Institut für Visualisierung und Interaktive Systeme
Abteilung Intelligente Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3073

# Development of a two layer classification system for situations in meetings

Antoniya Tyaneva

| | |
|---|---|
| **Studiengang:** | Informatik |
| **Prüfer:** | Prof. Dr. Gunther Heidemann |
| **Betreuer:** | Dipl.-Inf. Julia Möhrmann |
| **begonnen am:** | August 16, 2010 |
| **beendet am:** | February 15, 2011 |
| **CR-Klassifikation:** | I.2.10, I.4.7, I.4.8, I.4.10, I.5.0 |

# Contents


**1 Introduction** **9**
- 1.1 Motivation . . . 9
- 1.2 Prerequisites . . . 10
- 1.3 Document Overview . . . 11

**2 Person Tracking** **13**
- 2.1 Introduction to the Concepts of Tracking . . . 13
  - 2.1.1 Tracking Categories . . . 13
  - 2.1.2 Background Subtraction for Motion Detection . . . 15
- 2.2 Tracking Algorithm - Main Modules . . . 17
  - 2.2.1 Motion Detection . . . 18
  - 2.2.2 Person Region Tracking . . . 22
  - 2.2.3 Face Detection . . . 24
- 2.3 Experiments and Results . . . 28

**3 Face Pose Estimation** **31**
- 3.1 Face Pose Estimation Methods - Overview . . . 31
  - 3.1.1 Appearance template methods . . . 32
  - 3.1.2 Detector array methods . . . 32
  - 3.1.3 Nonlinear regression methods . . . 33
  - 3.1.4 Manifold embedding methods . . . 34
  - 3.1.5 Flexible models . . . 35
  - 3.1.6 Geometric methods . . . 35
  - 3.1.7 Tracking methods . . . 35
  - 3.1.8 Hybrid methods . . . 36
- 3.2 Eigenfaces for Face Pose Estimation . . . 36
  - 3.2.1 Calculation of Eigenfaces . . . 40
  - 3.2.2 Face Pose Classification . . . 42
- 3.3 Experiments and Results . . . 42

**4 Action Recognition** **47**
- 4.1 Feature Extraction . . . 48
- 4.2 Statistical Classification . . . 50
  - 4.2.1 Discrete Markov Processes . . . 51
  - 4.2.2 Hidden Markov Models . . . 52
  - 4.2.3 Parameter estimation . . . 54
- 4.3 Experiments and Results . . . 56

**5 Implementation structure** **61**

**6 Conclusion and Future work** **65**
- 6.1 Future work . . . 65

**Bibliography** **67**

# List of Figures


1.1 Incorporating current speaker . . . 10
1.2 Sample images from PETS-ICVS 2000 data set . . . 11
1.3 Sample pose set of cropped faces . . . 12

2.1 Phases of a typical motion detection approach . . . 16
2.2 Tracking Algorithm Overview . . . 18
2.3 Motion detection for person tracking in meeting rooms . . . 19
2.4 Foreground images obtained with approximate median approach . . . 22
2.5 Foreground images obtained with selective running average . . . 22
2.6 Blob Proximity . . . 24
2.7 Blob clustering and splitting . . . 25
2.8 Face detection over the PETS-ICVS data set . . . 25
2.9 Haar-like features . . . 26
2.10 Integral Image . . . 26
2.11 Summation Area Tables of Integral Images . . . 27
2.12 Rejection cascade . . . 27
2.13 Typical Haar-like features for frontal face . . . 28
2.14 Person tracking revisited . . . 30

3.1 Head Motion Range . . . 31
3.2 Appearance Template and Detector Array Methods . . . 33
3.3 Non-linear Regression and Manifold Embedding Methods . . . 34
3.4 Flexible and Geometric Methods . . . 36
3.5 Tracking and Hybrid Methods . . . 37
3.6 Discrete class division of the face training samples . . . 39
3.7 Eigenface samples for pose classes 'LEFT' and 'RIGHT' . . . 41
3.8 Error Quotes for Four Class Pose Division . . . 43
3.9 Labeling Interface for Large Image data sets . . . 43
3.10 Evaluation of face pose estimation . . . 44
3.11 Error Quotes Summary . . . 45
3.12 Incorporating the currently observed person . . . 46

4.1 Feature Stream Window . . . 48
4.2 Motion based features . . . 50
4.3 Weather Markov Chain . . . 52
4.4 Weather Hidden Markov Model . . . 53
4.5 Left-to-right HMM topology . . . 57
4.6 Visualization of recognized actions . . . 58
4.7 Evaluation of recognized actions . . . 59

# Abstract

Studying the human behavior in meetings requires the recognition of the individual actions of all participants. Typical situations like discussion and presentation can be characterized using different behavioral cues. In order to deviate an interpretation of meeting situations the persons have to be detected and reliably tracked. The estimation of the face pose can be used to infer who is the current focus of attention, because people naturally turn their faces to the current speaker. In this work we present an approach to recognize individual activities based on processing videos of group meetings. Persons are detected using the selective running average approach for background subtraction. The Viola-Jones method is applied to locate the faces in the image plane. For each of the detected faces a feature vector is calculated using principal component analysis. Face pose estimation is achieved with distance based comparison in the feature space. For action recognition a set of global motion features is extracted for each participant. The resulting vector sequences are used for action recognition using hidden Markov models. The correct recognition rates of the developed system seem promising with rates for individual activities between 62% and 100%.

# 1 Introduction

## 1.1 Motivation

Meetings are an inevitable part of the organizational life and modern business processes. During meetings people discuss current topic issues, share opinions or try to reach an agreement about a common issue. Such interpersonal communication events provide a rich resource of information, which can be used for various research tasks. Studying of human behavior in meetings and the development of scientific models based on human interaction can change many processes and return to positive impact on businesses or public services. The automatic extraction and interpretation of human behavior during meetings finds wide application in automated meeting annotation and browsing systems.

In order to enable the research and focus on specific content extraction from meetings, recordings of predefined meeting scenarios such as PETS-ICVS 2000 dataset are provided [1]. The scenarios are relatively simple, because they contain only predefined courses of action and do not cover all real life meeting situations. The meetings are also recorded under constant lightning conditions. Despite these limitations the content of such 'controlled' meetings is easier to understand. Such scenarios also provide the possibility of defining ground truth data, which includes the preferred design for the outcome. Evaluating different models and applications on the same data set makes the results comparable for further analysis.

In this thesis we are interested in the automatic video analysis of such scenario meetings. Video data provides information about the room occupation, participants' actions and the course of actions in meetings. The ability to recognize human interactions and activities, such as 'sitting down', 'raising hand', 'nodding' or 'head shaking', allows the deduction of high level information, such as the interpretation of scenarios. We could, for example, automatically make decisions about the observed situation, like do we see a presentation, a discussion, or voting. Such key events may prove to be appropriate search criteria, which enable automated structuring and summarizing of the recorded video content. In order to classify communication scenarios the following central questions should be considered and automatically answered:

- Where are the faces of the meeting participants?
- Where is the focus of attention?
- Who is moving during the meeting?
- What kind of actions perform the individual persons?
- What conclusions can be drawn for the overall situation?

Solving these problems provides the possibility of drawing conclusions about situations in the meeting. Since previous work focuses on the recognition of activities based on static frames only, we set our focus on the following objectives:

- Detection of moving persons and tracking over time;
- Face detection and estimation of the face pose orientation;
- Recognition of activities;

Face detection provides some information about the number of participants and the room occupation. The estimation of the face orientation can be used to estimate the relation to the person who is currently talking, as shown in Figure 1.1. Actions like 'sitting down', 'standing up' can refer to the beginning or end of the meeting. 'Raising hand' can indicate the role change from listener to moderator or voting respectively [2, 73].

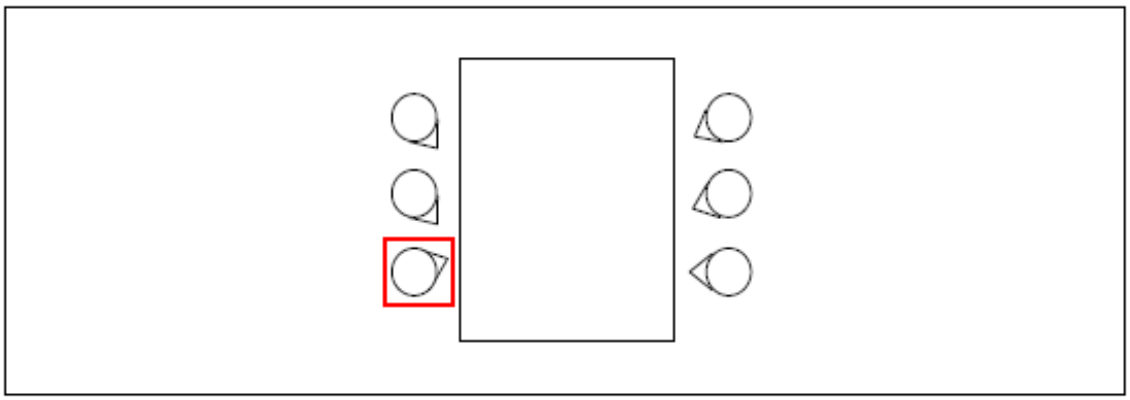


**Figure 1.1: Incorporating current speaker:** *Face pose estimation provides important information about the current focus of attention. People naturally turn their faces to the currently speaking person.*

One important requirement for this thesis was the development of a generic recognition system, i.e we want to integrate as few assumptions on the videos as possible. This is due to the fact that one goal of this thesis is the employment of developed algorithms in other recognition tasks than meetings.

## 1.2 Prerequisites

The goal of this work is to test the implementation over meeting video sequences from the PETS-ICVS 2000 dataset [1]. This dataset has been developed for the European project FGnet (IST-2000-26434) [3]. Four different scenarios with predefined courses of action are recorded and (partially) annotated. The scenarios are recorded from two wall fixed cameras and one omnidirectional table camera. The images obtained from the omnidirectional camera shown in Figure 1.2 require specific transformation before further image analysis can be performed.

Our focus lies on evaluating image sequences from scenario B1 obtained from the wall mounted camera one and camera two. In scenario B1 six meeting participants are observed entering the room, sitting down and raising hands. In addition facial expressions and head gestures like head shaking, nodding and yawing are performed.

We concentrate first on detecting and tracking the meeting participants. The tracking should provide the information which we can use to recognize a subset of single actions like 'sitting down' and 'raising hand'. Face detection and face pose estimation will be used to infer persons, who are currently the focus of attention. In scenario A1 the participants use different facial expressions and raise their hands multiple times. In scenario C1 each participant stands up and goes to the whiteboard. Because no prior training data was available, we applied the tracking approach presented in Chapter 2 on the video material from scenarios A1 and C1 to extract training sequences for several actions.

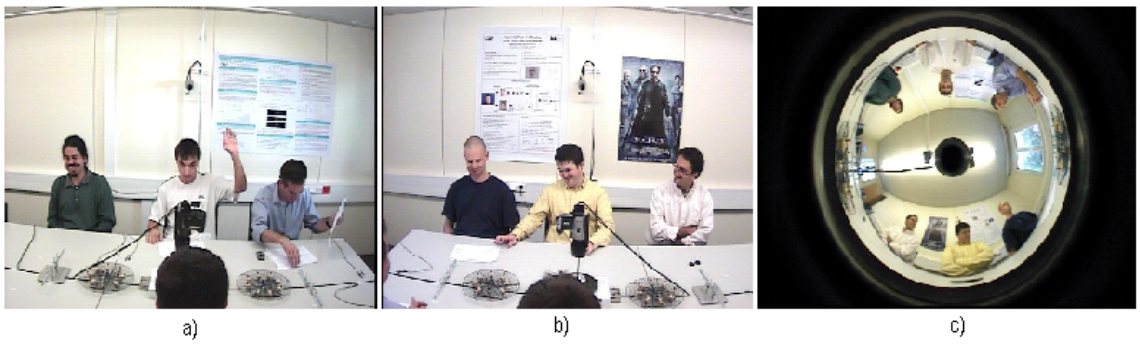



**Figure 1.2: Sample images from PETS-ICVS 2000 data set [1]:** *a). and b). show the images recoded by the wall mounted cameras 1 and 2. In c). we see the image from the omnidirectional camera, which was not used in this work.*

We also used the head pose image database Pointing 2004 [54, 29] set to obtain training samples for the face pose estimation. We chose not to train the system on faces from the video to create more generic recognition system. This set provides images of 15 persons, taken under diffuse ceiling light. It also contains images of persons wearing glasses and with different skin color. Two series of 93 images for each person at different poses are available. Images are labeled with two - horizontal and vertical - angles, which vary from -90° to +90°. In order to make this database applicable to our scenario, we decided to manually crop the faces for each person. As a result we obtained 1395 images with size 122x122 of 15 persons. Figure 1.3 represents an exemplary subset of resulting faces which were included in the face pose training set:

## 1.3 Document Overview

The rest of this work is structured as follows:

Chapter 2 – Person Tracking: contains a short introduction to different categories of tracking approaches and describes background subtraction as a method for motion detection. The rest of the chapter represents the main modules of the implemented tracking algorithm and the experimental results.

Chapter 3 – Face Pose Estimation: gives an overview of available face pose estimation methods and concentrates on the question how an eigenface approach can be applied to obtain facial orientation. The experimental results are discussed at the end of the chapter.

**Figure 1.3: Sample pose set of cropped faces [54, 29] :** *Original images were taken from the Pointing 2004 database and were manually cropped.*

Chapter 4 – Action Recognition: describes how single action recognition is achieved. The processes of appropriate feature extraction and the usage of hidden Markov models for statistical classification are discussed. Introduction to the principles of hidden Markov models is also included.

Chapter 5 – Implementation structure: contains a compact overview of the implementation structure and describes application flow.

Chapter 6 – Conclusion and Future work: provides suggestions for future work.

# 2 Person Tracking

Meetings are group events. Situations like discussions or presentations can be characterized by typical behavior of the participants. In presentations people normally direct their attention to the currently speaking person and typically raise their hands if they have a question. Changing the current speaker during a presentation can involve actions like 'stand up', 'going to whiteboard' and 'sit down'. Recognition of group events in video data requires first to find out what the individual participants are doing. For this important task it is necessary to locate the persons and their faces, and track them throughout the video. This chapter describes briefly which categories of tracking methods exist and how moving objects can be detected using *background subtraction*. In Section 2.2 follows the detailed description of the tracking mechanism implemented for this work.

## 2.1 Introduction to the Concepts of Tracking

### 2.1.1 Tracking Categories

*Object tracking* is an important field of computer vision. It involves detection of moving objects and estimation of their motion trajectory over video sequences. Many criteria can be used to group tracking methods in different categories [65, 11]. For example the tracked objects can have various representations. Special feature points like centroids, or primitive geometric shapes like ellipse and rectangle are typically used. Another more complex representations are any form of templates or 3D models. Tracking methods can also be distinguished in the way that the objects are being detected. For instance color segmentation or background subtraction are widely used. Hu et al. [46] grouped tracking methods in the following four categories:

- **Region-based tracking**
  Here tracking is performed based on the variation of the image regions in motion. For these algorithms, the background subtraction is used to detect motion regions. Each region has a bounding box and regions can merge and split. A person can be represented as combination of one or more regions so that the geometric structure constraints given by the human body are fulfilled.
- **Active-contour-based tracking**
  The tracked objects are represented by their contours. Tracking is achieved by updating these contours dynamically in consecutive frames. Active contour-based algorithms are highly sensitive to the initialization, making it difficult to start tracking automatically.
- **Feature-based tracking**
  Here certain features are extracted from each incoming frame. Then inter-frame object matching can be performed based on these features. The methods in

this category typically use centroids, perimeters, areas, line segments, curve segments, and corner vertices.

- **Model-based tracking**
  For model-based tracking developing a 2D or 3D model of the object is required. Also modeling of rigid and non-rigid objects is quite different and requires specific motion models. For example, rigid object motion can be modeled as affine or projective transformations. Non-rigid object can be covered with a mesh. The position of the nodes in the mesh, represent the final motion.

Each tracking application needs an object detection mechanism. Tracking approaches can be distinguished also by the object detection method, which they employ. Yilmaz et al. [65] grouped the object detection methods as follows:

- **Background Subtraction**
  Background subtraction methods use a model of the observed scene to detect changes in new incoming frames. Any significant difference in the current observation from the background model is captured as a moving foreground object. One very popular and simple method of detecting changes in consecutive images is *frame differencing*. This method applies the video frame at time $t-1$ as background model for the next frame at time $t$. We discuss background subtraction as a part of a motion detection process in the next Section 2.1.2 of this chapter.

- **Segmentation**
  Segmentation methods aim to subdivide an image into its constituent regions or objects. Different intensity properties are used for the partitioning. For example, discontinuity based approaches follow abrupt intensity changes, such as edges. Methods based on similarity partitioning use predefined criteria such as given threshold values. Once the regions or objects of interest are detected segmentation should stop [74].

- **Interest point detectors**
  Interest point detectors are used to find points, which have an expressive texture in their respective localities [65]. Typical points locations are edges or corners. These points should be invariant to lightning changes and variations of the camera viewpoint.

- **Supervised learning**
  Approaches which use appearance based object representation, such as *multiview appearance models* require a separate training phase in which different object views can be learned from a set of stored templates. For object detection a classification step is performed, which assigns a certain class label to an object.

Once the objects of interest are detected, the tracker should find out which objects from the current frame can be matched to any previously seen objects. After solving this correspondence problem the movement trajectories of already detected objects can be updated and new objects can be registered. The track normally stores the position and state of the object in every incoming image. More sophisticated methods which employ a motion model use probabilistic calculations, in order to predict the next position or state of the object. Commonly used are filter methods such as Kalman Filter or methods based on conditional probabilities [69]. Various approaches based

on Kalman filter or particle filter techniques, which allow modeling of linear or non-linear motion and prediction of the object position in the current frame can be found in the literature [19, 66, 14, 27]. All these methods are a subject of kinematic state estimation, which refers to filtering and prediction of features like object position, velocity and acceleration. For further detailed research over multiple target tracking systems based on kinematic state estimation the reader is referred to the literature [55, 18].

To achieve robustness, tracking methods have to deal with many problems, such as noise in the images, complex object shapes or complex motion, object occlusions, scene illumination changes and real-time processing requirements. Often various constraints are used to reduce the complexity of tracking approaches. For example, specifying the number of observed objects [59] or using a priori context information such as expected entry location and certain constant velocity or acceleration.

Detecting the objects of interest and calculating their tracks allows further analysis, in order to recognize certain object behavior. Understanding behavior involves the analysis and recognition of motion patterns, and the production of high-level description of actions and interactions.

### 2.1.2 Background Subtraction for Motion Detection

Extracting moving persons or other objects from the background requires the separation of the objects of interest from the rest. Since a semantic interpretation is impossible or at least very difficult for automatic recognition systems dynamics is often used as a classification criterion. For this purpose the incoming video image is compared with a scene model, so that significant changes can be captured as moving foreground objects. Thus by developing a good background subtraction method the specification of the two main phases *background modeling* and *foreground detection* plays a crucial role. Smooth object contours and temporal stability of the detection should be achieved. The method should be sensitive to capture insignificant changes, but at the same time it has to be stable for gradual and abrupt illumination changes [12]. For this reason separate pre- and post-processing steps can be employed in the overall motion detection mechanism. Various existing background subtraction methods were described and evaluated in recent years [25, 12, 16, 52]. Four processing steps outline a typical motion detection approach. These are shown in Figure 2.1 and are described in the following sections.

#### Pre-Processing

Before detection of moving objects is performed, some preparatory modifications can be applied to the original input image. Intensity normalization is used for compensating lightning changes. Noise reduction can be achieved with temporal or spatial smoothing. To align image fro multiple cameras frame registration can be applied [22].

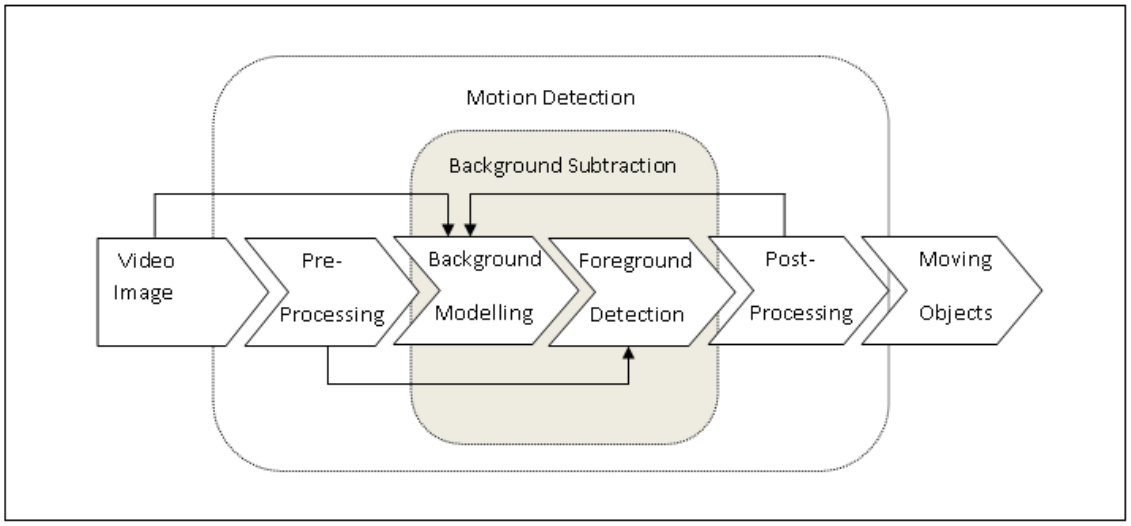



**Figure 2.1: Phases of a typical motion detection approach:** *Each incoming video image can be firstly modified in a pre-processing step. Simple methods often skip this step and directly apply background subtraction to classify each pixel as foreground or background. The resulting binary image can be further improved by post-processing operations. The output of the motion detection approach is a binary image which contains the moving foreground objects.*

### Background Modeling

Background modeling involves the creation and update of a model, which represents the background appearance [42]. The background model should be updated over the time, so that false detections due to abrupt illumination changes can be adopted [23]. Cheung et al. [42] define two categories of background adaptation methods - non-recursive and recursive. Non-recursive approaches use a buffer of the previous $K$ video images, and use the temporal variation of each pixel within the buffer to obtain the background image. Recursive methods do not require a buffer to maintain the background. Here the initialized model is updated recursively using each input image. Many approaches incorporate selectivity control to update only background pixels. We discuss two popular recursive methods implemented for this work in Section 2.2.1.

### Foreground Detection

Foreground detection refers to the process where each pixel from the current image is compared to the corresponding pixel in the background model. As a result each pixel is classified as part of the background or the foreground. Elhabian et al. [12] describe three categories of comparison techniques for foreground detection - statistical, difference and cluster based approaches. Difference based methods are popular and simple. The foreground image $I_f$ by difference based methods which is obtained after removal of the background image $I_b$ from the current observation frame $I_o$ is given by:

$$I_f = \begin{cases} 0 & |I_b - I_o| < threshold \\ 1 & |I_b - I_o| \geq threshold \end{cases} \tag{2.1}$$

The appropriate threshold value can be empirically selected or computed as an adaptive value and has a great impact on the final detection.

#### Post-Processing

Higher quality in the foreground image can be achieved through post-processing. Different techniques are used to diminish noise, eliminate objects of inappropriate size or reduce shadow effects. The simplest methods mask the foreground with standard binary image processing operations. Typical post-processing include operations like:

- **Morphological image processing**
  Isolated zero and non-zero pixels and small blobs or holes which result from false positive detections can be removed with binary morphological operations. Here the object structure is modified with help of so called structuring element. We discuss morphological image processing in Section 2.2.1.2.
- **Elimination of too small or too large binary objects**
  Undesirable detections can be also eliminated depending on their size. Through selection of appropriate size threshold, regions that are too small or too big can be filtered from the foreground image. Here connected components approach is applied to obtain all heuristics for the objects seen in the image. As part of the post-processing phase in our approach the elimination of small or large binary objects is described in Section 2.2.1.2.
- **Noise reducing processing - denoising**
  Noise is understood as undesirable random variation of pixel brightness or color information. Noise occurs by variety of environmental factors such as lightning and temperature conditions which impact the quality of image acquisition. Spatial and frequency filtering offer many approaches applicable respectively to reduce random or periodic noise effects.
- **Reducing of shadow effects**
  Shadows thrown from moving objects are captured by background removal and are often undesirable detections. Such effects are difficult to handle, because it is not immediately obvious if the foreground pixels are part of the moving object or part of the shadow. To detect shadows some approaches use *shadow point mask*, which is defined following a set of rules and the pixel hue and saturation information from the background model and the current image [56].

## 2.2 Tracking Algorithm - Main Modules

In this work, we present a person tracking algorithm which is designed to detect and track multiple persons in image sequences. The goal was to test the tracking mechanism on the PETS-ICVS 2000 dataset described in Section 1.2. Image data is obtained from indoor wall-fixed static cameras under diffuse ceiling lighting conditions. In this section we give a brief overview of the implemented tracking approach, followed by detailed description of the single modules.

The first phase of the tracking process, shown in Figure 2.2 is *motion detection*.

Background subtraction is applied to separate persons from the static background objects, such as chairs and tables. As a result the moving persons are captured as non-zero *binary large objects* (blobs) in the foreground image. In order to reduce noise

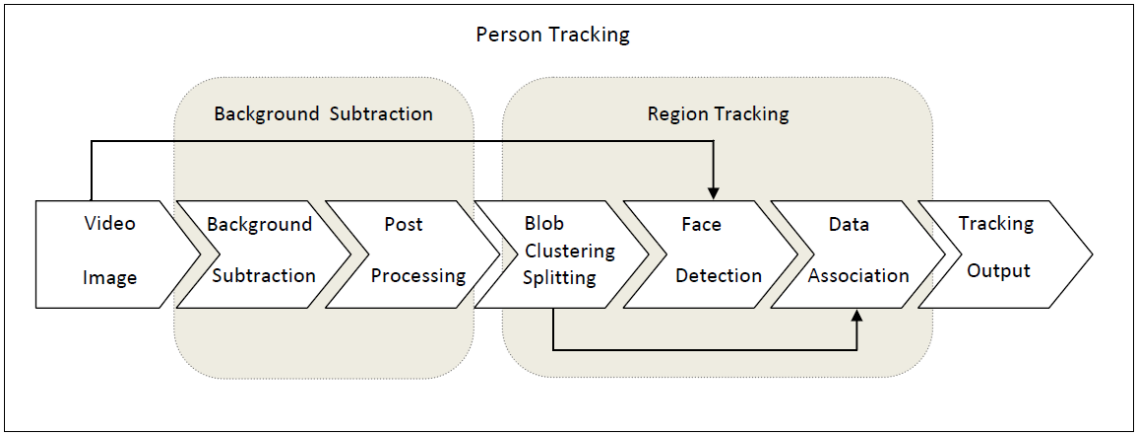



**Figure 2.2: Tracking Algorithm Overview:** *Motion detection is applied on each incoming video frame. The resulting blobs are clustered and split so that individual persons finally are represented as group of smaller blob or one single blob. The person regions specify the area in the origin input image where face detection is applied. Faces and blobs are associated according to their positions. In the next step for all existing persons the blob and face positions are updated with the current coordinates. New persons are instantiated if no match exists between the current person regions and the already registered blobs.*

and to obtain clear blob contours the binary image is further post-processed with combination of morphological operations and additional size-filters. The final blobs, are then analyzed, in order to group them in regions of interest that represents the detected persons. Hierarchical clustering is applied to group blobs that represent body parts, like heads and hands. A splitting step is also needed to separate the groups of blobs for each individual person. The bounding box of these clusters specify the person regions. These person regions specify the area in the current video frame where a face detection method is applied. In this way the number of false positive face detections can be reduced. The detected faces are then mapped to the person regions according to their position. The resulting cluster-face pairs are associated with the previously detected persons. For pairs which could not be matched a new person object is instantiated. Each person object stores the associated bounding box, face and centroid coordinates obtained from the evaluation of the current frame. Extracted data is finally used in the action recognition step. In the following sections the functionality of the main tracking modules is discussed.

### 2.2.1 Motion Detection

The image sequences, which were tested for our tracking method, are obtained from indoor static cameras under almost constant lighting conditions. Thats why, in this first development stage, no separate pre-processing step was applied in the motion detection phase of the tracking algorithm. For further refinement and achievement of better tracking results in real life applications the benefits of the pre-processing step should be used. This makes background subtraction the first relevant step in the motion detection of this tracking algorithm. Two common recursive background subtraction methods *approximate median* and *selective running average* were implemented and tested. These are

described in the sections below. The output foreground image is then post-processed by a combination of morphological opening and closing with a square structure element. Blobs which are too small to represent any person body part are eliminated. Figure 2.3 gives a compact overview of the performed steps in the motion detection process. The individual steps are described in detail in the following sections.

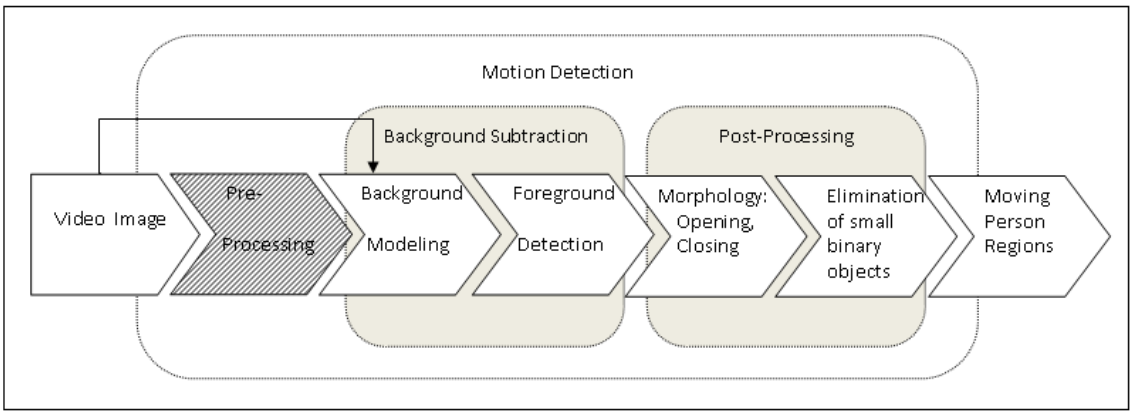



**Figure 2.3: Motion detection for person tracking in meeting rooms:** *Starting with loading images from the video sequence background subtraction method is applied without further pre-processing. Morphological opening, followed by closing is used to improve the binary image. Small blobs are then also eliminated. The final binary image now contains only the person regions.*

#### 2.2.1.1 Background Subtraction

**Approximate Median**

McFarlane and Schofield [64] presented a recursive approximation of the median filter for background subtraction as part of a piglet tracking application. This approach subtracts each incoming frame from a time averaged reference image, followed by global thresholding. The reference image represents the background model. The background model is updated pixelwise with each incoming image frame. If the pixel values in the current video image are greater than those in the reference image then the values in the reference image are incremented by one, in other case decreased by one, see Algorithm 2.1. It is expected that the number of the pixels is almost equally distributed between these which are incremented and these which decremented. This represents the idea of median approximation. *Approximate median* approach is simple and requires the storage of only one reference image which makes it computationally inexpensive. Disadvantage given by Cheung and Kamath [42], is that adoption of large changes in the background needs many frames. For example learning of regions revealed by a car that moves away after being parked for a long time. Another disadvantage observed for detecting sitting persons in a meeting room is the fast adoption of the body parts or whole persons into the background.

**Selective Running Average**

The *running average* approach is based on the so called *exponentially weighted moving average* filter (EWMA) [36]. This filter can be applied for noise reduction in dynamic systems. Explicit initialization is not required. In this case it will take a

**Algorithm 2.1** Approximate Median

**procedure** ApproximateMedian(*thresh*,*imageseq*)
  **for all** $imagesI_t \in imageseq$ **do**
    **for all** $pixels(x,y)$ **do**
      $distance \leftarrow dist(Background_t(x,y), I_t(x,y))$
      $$Foreground_t(x,y) \leftarrow \begin{cases} 1 & distance >= thresh \\ 0 & \text{otherwise} \end{cases}$$
      $$Background_{t+1}(x,y) \leftarrow \begin{cases} Background_t(x,y)+1 & I_t(x,y) > Background_t(x,y) \\ Background_t(x,y)-1 & \text{otherwise} \end{cases}$$
    **end for**
  **end for**
**end procedure**

period of time to learn the background objects, depending on the learning rate value. In comparison to approaches that need history buffer this method uses less storage resources, because it keeps only the current background and the difference image for further calculations. The difference image is obtained after calculating the color difference in each pixel position between the background model and the current image followed by global thresholding, where each pixel is compared with a threshold value, specified for the whole image. The result is the final classification of each pixel as fore- or background. The background model is updated with the following equation:

$$Background_{t+1} = \alpha * Image_t + (1+\alpha) * Background_t$$

The learning rate $\alpha$ specifies the importance weight of the current image data. This rate represents the percentage of adoption of the current image changes in the background. If the value of $\alpha$ is closer to 1, the filter adopts changes faster than with small $\alpha$. The value of $\alpha$ has a great impact on the pixel classification [36]. Updating the background model with each incoming video image can result in misclassifications if the color values of the foreground overwrite the background values. To overcome this problem *selectivity* control can be applied, where the color of a background pixel is updated only if this pixel in the difference image is classified as background, in other case the previous value is kept and the foreground objects cannot be adopted, see Algorithm 2.2. This can be also a disadvantage, because it takes longer until false detections are integrated in the background [23].

#### 2.2.1.2 Post-Processing

In order to reduce the noise in the resulting image obtained after background subtraction we employ two post-processing steps:

##### Morphological image processing

Isolated pixels and small binary objects can result from false positive detections,see Figures 2.4 and 2.5. False negative detections result in zero pixels, which form holes in correct foreground regions. Such effects can be removed with binary morphological

**Algorithm 2.2** Selective Running Average

**procedure** SELECTIVERUNNINGAVERAGE(*thresh*,$\alpha$,*imageseq*)
  **for all** $imagesI_t \in imageseq$ **do**
    **for all** $pixels(x,y)$ **do**
      $distance \leftarrow dist(Background_t(x,y), I_t(x,y))$
      $$Foreground_t(x,y) \leftarrow \begin{cases} 1 & distance >= thresh \\ 0 & \text{otherwise} \end{cases}$$
      $$Background_{t+1}(x,y) \leftarrow \begin{cases} \alpha.I_t(x,y) + (1-\alpha).Background_t(x,y) & I_t(x,y) = 0 \\ Background_t(x,y) & \text{otherwise} \end{cases}$$
    **end for**
  **end for**
**end procedure**

operations. This type of processing concerns structure modification of objects in the image. Basic operators are *erosion* and *deletion*. Erosion shrinks elements in the binary image. Image objects that are smaller than the structuring element are filtered from the image. By dilation, an object grows in the spatial extent [57]. The specific manner of growing is controlled by the shape of the structuring element. All of the morphological algorithms are based on these two primitive operators. We apply a combination of two fundamental operators - opening, followed by closing, which combine dilation and erosion with the same structuring element. Opening achieves smooth contours, eliminates thin protrusions, and breaks isthmuses. Closing fuses narrow breaks, fills small holes and gaps in the contour [74].

#### Elimination of too small or too large binary objects

Abrupt illumination changes result can be detected as regions of different size, which do not correspond to any moving object. Often the moving objects of interest are expected to be larger than a certain size. The moving objects are represented through non-zero pixels which are grouped into components based on their connectivity. Extracting and labeling of such disjoint connected components is central for the image analysis. All pixels in a *connected component* obtain the same label. Typically they share similar pixel intensity values and are connected with each other. Connected components approach scans an image and labels pixels in the same group with a gray level or a color. The connected component approach can deliver all image heuristics of each found component, such as area, pixel identification list, perimeter etc. Through selection of appropriate size threshold, regions that are too small or too big can be filtered from the foreground image. This process can help to avoid additional computational effort of analyzing wrong classified regions [74].

In Figures 2.4 and 2.5 we see the post-processed foreground images obtained after application of *approximate median* and *selective running average* approach. The approximate median approach do not provide an external parameter to control the temporal integrity. Persons which sit still disappear gradually in the background. With *selective running average* this problem was avoided.

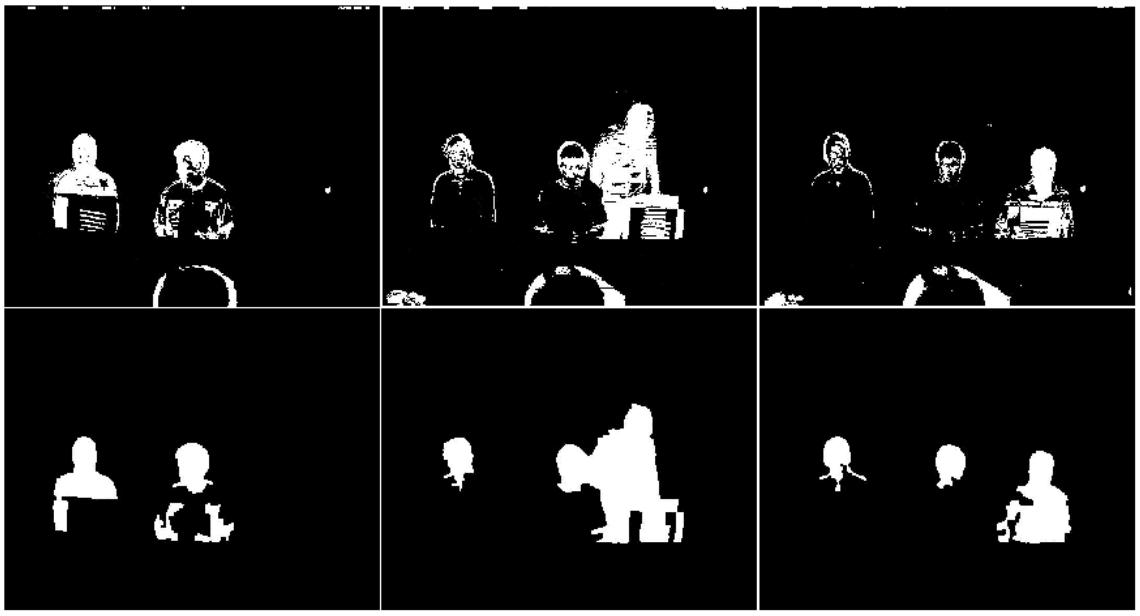


**Figure 2.4: Foreground images obtained with approximate median approach:** *The foreground images after applying approximate median approach show the temporal integrity of the static objects. Persons which sit and make only small body motion are gradually integrated in the background.*

**Figure 2.5: Foreground images obtained with selective running average:** *The foreground images obtained from selective running average approach is modified with morphological opening and closing. The images are denoised and the resulting blobs have sooth contours.*

### 2.2.2 Person Region Tracking

All moving regions detected by the *motion detection* have to be associated with one new or already existing person object. These regions differ in size and location. Sometimes body parts appear as disconnected blobs. Adequate representation of the person requires an association mechanism, which merges the body parts with the person region. At the same

time when persons are near to each other, they can be detected as one connected blob, which requires separate splitting mechanism, see Figure 2.7. To handle these situations hierarchical clustering, followed by a splitting step is applied. The clustering approach is based on a blob distance metric, which represents the relation between size and distance of the individual blobs. The splitting step is based on a simple comparison between the current person regions and and these in the previous frame. The resulting person regions, which contain one or more blobs, contain the information used to update the person tracks. Each region from the current frame is compared in size and position with the person regions from the previous frame. If an appropriate match was found the person region position and its centroid are used to update the track. In the other case a new person object is instantiated.

#### 2.2.2.1 Blob Clustering

Clustering techniques are used to divide data instances into natural groups, so that they share common characteristics [33]. Cluster analysis is the study of techniques for automatically finding potential classes. This section gives a short description of hierarchical clustering and its application as a part of the people tracking algorithm. Hierarchical clustering techniques fall into two general classes:

- **Agglomerative Hierarchical Clustering**
  This is a 'bottom up' approach, where pairs of clusters are merged, as they move up in the hierarchy. Here the data instances are directly divided into predefined number of clusters.
- **Divisive Hierarchical Clustering**
  Here a 'top down' mechanism groups data with a sequence of nested partitions. The process starts in one big cluster, which contains all data.

Hierarchical clustering organizes the data based on distance calculations, given by a proximity matrix. The results are represented by a binary tree or dendrogram. Each leaf is regarded as a data instance. The root, as top of the hierarchy, represents the complete data set. Clustering can be achieved by cutting the dendrogram on a certain level [75].

To cluster the blobs from the final foreground binary image, first the distance relation between each pair of blobs is calculated. This distance is stored in a proximity matrix. The measurement is based on the relation of the distance between the two nearest points of the blob bounding boxes and the difference in their sizes, see Figure 2.6. For disconnected blobs, which have overlapping bounding boxes the proximity values is set to zero. The distance metric is given by:

$$Proximity_{b_1,b_2} = \begin{cases} \frac{\Delta x+\Delta y}{\Delta w+\Delta h} & b_1 \cap b_2 = 0 \\ 0 & b_1 \cap b_2 > 0 \end{cases} \tag{2.2}$$

After the proximity matrix is calculated, the blobs are paired in a binary tree, based on the smallest distance. At last the tree is cut by a specified threshold value. This means that two blobs are clustered together if the ratio of their distance to their size difference is less than a given threshold value. In this way smaller disconnected blobs, which represent body parts can be merged to bigger ones, and in the same time large blobs, which represent persons and are near to each other, remain separated. For more

detailed information about different categories, algorithms and application of clustering approaches the reader is referred to the literature [75, 45, 35].

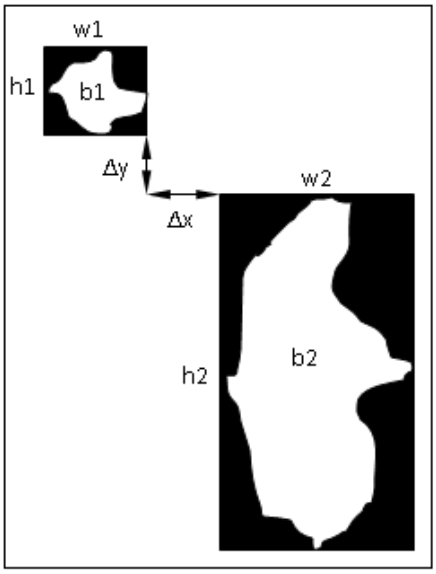



**Figure 2.6: Blob Proximity:** *The smallest distance between two regions is related to the difference of their sizes. If two blobs are similar in size the expected metric values should be greater than for two blob which have a large difference in their size. In this way two person blob which have similar size remain separated. At the same time body parts can be clustered.*

#### 2.2.2.2 Blob Splitting

In a meeting room people sit often near to each other, so that small movements, can cause intersection of their regions. This results in zero value in the proximity matrix. Thus the regions clustered together. Another situation which require additional treatment occurs by overlapping of the persons in the image plane. In this case the motion detection returns one connected blob which corresponds to two persons. Normally, the time interval between two successive frames is small, therefore it is expected that changes are limited to a certain extent. Size constraints are used here for simplification. For example it is assumed that the size of a person cluster can not increase more than sixty percent from frame to frame. In this case, two people which occlude each other, can be simply separated by keeping the their previous regions.

### 2.2.3 Face Detection

Developing a face detection method was not an objective for this thesis. To find the faces of the meeting participants an open source implementation [4] of the Viola-Jones [40] face detection method was used. The method was applied in each incoming frame, only upon the person regions. In this way the number of false positive detections were highly reduced. Multiple detections per person region were resolved, depending on the face position relative to the region bounding box. Detected faces in the lower part of the bounding box and faces which intersect the boundaries of the bounding box were

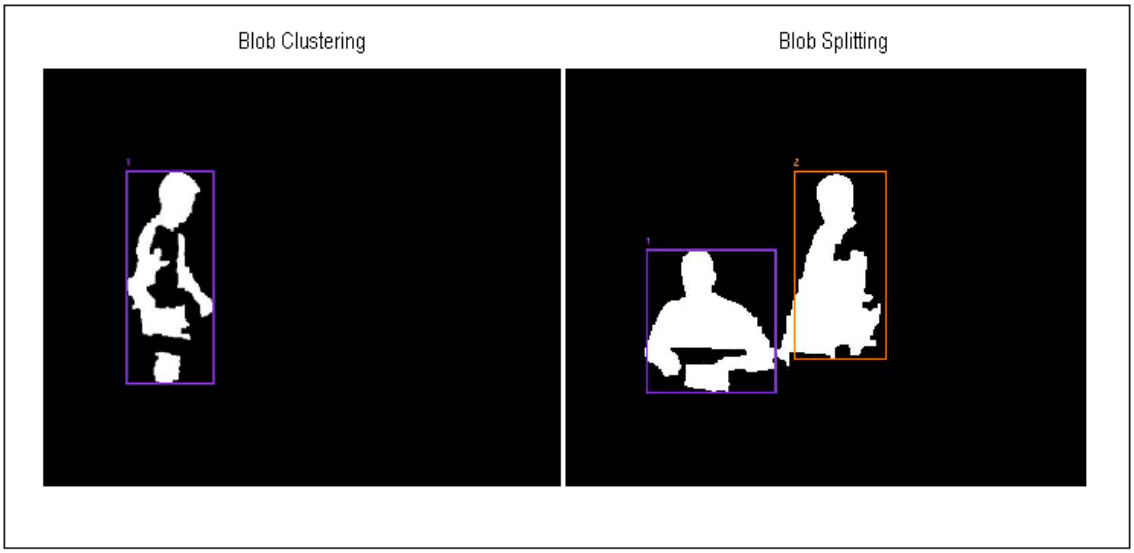



**Figure 2.7: Blob clustering and splitting:** *Hierarchical clustering is applied to group disconnected blobs in one person region. Splitting step is achieved by keeping the previous region positions.*

discarded. If two faces are detected in the upper part of the person region, the face with the upper position wins. If all faces were rejected or no faces were detected in current region, the previous face position is kept. In order to achieve completeness a short description of the Viola-Jones face detection approach is provided below. Figure 2.8 represents the results after applying the face detection method.

**Figure 2.8: Face detection over the PETS-ICVS data set**

The adopted method classifies images as faces or non-faces based on the value of so called Haar-like features. Papageorgiou et al. [53] describes this features as a natural set of basis functions which encode differences in average intensities between different regions. The set of features for object detection was extended from Rainer Lienhart and Jochen Maydt [47]. Figure 2.9 represents a typical set of such features.

In order to find Haar-like features, the input image is represented as so called *integral image*. *Integral images* store in each pixel position the sum value of the texture intensities of all pixels contained in the region given by the pixel of interest and the upper left corner

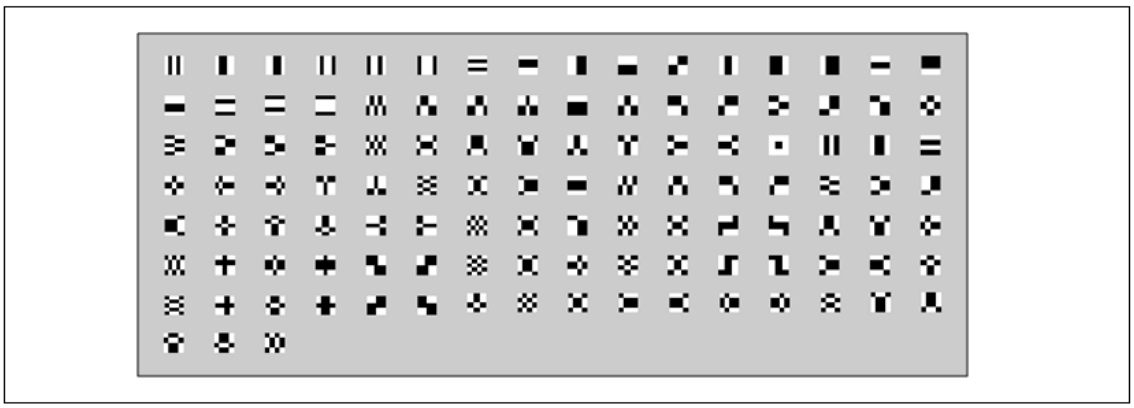


**Figure 2.9: Haar-like features** [5]

of the input image, see Figure 2.10. The sum of all intensities in any rectangular area of

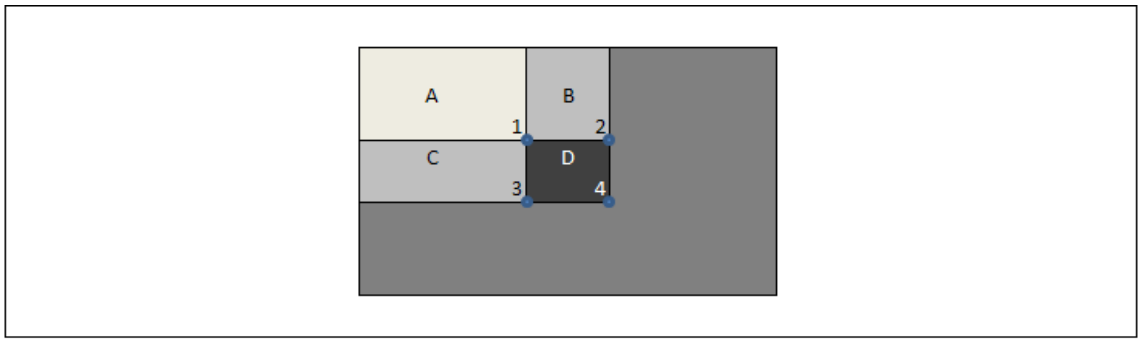



**Figure 2.10: Integral Image:** *In position 4 is stored the sum of all intensities in A, B, C and D. The sum of intensities in D can be computed by 4-2-3+1.*

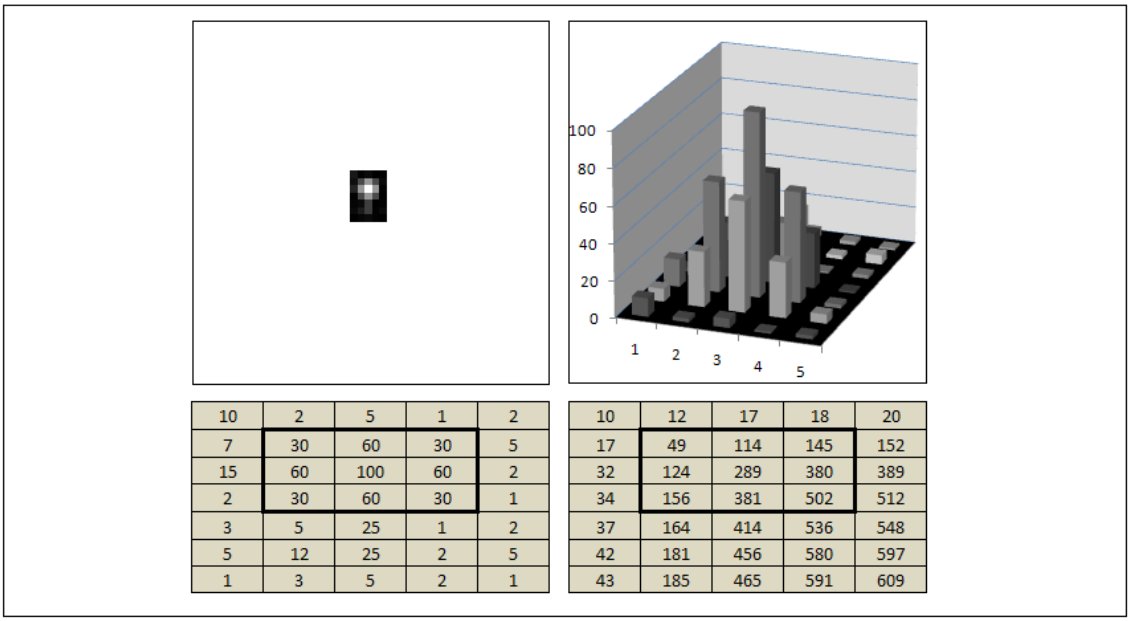
the image can be recovered each time from the integral image [28]. Integral images $I'$ for each position *(x,y)* of the original image $I$ are calculated row by row using the previous calculations:

$$I'(x,y) = I(x,y) + I'(x-1,y) + I'(x,y-1) - I'(x-1,y-1)$$

In order to avoid double counting, which occurs adding the second and the third term we need to subtract the last term [41].

Foe example if we have a small image of 7x5 pixel, as shown in Figure 2.2.3, we can represent it as a bar chart of intensity value. The same information is stored in the tables below on the left side numerical and on the right the integral values. Assume we want to calculate the integral image of the bounded Area in the left table in Figure 2.2.3. This area is given by region D in Figure 2.10. The sum of the pixels within rectangle D can be computed with four references, given the integral image table on the right in Figure 2.2.3. The value of the integral image at location 1 in Figure 2.10 is the sum of the pixels in rectangle A. The value at location 2 is A + B, at location 3 is A + C, and at location 4 is A + B + C + D. Thus the sum within D can be computed as 4 -2- 3 + 1. In our example the sum of the bounded area is given by 502 – 34 – 18 + 10 = 460.

The number of all possible rectangle subregions for each input image is large. For example for 384x288 image, with detector of size 24x24, there are more than 180.000




| | | | | |
|---|---|---|---|---|
| 10 | 2 | 5 | 1 | 2 |
| 7 | 30 | 60 | 30 | 5 |
| 15 | 60 | 100 | 60 | 2 |
| 2 | 30 | 60 | 30 | 1 |
| 3 | 5 | 25 | 1 | 2 |
| 5 | 12 | 25 | 2 | 5 |
| 1 | 3 | 5 | 2 | 1 |

| | | | | |
|---|---|---|---|---|
| 10 | 12 | 17 | 18 | 20 |
| 17 | 49 | 114 | 145 | 152 |
| 32 | 124 | 289 | 380 | 389 |
| 34 | 156 | 381 | 502 | 512 |
| 37 | 164 | 414 | 536 | 548 |
| 42 | 181 | 456 | 580 | 597 |
| 43 | 185 | 465 | 591 | 609 |

**Figure 2.11: Summation Area Tables of Integral Images:** *A small gray scale image and its bar chart of intensity values can be seen in the upper part of the figure. The intensities values are stored in the left table. The right table represents the values in the integral image.*

possibilities [40]. Viola-Jones method constructs a supervised Haar-classifier using *adaptive boosting*-(AdaBoost) in order to achieve appropriate feature selection.

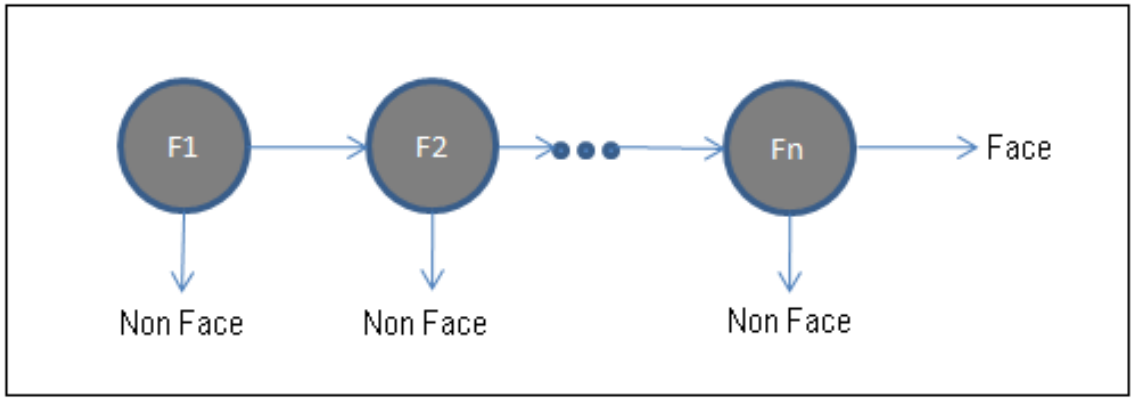



**Figure 2.12: Rejection cascade:** *An input image is classified as a face if the computation passes through the entire cascade.*

AdaBoost is a machine learning mechanism that combines multiple classifiers. The main idea of boosting approaches is to generate complementary classifies by training each following classifier on the mistakes of the previous ones. The resulting set of weak-classifiers can be combined to one strong classifier. For example we can divide any training set randomly into three. We use the first part of the data P1 to train one classifier C1. The second data set P2 is the used to test C1. All data instances which could not be classified, together with the P2 are then used to train a second classifier C2. The last data set P3 is used to test C1 and C2, the data which deliver classification error form the training set of the last classifier C3. Combination of such three classifiers forms a

boosting system, which can be recursively used to form a higher system. AdaBoost can combine multiple simple classifies, not only three. In addition the same training set, can be used many times.

The AdaBoost mechanism in the Viola-Jones approach uses many weak classifier, trained on single Haar-feature. These are the used to train a strong classifier, which learns frontal face view with a small set of certain features. The typical features for frontal face are based on the idea that the region of the eyes is darker than the upper part of the cheeks or nose bridge as shown in Figure 2.13. Input for the classifier are histogram- and size-equalized image patches, which are then labeled as face or non face. The used

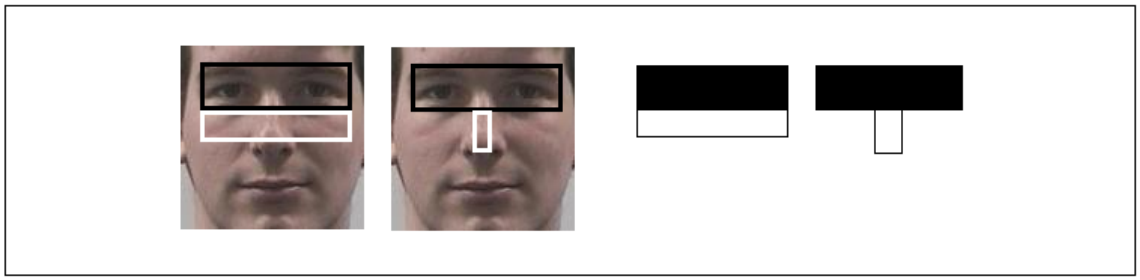


**Figure 2.13: Typical Haar-like features for frontal face:** *The typical features for frontal face expect that the eyes region is darker than the nose bridge or the upper part of the cheeks.*

AdaBoost mechanism is organized as a rejection cascade of nodes, see Figure 2.12. A non-face result at any stage of the cascade terminates the computation, and the algorithm then declares that no face exists at that image location. Thus, in order to classify an image as face the computation should pass through the entire cascade. For further information about *supervised learning* and *boosting theory* the reader is referred to the literature [13].

## 2.3 Experiments and Results

The presented tracking method, uses a background subtraction method to capture motion. For this work the background model is obtained with the *selective running average* approach. The choice of the method and the corresponding parameters was made after direct behavior comparison on the PETS-ICSV Dataset - Scenario B1 between simple *frame differencing*, *approximate median* and *selective running average* method.

With *frame differencing* visible moving persons in the scene were easily detected, but it showed weaknesses by relative small illumination changes caused by shadows thrown on the wall from persons who enter the room and walk near the wall or in front of the camera. These problems could be avoided with the *approximate median* approach. This method showed stable results against small lightning changes. However, the temporal integration of static objects in the background was relatively fast, so that sitting people, who exhibit only small scale body motion such as head or hand movements, disappear partly or completely from the captured foreground. Better results were achieved with *selective running average* approach. Here the sitting people were kept in the foreground and the small light changes were integrated in the background.

One main limitation of all these methods is that they are sensitive to the choice of the threshold and they do not provide a method for adaptive selection of this parameter. In

this case one remaining problem is the strong illumination change, caused by sunshine or light switching, which can abruptly occur during the meeting. Despite this restrictions the simplicity and the efficiency of these methods makes them attractive for various research tasks. The application of *selective running average* to images from scenario B1 of the PETS-ICVS data set delivered smooth person contours, which was enough on this stage of development to proceed with the analysis of the resulting foreground image.

When the final foreground image is calculated the resulting blobs are clustered or split to obtain person regions. Still remaining problems here are occlusions caused by background objects, such as chairs, tables or white boards. In this case greater parts of the body can be occluded by background objects. In the worst case the person can disappear completely. Sudden revealing of the occluded parts could lead to blob enlargement, which is discarded from the splitting mechanism, so that the person can no more be adequately tracked. In case of full occlusion, followed by revealing in another location in the image plane new person object will be instantiated. To achieve better tracking results, blob clustering and blob splitting should be further improved. Also information from multiple cameras can be used to handle occlusions.

Upon all person regions, measured in the current frame, Viola-Jones face detection is applied. Detected faces, are associated with the corresponding person region, depending of the percentage of intersection. Region matching is also performed, to associate the current measurement with the data from the previous frame. Face bounding box and center coordinates are stored in each person track measurement. In general the face detector works well on faces near frontal view. Explicit face tracking approaches require face detection as an initialization or reinitialization step in case of lost tracker. To achieve improvement in the face detection separate classifier should be trained to detect profile faces [41]. Because of the simplicity of the scenarios provided in the PETS-ICVS 2000 data set we only integrated the face detection in the overall tracking mechanism.

The presented tracking algorithm do not apply any probabilistic calculations. Though the tracked features can be also used to generate predictions for the person movements. In order to improve this tracking mechanism by using any probabilistic method the development of a motion model which adequate represent person movements is recommended.

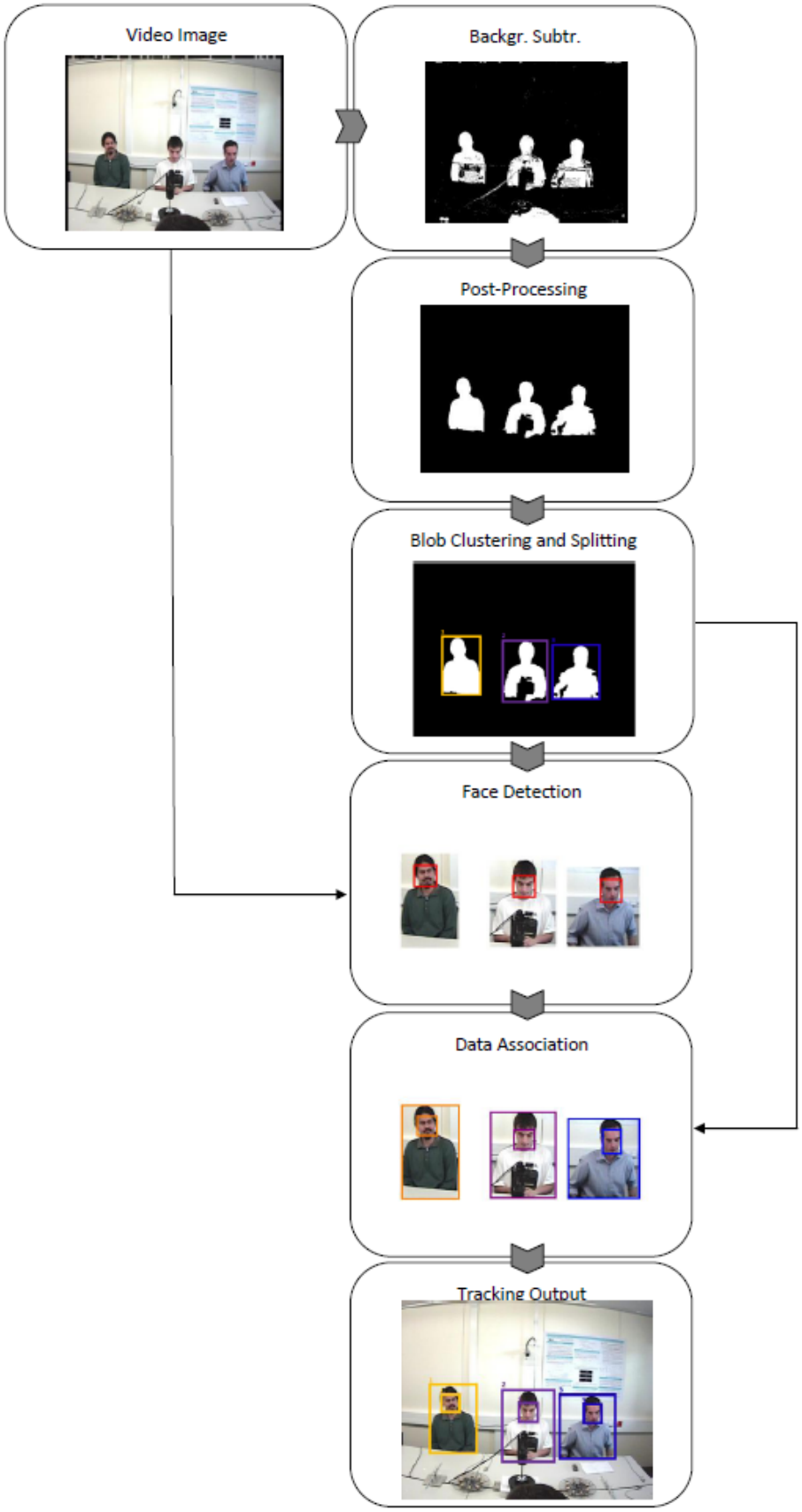



**Figure 2.14: Person tracking revisited:** *The input video image is processed with a background subtraction method. Noise and small blob are eliminated with a pose-processing step. The final blobs are grouped and split to represent the person regions. Face detection method is applied over these regions. The detected faces are paired with the regions and the associated to the corresponding person object. On demand new persons are instantiated.*

# 3 Face Pose Estimation

Face pose estimation is a major criterion when analyzing visual meeting data. Head orientation accompanies conversations as a non-verbal communication feedback [30]. During interpersonal interaction people turn their faces to observe the current speaker or to target the object of the discussion. Head gestures such as nodding and head shaking can express agreement, understanding, confusion or dissent [2].

The natural head movement range for an average adult male encompasses a frontal lateral bending from -40.9° to 36.3°, a horizontal axial rotation from -79.8° to 75.3°, and a sagittal flexion and extension from -60.4° to 69.6° [32]. Figure 3.1 shows the natural head movements. A lot of research is done in the area of *face pose estimation*. This chapter gives

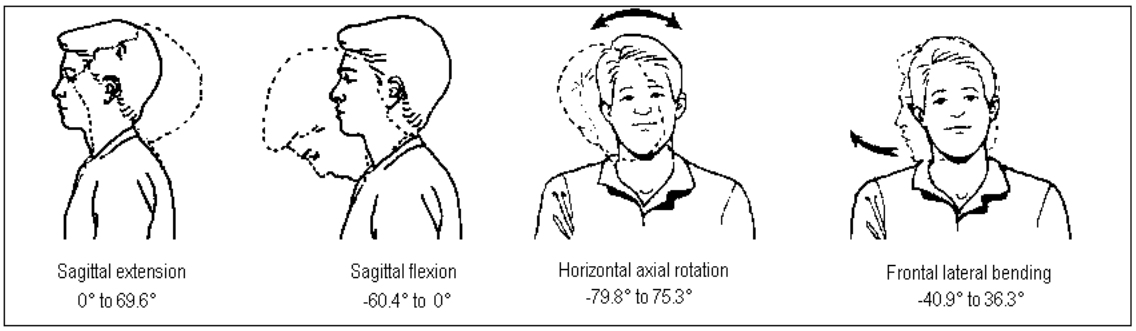



**Figure 3.1: Head motion range:** *The natural head movement includes 'frontal lateral bending' and 'horizontal axial rotation' from left to right. The head movement from frontal position downwards to the breast is called 'sagittal flexion'. The motion from frontal position upwards is called 'sagittal extension'. For simplicity in this work it is not distinguished between 'frontal lateral bending' and 'frontal side view' of the head.* [6, 32]

a short overview of available methods and describes in detail an eigenface approach [51], based on *principal component analysis* (PCA), adopted for this work. The experimental results are presented in Section 3.3.

## 3.1 Face Pose Estimation Methods - Overview

Murphy-Chutorian et al. [71] classified head pose estimation methods into eight categories depending on the fundamental approach underlining their implementation. These categories are:

- Appearance template methods
- Detector array methods
- Nonlinear regression methods

- Manifold embedding methods seek
- Flexible models
- Geometric methods
- Tracking methods
- Hybrid methods

### 3.1.1 Appearance template methods

Face appearance templates are prototype exemplars annotated according their discrete pose. Templates can be obtained from different representations. Beymer [17] used face model database, which contain manually annotated locations of the two eyes, nose lobes and corners of the mouth. A new input image is then geometrically aligned with each model, based on these anchor points. The model set is then scanned comparing the input to each model face. Weighted Euclidean distance is used by this measurement. The smallest distance outlines the model which is reported as result.

Another appearance models are based on the idea that a new face can be represented as a linear combination from a set of training samples. In order to obtain a proper decomposition, methods such as Principle Component Analysis (PCA) or Linear Discriminant Analysis (LDA) are used. Sherrah et al. [49] investigated this issue on the use of Gabor filters and PCA. The authors tried to emphasize differences in the pose using orientation-selective Gabor filters and in the same time to suppress identity information using PCA, while describing changes in the face orientation.

In general the simplicity of these methods makes them attractive. Training sets can be built only from cropped faces, which allows easy extension. But more templates result in more computational expensive comparisons, which may concern the efficiency. In addition these methods require a separate face detection step. False positive detections can impact the accuracy of the face pose estimation. Another limitation is that without additional mechanisms these approaches can treat only discrete set of poses.

### 3.1.2 Detector array methods

Head pose can be estimated by training different face detectors, each to a specific discrete pose. This allows fusing of face detection and pose estimation into one process. Different approaches such as Support Vector Machines, Neural Networks or Adaptive Boosting can be used to detect faces over variety of poses. Successful face detection specifies the pose, passing a set of binary classifiers, where no two classifiers are in disagreement. Detector arrays work direct with the image. The image is evaluated by a detector trained on many images with a supervised learning algorithm. For this reason training data should contain also many non-face examples.

Rowley et al. [44, 43] presented non-upright face detection using neural networks. Two neural network classifiers ware trained to estimates the pose of a face and then to detect faces in a detection window. Detection requires three steps: for each image window the face pose is first estimated. The pose estimate is then used to derotate the image window; the window is then classified by the second detector. The basic approach first applies

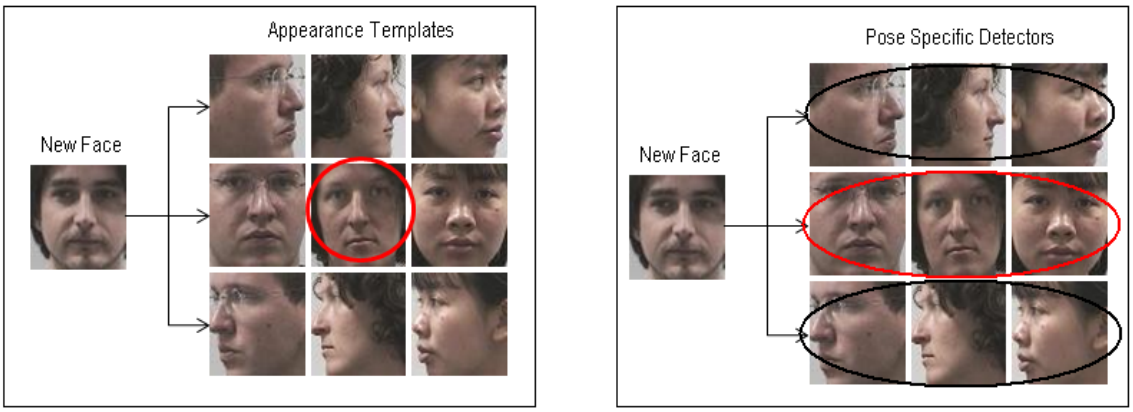



**Figure 3.2: Appearance Template and Detector Array Methods:** *Appearance template methods scan a set of labeled training samples to find the most similar one to the new input face. Detector array methods contain series of head detectors, which are trained to a specific pose. The detector with the greatest support specifies the label for anew input face.*

a set of neural network-based filters to an image and then combines the outputs. The filters examine each location in the image at several scales, looking for locations that might contain a face. The arbitrator then merges detections from individual filters and eliminates overlapping detections.

Another popular approach, presented from Viola and Jones [72] based on detection over 12 different pose classes, uses integral images for facial feature extraction. Appropriate feature selection for each class is achieved with AdaBoost. Frontal face detection was described in the previous chapter.

Detector arrays with binary outputs which successfully find faces require that all classifiers for a certain pose have the same evaluation. Since the majority of the detectors are based on binary classifiers, the pose estimation is restricted over discrete set of poses. Finally, increasing the number of detectors requires more computational resources, which restricts the implementation of real-time solutions with a large set of detectors.

### 3.1.3 Nonlinear regression methods

Non-linear regression techniques can be used to directly map annotated face images to a discrete or continuous pose measurement. Different techniques exist already. Murphy-Chutorian et al. [70] presented method for estimation of a driver head orientation. First facial regions are found by three cascaded-AdaBoost face detectors applied to the grayscale video images. The detected face is then normalized to a fixed size and used to compute a Localized Gradient Orientation (LGO) histogram. Two Support Vector Regressors (SVRs) trained for head rotations in pitch and yaw obtain the histogram as input. Neural networks find wide application as non-linear regression techniques. Multi Layer Perceptron can be trained either on discrete or on continuous pose measurement. Neural Networks that contain many locally linear maps (LLM) offer another solution. Robert Rae et al. [61] worked on an approach where color segmentation is done to detect skin colored areas in the image. The detected regions of interest are passed to LLM-network which is trained to fit an ellipse around the segmented face. This information is the used in the final estimation step where another neural network identifies the face

orientation. Also these methods can use training data containing cropped images. Face

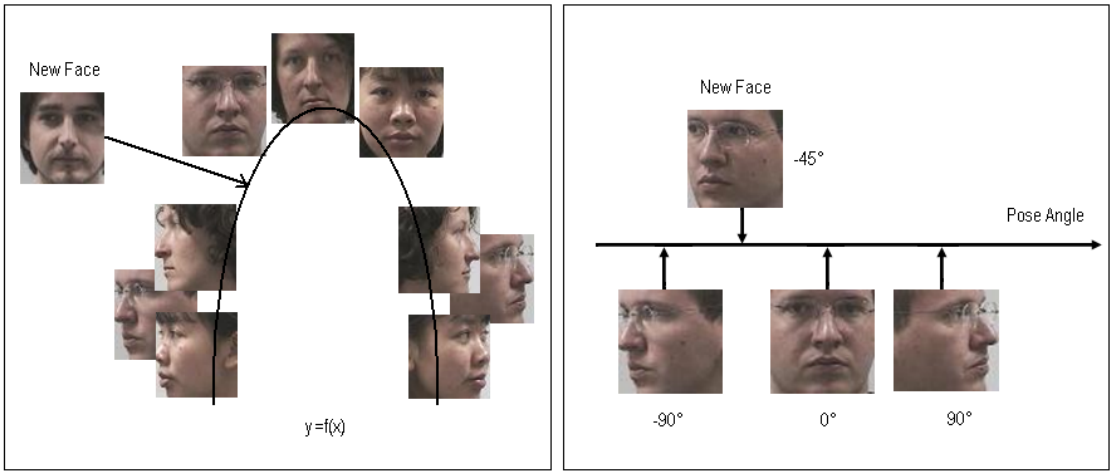



**Figure 3.3: Non-linear Regression and Manifold Embedding Methods:** *Non-linear regression methods perform functional mapping of features extracted from the new image to a head pose measurement. Manifold embedding methods model the head pose as a manifold and project new faces in the subspace given from the manifold using linear and nonlinear techniques.*

localization errors here lead to misclassifications which impacts the accuracy. It is difficult to determine the success rate of map learning for these methods.

### 3.1.4 Manifold embedding methods

Face images can be represented as data samples in high-dimensional space, given by the number of their pixels. The pose of a human head as a solid three dimensional model can vary in fewer dimensions - three dimensions for orientation and three for position. This makes possible the embedding of high-dimensional image samples in a low-dimensional continuous manifold constrained by specified pose variations. Regression or template matching in the embedding subspace can be used to obtain the face orientation.

Various dimensionality reduction methods can be used here in order to represent face images in some subspace. The main difficulties are to bring out the head pose while suppressing other sources of image variation such as identity. McKenna et al. [37] describes how head pose can be estimated by projection of an image into a PCA subspace. Gabor-wavelets are then used for comparison of the projection with a set of appearance-based templates. These approaches are unsupervised techniques which don't use the pose class labels, provided with the training set. If the pose space is divided in discrete pose classes and each class is represented by image samples of different people, then the identity can be decoupled from the pose. PCA can be applied to obtain pose-eigenspaces for each pose class. The pose eigenspace is given by the eigenvectors of the covariance matrix of the training data set. The norm of the projection coefficients vector gives the fraction of energy of the input face, embedded in the eigenspace. This can be understood as similarity measurement between the input face and the templates given by a certain pose class. Finding the class with the highest projection energy delivers the pose [21].

### 3.1.5 Flexible models

Flexible models apply non-rigid templates are specified by a set of parameters. A priori information about the expected shape is used to align these templates with the faces found in the image. The final parameter values obtained after alignment can be used to describe the facial features. During the training phase pose labels and annotations for typical features are required. This allows to determine the pose by comparisons at the feature level rather than at the global appearance level. In this method category fall mechanisms like Elastic Bunch Graphs and Active Appearance Models.

By Elastic Bunch Graphs the features can be represented by so called Gabor jets. These jets are extracted from images with annotated landmark locations. Every facial landmark represents a node from the graph. Every node stores a set of jets for the corresponding landmark. The landmark descriptions provided in the bunch graph can be used to locate landmarks in a new image [20].

Active Appearance Models try to describe a new input image by generating synthetic images that are as similar as possible, using a parameterized model of appearance. The model require fitting parameters in order to be matched to an image. Good selection of parameters is the one which minimizes the difference between the image and the synthesized model example, projected into the image. The high number of parameters makes this a difficult problem [68].

### 3.1.6 Geometric methods

Facial landmarks and their specific configuration are used in geometric approaches for head pose estimation. If the location of the features is known any geometrical relation can be calculated. For example if mouth corner points, eye corner points and the nose tip are given the facial symmetry axis can be found by connecting a line between the midpoint of the eyes and midpoint of the mouth. Weak perspective geometry is assumed when depth changes on the face are small compared with the distance between the face and the camera. The face pose can be determined from the 3D angle of the nose assuming fixed ratio between the facial feature points and a fixed length of the nose [26].

### 3.1.7 Tracking methods

Tracking methods observe inter video frame head movements in order to achieve pose estimation. This includes a set of measurements such as the position of any points of interest or the position of whole image regions. Various tracking methods for head pose estimation exist already. The most of the techniques require initialization step, for example face frontal-view to maintain a lost tracker.

Yao et al. [24] presented an approach where the face is considered to be planar and remain rigid. In this case the motion of the facial features - eyes, mouth and nose can be modeled as an affine approximation. Consistent relationship among the feature locations is achieved with linear regression and a Kalman filter. If a frontal face is represented as a circle then this circle will be projected to an ellipse if the head pose changes. Estimating the pose then becomes a matter of determining the 3D orientation of a circle whose projection corresponds to this ellipse. Further model-based tracking approaches can

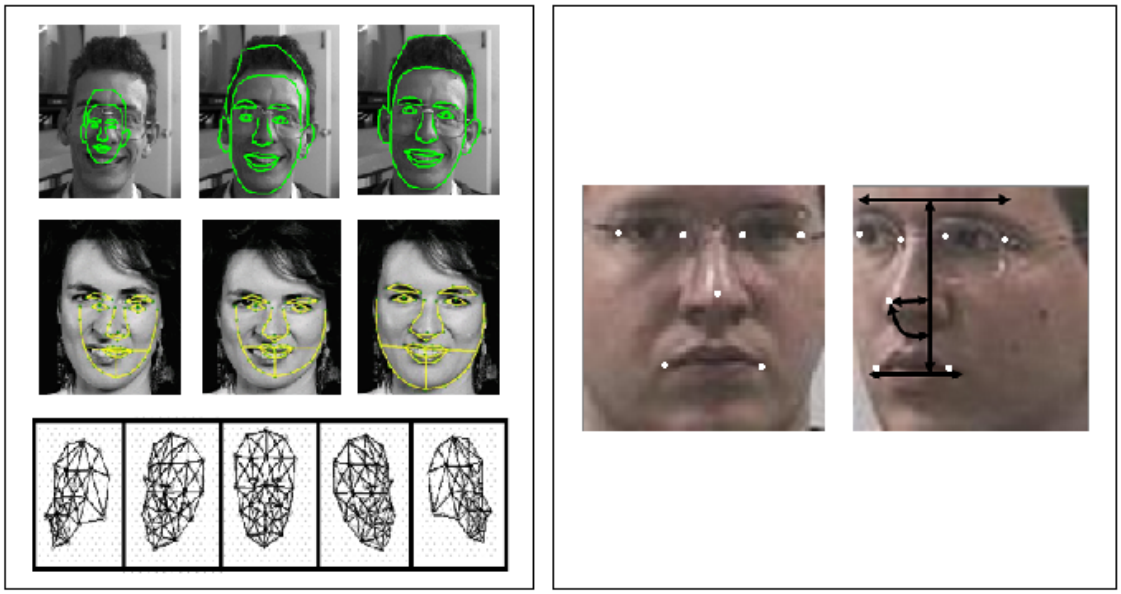


**Figure 3.4: Flexible and Geometric Methods:** *Flexible methods use models which are iteratively fitted to the facial structure. Head pose is estimated by feature comparison or from the fitting model parameters [7]. Geometric methods try to determine the pose from the relative configuration of typical facial features like eye and mouth corners.*

employ cylinders or shape appealing 3D models for head pose estimation. Here the texture mapping from the face image to the model is done, so that the rotation and translation that best match the new observation, can be found.

These approaches require accurate initialization of position and pose to generate or fit a model. Without a detection and pose estimation step, only the relative transformation between frames can be obtained. In this case head pose is determined rather as a tracking of head movement than in the absolute sense.

### 3.1.8 Hybrid methods

One or more of methods can be combined in order to improve the pose estimation. The usage of multiple methods gives the possibility to overcome the limitations given by a single approach and to achieve higher accuracy of the results. To achieve automatic initialization tracking systems can be combined with any static approach, which is also responsible for a reinitialization step, in case that the tracker is lost.

## 3.2 Eigenfaces for Face Pose Estimation

This work dispenses with approaches, which depend on excessive 3D geometry or calculations of flexible models. The eigenface approach seemed to be an attractive method for face pose estimation, because of its simplicity and learning capability.

Originally eigenfaces were presented by M. Turk and A. Pentland [51] in the area of face recognition. The recognition process has two main phases - detection and identification.

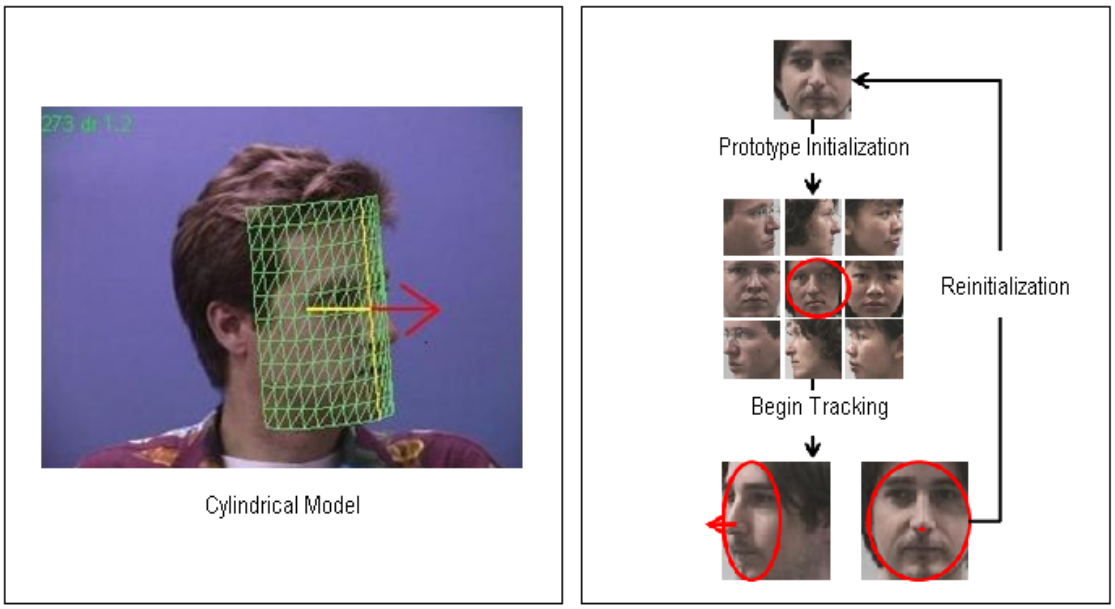



**Figure 3.5: Tracking and Hybrid Methods:** *On the left an example of 3D cylindrical model for head tracking [8]. To estimate the pose the relative movement between video frames has to be estimated. Hybrid methods combine one or more approaches to estimate pose. Here is shown how an example of an appearance template method can be combined with a tracking system based on planar affine transformation.*

After detecting a face in the image it can be compared with a database of existing faces in order to be classified as known or unknown. The comparison is based on the idea that each face can be represented as a linear combination of characteristic images called eigenfaces. The eigenfaces are given by the eigenvectors of the covariance matrix calculated from the initial training set of face images. Recognition is performed by projecting a new image onto the subspace spanned by the eigenfaces and then classifying the face by comparing its position in the face space with the positions of known individuals.

The objective of this work is the estimation of face pose, which is invariant to identity. This requires decoupling of the pose information from the identity and can be achieved by specifying a discrete set of poses and building a training set of different individuals for each pose. Each training set is then used to calculate the corresponding pose eigenspace. The pose of new images can be obtained by computing its projection in each pose egenspace and applying distance based comparison or any other classification approach.

The implementation of the face pose estimation method outlines three main steps:

- Building face pose training set
- Training step
- Classification step

### Building face pose training set

Providing appropriate training data is a tedious task. Various data sets, which contain images of varying head pose from people of different nationality are available [34, 50]. Most of them contain images from the upper body part of male and female persons. Images differ also in scale and are taken under changing lightning conditions. To build our training set of different poses we we cropped faces from the Pointing'04 dataset [54, 29] described in Section 1.2.

Neighbor poses with small differences look very similar. For example pose variance of 1° can be almost indistinguishable. Thus the view space can be divided in pose classes according to a specified angular resolution - for example 20° [48]. In the Pointing'04 dataset provided horizontal resolution of -90 to +90° with step of 15°, and vertical resolution from -90 to +90° with step of 30°, inclusive images taken at -15° and +15°.

Because of the moderate number of training samples for each pose we decided to test three coarse divisions of the training faces in four, six and nine disjoint pose classes. The class labels from left to right and from top to bottom in Figure 3.6 are given by:

- Nine class division
  'Up Left','Up Front','Up Right','Center Left','Center Front','Center Right', 'Down Left','Down Front', 'Down Right';
- Six class division
  'Center Left','Center Front','Center Right', 'Down Left','Down Front', 'Down Right';
- Four class division
  'Left','Front','Right','Down';

### Training step

After specifying the training sets for each pose class the training step can be performed. The images from each class are loaded, converted in gray-scale and scaled down to an arbitrary size appropriate for different image sequences. After this pre-processing step the eigenfaces, which span each pose eigenspace are calculated and stored for further usage. On system run only a subset of this eigenfaces, which correspond to the highest eigenvalues of the pose class covariance matrix are loaded. These eigenfaces span then the pose eigenspace. For each face sample in one pose set the projection onto the corresponding pose eigenspace is also calculated. Now the system is ready for the classification step.

### Classification step

Once new faces are detected in the current frame, they first are converted to gray-scale images and then rescaled to scaled down to size of N x N pixels. The resulting image is then projected in each class eigenspace in order to obtain the weight vector for each class. The projection results are then compared to the averaged weight vector of each class. Different distance metrics can be applied for comparison in the pose eigenspace. The class yielding the smallest distance represents the classification result.

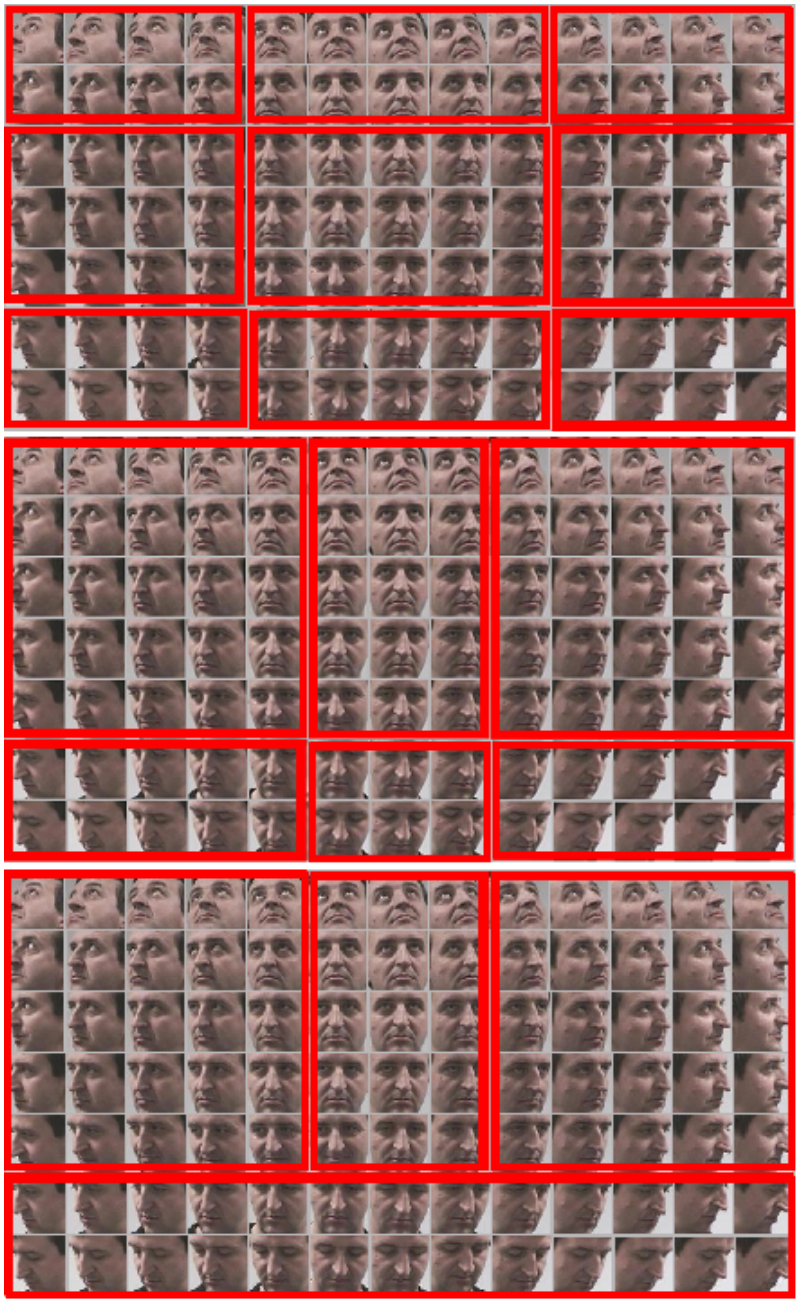


**Figure 3.6: Discrete class division of the face training samples:** *Cropped face are divided into nine, six and four discrete pose classes.*

The *eigenface* approach of M. Turk and A. Pentland [51] is based on the technique of principal component analysis (PCA) which is a subject of *multivariate analysis*. This technique finds application in statistical inference, but the major application area is as data-analytical technique, which tries to describe the multivariate structure of given data [38].

Classification systems obtain as input observation data which are expected to contain information. Ideally the system should be able to make a decision using whichever features are necessary, discarding the irrelevant. However, data-analytical methods are often used in order to reduce the dimensionality of the input data. These methods

concentrate on the extraction of a set of features which best represents the data set without lost of significant information. Several reasons why dimensionality reduction as a separate preprocessing step is useful are given by E. Alpaydin [13]:

- When data can be described with fewer features, the process that underlies the data can be easier to understand, which allows knowledge extraction.
- When data can be represented in a few dimensions without loss of information, it can be plotted and analyzed visually for structure and outliers.
- The complexity of learning algorithms depends often on the number of input dimensions, as well as on the size of the data sample. Memory and computation reduction can be achieved through dimensionality reduction.
- If the input data are decided to be irrelevant, the cost of evaluating can be saved.

Methods for reducing dimensionality can be classified as: feature selection and feature extraction. By feature selection, the focus lies on finding a subset of the prior dimensions that represent the most information and the rest dimensions are then discarded. By feature extraction, a new set of less dimensions that are combinations of the prior dimensions is being searched. These methods may be supervised or unsupervised depending on whether or not they use the output information. Principal components analysis (PCA) counts for one of the best known and widely used unsupervised method.

### 3.2.1 Calculation of Eigenfaces

Each N x N face image of 8-bit gray values can be represented as a vector of $N^2$ dimensions. An image of size 100 x 100 becomes a vector of 10.000 dimensions, i.e. the face can be represented as a point in 10.000-dimensional space. So a training set of 100 sample images from different individuals looking left can be represented as an 100 x 10.000 array . It is expected that images of faces, which are similar in their orientation, will not be randomly distributed in this huge image space. Thats why they can be described by a relatively low dimensional subspace. Principal component analysis can be applied to find the vectors that best describe the distribution of a certain face pose within the entire image space. The vectors that best describe one face pose, span the pose eigenspace. These vectors are the eigenvectors of the covariance matrix corresponding to the original face images of the pose set. They have length of $N^2$ and are face-like in appearance, thus they are called eigenfaces, see Figure 3.7.
Assume that a pose class given by $\Gamma_1, \Gamma_2, \ldots, \Gamma_{l-1}\Gamma_l$ contain faces of L different persons with size M x N pixels. Each face image $\Gamma_i$ is then reshaped to a K = M x N vector:

$$\Gamma_i = \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_k \end{bmatrix}$$

The face vectors, obtained from the training samples in one pose class, form a K x L matrix P:

$$P = \begin{bmatrix} \gamma_{11} & \cdots & \gamma_{1l} \\ \vdots & \ddots & \vdots \\ \gamma_{k1} & \cdots & \gamma_{kl} \end{bmatrix}$$


**Figure 3.7: Eigenface samples for pose classes 'LEFT' and 'RIGHT'**

In order to calculate the covariance matrix $C$ of the input data $P$ the mean vector $\Psi$, which represents the average face of this set is calculated.

$$\Psi = \frac{1}{L}\sum_{i=1}^{L}\Gamma_i$$

The K x K covariance matrix $C$ of this training set is then given by

$$C = \frac{1}{L-1}\sum_{i=1}^{L}\left(\Gamma_i - \Psi\right)\left(\Gamma_i - \Psi\right)^T$$

where L-1 is used for matrix normalization in order to obtain an unbiased estimate of $C$. The value $c_{ij}$ represents the covariance between the $i-th$ and $j-th$ variable. If $c_{ij}$ is non-zero value, this indicates that a linear relationship exist between these two variables. Subject to principal component analysis, to find a set of L orthonormal vectors, $u_i$ , which best describes the distribution of the data. These vectors are the eigenvectors of the covariance matrix $C$. The search is focused on the eigenvectors with the highest corresponding eigenvalues $\lambda_j$:

$$\lambda_j = \frac{1}{L}\sum_{i=1}^{L}\left(u_j^T\left(\Gamma_i - \Psi\right)\right)^2$$

Because $C$ is real and symmetric the vectors $u_j$ of $U$ and are orthonormal:

$$u_j^T u_k = \begin{cases} 1 & \text{if } j = k \\ 0 & otherwise \end{cases}$$

The principal component transformation is performed by projecting the values $\gamma_1, \gamma_2 \ldots \gamma_k$ of the mean adjusted face $\Gamma$ into the pose eigenspace spanned by the principal component axes:

$$\Omega = U^T (\Gamma - \Psi)$$

The transformed values are called principal components of $\Gamma$. The *ith* principal component of $\Gamma$ is given by:

$$\omega_i = u_i^T (\Gamma - \Psi)$$

where i = 1,..,D and D < K, if the pose eigenspace is described by D-dimensional subset of $U$. Then the projection of a face in the pose eigenspace results in D-dimensional component vector $\Omega = \omega_1, ..., \omega_D$. This vector can be used in any pattern recognition approach to find which pose class describes this pattern sample in the best way.

### 3.2.2 Face Pose Classification

After calculation of the pose egenspaces and the class principal components the system is ready for classification. Distance metrics are very simple method to determine which class provides the best description for a new face. Here we need to find the pose class $k$ that minimizes the distance between the face component vector $\Omega$ and the vector $\Omega_k$ that represents the pose class.

$$Class_K = \arg\min Dist(\Omega, \Omega_k) \qquad \Omega_k \in ClassSet$$

The class component vector $\Omega_k$ is calculated by averaging the component vectors calculated from the original training images in the pose class.

$$\Omega_k = \frac{1}{D} \sum_{i=1}^{D} \Omega_i$$

## 3.3 Experiments and Results

In order to classify the pose of the detected faces two simple distance metrics were compared:

- Euclidean Distance

$$D_{eucl}(\Omega, \Omega_k) = \sqrt{\sum_{i=1}^{D} (\omega_i - \omega_{ki})^2}$$

- City Block Metric

$$D_{city}(\Omega, \Omega_k) = \sum_{i=1}^{D} |\omega_i - \omega_{ki}|$$

We first applied the face pose estimation over the training faces from Pointing 2004 data set. In Section 3.2 three different pose class divisions were specified. For each of these class divisions the two distance metrics were compared. After grouping the training faces for example in four classes 'Down','Front','Left' and 'Right' we calculate the four pose eigenspaces. Each training face is then projected in the four pose eigenspaces. The

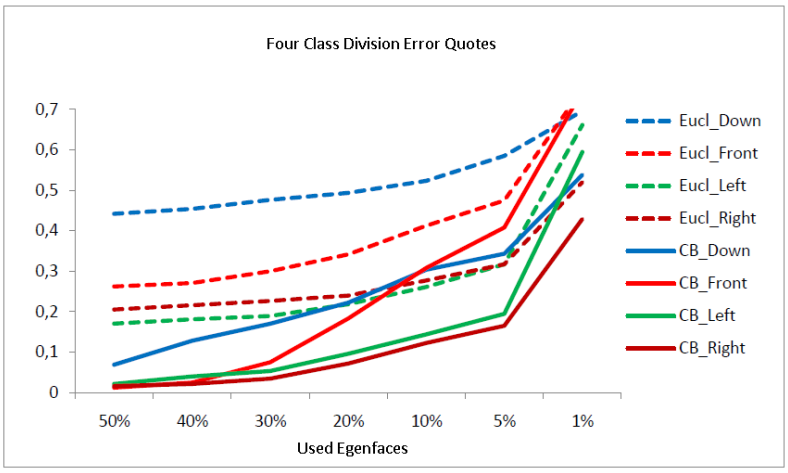



**Figure 3.8: Error Quotes for Four Class Pose Division:** *The number of misclassified faces increases when the number of used eigenvectors, which span the pose eigenspaces is decreased. It can be also seen that euclidean distance performs considerably worse than the city block metric for all poses.*

projection component vector $\Omega$ is then compared against the average class component vector $\Omega_k$. The class yielding the minimal distance is returned as classification result. If the classification response do not correspond to the class in which the training face is grouped the result is counted as wrong. Figure 3.8 shows the error quotes resulting from the four class division and Figure 3.11 summarizes the results from the three different class divisions. The selection of an appropriate number of eigenfaces which span the pose

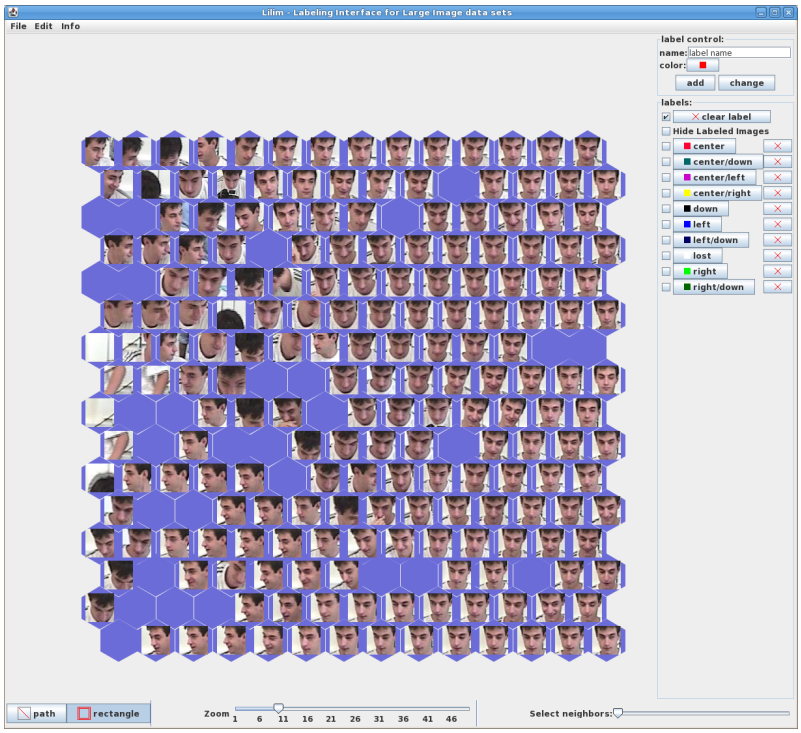



**Figure 3.9: Labeling Interface for Large Image data sets [15]:** *The faces in the selected image sequences of the PETS-ICVS Scenario B1 were labeled with a graphical interface where the visualization is based on self-organizing maps. Some faces were labeled with two classes, for example 'center/right'. This was done in cases where the user was unable to decide in favor of one class.*

eigenspaces plays also an important role for the recognition rate. As we can see in Figure 3.11 the error rate increases with decreasing the number of used eigenfaces. We can also observe that the *city block* metric delivers lower error rates in the three class divisions. Error rates fall under 25 percent with increasing the number of used eigenvectors starting by 20 percent. On the base of these observations we decided to evaluate the face pose in the image sequences from PETS-ICVS 2000 data set with the city block metric under four class division, by keeping 20 percent of eigenvectors for each pose class. Annotations for the exact face orientation in angles was not provided with the PETS-ICVS 2000 dataset. The selected image sequences from scenario B1 contain more than 4550 images. For the evaluation of every second frame of these sequences more than 10000 faces should be labeled corresponding an appropriate face pose class. This large amount of faces was visualized and labeled with a graphical interface based on self-organizing maps. Figure 3.9 shows the application of 'Lilim' - *Labeling Interface for Large Image data sets* on the faces of PETS-ICVS Scenario B1 [15]. Figure 3.10 represents the results of the evaluated frame sequences from camera 1 and camera 2 of the PETS-ICVS 2000 data set. Unfortunately the correct classification rate is not as good as expected. However the

| Face Pose Estimation PETS-ICVS 2000 | | | | |
|---|---|---|---|---|
| CameraID | PersonID | Correct | Wrong | Lost |
| Cam1 | 1 | 1214 | 531 | 162 |
| Cam1 | 2 | 1157 | 423 | 233 |
| Cam1 | 3 | 913 | 643 | 22 |
| Cam2 | 4 | 1631 | 491 | 117 |
| Cam2 | 5 | 1371 | 614 | 172 |
| Cam2 | 6 | 495 | 1012 | 315 |
| Total | | 6781 | 3714 | 1021 |
| | | 64% | 36% | - |

**Figure 3.10: Evaluation of face pose estimation:** *The results are based on the evaluation of every second frame from 4515 images obtained from camera 1 and 4550 images from camera 2 in scenario B1 from PETS-ICVS 2000 data set. The classification of each face is compared with the with the previously assigned labels according to the four class division. Lost faces were ignored for the face pose estimation. Person IDs correspond to the order in which the three persons for each enter the camera field of view.*

recognition technique used is very basic and no abstraction of the faces is performed before PCA calculation, for example using filters. Additionally the recognition rate for person 3 significantly decreases the recognition rate from approx. 70 to 60 percent. We assume that this person was not sufficiently represented by the training set. Face pose classification is performed in each video frame after updating the individual person tracks. A simple error correction function is used to recover face pose changes which are irrelevant for two consecutive frames, for example 'Left'-to-'Right' turning without passing any intermediate pose. When the face pose is finally evaluated we can deduct which person in the image plane is the current focus of attention, as shown in Figure 3.12. In order to improve the face pose estimation the coarse division of pose classes should be refined. Additionally the video data from both cameras can be synchronized so that observed persons which are not in the current image plane can be deducted.

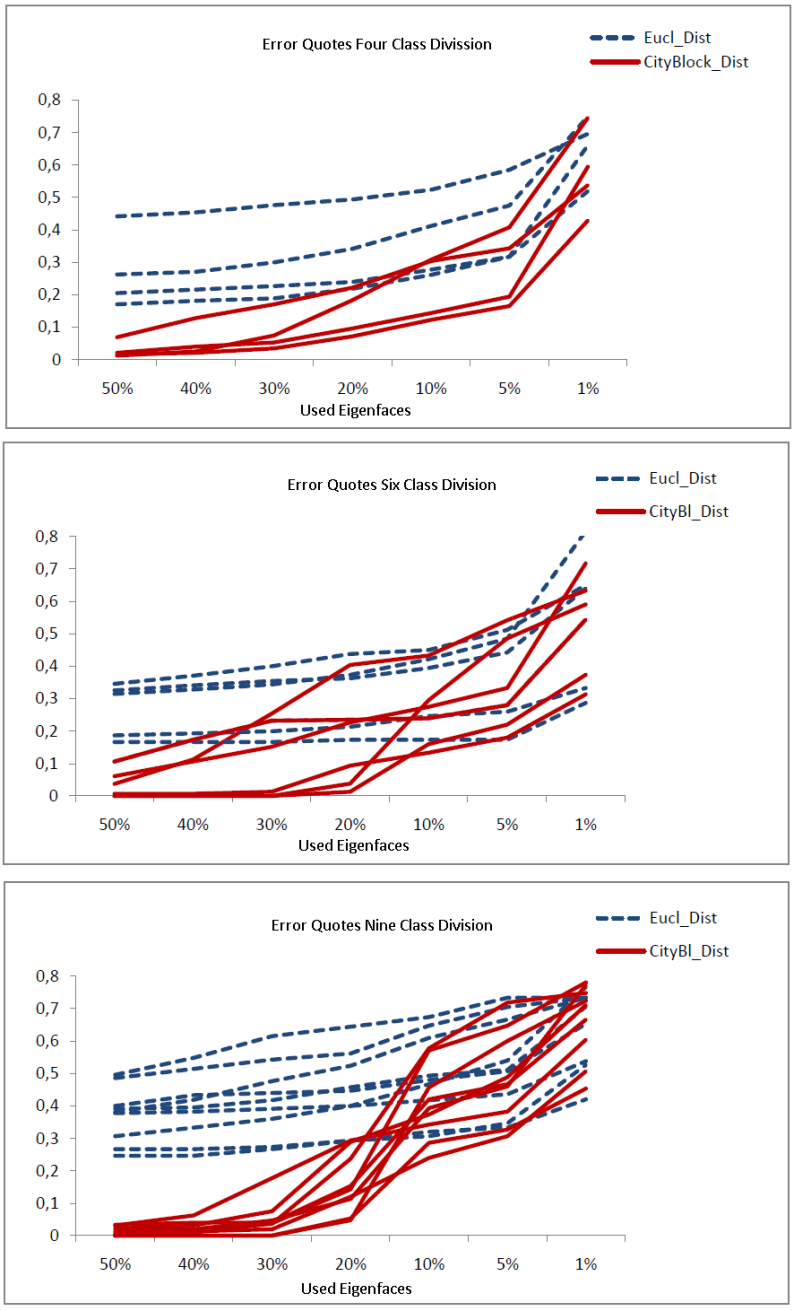



**Figure 3.11: Error Quotes Summary:** *The face pose estimation was tested on the three different class divisions of the training faces. Each line in the diagram represents the error rate of a certain pose class labeled corresponding to distance metric used for the classification. The error rates by 20 percent of used eigenvectors in the nine and six class division were partially higher than these of four class division. Thats why we chose the four class division with the assumption that the provided classes will be enough to estimate the proper direction in which the persons are looking.*

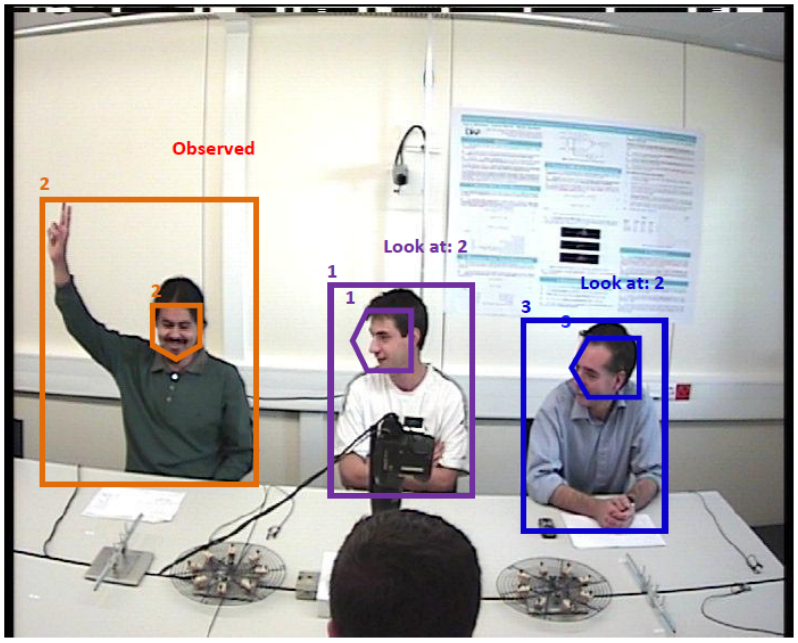



**Figure 3.12: Incorporating the currently observed person:** *Face pose estimation was applied to deviate the currently observed person in the image plane.*

# 4 Action Recognition

After meeting participants are detected their positions can be successfully tracked. The generated tracks can be analyzed in order to draw conclusions about their activities. Tracking provides information which allows to gain understanding of the persons' behavior. This includes extraction of a set of feature from each frame in an image sequence. These features are used to train specific classifiers and to perform recognition of the individual actions. Recognition of the individual activities performed during meeting can be then applied to extract high level group events, such as presentations and discussion. Zobl et al. [60] presented an approach for individual action recognition based on global motion features extracted and calculated from the gray-scale values of subsequent difference images. A segmentation step is performed in order to obtain the boundaries of an action and as final processing step statistical classification is achieved with hidden Markov models trained on specific actions.

In this chapter we present a set of features based on the work of Zobl et al. [60] but extends their work in some aspects. We also describe how single actions can be recognized using hidden Markov models (HMM). The approach was tested on the images from the scenarios described in Section 1.2.

Two main processing levels are outlined by the action recognition approach - *feature extraction* and *statistical classification*. All methods described in the previous chapters are focused on evaluating the current image. The performance of each human action takes a certain time interval. Thus action recognition requires a sequence of observations obtained over several video frames. Additionally to the position and face coordinates of each person a set of global motion features is extracted from each incoming frame and stored in the tracking history. The action recognition can be integrated in the tracking mechanism so that, as soon as enough data is stored in the tracking history, action classification can be performed. An arbitrary time step specifies when the next recognition step for the currently observed person should be done, as shown in Figure 4.1. Although HMMs do not require feature vectors with a fixed length (over a specific time interval) we chose not implement sophisticated segmentation techniques. Firstly, because it is beyond the scope of this thesis and secondly, because the implementation wold here required us to introduce some assumptions about the nature of the videos. This would here led to a less generic recognition system. However the recognition rate could have been improved.

The tracking approach extracts and stores the features in the person object history. For the action recognition we use sequences which cover at about 1.8 seconds video material by camera frame rate of 30fps on a time step of about 1 second. The mechanism outcome is the recognized action of one unknown feature segment.

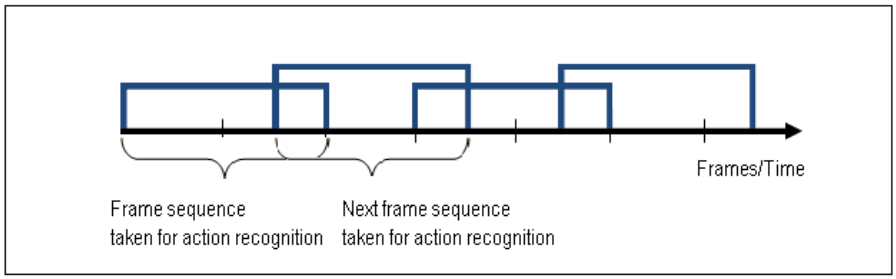



**Figure 4.1: Feature Stream Window:** *The tracking mechanism delivers all required features for the action classification. These features are store in the tracking history of each detected person. As soon as enough data are collected action classification can be performed. The classification can be performed on each new incoming frame or on an arbitrary time interval by modifying the overlap length.*

## 4.1 Feature Extraction

The feature extraction is a part of the tracking process. For each incoming video frame, background subtraction is performed in order to capture motion. The resulting binary image contains the regions of moving persons in the foreground. This regions are individually used to extract a set of global motion features as described in [60]. The moving regions in the foreground image $I_f$ (see Equation 2.1) change their size and position, which can be interpreted as a motion distribution over the image space in the horizontal and vertical direction. Different movements can be characterized by a specific distribution. If the distribution can be described with a set of features this can contribute to obtain motion patterns appropriate for action recognition. The set of features we used to depict an approximation of the moving direction of the human body are:

- Person center of mass $m$,
- The relative change of the center of mass between two consecutive frames $\Delta m$,
- Motion wideness $\sigma$,
- Motion intensity $i$,
- Size of the person region,
- Face position relative to the person region.

The first feature extracted from the foreground image $I_f$ at time $t$ is the person centroid, which specifies the center of mass. The center of mass $\vec{m}(t) = \left[m_x(t), m_y(t)\right]^T$ is calculated relative to the bounding box of the region $R_i$:

$$m_x(t) = \frac{\sum\limits_{(x,y)\in R_i} x \times I_{f_t}(x,y)}{\sum\limits_{(x,y)\in R_i} I_{f_t}(x,y)} \qquad m_y(t) = \frac{\sum\limits_{(x,y)\in R_i} y \times I_{f_t}(x,y)}{\sum\limits_{(x,y)\in R_i} I_{f_t}(x,y)}$$

Movement direction variations can be considered by calculating the relative change of the center of mass between two frames.

$$\Delta m_x(t) = m_x(t) - m_x(t-1)$$
$$\Delta m_y(t) = m_y(t) - m_y(t-1)$$

In order to distinguish between actions where large parts of the body move such as 'sitting down' and actions which concern only small body parts movement such as 'head shaking' the mean absolute deviation of a pixel $\vec{\sigma(t)} = \left[\sigma_x(t), \sigma_y(t)\right]^T$ relative to the center of motion is calculated. This feature describes the motion wideness.

$$\sigma_x(t) = \frac{\sum\limits_{(x,y)\in R_i} (x - m_x(t)) \times I_{f_t}(x,y)}{\sum\limits_{(x,y)\in R_i} I_{f_t}(x,y)} \qquad \sigma_y(t) = \frac{\sum\limits_{(x,y)\in R_i} \big(y - m_y(t)\big) \times I_{f_t}(x,y)}{\sum\limits_{(x,y)\in R_i} I_{f_t}(x,y)}$$

The absolute average height of the motion distribution is determined by:

$$i(t) = \frac{\sum\limits_{(x,y)\in R_i} I_{f_t}(x,y)}{\sum\limits_{(x,y)\in R_i} 1}$$

This feature handles differences between very intensive movements and almost stationary images. Large values of $i(t)$ express large image variances i.e. intensive motion and small value characterize slow motion changes.

Actions like 'standing up' and 'hand up' where the motion direction and intensity are similar may obtain similar characteristic vectors. Four additional features are used to emphasize the differences of these actions. When persons stand up or sit down, the face position moves down or up respective to the bounding box as shown in Figure 4.2. When persons move their hands upwards or sidewards the face position remains central relative to the bounding box. Thats why we used the width $w$ and hight $h$ of the person bounding box and the horizontal an vertical relation between the relative face center coordinates $\left[c_x(t), c_y(t)\right]^T$ and the region size, given by:

$$f_x(t) = \frac{c_x(t)}{w(t)} \qquad f_y(t) = \frac{c_y(t)}{h(t)}$$

One of the main advantages of this approach is that the information in the high dimensional person region image can be reduced to an eleven dimensional feature vector $\vec{v}$ and at the same time preserving the characteristics of the currently observed motion.

$$\vec{v}(t) = \left[m_x, m_y, \Delta m_x, \Delta m_y, \sigma_x, \sigma_y, i, w, h, f_x, f_y\right]^T$$

The extracted feature vectors from each frame form a sequence of motion vectors $\vec{V}_n = [v_1, v_2, ..., v_n]^T$, where each vector carries important information about the currently performed action. The feature extraction begins with calculating the center of mass for each person in each incoming frame. First all centroid positions $\vec{M}_n = [m_1, m_2, ..., m_n]^T$ of $\vec{m}(t) = \left[m_x(t), m_y(t)\right]^T$ are extracted. Moving average filter is applied to smooth the data. Moving average filter is a lowpass filter that applies coefficients equal to the reciprocal of a specified span value. For each $m(t)$ of the smoothed sequence the rest of the motion features are calculated and together with the region size and face position finally stored in the sequence $\vec{V}_n$ of feature vectors. The sequence $\vec{V}_n$ is then used to classify the current action.

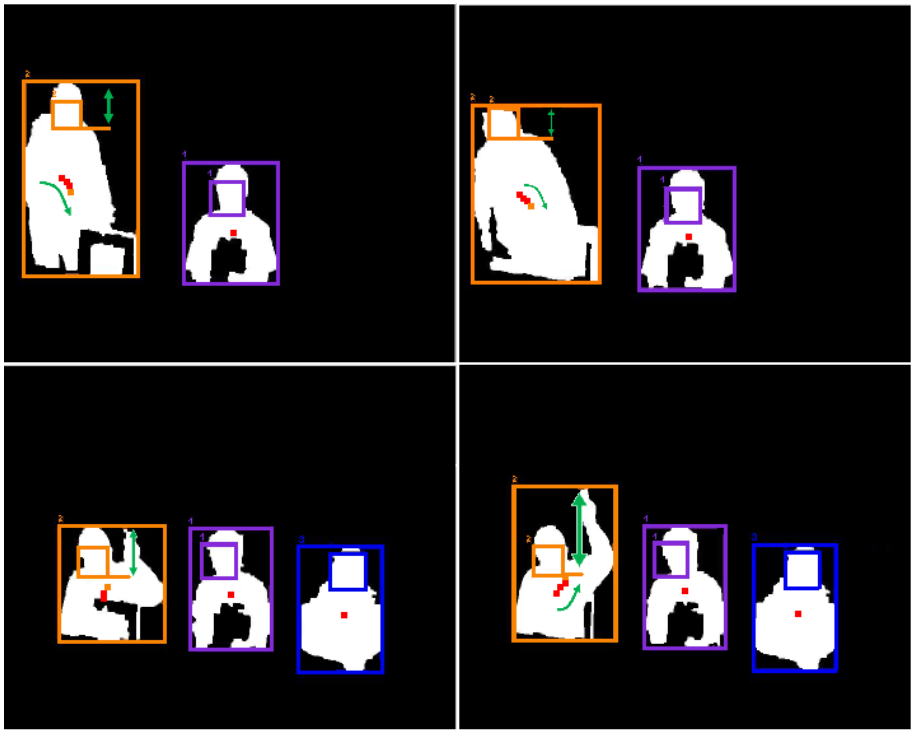


**Figure 4.2: Motion based features:** *For each person region in the foreground image the center of mass is calculated. We can see here the centroids obtained from three previous frames marked in red, which indicate the motion direction. For the sitting persons who don't move very intensely the centroid coordinates coincide over three frames. We can also observe that for actions like 'hand up' the centroid is moving upwards or downwards, depending in which direction the hand is currently moving. This could lead to similar feature vectors between actions like 'hand up' or 'stand up'. Using the relative face position and the size of the person regions can indicate the differences between these actions. As we can see the face location stays relative centered to the bounding box by raising hand. By 'sitting down' or 'standing up' the face moves together with the person region and remains in the upper part of the region.*

## 4.2 Statistical Classification

For many temporal inherent problems which represent a process that unfolds in time the knowledge about the exact step sequence which is passed in order to obtain the actual current state may be negligible. More important to know is how the system at time $t$ was influenced by the directly preceding state at time $t-1$. *Hidden Markov models* (HMMs) are known to find wide usage in such problems for instance in speech or gesture recognition. Hidden Markov models obtain a number of parameters which try to 'explain' the set of training samples from a certain category. New samples can be then classified by the model that has the highest posterior probability, or which best 'explains' the test sample [67]. In this work an open source library [9] was used to create and train a number of hidden Markov models in order to recognize individual actions. The following sections contain a short introduction to the theoretical aspects of hidden Markov models and describe how they find an application for action recognition.

### 4.2.1 Discrete Markov Processes

Hidden Markov models are based on so called *Markov models*, also known as *Markov chains*. Markov models can be described as a system that at any discrete time step is in one of $N$ different states $S_1, S_2, ..., S_N$. The state is denoted as $q_t = S_i$, $t = 1, 2, ..., T$ which means that at time step $t$, the system is in state $S_i$. The system moves to a state with a given probability, depending on the values of the previous states:

$$P(q_{t+1} = S_j \,|q_t = S_i, q_{t-1} = S_k, ...)$$

Special case are *first-order Markov chains* where the state to which the system moves next depends only on the current state:

$$P(q_{t+1} = S_j \,|q_t = S_i, q_{t-1} = S_k) = P(q_{t+1} = S_j \,|q_t = S_i)$$

This means that the system is time independent. A set of *transition probabilities* $a_{ij}$ with which system switches in $S_j$ being in state $S_i$ can be expressed as:

$$a_{ij} = P(q_{t+1} = S_j \,|q_t = S_i \,) \qquad 1 \leq i, j \leq N$$

The transition probabilities should satisfy the following conditions:

$$a_{ij} \geq 0 \quad and \quad \sum_{j=1}^{N} a_{ij} = 1$$

The system can revisit a state at different time steps. The transition from $S_i$ to $S_j$ has the always the same probability. The transition probabilities $\left[a_{ij}\right]$ specify a N x N matrix $A$ whose rows sum to 1. The system requires the definition of a set of initial probabilities $\pi_i$. They specify the probability for each state $S_i$ being the first state in the sequence:

$$\pi_i = P(q_1 = S_i) \quad where \quad \sum_{i=1}^{N} \pi_i = 1$$

The initial probabilities are stored in a vector $\Pi = [\pi_i]$ of length N. *Observable Markov models* have states which can be observed at any time. This means that when the system switches from one state to another, the result is an observation sequence of states which correspond to physically observable events.

One very popular example which describes the weather conditions [63] is shown in Figure 4.3. Three main states are defined $S_1 = Sunny, S_2 = Cloudy, S_3 = Rainy$. The initial probability for all states is equal:

$$\pi_i = \tfrac{1}{3} \qquad i \in \{1, 2, 3\}$$

The transition probability matrix of this model is given by:

$$A = \begin{pmatrix} 0.3 & 0.7 & 0.0 \\ 0.3 & 0.5 & 0.2 \\ 0.1 & 0.2 & 0.7 \end{pmatrix}$$

The probability that a sunny day follows after one rainy day can be then expressed as:

$$a_{31} = P(S_3 = Sunny \,|S_1 = Rainy \,) = 0.1$$

In this model it is assumed that the weather tomorrow depend only on the weather today and not from the weather yesterday. In order to calculate the probability with which the model generates a particular sequence we simply multiply the successive probabilities.

$$P(O\,|\Pi, A\,) = P(q_1)\prod_{t=2}^{T} P(q_t\,|q_{t-1}\,) = \pi_{q1}a_{q1q2}...a_{qT-1}a_{qT}$$

For instance, to find the probability of a sample observation sequence $O = (S_3, S_3, S_2, S_1, S_2)$ given the initial and transition probabilities $\Pi$ and $A$ we have:

$$P(O\,|\Pi, A\,) = P(S_3).P(S_3\,|S_3\,).P(S_2\,|S_3\,).P(S_1\,|S_2\,).P(S_2\,|S_1\,) = \pi_3 a_{33}a_{32}a_{21}a_{12} = 0.0098$$

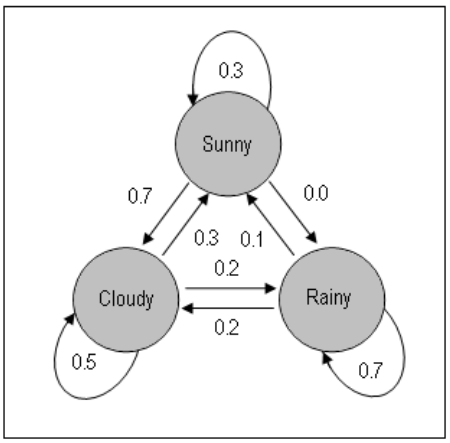



**Figure 4.3: Weather Markov Chain:** *The states in the Markov chain are directly observable. The weather changes can be directly seen.*

### 4.2.2 Hidden Markov Models

The states of a hidden Markov model (HMM) in comparison to Markov chains are not observable. However visiting a state produces an output that can be observed i.e the observation can be represented as a probabilistic function of the state. Hidden Markov models can be characterized similar to Markov chains. According to Rabiner [58] a HMM is defined by a set of parameters $\{N, M, A, B, \Pi\}$ which are given as:

- **N**- specifies the number of states in the model, which are individually denoted by $S = \{S_1, S_2, ...S_N\}$ and states at time $t$ by $q_t$. In the rule each state can be reached from any other state in so called ergodic models, but often other more restricted connections are from interest. For instance left-to-right interstate connection structure which is widely applied in the speech recognition. The states are hidden but they can assigned with some physical significance.
- **M** - describes the discrete alphabet size. The system that is being modeled produces some physically observable output. The distinct observation symbols produced in the system states are denoted by $V = \{v_1, v_2, ...v_M\}$.

- **A** - represents the transition probability distribution matrix where

$$a_{ij} = P(q_{t+1} = S_j \,|q_t = S_i) \quad 1 \leq i, j \leq N$$

  For ergodic models $a_{ij} > 0$ for all $i, j$ and for other HMM types the transition probability for some state pairs $(i, j)$ can be set to o.

- **B** - represents the probability distribution of the observation symbols in state $j$.

$$B = \{b_j(k)\}, \quad where$$
$$b_j(k) = P(v_k \,|q_t = S_j), \quad 1 \leq j \leq N \quad and \quad 1 \leq k \leq M$$

- $\Pi$ - defines the initial state distribution $\{\pi_i\}$, where

$$\pi_i = P(q_1 = S_i) \quad 1 \leq i \leq N \ .$$

The parameters $N, M$ are implicitly specified which allows to denote HMM in more convenient way:

$$\lambda = (A, B, \Pi)$$

The model $\lambda$ can be used to generate observation sequences of arbitrary length, but often the other direction of estimating the parameters of the model given a training set of sequences. If we consider the weather example of a Markov chain from

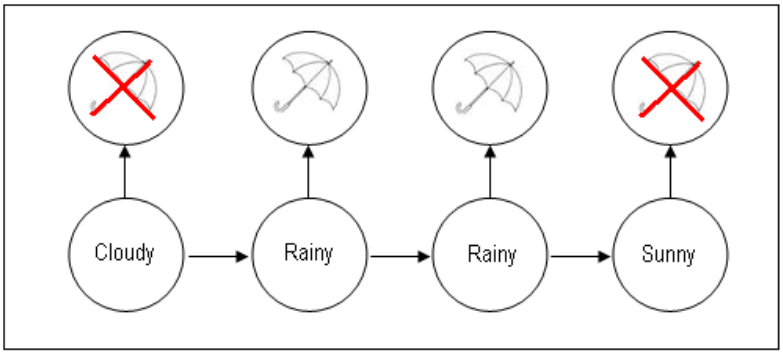



**Figure 4.4: Weather Hidden Markov Model:** *The states of the hidden Markov models are not observable. In the weather example, if we could not see though the window, we could make decision about the weather, depending on the fact if other people in the room wear an umbrella or not.*

the previous section and transform it to a hidden Markov model then the states $S_1 = Sunny, S_2 = Cloudy, S_3 = Rainy$ would not change. However they would become hidden, and so no more observable. What we could see are the events initiated by the states. We can extend the example so that a man who works in an underground train station and sells transport tickets can make assumptions about the current weather based only the clothes of the people who enter the station. If the persons wear umbrellas he could assume that outside rains. If the persons don't wear umbrellas the weather outside could be sunny or cloudy but with sure not rainy. Then he can think over which combination $P(Weather\_today = S_j \,|Weather\_yesterday = Sunny)$ and $P(Weather\_today = S_j \,|Clothes = No\_umbrella)$ is the most probable.

Three main problems are object of interest, given a number of sequences of observations [13]:

1. Given a model $\lambda$ and any observation sequence $O = O_1O_2...O_T$, how can we evaluate the probability $P(O\,|\lambda\,)$of the given observation sequence.
2. Given a model $\lambda$ and an observation sequence O, which state sequence $Q = q_1q_2...q_T$ maximizes the probability of generating O - $P(Q\,|O,\lambda\,)$.
3. Given a training set of observation sequences, $X = \left\{O^k\right\}_k$, how we can find $\lambda$ that maximizes $P(X\,|\lambda\,)$. Namely we want to learn the model that maximizes the probability of generating X.

Since the direct implementation of a hidden Markov model is not an objective in this thesis we will constrain us only over a brief description of the third problem. For detailed theory fundamentals of HMM so as mathematical proofs of the convergence behavior the reader is referred to the literature [39, 62, 58].

### 4.2.3 Parameter estimation

The third problem describes the attempt to optimize the model parameters $(A, B, \Pi)$. The observation sequence used to adjust the model parameters is called training sequence. The training problem is one of the most critical one since it allows us to adjust the model parameters in order to create models that 'best' represent some real phenomena. For any given finite observation sequence the model parameters can not be estimated efficiently [58]. However local maximization of $P(O\,|\lambda\,)$ can be achieved by Baum-Welch algorithm which is an iterative expectation-maximization (EM) approach.

The main steps of the Baum-Welch iteration are:

1. Calculate $\alpha$ and $\beta$
2. Calculate $\gamma$ and $\xi$
3. Reestimate $A$,$\Pi$,$c_{ij}$,$\mu_{ij}$ and $\sigma_{ij}$
4. Terminate if exit criteria is satisfied, else go to $(1.)$

The Baum-Welch algorithm is based on the *Forward-Backward* mechanism, where $\alpha_t(i)$ is so called *Forward-Variable* and represents the probability by observing particular sequence $O = O_1O_2...O_t$ until time $t$ and reaching the state $S_i$ at time $t$ by given the model $\lambda$. The *Backward-Variable*$\beta_t(i)$ defines the probability that the following or the rest of the observation sequence $O = O_{t+1}O_{t+2}...O_T$ could be 'triggered' from state $S_i$ at time $t$. Then with the help of these two variables we can compute $\gamma_t(i)$ and $\xi_t(i,j)$, where $\gamma_t(i)$ defines the probability that the system is in state $S_i$ at time $t$ and $\xi_t(i,j)$ describes the probability with which the system at time $t$ moves from state $S_i$ to state $S_j$ in time $t+1$.

The estimation of these variables can be done iteratively:
**Calculate** $\alpha$

$$\alpha_t(i) = P\left(O_1...O_t, q_t = S_i\,|\lambda\,\right)$$

- Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$$

- Iteration:

$$\alpha_{t+1}(j) = \left[\sum_{j=0}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1, 1 \leq i \leq N$$

**Calculate** $\beta$

$$\beta_t(i) = P(O_{t+1}...O_T \,|q_t = S_i, \lambda\,)$$

- Initialization:

$$\beta_T(i) = 1 \quad 1 \leq i \leq N$$

- Iteration:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad 1 \leq t \leq T-1, 1 \leq i \leq N$$

**Calculate** $\gamma$

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

**Calculate** $\xi$

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b(O_{t+1}) \beta_{t+1}(j)}$$

The goal of the parameter adjustment in the iterative process of observing data is to achieve more accurate predictions as more data become available. This can be understood as 'learning', because the parameters are not available a priori but are calculated through a given sequence. In this way obtained 'knowledge' can be further used to analyze new data. It is not sufficient to obtain optimal parameters from a single data observation, thats why the parameters should be set in relation to the current data, so that they can be newly optimized to a certain extent. This new parameters can be used again to compare data and then again to perform optimization. This steps of calculation and optimization can be repeated until the optimization reaches a (local) maximum or an exit condition specified by the number of iterations is reached. The proof that this method shows convergence against a local maximum and delivers better parameter values on each step can be seen in the literature [39, 62].

The models that we discussed above are discrete in nature. The observable events can be clearly distinguished and no smooth transitions or intermediate steps were applied. To handle continuous observations the usage of a simple discrete distribution is not sufficient. Here a probability density function such as Gaussian normal distribution with $N(\mathrm{x}, \mu_i, \sigma_i^2)$ with mean value $\mu_i$ and variance $\sigma_i^2$ is adopted. In order to achieve better results mixture of Gaussians are often applied in the praxis. We obtain an observation density of the form:

$$b_j(O) = \sum_{m=1}^{M} c_{jm} N(O, \mu_{jm}, U_{jm})$$

Here $c_{jm}$ are the mixture weights, $\mu_{jm}$ is the mean vector and $U_{jm}$ the covariance matrix of the $mth$ mixture component in state $j$. The coefficients should satisfy the following conditions:

$$\sum_{m=1}^{M} c_{jm} = 1, \quad 1 \leq j \leq N$$

$$c_{jm} \geq 0, \quad 1 \leq j \leq N, 1 \leq m \leq M$$

To obtain the parameters of the mixture of Gaussians we have to extend the variable $\gamma_t(j)$ with one mixture index, so that we obtain $\gamma_t(j,k)$, where $t$ is the time, $j$ represents the state, and $k$ is the index of the mixture element. We obtain:

$$\gamma_t(j,k) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \left[ \frac{c_{jk} N(O_t, \mu_{jk}, U_{jk})}{\sum_{m=1}^{M} c_{jm} N(O_t, \mu_{jm}, U_{jm})} \right]$$

The mixture weight coefficients $c_{jk}$ can be computed by:

$$c_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)}$$

The mean values $\mu_{ik}$ are obtained with:

$$\mu_{ik} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) O_t}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

And finally for the covariance matrix $U_{jk}$ we have:

$$U_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(O_t - \mu_{jk})^2}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

## 4.3 Experiments and Results

In a fully connected - ergodic HMM, every state can be reached from any other state, which results in a full $N \times N$ matrix $A$.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

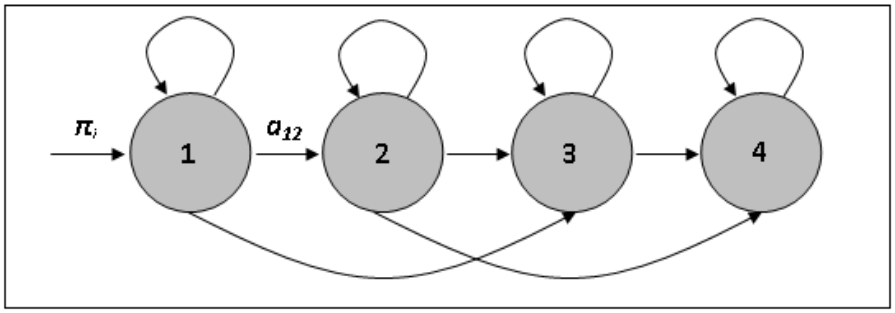



**Figure 4.5: Left-to-right HMM topology:** *Here the system never moves to a state with a smaller index. This kind of HMMs find wide usage in the speech recognition.*

For many applications it is useful to balance the complexity with the state topology, disallowing some transitions setting their $a_{ij} = 0$. For instance in the left-to-right topology the states are ordered in time so that as time increases, the state index increases or stays the same. This constraints allow modeling of sequences whose properties change over time as in speech. In this case the system also never moves to a state with a smaller index, namely, $a_{ij} = 0$, for $j < i$. Transitions between not subsequent states can be constrained by $a_{ij} = 0$, for $j > i + \tau$. The example of the left-to-right HMM given in figure 4.5 with $\tau = 2$ has the state transition matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{23} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

The number of states N is another factor that impacts the complexity of an HMM. Because the states are hidden, their number is not known and should be chosen before training. Prior information can be used and optimized by cross-validation through checking the likelihood of validation sequences.

In order to recognize person actions, we trained separate HMMs for each action. The prior parameters were set to random. The number of states and mixtures depend on the variety of the available training material. The number of states and mixtures were selected after comparing the learning curves of the HMMs with different parameter combinations. For three mixtures of Gaussians and six states we obtained the most optimal learning curves with the available training data. We applied left-to-right topology of six states and three mixtures of Gaussians for all action models. After specifying the model topology and reestimating the parameters with the training data the system is ready to classify action sequences. The probability with which a given HMM is likely to generate the given sequence is calculated with the *Viterbi algorithm*. Viterbi algorithm is similar to the estimation of the Forward-Variable, where instead to use summation, maximum function is applied at each step. Given an observation sequence O of extracted features and a set of trained models $\lambda_i$ for $i$ different actions, the model $\lambda$ with the highest production probability should be found.

$$\lambda^* = \arg\max_i \quad P(O|\lambda_i) \qquad \lambda i \in Actions$$

Due to the small amount of training data only a subset of four actions were covered in our training set. These actions are 'walk', 'sitting down', 'sit' and 'hand up'. Figure 4.6

shows how the recognized actions were visualized.
In order to evaluate the action recognition it would be favorable to implement an appropriate evaluation mechanism. Due to the lack of time we could only estimate the recognition rate on the basis of simple counting on selected video sequences. However this estimation gives an overview of the recognition capabilities of the developed system. Figure 4.7 shows the partial results of the individual action recognition. The second part of the scenario B1 delivers very low recognition rate. In this part of the scenario all six people raise their hands at the same time. This results in occlusions which are handled by the splitting step in the tracking mechanism, so that the previous positions are used to keep the person objects separated. In this case we cannot obtain optimal regions from which we can extract the motion features and we use only the information visible in the previous regions. Additionally the occlusions impact the motion distribution in the observed range. In order to improve the action recognition it is recommended to refine the tracking approach. Extending the training material to a reasonable amount of data from different camera setups, lighting-, and room conditions will also allow dealing with misclassification. Additionally an approach for automated evaluation should be provided in order to generate reasonable results which can be compared when different features and classification methods are being applied.

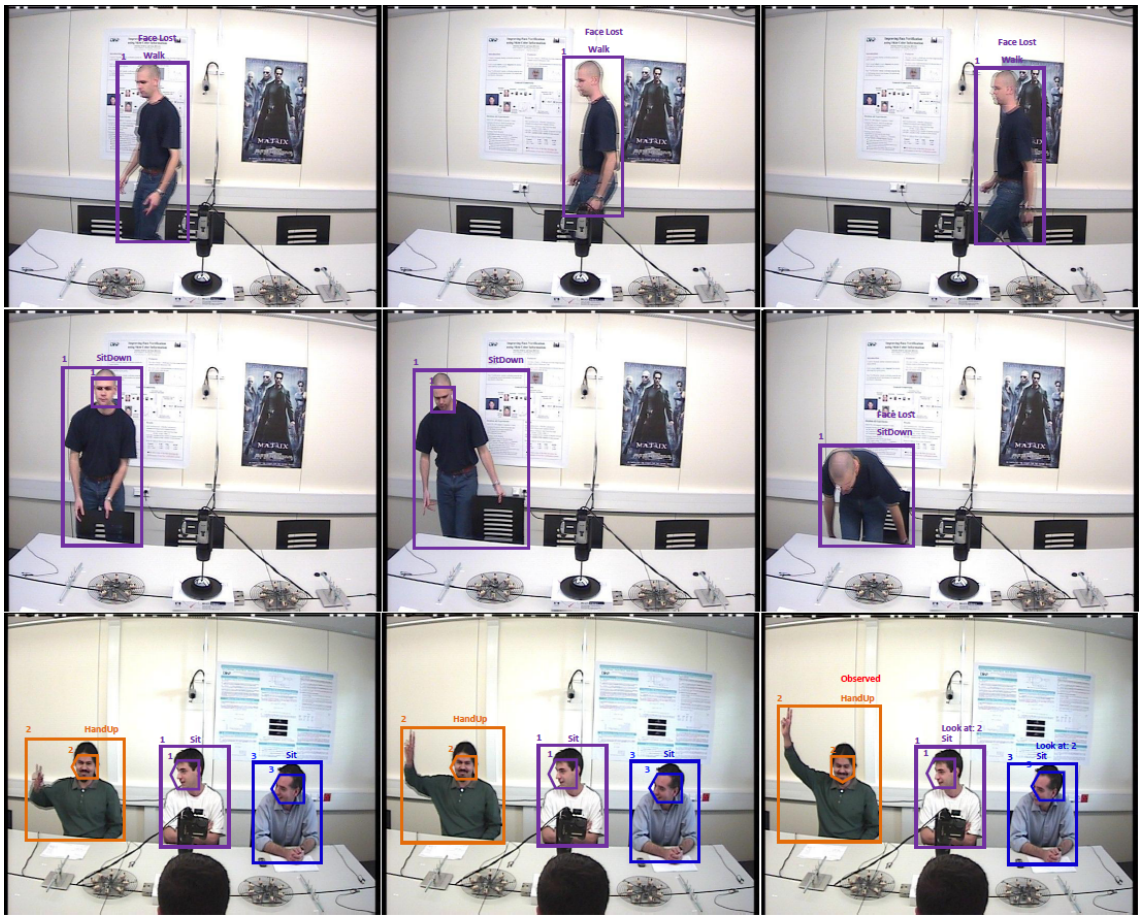



**Figure 4.6: Visualization of recognized actions:** *For each current frame the recognized actions are stored in the person tracks. These can be then printed out as action label for each person object.*

| Action Recognition PETS-ICVS 2000 | | | |
|---|---|---|---|
| Action | Nr. of sequences | Correct | Wrong |
| walk | 3 | 100% | 0% |
| sit down | 6 | 100% | 0% |
| hand up | 21 | 62% | 38% |
| sit | 14 | 65% | 35% |

**Figure 4.7: Evaluation of recognized actions:** *The action recognition is evaluated on selected image sequences from PETS-ICVS 2000 dataset. Depending on camera field of view only 3 persons were observed by 'walking' and respectively correct recognized. For all six participants 'sitting down' was also correctly recognized. The persons sit very near to each other so that by moving their hands their regions intersect or blobs just get connected. The mechanism which handles these situations splits the regions based on their previous position. In this case the feature extraction mechanism observes motion changes only in this region. That's why not the complete motion variance can be used for feature extraction, which results in lower recognition rate for the action 'hand up'. For 'sitting' most errors are caused by segmentation errors in the tracking process, which results in similar motion features as real activities.*

# 5 Implementation structure

In this work we presented a method to track persons in a meeting room and recognizing some of their activities, based on feature extraction from video data. The main modules of the application are:

- **'Configuration'** - module sets all required parameters and initializes the main classes.
- **'ImageLoader'** - is responsible for loading image sequences of arbitrary length and file type in the system.
- **'FaceDetect'** - integrates an open source implementation of Viola-Jones face detection approach[4] and includes additional routines to configure the results for further usage.
- **'FacePose'** - implements separately the training and recognition phase for face pose estimation. In the first phase the training samples of an arbitrary number of classes are loaded, scaled down to a certain size, which can be different for different applications and the pose eigenspaces are calculated. Then for each pose class only a small arbitrary number of eigenfaces is stored for further usage. In the second phase, new faces are projected into the subspace of each pose class. Distance metric based comparison is applied to label the new face with one of the available pose class.
- **'BackgroundSubtraction'** - is the heart of the tracking method. Here different background subtraction approaches are implemented in separate classes which can be selected in the configuration module.
- **'BinaryProcessing'** - specifies all post-processing procedures which are applied to improve the foreground image, so that smooth contours of the person objects can be achieved.
- **'BlobManager'** - is in fact the region tracking mechanism, which clusters and splits the blobs, depending on the previous measurement.
- **'FeatureExtraction'** - calculates and prepares the motion features of each person region in the observed frame.
- **'Person'** - instantiates person objects. All main person attributes are updated with the information from the current frame which is being measured. A separate attribute stores all relevant features of the current measurement in the tracking history of the observed person object. Additional class method here is responsible for the classification of the action that is currently performed by the particular person.

- **'PersonManager'** - deals with the handling of multiple person objects and the updating their tracks from the current frame. This includes instantiation of new person objects, data association, handling of lost faces, removing from false detected person objects, estimation of the currently observed person. For convenience the implementation of routines for visualizing the results is also covered in this package.
- **'TrainHMM'** - implements the training of the different HMMs under the usage of an open source HMM Toolbox. Here the extracted training sequences are loaded, the number of model states and mixtures are specified. The model transition probabilities are set to random and reestimation follows applying the Baum-Welch (EM) approach implemented in the HMM Toolbox. The trained models are stored and the loaded by the tracking main.
- **'ThirdParty'** - contains the source of the face detection method and the HMM Toolbox.

The application modules can be easily extended or replaced in order to achieve better results. Most parameters are configurable, which enables flexible setup for usage on other test image sequences.

The application process and the module dependencies can be summarized as follows. Starting with loading of images, we apply a selective running average mechanism to each currently observed frame. The output is a binary image which contains all detected objects which are part of the moving foreground. This foreground image is post-processed with morphological operators and size filters in order to obtain smooth person boundaries. The resulting image can contain blobs of different sizes, which correspond to body parts, individual persons, multiple persons which are near to each other and overlap in the view of the image plane. This requires separate handling based on size and position of the blobs. Here hierarchical clustering is applied to associate smaller blobs, which could represent hands or heads, to the nearest person object. Blob splitting handles partial overlapping of near standing persons, depending on the relation of the current region size to the previously detected regions. The position of the final person regions specify the area of the current image where the face detection is applied. Detected faces are then assigned to the person regions, which results in face-region pairs. Now the tracks of each already existing object can be updated. This includes the region positions, the face coordinates and the motion features. For all new objects that could not be matched to persons from previous frames, new person objects are created. After the person tracks are updated with the current regions and faces estimation of the face pose can be performed. If the face detection method does not provide a face for a particular person in the current measurement, it is assumed that the face is still located at the previously detected position. After the face pose of all detected persons is calculated, follows estimation of which person is being currently observed is performed. This is done knowing the current person positions and their face orientation. If the current tracking history segment of each person contains enough data then action recognition can be performed. The sequences observe at about 1.8 seconds image material of 30 fps frame rate acquisition. The recognized action of the current tracking history segment is then also stored for each individual person. If person objects are just false detections in individual frames they are marked and deleted. This is implemented to serve as a basis error correction.

The prototype is implemented as source for Matlab 7.10.0 (R2010a). Matlab is untyped interpreted language and it is slow. The object system is possibly less advanced as than those of C++ or Java. However as a scripting language it is very suitable for rapid prototyping. Matlab code is also high level and can be easily read. Excellent numerical algorithms, such as SVD - *singular value decomposition* which we apply in the principal component analysis are provided. Various data visualization and plotting tools offer convenient ways to represent the results. Comparisons between Matlab and other interactive languages can be found online [31, 10]. The coarse performance estimation of the presented approach, including the tracking, face pose estimation and action recognition without screen visualizing of the tracking results lies by 4.6 seconds per image frame on an Intel Penryn - Celeron processor 2x1.7 GHz. The optimization of the developed code or the implementation in other programming language is not a research topic and will be task in future work.

# 6 Conclusion and Future work

In this work, a recognition system for situations in group meetings was developed. For this purpose videos of prerecorded meetings were used and a tracking algorithm was applied to identify persons. A standard face detection algorithm was employed and global motion features, like the center of mass or the relative position of the face with respect to the bounding box, were calculated. Additionally, a face pose estimation was implemented with the goal of identifying the speaker. This information can be useful to recognize situations like discussions with alternating speakers.

The initially intended situation recognition could not be realized in the coarse of this work due to the limited time. However the implementation of a classification stage which employs the individual actions, as well as information about the focus of attention, can be implemented quite easily. The goal of this thesis was the development of a system for recognizing activities in meetings without integrating too many assumptions about the nature of meetings. Although this could hare improved the recognition rate, it was considered more important to develop a generic system with reusable components. However, it has to be admitted that the global motion features seem rather specific. Their applicability to the recognition in other scenarios will have to be investigated.

The correct recognition rates of the developed system seem promising with rates for individual activities between 62% and 100%. The recognition rates could be improved by implementing more sophisticated methods for all steps in the recognition process, however, the goal was the development of a generic system with only few assumptions and no dependencies on context information. Considering this requirement the achieved results are acceptable.

## 6.1 Future work

The results from this work can be used for further development in the area of person tracking and action recognition. Here the main subtasks of the overall problem of situation classification won on clarity and structure, so that further experiments and analysis can be done. The presented methods are based on the fundamentals described in the previous chapters. The tracking approach can be extended with probabilistic estimations of the extracted features, which could help to make predictions about the performed motion. This can also help to increase the accuracy of the blob clustering, splitting and eventually of the action recognition. Extending the face pose training set would help to handle face pose under changing lightning conditions. To achieve better results in the face pose estimation and recognition of more pose classes the face projection patterns can be used as input for more sophisticated classification methods than the simple distance based metrics. Synchronization of video data obtained from multiple cameras will eventually help to handle occlusions and to refine the estimation of focus of attention. In order to obtain high level situation information such as classification of

group activities, not only refinement and improvement of the applied methods but also high amount of training data, which are also competently managed, are required.

The developed technique can be extended to increase the number of reasonable activities. The applicability of the developed features to the recognition of other activities will have to be investigated, as well as the development of additional features. Additionally, the activity recognition, i.e. tracking and motion detection algorithms will have to be extended in the future to deal with more complex meeting situations with non-constant conditions, i.e. real life situations. The goal of future work in the area of meeting recognition will also be the adoption of these concepts to recognition task outside of meeting rooms.

# Bibliography


[1] http://petsicvs.visualsurveillance.org/. (Cited on pages 9, 10 and 11)

[2] http://www.amiproject.org/newsletter/issue-20/. (Cited on pages 10 and 31)

[3] http://www-prima.imag.fr/FGnet/. (Cited on page 10)

[4] http://www.noulo.net/index.php?option=com_content&view=article&id=37&Itemid=60&af9448e7c3876ad3024976acabe1be11=1f5fafb1988c981201d2fb9b71ebe18c. (Cited on pages 24 and 61)

[5] http://www.codeproject.com/KB/audio-video/haar_detection.aspx. (Cited on page 26)

[6] http://www.brain-spine.com/cervical_activities/six_movements_of_the_neck.html. (Cited on page 31)

[7] http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/Models/asms.html. (Cited on page 36)

[8] http://www.consortium.ri.cmu.edu/projAAMCyl.php. (Cited on page 37)

[9] http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html. (Cited on page 50)

[10] http://www.mathworks.com/help/techdoc/matlab_oop/brqzfut-1.html. (Cited on page 63)

[11] S. Kang; J. Paikab; A. Koschana; B. Abidia; M. A. Abidia. Real-time video tracking using ptz cameras. *Proc. of SPIE 6th International Conference on Quality Control by Artificial Vision*, Vol. 5132:pp. 103–111, 2003. (Cited on page 13)

[12] S. Elhabian; K. El-Sayed; S. Ahmed. Moving object detection in spatial domain using background removal techniques - state-of-art. *Recent Patents on Computer Science*, pages 32–54, 2008. (Cited on pages 15 and 16)

[13] E. Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. Mit Press; 2nd Revised edition, 2010. (Cited on pages 28, 40 and 53)

[14] A. Baumberg. *Learning Deformable Models for Tracking Human Motion*. PhD thesis, University of Leeds, 1995. (Cited on page 15)

[15] S. Bernstein. Entwicklung einer grafischen oberfläche zum browsen und labeln von trainingsbildern. Master's thesis, Fachhochschule Mittweida (betreut an der Universität Stuttgart), 2010. (Cited on pages 43 and 44)

[16] S. Herrero; J. Bescós. Background subtraction techniques: Systematic evaluation and comparative analysis. *J. Blanc-Talon et al. (Eds.): ACIVS*, LNCS 5807:33–42, 2009. (Cited on page 15)

[17] D. J. Beymer. Face recognition under varying pose. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 756–761, 1994. (Cited on page 32)

[18] Y. Bar-Shalom; W. D. Blair. *Multitarget-Multisensor Tracking: Applications and Advances*. Artech House Inc, 1990. (Cited on page 15)

[19] M. Isard; A. Blake. Condensation-coditional density probagation for visual tracking. *Int'l J. Computer Vision*, Vol. 29, No. 1:5–28, 1998. (Cited on page 15)

[20] D. S. Bolme. Elastic bunch graph matching. Master's thesis, Colorado State University, 2003. (Cited on page 35)

[21] S. Srinivasan; K. Boyer. Head pose estimation using view based eigenspaces. *Proc. 16th Int'l Conf. Pattern Recognition*, 4:302–305, 2002. (Cited on page 34)

[22] L. G. Brown. A survey of image registration techniques. *ACM Computer Survey*, pages 325–376, 1992. (Cited on page 15)

[23] S. Brutzer. Background subtraction in der videoüberwachung. Master's thesis, University of Stuttgart, 2010. (Cited on pages 16 and 20)

[24] P. Yao; G. Evans; A. Calway. Using affine correspondence to estimate 3-d facial pose. *Proc. IEEE Int'l Conf. Image Processing*, pages 919–922, 2001. (Cited on page 35)

[25] C. H. Chen. *Handbook of Pattern Recognition and Computer Vision*. World Scientific Publishing Co., 2010. (Cited on page 15)

[26] A. H. Gee; R. Cipolla. Determining the gaze of faces in images. *Image and Vision Computing*, 12:639–647, 1994. (Cited on page 35)

[27] M. S. Arulampalam; S. Maskell; N. Gordon; T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, Vol. 50, No. 2, 2002. (Cited on page 15)

[28] F. C. Crow. Summed-area tables for texture mapping. *Proceedings of the 11th annual conference on Computer graphics and interactive techniques SIGGRAPH '84*, Volume 18 Issue 3, 1984. (Cited on page 26)

[29] N. Gourier; D. Hall; J. L. Crowley. Estimating face orientation from robust detection of salient facial structures. *Proceedings of Pointing 2004, International Workshop on Visual Observation of Deictic Gestures - ICPR*, 2004. (Cited on pages 11, 12 and 38)

[30] L.-P. Morency; C. Sidner; C. Lee; T. Darrell. Contextual recognition of head gestures. *Proceedings of the 7th international conference on Multimodal interfaces, ICMI'05*, 2005. (Cited on page 31)

[31] H. Fangohr. A comparison of c, matlab and python as teaching languages in engineering. `http://eprints.soton.ac.uk/22811/1/Fang_04.pdf`. (Cited on page 63)

[32] C. Sforza; G. P. Grassi; N. Fragnito; M. Turci; V.F. Ferrario. Three-dimensional analysis of active head and cervical spine range of motion: effect of age in healthy male subjects. *Clinical Biomechanics*, 17:611–614, 2002. (Cited on page 31)

[33] I.H Witten; E. Frank. *Data Mining : Practical Machine Learning Tools and Techniques*. Elsevier, 2005. (Cited on page 23)

[34] R. Frischholz. `http://www.facedetection.com/facedetection/datasets.htm`. (Cited on page 38)

[35] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2 edition edition, 1990. (Cited on page 24)

[36] B. Shoushtarian; N. Ghasem-aghaee. A practical approach to real-time dynamic background generation based on a temporal median filter. *Journal of Sciences, Islamic Republic of Iran*, 14(4):351–362, 2003. (Cited on pages 19 and 20)

[37] S. McKenna; S. Gong. Real-time face pose estimation. *Real-Time Imaging*, vol. 4, no. 5:333–347, 1998. (Cited on page 34)

[38] J.E. Jackson. *A user's guide to principal components*. John Wiley & Sons, 2003. (Cited on page 39)

[39] F. Jelinek. *Statistical Methods for Speech Recognition*. Mit Press, 1997. (Cited on pages 54 and 55)

[40] P. Viola; M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on CVPR*, 1, 2001. (Cited on pages 24 and 27)

[41] G. Bradski; A. Kaehler. *Learning OpenCV Computer Vision with the OpenCV Library*. O'Reilly Media, 2008. (Cited on pages 26 and 29)

[42] S-C. Cheung; C. Kamath. Robust techniques for background subtraction in urban traffic video. *In S. Panchanathan and B. Vasudev, editors, Proc Elect Imaging: Visual Comm Image Proce 2004 (Part One)*, Vol. 5308:881–892, 2004. (Cited on pages 16 and 19)

[43] H. Rowley; S. Baluja; T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, Issue 1:23, 1998. (Cited on page 32)

[44] H. Rowley; S. Baluja; T. Kanade. Rotation invariant neural network-based face detection. *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 963, 1998. (Cited on page 32)

[45] P.-N. Tan; M. Steinbach; V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006. (Cited on page 24)

[46] W. Hu; T. Tan; L. Wang; S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Volume 34, Issue 3:334, 2004. (Cited on page 13)

[47] R. Lienhart; J. Maydt. An extended set of haar-like features for rapid object detection. *Proc. International Conference on Image Processing*, 1, 2002. (Cited on page 25)

[48] J. Sherrah; S. Gong; E.-J. Ong. Understanding pose discrimination in similarity space. *10 th British Machine Vison Conference, BMVA Press*, pages 523–532, 1999. (Cited on page 38)

[49] J. Sherrah; S. Gong; E.-J. Ong. Face distributions in similarity space under varying head pose. *Image and Vision Computing*, vol. 19, no. 12:807–819, 2001. (Cited on page 32)

[50] Face Recognition Home Page. `http://www.face-rec.org/databases/`. (Cited on page 38)

[51] M. Turk; A. Petaland. Eigenfaces for recognitio. *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, 1991. (Cited on pages 31, 36 and 39)

[52] M. Piccardi. Background subtraction techniques: a review. *IEEE International Conference on Systems, Man and Cybernetics*, pages 3099–3104, 2004. (Cited on page 15)

[53] C.P. Papageorgiou; M. Oren; T. Poggio. A general framework for object detection. *Sixth International Conference on Computer Vision*, page 555, 1998. (Cited on page 25)

[54] Pointing'04. `http://www-prima.inrialpes.fr/Pointing04/data-face.html`. (Cited on pages 11, 12 and 38)

[55] S. Blackman; R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House Inc., 1999. (Cited on page 15)

[56] R. Cucchiara; M. Piccardi; A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25, NO. 10,:1337–1342, 2003. (Cited on page 17)

[57] W.K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. John Wiley & Sons, fourth edition edition, 2007. (Cited on page 21)

[58] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, Vol: 77 , Issue: 2:257, 1989. (Cited on pages 52 and 54)

[59] A. T. Rasmussen; H. M. Rasmussen. Tracking people in sports using video analysis. Master's thesis, Technical University of Denmark, 2008. (Cited on page 15)

[60] M. Zobl; F. Wallhoff; G. Rigoll. Action recognition in meeting scenarios using global motion features. *Proc. of Fourth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-ICVS)*, Ferryman, J.:. 32-36. University of Reading, UK., 2003. (Cited on pages 47 and 48)

[61] R. Rae; H. Ritter. Recognition of human head orientation based on artificial neural networks. *IEEE Transactions on Neural Networks*, Vol. 9, No. 2:257–265, 1998. (Cited on page 33)

[62] A. P. Dempster; N. M. Laird; D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society, Series B*, Vol.39, No.1:1–38, 1977. (Cited on pages 54 and 55)

[63] D. Meintrup; S. Schäffler. *Stochastik:Theorie und Anwendungen*. Springer Verlag, 2005. (Cited on page 51)

[64] N.J.B. McFarlane; C.P. Schofield. Segmentation and tracking of piglets in images. *Machine Vision and Applications, Springer-Verlag*, 8:187–193, 1995. (Cited on page 19)

[65] A. Yilmaz; O. Javed; M. Shah. Object tracking: A survey. *ACM Comput. Surv. 38, 4, Article 13*, 2006. (Cited on pages 13 and 14)

[66] N. T. Siebel. *Design and Implementation of People Tracking Algorithms for Visual Survaillance Applications*. PhD thesis, The University of Reading, 2003. (Cited on page 15)

[67] R. O. Duda; P. E. Hart; D. G. Stork. *Pattern Classification: Pattern Classification Pt.1*. John Wiley & Sons, second edition edition, 2000. (Cited on page 50)

[68] T.F. Cootes; G.J. Edwards; C.J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, Issue: 6:681, 2001. (Cited on page 35)

[69] F. Brémond; M. Thonnat. Tracking multiple non-rigid object in a cluttered scene. In *In Proc. of the Scandinavian Conference on Image Analysis (SCIA'97)*, 1997. (Cited on page 14)

[70] E. Murphy-Chutorian; M. Trivedi. Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. *Proc. 10th Int'l IEEE Conf. Intelligent Transportation Systems*, pages 709–714, 2002. (Cited on page 33)

[71] E. Murphy-Chutorian; M.M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 4:607–626, 2009. (Cited on page 31)

[72] M. Jones; P. Viola. Fast multi-view face detection. Technical report, Mitsubishi Electric Research Laboratories, Inc., 2003. (Cited on page 33)

[73] R. Stiefelhagen; J. Yang; A. Waibel. Modeling focus of attention for meeting indexing based on multiple cues. *IEEE Transactions on Neural Networks*, Vol. 13, No. 4:928–938, 2002. (Cited on page 10)

[74] R. C. Gonzalez; R.E. Woods. *Digital Image Processing*. Pearson Prentice Hall, third edition edition, 2008. (Cited on pages 14 and 21)

[75] R. Xu; D. C. Wunsch. *Clustering (IEEE Press Series on Computational Intelligence)*. John Wiley & Sons, 2009. (Cited on pages 23 and 24)

Alle URLs wurden zuletzt am 14.02.2011 geprüft.

**Erklärung**

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

---

(Antoniya Tyaneva)