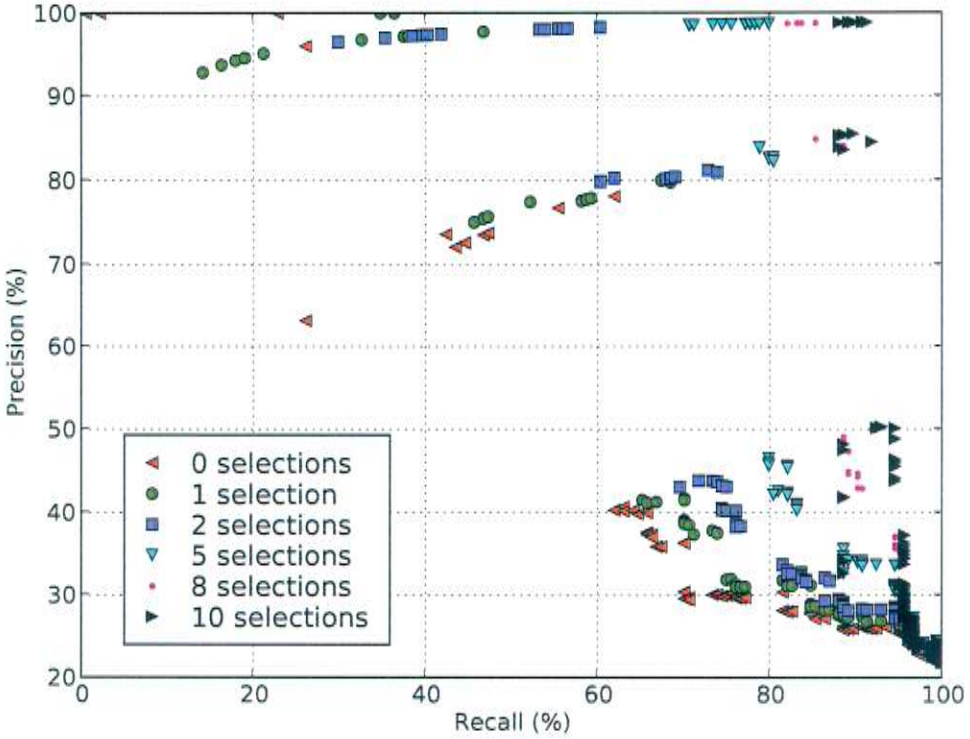


Figure 6.9: Results of interactive retrieval

Figure 6.9. It can be observed that there are gaps in the plot, which come from cases where a different person gets into query set by error, which leads to many easy automatic matches of more face images of this incorrect person, in turn decreasing the precision considerably.

If only the cluster with high precision around 90-100% is considered, the positive effect of the interactive selections becomes clear. Without any interactive selections, the recall only reaches 26%. With increasing number of interactive selections, the recall increases dramatically, reaching already 60% for two selections and 80% for five selections. With ten selections a very high recall rate of 91.3% at a precision of 98.8% is reached.

These results are also encouraging, because for example in search engines, users might be willing to put in some effort to find what they are looking for, by selecting matching images. The results could even be stored and used for subsequent searches, possibly after cross-checking with selections of other users.

7 Conclusion

In this work a system for automatic video analysis has been presented. The system can reliably segment a video into shots, which is a necessary step for any video-analysis task. This has been demonstrated on TV series data and also on the more difficult TRECVID 2007 data.

Furthermore, the system can automatically detect and reliably track faces in the video. It has been shown that the system detects most of the persons in the video while the number of false detections stays reasonably low. The precision of tracking has been found to be very high. That is in the large majority of cases, the track covers the correct face all the time and all frames in a shot that contain the face are covered by the track. As shown, the tracker also very rarely confuses two tracks when they cross. The result is a face tracker that can reliably extract face images from a video, which can then be further processed in many ways.

For this, three application scenarios have been presented. A closed-set identification task was used to compare the performance of our DCT face recognition algorithm to other well-known approaches using the same data. It was shown that our approach gives significantly better results. However, it also turned out that the data is quite difficult for all algorithms, mainly because of problems with the eye localization with Haar cascade-based eye detectors, that lead to misalignment in some of the face images. Also, there were significant pose variations in the data set, which are also not handled well by any algorithm. Finally, the training set was found to be heavily imbalanced, leading to low performance for one of the six subjects.

An automatic retrieval task was explored, to see how our face recognition approach, that was previously mainly used in closed-set or open-set identification scenarios would perform in a retrieval setting. The results are encouraging, but the same problems as in the closed-set scenario affected the system performance negatively. These problems have to be addressed to improve the performance further.

Finally, an interactive retrieval scenario was used to explore the performance that can be achieved with unaligned images on the one hand and interactive assistance on the other hand. The results of this experiment indicate that unaligned images can be used for face recognition, although the performance is of course worse than with correctly aligned frontal images. On the other hand, much more data is available this way and because of the reliable tracks provided by the face tracker, query sets can be built by using all face images from a track.

It was shown that interactive assistance can help significantly in the retrieval process. With only five to ten selections of the user (corresponding to 100-200 face images that have to be classified), high recall rates could be reached, with a precision close to 99%.

8 Future Work

There are several ideas on how to improve the system developed in this work.

For shot boundary detection, a better approach for long dissolve detection has to be developed, as well as detectors for other types of shot boundaries, that are found in TV series, like wipes. Previous systems have successfully used SVMs for dissolve detection and model based approaches for wipe detection [LGZ+06]. These would be viable candidates for detectors that could make the shot boundary system more robust. Also, the speed of the shot boundary detection module could surely be further reduced, since no effort apart from the system design has been put into optimization.

The tracker clearly suffers a bit from the sub-optimal color modelling. More effort has to be spent on finding a way to robustly detect skin-colored regions in TV series data. But even with perfect skin color segmentation, the tracker would still produce false detections, resulting for example from necks or ears. To address these, new features have to be developed, that can be used in addition to the confidences returned by the face detectors. Since most of the track switches resulted from non-tracked persons, it is also clear that the initialization can still be improved. A possible approach could be to use a "super"-tracker, that scans the whole image, possibly using importance sampling [IB98b], and creates new tracks for areas with high confidence. Also, more sophisticated approaches to resolve overlaps of tracks should be tested. Although the results of this work are already very good, track switching is a very serious error in this domain and should be avoided altogether if possible. It could also be tried to establish track correspondences between tracks in the same shot, based on non-facial information, such as color of clothing or hair. The tracker was in no way optimized for performance and could probably be made much faster, probably faster than real-time. That way, it could also be used for online systems such as our portable face recognizer [ST07].

Another idea would be to not only segment the video into shots, but also group related shots that depict the same location at approximately the same time into scenes. From this information, the presence of persons could be inferred, and track association could be performed using hair or clothing information, as described above.

For the classification part, clearly this work is only a first step. Most importantly more labeled data is needed to perform more experiments, also across different movies, for example. A major problem for this kind of data is the need

for alignment of the face images. The approach used in this work is clearly not optimal. Active appearance model (AAM)-based approaches such as [Gao08] could be a solution and allow much more data to be used with better alignment than is possible in this work. The proposed approach not only aligns the image but can also perform pose correction and has been shown to significantly improve identification results for non-frontal faces. Another approach to reduce the effect of misalignment would be to augment the training set with artificial samples generated by varying the eye localizations. This has been shown to improve the results on well-known frontal face databases but has the disadvantage that the amount of training data increases dramatically.

For the use of unaligned images it would be useful to investigate approached to remove the background that is invariably present in non-frontal images. A promising approach can be found in [AZ05].

Finally it would be beneficial to develop a graphical user interface to let users search for persons in videos. That way the performance of the system can be demonstrated in practice and not only with numerical results.

Bibliography

[AHP04] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *Proceedings of the 8th European Conference on Computer Vision*, May 2004, pp. 469–481.

[ALGS07] D. Anguelov, K.-c. Lee, S. B. Gökturk, and B. Sumengen, "Contextual identity recognition in personal photo albums," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2007, pp. 1–7.

[AMGC02] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[AZ05] O. Arandjelović and A. Zisserman, "Automatic face recognition for film character retrieval in feature-length films," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 860–867.

[BA93] M. Black and P. Anandan, "A framework for the robust estimation of optical flow," in *Proceedings of the 4th International Conference on Computer Vision*, 1993, pp. 231–236.

[BHK97] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[Bir97] S. Birchfield, "An elliptical head tracker," in *Conference Record of the 31st Asilomar Conference on Signals, Systems & Computers*, vol. 2, 1997, pp. 1710–1714.

[Bir98] ——, "Elliptical head tracking using intensity gradients and color histograms," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1998, pp. 232–237.

[Bor86] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, vol. 34, no. 3, pp. 344–371, 1986.

[Bra98] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision*, 1998, pp. 214–219.

[BS08] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, 2008.

[Che95] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 790–799, 1995.

[CM02] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[DDCF01] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb, "Plan-view trajectory estimation with dense stereo background models," in *Proceedings of the 8th IEEE International Conference on Computer Vision*, vol. 2, 2001, pp. 628–635.

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, Oct. 2000.

[DTK+02] D. Demirdjian, K. Tollmar, K. Koile, N. Checka, and T. Darrell, "Activity maps for location-aware computing," in *Proceedings of the 6th IEEE Workshop on Applications of Computer Vision*, 2002, pp. 70–75.

[ES05] H. K. Ekenel and R. Stiefelhagen, "Local appearance based face recognition using discrete cosine transform," in *Proceedings of the 13th European Signal Processing Conference (EUSIPCO 2005)*, Sep. 2005.

[ES06] ——, "Analysis of local appearance-based face recognition: Effects of feature selection and feature normalization," in *Proceedings of the CVPR Biometrics Workshop*, Jun. 2006.

[ESZ06] M. Everingham, J. Sivic, and A. Zisserman, "'Hello! My name is... Buffy' - Automatic naming of characters in TV video," in *Proceedings of the 17th British Machine Vision Conference*, Sep. 2006, pp. 889–908.

[FFM] "Ffmpeg video codec library." [Online]. Available: http://ffmpeg.mplayerhq.hu/

[FH00] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient matching of pictorial structures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 66–73.

[FHT00] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *The Annals of Statistics*, vol. 28, pp. 337–374, Apr. 2000.

[FLK+02] J. Fritsch, S. Lang, A. Kleinehagenbrock, G. Fink, and G. Sagerer, "Improving adaptive skin color segmentation by incorporating results from face detection," in *Proceedings of the 11th IEEE International Workshop on Robot and Human Interactive Communication*, 2002, pp. 337–343.

[FS97] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[FS99] ——, "A short introduction to boosting," *Journal of the Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, Sep. 1999.

[Gao08] H. Gao, "Face registration with active appearance models for local appearance-based face recognition," Diplomarbeit, Interactive Systems Labs, Universität Karlsruhe (TH), Jun. 2008.

[GG05] S. Gangaputra and D. Geman, "A unified stochastic model for detecting and tracking faces," in *Proceedings of the 2nd Canadian Conference on Computer and Robot Vision*, 2005, pp. 306–313.

[Gor06] D. O. Gorodnichy, "Seeing faces in video by computers," *Image and Vision Computing*, vol. 24, no. 6, pp. 551–556, Jun. 2006.

[Goy01] V. K. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 9–21, Sep. 2001.

[GW01] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Pearson Education, 2001.

[HCJW07] M. Horton, M. Cameron-Jones, and R. Williams, "Multiple classifier object detection with confidence measures," in *Proceedings of the 20th Australian Joint Conference on Artificial Intelligence (AI)*, Dec. 2007, pp. 559–568.

[HL01] E. Hjelmås and B. K. Low, "Face detection: A survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236–274, Sep. 2001.

[HSE+95] J. Hafner, H. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient color histogram indexing for quadratic form distance functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 729–736, 1995.

[IB98a] M. Isard and A. Blake, "CONDENSATION—conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, Aug. 1998.

[IB98b] ——, "ICONDENSATION: unifying low-level and high-level tracking in a stochastic framework," in *Proceedings of the 5th European Conference on Computer Vision*, 1998, pp. 893–908.

[JR02] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, vol. 46, no. 1, pp. 81–96, Jan. 2002.

[JV03] M. J. Jones and P. Viola, "Fast multi-view face detection," Mitshubishi Engineering Research Laoratories (MERL), Tech. Rep., Aug. 2003.

[Kal60] R. E. Kalman, "A new approach to linear filtering and predictive problems," *Transactions of ASME, Journal of basic engineering*, vol. 82, pp. 34–45, 1960.

[Kha03] S. A. Khayam, "The discrete cosine transform (DCT): Theory and application," Department of Electrical & Computing Engineering, Michigan State University, Tech. Rep., 2003.

[KHDM98] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, Mar. 1998.

[KK96] R. Kjeldsen and J. Kender, "Finding skin in color images," in *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 312–317.

[KMB07] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern Recognition*, vol. 40, no. 3, pp. 1106–1122, Mar. 2007.

[LGZ+06] Z. Liu, D. Gibbon, E. Zavesky, B. Shahraray, and P. Haffner, "AT&T Research at TRECVID 2006," in *Proceedings of the NIST TRECVID Workshop*, Nov. 2006.

[Lie99] R. Lienhart, "Comparison of automatic shot boundary detection algorithms," in *Proceedings of the 7th Conference on Storage and Retrieval for Image and Video Databases*, Jan. 1999, pp. 25–30.

[Lie01] ——, "Reliable transition detection in videos: A survey and practitioner's guide," *International Journal of Image and Graphics*, vol. 1, no. 3, pp. 469–486, 2001.

[LKL03] H.-S. Lee, D. Kim, and S.-Y. Lee, "Robust face-tracking using skin color and facial shape," in *Proceedings of the 4th International Conference on Audio- and Video-Based Biometric Person Authentication*, 2003, pp. 1060–1061.

[LKP03] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," in *Proceedings of the 25th Symposium of the German Association for Pattern Recognition*, 2003, pp. 297–304.

[Low04] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[MDWW04] Y. Ma, X. Ding, Z. Wang, and N. Wang, "Robust precise eye location under probabilistic framework," in *Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition*, May 2004, pp. 339–344.

[MJP00] B. Moghaddam, T. Jebara, and A. Pentland, "Bayesian face recognition," *Pattern Recognition*, vol. 33, no. 11, pp. 1771–1782, Nov. 2000.

[Nef99] A. Nefian, "A hidden Markov model-based approach for face detection and recognition," Ph.D. dissertation, Georgia Institute of Technology, 1999.

[NGSM05] K. Nickel, T. Gehrig, R. Stiefelhagen, and J. McDonough, "A joint particle filter for audio-visual speaker tracking," in *Proceedings of the 7th International Conference on Multimodal Interfaces*, 2005, pp. 61–68.

[Ope] "Open source computer vision library (OpenCV)." [Online]. Available: http://www.intel.com/technology/computing/opencv/

[PBC05] S. L. Phung, A. Bouzerdoum, and D. Chai, "Skin segmentation using color pixel classification: analysis and comparison," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 148–154, 2005.

[PFS+05] P. Phillips, P. Flynn, T. Scruggs, K. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 947–954, Jun. 2005.

[POP98] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proceedings of the 6th International Conference on Computer Vision*, Jan. 1998, pp. 555–562.

[PWHR98] P. J. Phillipsa, H. Wechslerb, J. Huangb, and P. J. Raussa, "The feret database and evaluation procedure for face-recognition algorithms," *Image and Vision Computing*, vol. 16, no. 5, pp. 295–306, Apr. 1998.

[Qua] "The Quaero project." [Online]. Available: http://www.quaero.org/

[RBK07] D. Ramanan, S. Baker, and S. Kakade, "Leveraging archival video for building face datasets," in *Proceedings of the IEEE 11th International Conference on Computer Vision*, Oct. 2007, pp. 1–8.

[RJ07] J. Ren and J. Jiang, "Statistical classification of skin color pixels from MPEG videos," in *Proceedings of the 9th International Conference on Advanced Concepts for Intelligent Vision Systems*, Aug. 2007, pp. 395–405.

[RMG98] Y. Raja, S. J. McKenna, and S. Gong, "Tracking and segmenting people in varying lighting conditions using colour," in *Proceedings of the 3rd. International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 228–233.

[SB90] M. J. Swain and D. H. Ballard, "Indexing via color histograms," in *Proceedings of the 3rd International Conference on Computer Vision*, Dec. 1990, pp. 390–393.

[SB91] ——, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, Nov. 1991.

[SBB02] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression (PIE) database," in *Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition*, May 2002, pp. 46–51.

[Sch99] R. Schapire, "A brief introduction to boosting," in *Proceedings of the International Joint Conferenceon on Artificial Intelligence*, 1999, pp. 1401–1405.

[SCT02] M. C. Shin, K. I. Chang, and L. V. Tsap, "Does colorspace transformation make any difference on skin detection?" in *Proceedings of the 6th IEEE Workshop on Applications of Computer Vision*, 2002, pp. 275–279.

[SEZ05] J. Sivic, M. Everingham, and A. Zisserman, "Person spotting: video shot retrieval for face sets," in *Proceedings of the 4th Conference on Image and Video Retrieval*, 2005, pp. 226–236.

[SM05] G. Shakhnarovich and B. Moghaddam, "Face recognition in subspaces," in *Handbook of Face Recognition*, S. Z. Li and A. K. Jain, Eds. Springer, 2005, pp. 141–168.

[SMHL00] M. Soriano, B. Martinkauppi, S. Huovinen, and M. Laaksonen, "Skin detection in video under changing illumination conditions," in *Proceedings of the 15th International Conference on Pattern Recognition*, vol. 1, 2000, pp. 839–842.

[SSZ04] J. Sivic, F. Schaffalitzky, and A. Zisserman, "Object level grouping for video shots," in *Proceedings of the 8th European Conference on Computer Vision*, May 2004, pp. 189–210.

[ST94] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 1994.

[ST07] L. Szasz-Toth, "Open-set face recognition," Studienarbeit, Interactive Systems Labs, Universität Karlsruhe (TH), Oct. 2007.

[Sta06] J. Stallkamp, "Video-based face recognition using local appearance-based models," Diplomarbeit, Interactive Systems Labs, Universität Karlsruhe (TH), Nov. 2006.

[SZS06] J. Sivic, C. L. Zitnick, and R. Szeliski, "Finding people in repeated shots of the same scene," in *Proceedings of the 16th British Machine Vision Conference*, 2006, pp. 909–918.

[TP91a] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 1991, pp. 586–591.

[TP91b] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[TSFA00] J.-C. Terrillon, M. N. Shirazi, H. Fukamachi, and S. Akamatsu, "Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images," in *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, 2000, pp. 54–61.

[VJ01] P. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 511–518.

[VJ04] ——, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[VSA03] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A survey on pixel-based skin color detection techniques," in *Proceedings of the International Conference on Computer Graphics & Vision (GraphiCon)*, Sep. 2003.

[VSM03] R. C. Verma, C. Schmid, and K. Mikolajczyk, "Face detection and tracking in a video by propagating detection probabilities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1215–1228, 2003.

[WB04] G. Welch and G. Bishop, "An introduction to the Kalman filter," Department of Computer Science, University of North Carolina at Chapel Hill, Tech. Rep., 2004.

[YKA02] M.-H. Yang, D. Kriegman, and N. Ahuja, "Detecting faces in images: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 34–58, 2002.

[YO99] T.-W. Yoo and I.-S. Oh, "A fast algorithm for tracking human faces based on chromatic histograms," *Pattern Recognition Letters*, vol. 20, pp. 967–978, Oct. 1999.

[YWX+05] J. Yuan, H. Wang, L. Xiao, D. Wang, D. Ding, Y. Zuo, Z. Tong, X. Liu, S. Xu, W. Zheng, X. Li, Z. Si, J. Li, F. Lin, and B. Zhang, "Tsinghua

University at TRECVID 2005," in *Proceedings of the NIST TRECVID Workshop*, 2005.

[YZC+04] J. Yuan, W. Zheng, L. Chen, D. Ding, D. Wang, Z. Tong, H. Wang, J. W. J. Li, F. Lin, and B. Zhang, "Tsinghua University at TRECVID 2004: Shot boundary detection and high-level feature extraction," in *Proceedings of the NIST TRECVID Workshop*, 2004.

[ZCK98] W. Zhao, R. Chellappa, and A. Krishnaswamy, "Discriminant analysis of principal components for face recognition," in *Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, Apr. 1998, pp. 336–341.

[ZCPR03] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: a literature survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, 2003.

[ZKC03] S. Zhou, V. Krueger, and R. Chellappa, "Probabilistic recognition of human faces from video," *Computer Vision and Image Understanding*, vol. 91, no. 1–2, pp. 214–245, Aug. 2003.