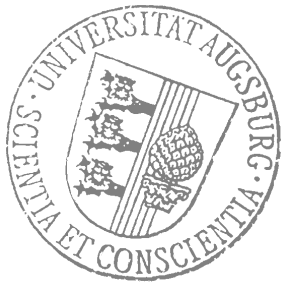
Diplomarbeit

Learning People's Appearances from Multiple Views

Jochen Lux

Fakultät für Angewandte Informatik
Universität Augsburg



Supervisor: Prof. Dr. Rainer Lienhart

Augsburg, August 2006

Zusammenfassung

In diesem Dokument wird ein System zur Erkennung von Personen in Bilddaten vorgestellt. Es basiert darauf, das äußere Erscheinungsbild einer Person aus Bildsequenzen zu lernen. Dabei wird ein statistisches Modell von jeder Person erstellt, die im Folgenden erkannt werden soll. Für die Beschaffung von positiven Trainingsdaten wird ein verteiltes Netzwerk von Videokameras verwendet, welches in einem Raum in verschiedenen Höhen und Positionen angebracht wurde. Die aufgenommenen Videodaten werden automatisch mit dem Namen der gezeigten Person versehen, um den Grad an manueller Interaktion seitens des Benutzers möglichst gering zu halten.
Im Anschluss daran wird eine Vordergrund-Hintergrund Segmentierung durchgeführt, um das Aussehen einer Person unabhängig vom dargestellten Hintergrund lernen zu können. Verschiedene lokale Merkmalspunkte werden für den Bildvordergrund extrahiert und zu einem globalen Bildmerkmal kombiniert. Dieses globale Bildmerkmal wird im Verbund mit dem Personennamen verwendet, um einen Trainingsdatensatz zu erstellen.
Die Elemente dieses Datensatzes dienen als Eingabedaten für ein maschinelles Lernverfahren, welches nach Abschluss der Trainingsphase ein statistisches Modell der zu erkennenden Person ausgibt. Nachdem ein solches Modell für jede Person erstellt und dem Klassifizierer übergeben wurde, können Testbilder mit Personennamen versehen werden. Da eine automatisierte Trennung des Vordergrundes vom Hintergrund bei statischen Bildern ohne a priori Wissen über den Bildinhalt nicht möglich ist, werden vom Klassifizierer Teile des Testbildes sukzessive bezüglich ihrer Ähnlichkeit mit den gelernten Personen bewertet. Nach Beendigung des Algorithmus wird eine Gesamtklassifikation erstellt und zusammen mit einer Schätzung über die Position der Person im Testbild zurückgegeben.
Experimentelle Ergebnisse bescheingen dem gewählten Ansatz gute Erkennungsraten.

Abstract

In this document, a system for recognizing persons in images is proposed. It is based on learning the outer appearance of a person from image sequences. In order to accomplish this, a statistical model of every person, we want to recognize, is being learnt. Positive training data is acquired by using a distributed camera network. Video data is labeled automatically to minimize user interaction. Foreground segmentation is performed to seperate the person of interest from the background. Various features are extracted and combined to form a training data set for each individual. Person recognition is performed by using a multi-scale classifier, that iteratively classifies image parts to create an overall recognition result. Finally, the position of a person in the image is estimated. Experimental results show the system to perform well.

Contents

Chapter 1

Introduction

Digital cameras, camcorders and webcams have lately become a part of modern life. The growing amount of digital media data raises the question, how images can be organized efficiently. A valuable source of information for classification and later recovery of an image are the identities of the persons, it displays. The conventional way to build such a system, would be to label the image data manually. This is inefficient and tiresome. In order to overcome this discomfort, we propose an approach, that is able to recognize people in images automatically, based on their outer appearance.
The unconstrained recognition of persons is generally faced with a number of challenges. The basic problem is, that the similarity between images of different people under the same conditions is often bigger than between images of the same person under different circumstances. Such conditions can be illumination parameters, the background of a scenery or the pose and clothes of a person.
To be able represent the outer appearance of a person despite those difficulties, a statistical model is created for each individual, that shall be recognized. This approach induces the assumption, that the appearance of a person does not change significantly between the learning and the recognition phase. It is therefore reasonable to gather a huge amount of training data, that shows the person in many different situations, in order to cover the variability in visual appearance and to represent the individual realistically. The training data has to be labeled with the person it displays and stored efficiently.
We decided to collect our training data by using a visual sensor array consisting of cheap webcams, which are installed in a room at various positions and heights. The room is mostly used by the person, whose model shall be learnt and contains no other moving objects. Therefore, the foreground of the captured videos can be segmented to ensure, that the person's model is learnt independently of the background it displays. This data acquisition approach has a number of advantages: first of all, training data can be acquired non-intrusive and labeled automatically, resulting in minimal user effort. Second, the scenario is a straightforward way to gather training data over a long period of time, thereby obtaining variance in

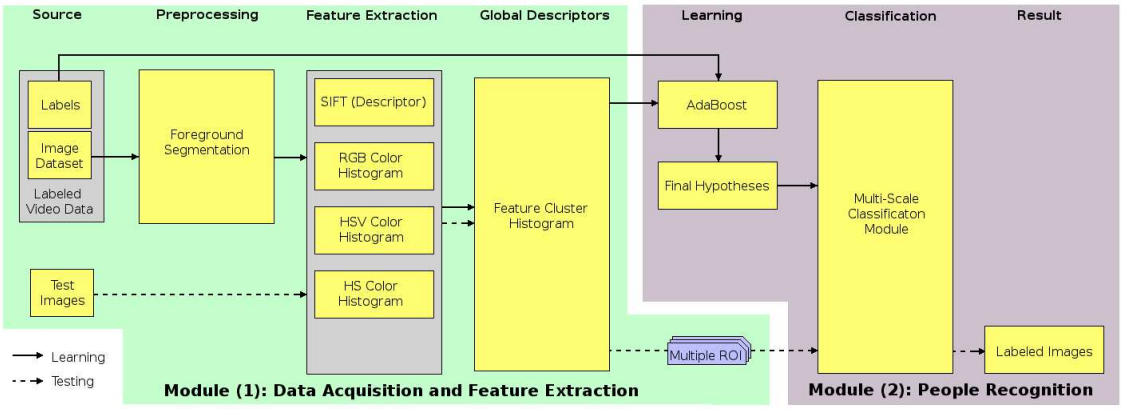


Figure 1.1: System outline

the appearance of a person. This effect is additionally amplified by the use of multiple cameras.
There has been some research on the subject of person recognition in images or video sequences. A widely spread approach to accomplish this is face recognition ([18], [9]), which only performs well for frontal faces and is sensitive to expression and illumination changes. In order to increase recognition rates, face recognition should be combined with a statistical model, that has been created by an appearance based approach, like the one presented in this document. Alternative recognition systems, that rely on people's appearances, have been proposed by Nakajima et al. ([1]) and Hähnel et al. ([15]). Both approaches assume, that people can be segmented in the learning and in the recognition phase, which simplifies the task. Furthermore, images of persons, that have not been learnt, cannot be handled by both approaches.

The system, we propose in this document, is partitioned in two modules: Data acquisition and feature extraction (1) and people recognition (2). The system is outlined in figure 1.1. Initially, video frames, that are captured by the visual sensor array are passed on to the foreground segmentation algorithm. After the background subtraction has been performed, the image and an associated segmentation mask is handed over to the feature extraction module. Depending on parameter settings, basic features are extracted for the image area, where the person is displayed. After that, a global feature is calculated and forwarded to the learning algorithm.
In combination with the assigned label, each global feature is used as a positive training sample. After the learning phase is completed, a statistical model of the person is handed over to the classification module. This procedure is iterated, until a model has been created for every person, that shall be recognized.
At the beginning of the recognition phase, features are being extracted from the entire image, as foreground segmentation is not possible for a static image without a priori knowledge of the image's contents. Next, global features are iteratively

calculated for image parts and passed on to the classification module. Each global feature is classified with every person's model. A decision is made, which person is displayed on the overall image. Finally, the classified image is returned to the user, including an estimation of the recognized person's position in the image.
The document is structured as follows: Chapter 2 describes data acquisition and the implementation of our distributed camera network. Chapter 3 presents basic and global feature extraction and explains its theoretical basis. In chapter 4, the machine learning algorithm and our multi-scale classification module are introduced. Chapter 5 shows experimental results and chapter 6 concludes the paper with a short summary and by stating directions of future research.

Chapter 2

System Design for Data Acquisition

The description of a person's appearance depends strongly on the perspective of the observer. In order to induce robustness to so-called "out-of-plane rotations" (i.e. tranformations, that rotate an object along a three-dimensional axis) and to add variability to our training data (e.g. different lighting conditions and sensor hardware), we created a network of visual sensors for data acquisition. These sensors are used to collect a vast amount of positive training data from several persons over a long period of time with the intention of learning a statistical classification model for each test person. In order to get multiple perspectives of the same person at the same time, the cameras are installed in a room at various positions and heights. Video recording can be controlled from a single computer, the master. Before the video capturing process is started, the user is requested to input his name. In combination with the assumption, that the person of interest is the only moving object in the video, it is not necessary to label the training data manually, thus ensuring minimal user effort.
In this chapter, the system design of a distributed camera network for data acquisition is presented and our implementation is described in further detail.

2.1 Distributed Camera Network

The distributed network used for capturing training data consists of multiple visual sensors. There exits no upper limit for the number of cameras in the network. However, due to the amount of computing power necessary for video encoding and due to camera driver restrictions, only two cameras are connected to each PC. The PCs are linked by using an ethernet network interface, thereby ensuring the platform's profound scalability. Camera control is accomplished by using the Intel Universal Plug and Play (UPnP) technology (e.g. [16], [19] and [4]) as a communication protocol.
We chose UPnP for the following reasons: Firstly, UPnP features a distributed,

transparent networking architecture. In this context, transparency describes the architecture's ability to identify resources independently of both the user's location and the resource location. A second advantage of UPnP is it's independence of platform, protocol, device and programming language. It is therefore possible to use any visual sensor or personal computer hardware, as long as a camera driver and a UPnP protocol implementation is available for the platform in question. Thirdly, UPnP is a widely accepted industry standard, which ensures community support and prevents the implementation from being out-of-date in the near future. Finally, the UPnP architecture is based on internet related communication techniques, is therefore reliable and can easily be integrated in an existing networking environment.
Our implementation focus is put on minimal user effort while retaining maximal flexibility. Hence, a control program is installed on a single network PC, the master. There is no other precondition for the choice of the specific master except that the PCs with the camera devices installed have to be reachable on the UPnP network ports. In computer networking, ports are typically used to map data to a particular process running on a computer. Client PCs with installed camera devices are refered to as *"UPnP devices"* and run a background process, that multicasts their ID and installed *"UPnP services"* when joining the network. In our implementation, UPnP services represent the visual sensors, that are connected to each UPnP device. The master PC (a *"UPnP control point"*) receives these UPnP messages and sends commands to each of the registered devices on user request.
The UPnP architecture defines several protocols for the communication between the control point and the UPnP devices. The following paragraphs give a brief review of the UPnP networking architecture by introducing the steps, that are performed during program execution. We will furthermore present the UPnP protocol stack, which shows the abstraction levels, that are used by the UPnP architecture. Furthermore, significant differences in our approach with respect to the standard implementation as well as necessary customizations are marked in the related paragraph.

2.1.1 The UPnP Communication Process

First, an IP address is assigned to all UPnP devices via DHCP or AutoIP. The UPnP communication procedure implies two consecutive steps, Discovery and Description, and then allows for three processes (Control, Eventing and Presentation) to iterate on demand. This is illustated in table 2.1 and is described in further detail below.

Discovery		
Description		
Control	Eventing	Presentation

Table 2.1: UPnP Communication Process with two consecutive steps (Discovery, Description), followed by three processes, that are called on demand (Control, Eventing and Presentation)

Discovery

The first step in the UPnP communication process is the discovery of UPnP devices by the control point. When a device is added to the network, it advertises its services by multicasting a Simple Service Discovery Protocol (SSDP) message on a specified network channel and port (239.255.255.250:1900). Every UPnP device may act as multiple devices, each of which having the same functionality as a real device. Control points in the network cannot distinguish between a virtual device and a real one. If a device contains more than one logical device or service, a number of messages is sent to cover the full extent of the device's capabilities.Discovery messages typically contain the device's or service's UPnP type, a unique identifier, an URL to the device's UPnP description and a duration, after which the advertisement expires. If a device becomes unavailable, a `ssdp:byebye` message needs to be sent, otherwise it will be assumed available until the advertisement's expiry has been reached.
If a control point is added to the network, it scans for services or devices of interest by multicasting a message including a search pattern. The responses are a unicast version of the messages sent by newly connected devices.

Description

After a successful discovery, the control point needs to gather information about a device's interface to interact with the advertised services. The UPnP standard uses two different types of descriptions, device descriptions and service descriptions.
A device description contains vendor specific information, such as URLs to the manufacturer's website, serial or model numbers, etc. It also features information about the embedded services, e.g. service type, name and URLs for the service description, control and eventing mechanisms. Device descriptions can contain more than one logical device or service. This is important for our implementation, as each of our devices (a network PC) encloses two services (webcams).
Service descriptions on the other hand include a set of zero or more commands ("*actions*" in UPnP nomenclature). Actions model the interface used by the control point to communicate with the various services. They may have zero or more associated parameters ("*arguments*"), which can be marked as either input

or output variables. One argument can specifically be labeled as a return value. Finally, service descriptions feature a list of variables, that represent the state of a service at run time. They are defined by their data type, data range and event characteristics.
Both descriptions are written in XML syntax and based on a UPnP Device Template or UPnP Service Template (defined by a UPnP Forum working committee), respectively.

Control

Until now, the control point learned about the available devices and services due to the discovery step and obtained the information necessary to invoke actions during the description process. In the control step, commands can be sent to the various services. UPnP uses SOAP (Simple Object Access Protocol) to deliver control messages to devices and return responses back to the control point.
As the deadline for responses of services is 30 seconds (including the expected transmission time) and as availabilty of the control point during execution needs to be preserved, a new thread is created by the control point for every action request. To be able to record videos that are longer than 30 seconds, our implementation abstains from immediately streaming the encoded video back to the control point. After a `startRecording` command, services rather respond with a boolean value, that indicates the beginning of a recording, and store the videos on a network share. Additionally, a UPnP event could be called after the action has been performed to indicate the successful execution and to submit the video filename (see also section 2.3).

Eventing

In the control section, we dealt with actions and their arguments. The event section focuses on state variables, which we already mentioned in the discovery step. UPnP uses a typical Publisher-Subscriber pattern (an asynchronous messaging paradigm, that allows for better scalability than a naive message passing approach) to keep the control point informed about state changes of services. Each state variable can be declared "envented" in its UPnP service description. Thus, control points are enabled to subscribe for the variable by sending a subscription message to the respective service. If a subscribed state variable changes, the service publishes an update to all interested control points. To accomplish that, the GENA (General Event Notification Architecture) standard is used.
As already mentioned earlier, this method can be used to circumvent the 30 seconds deadline an action response is limited to, in case a command takes a longer time to be performed (such as video recording).

Presentation

This last step, addresses the interaction between the user and the UPnP architecture. The device description may contain an URL, by which devices can be controlled using an internet browser. The current system does not feature a presentation layer. Instead, a user interface is implemented directly on the control point to operate the system.

2.1.2 The UPnP Protocol Stack

The previous sections focused on the description of the UPnP architecture from a functional point of view. This section describes the interaction of various protocols to give an impression, how the described processes are implemented. The six layers of the UPnP protocol stack are shown in table 2.2.

<table>
<tr><td>1.</td><td colspan="4">UPnP Vendor</td></tr>
<tr><td>2.</td><td colspan="4">UPnP Forum</td></tr>
<tr><td>3.</td><td colspan="4">UPnP Device Architecture</td></tr>
<tr><td>4.</td><td>HTTPMU (multicast) [GENA] [SSDP]</td><td>HTTPU (unicast) [SSDP]</td><td>SOAP
HTTP</td><td>HTTP [GENA]</td></tr>
<tr><td>5.</td><td colspan="2">UPD</td><td colspan="2">TCP</td></tr>
<tr><td>6.</td><td colspan="4">IP</td></tr>
</table>

Table 2.2: UPnP Protocol Stack, Layer 1-6

The discovery process uses the SSDP protocol to transport messages. The description step acquires device/service descriptions by HTTP GET requests. In the control process SOAP and HTTP is used for remote procedure calls. As mentioned earlier, the GENA standard is applied for event subscription and notification. Finally, the presentation layer relies only on the HTTP protocol.

2.2 Video Encoding and Storage

The collection of huge amounts of positive training data is a core aspect of our person recognition approach. Therefore we need to store the recorded videos at a central location. In order to keep this amount of data manageable, a video lossy encoding algorithm becomes necessary. In our system, recorded videos are encoded using the DIVX v6.0.3 codec. The DIVX codec is based on the lossy MPEG-4 Part 2 compression standard and provides good compression rates while maintaining high visual quality. It should be noted, that the first three and the last two frames of every video are identical when using the mentioned codec.

2.3 Our Setup

This section describes the setup, as we used it for data acquisition. Our visual sensor array consists of six cheap webcams of the type "Philips SPC 600NC" [3]. The total costs of the array lie between 420$ and 440$. The array is used to collect video data, where each frame has a resolution of 640x480 pixels. The cameras in the network are not synchronized. Figure 2.1 illustrates the different camera perspectives used for gathering video material. Each PC of our distributed camera network is capable of simulaneously encoding two video streams with a rate of 9-12 frames per second. Thus, three PCs are used for our six cameras. Most of the captured videos have varying lengths ranging from 1000 to 3000 frames. To compensate some illumination changes, the cameras use automatic white balancing and automatic brightness control. Each camera is controlled by the PC, it is locally connected to. The connection between the visual sensor and the PC is established using USB 1.1. Extension cords with integrated active repeaters have been used to gain flexibility for the configuration of camera positions. The communication layer between the camera and the UPnP client, called *camera control interface* in the following, has been implemented by using the "cvcam" module of the "Open Computer Vision Library (OpenCV)" (see also [2]). In our implementation, the control point's data structure handling (implemented as a queue of UPnPDevice structs) has been altered due to compatibility problems with the OpenCV camera control interface.

In order to remotely control all camera recording functions from the master PC, our implementation features the following functions:

- The recording of a video with a definable length and beginning.
- The shooting of a settable number of photos with a definable length, beginning and time interval between the shots.
- The requesting of information about the status of the registered devices.

Based on the the functional requirements listed above, device and service descriptions were designed on template basis and include the actions listed in table 2.3.

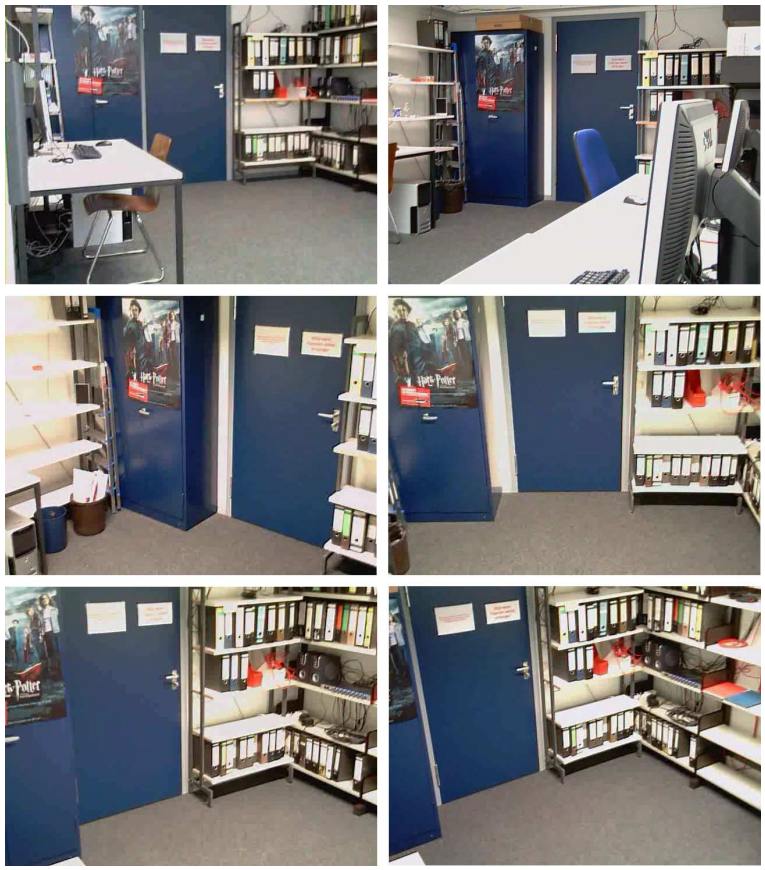


Figure 2.1: Examples of camera perspectives used for data acquisition

Action	Argument Direction	Argument Type	Argument Name
createSnapshot	Input	Integer	start
	Input	Integer	waitingPeriod
	Input	Integer	anzahl
	Input	String	acUserName
	Return	Boolean	success
startRecording	Input	Integer	startTime
	Input	Integer	duration
	Input	Integer	username
	Return	Boolean	success
stopRecording	Return	Boolean	success
getCameraCount	Return	Integer	cameraCount

Table 2.3: Implemented UPnP Actions

Chapter 3

Feature Extraction

As mentioned in the introduction (see Chapter 1), we plan to use the outer appearance of the human body for solving our person recognition problem in images. In order to learn a person's appearance, multiview videos are captured over several weeks. We assume that the overall training data we obtain for each person covers the actual variability in visual appearance of the individual. In order to extract features describing the appearance of the persons body, each captured video from our camera array is forwarded to the preprocessing module (see Figure 1).
Videos enable the segmentation of moving objects, representing persons to be learnt, by foreground segmentation methods. The applied foreground segmentation method is described in Section 3.1. The algorithm computes a binary mask indicating which parts of each frame represent foreground, i.e. the location of the person, and which parts represent background, i.e. parts of the video frame we are not interested in.
Features which describe a person are subsequently computed to describe the segmented regions representing the person to be learnt. Therefore we first extract local features from the entire image. A set of local features builds a sparse representation of the image content, that only considers some region with designated properties. As those features are extracted from the entire image and we only aim to obtain a description for the pictured person, we need to filter features describing the background. This is done by applying the binary mask obtained by the previously mentioned foreground segmentation algorithm.
The detection of regions of interest, that can be used for computing local features, is described in Section 3.2. Computing different descriptions of those regions of interest is discussed in Section 3.3. We extract color and gradient features. The computed desriptors form our feature set, building the desired description of the person's outer appearance.

3.1 Foreground Segmentation

In order to recognize persons in images, local features, that represent a person, are an important source of information for our approach. To be able to identify these features and use them as positive training data, the person to be recognized has to be separated from the background of a video. Otherwise, the background would be learnt, as it is static. To accomplish that, we apply a foreground segmentation algorithm on each of our videos. The algorithm takes a sequence of video frames as its input and creates a binary mask for every frame of the video, that differentiates between foreground and background using pixel color.
The location, where our videos are recorded, is a room, that is mostly used by the person, whose appearance is to be be learnt and contains no other moving objects. It is furthermore assumed, that only gradual illumination changes occur. To enable the segmentation algorithm to learn the background of a scene, a period of absent foreground is available in the beginning of every video. We require our foreground segmentation algorithm to be able to handle gradual illumination changes.
Basically, two different types of foreground detection algorithms can be distinguished:

- Algorithms, that use a frame-by-frame distance as a measure for segmentation (e.g [10]).
- Algorithms, that build a statistical model of the background and extract the foreground by subtracting the model from the original frame(e.g. [17]).

Our approach uses the algorithm proposed in [6], which falls into the second category. It performed best in the "VSSN 2005 Open Source Algorithm Competition" [11] and is in the following described in further detail.

Algorithm

The algorithm runs through the following steps, iterating step 2-5 for every video frame:

1. Initialization
2. Background subtraction
3. Thresholding the image of differences
4. Postprocessing
5. Updating the statistical model

The background model $B_{xy}(t)$ consists of one Gaussian distribution (uni-modal), which is sufficient for indoor scenes. The model is initialized with the corresponding pixel values at position (x,y) of the first video frame:

$$B_{xy}(0) = I_{xy}(0) \tag{3.1}$$

In the background subtraction step, a difference image $D_{xy}(t)$ is calculated by the pixel-wise subtraction of the background model from the frame N in question:

$$\forall xy : D_{xy}(N) = I_{xy}(N) - B_{xy}(N-1) \tag{3.2}$$

In the second iterative step, a Kapur threshold [7] is applied to each pixel of $D_{xy}(t)$, thereby classifying the corresponding pixel of $I_{xy}(N)$ as either foreground or background. The Kapur algorithm uses the entropy of the image's histogram to determine the balance between false positives and false misses. It is dynamic, as the entropy is different for each frame and will furthermore be addressed as a function t with respect to N.

The classification results in a binary image mask $M_{xy}(N)$, that shows the affiliation of each pixel of $I_{xy}(N)$ either to the set of white foreground pixels or to the set of black background pixels.

$$\forall xy : M_{xy}(N) = t_N(D_{xy}(N)) \tag{3.3}$$

During the next step, we will postprocess the classified binary image. If the background of our video data would be static, the procedure described above would a successful way of thresholding the error between the estimate of an image without moving objects and the current image. In typical (outdoor) scenes however, the background contains small movements, that cannot be handled by a uni-modal distribution and would cause false detections. Therefore, the algorithm incorporates a noise background layer using the thresholded image of differences $M(N)$ as its data basis and a small noise removal filter.
The reason for the separation into two different noise reduction methods is, that a mere filtering approach would also affect small foreground objects. First, very small noise is removed using a morphological operator non linear-filter based method (erosion, dilation). As a second step, the remaining pixels are clustered into neighborhood regions. If the number of pixels in a cluster is beneath a certain threshold, the cluster is automatically added to the noise layer. After the update process, the noise layer is subtracted from the image mask.

The last step of our foreground segmentation is the update of our statistical background model. It is a necessary step to be able to handle gradual illumination changes.

$$B_{xy}(N) = B_{xy}(N-1) \text{ , if } I_{xy}(N) \text{ is foreground} \quad (3.4)$$
$$B_{xy}(N) = (1-\alpha)B_{xy}(N-1) + \alpha I_{xy}(N) \text{ , if } I_{xy}(N) \text{ is background} \quad (3.5)$$

Each pixel is tested for being a member of the background and if so, the background model is being updated. In the update process, a learning rate $\alpha \in [0, 1]$ is used, which defines the model's speed of adaption to changes in the video frames' pixel values. If the value of α is set high, the system will adapt rapidly to changes in the video and therefore classify most of the incoming frames' pixels as background. If alpha nears 0, the model will behave like a statical foreground segmentation algorithm.

Results The results acquired by the foreground segmentation process usually include some noise, resulting from a small part of the background around the person being classified as foreground. On the first glance this seems to indicate problems for the recognition process, because, as already stated above, the classifier could recognize persons merely by the fact of them standing in front of the same background. As the background of the training and the recognition phase usually differs significantly, this should not pose a bigger problem to recognition performance. Figure 3.1 displays a video frame and its associated mask, where the foreground segmentation performed well.

Figure 3.1: Well performing foreground segmentation result; the left image shows the original frame, the right image displays the segmentation mask (foreground is marked in white, background in black

In figure 3.2 and figure 3.3 two further results of the segmentation algorithm are illustrated. The results show clearly, that the algorithm fails in presence of moving shadows. These are classified as foreground.

Figure 3.4 depicts two consecutive frames and their associated segmentation masks. The segmentation errors in the second frame are the result of sudden illumination changes. As mentioned, the background model (update) accounts only for gradual illumination changes. Even in (static) indoor environments, sudden illumination changes may occur due to light switches or camera properties

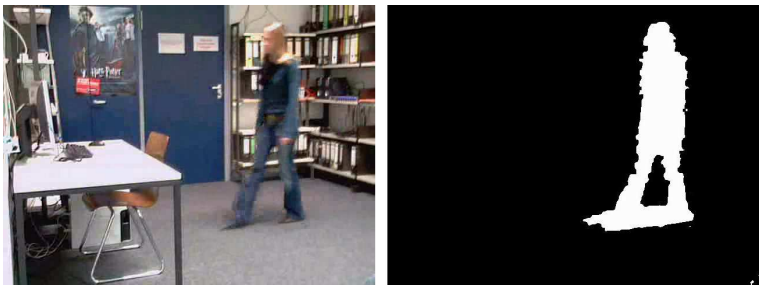


Figure 3.2: Foreground segmentation result displaying unhandled shadows

Figure 3.3: Another example of shadow-related segmentation errors

(automatic illumination control). As a result of a poorly segmented foreground, features of the background would be added to our positive training data set and therefore be learnt as the person, we want to recognize. This leads to poor recognition rates, if the machine learniing algorithm is not able to handle noisy training data.

3.2 Feature Detection

The extraction of features is generally considered a two-step procedure: first, regions of interest have to be identified in each image. We consider as regions of interest highly distinctive local regions in an image, that may be found under a wide range of circumstances. The desciption of those regions of interest then is used as a sparse representation of our image.
In order to detect such regions of interest, the algorithm proposed in [14] is used. This approach has been chosen due to its properties: the detected regions are invariant to image scale and rotaion and partially invariant to illumination changes, changes in 3D-viewpoint and additive noise. This is achieved by detecting regions of interest, so called keypoints. Those regions are transformed into local coordinates, that are invariant to translation, rotation and scale (as well as some other imaging parameters) and describe subsequently the regions of interest in these coordinates (see Section 3.3).
The first step in the detection of keypoints is to select points in location and

Figure 3.4: Example of a sudden illumination change resulting in foreground segmentation errors

scale that may be repeatably detected, thus obtaining invariance regarding those parameters. The necessity of obtaining scale invariance can be illustrated as follows: Real world objects, such as the persons we want to recognize, are generally composed of different structures at different scales. Every real world object has a scale, where its individual features can be learnt with the least amount of effort (the "appropriate" scale). The individual aspects of a mountain, for example, can most efficiently be described at kilometer level, whereas the appropriate scale for learning the features of a wristwatch would have to be much finer. Furthermore, if the object's distance to the camera changes between the learning and the classification phase, a scale invariant description is necessary to achieve a correct classification result. As our person recognition system does not make a priori assumptions about the nature or size of the objects to be recognized, there would be a high probability of missing vital features when using a predetermined scale. In order to achieve scale invariance, we use a scale space representation of the image obtained by convolving the image with a variable-scale Gaussian kernel. Scale space $L(x, y, \sigma)$ is then defined by:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.6}$$

with G being defined as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}. \tag{3.7}$$

An efficient choice to detect points in location and scale repeatably and thus ensuring invariance with respect to those parameters is, to detect peaks in the

difference of Gaussian pyramid:

$$D(x, y, \sigma) \quad = \quad (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

The difference of Gaussian function is a close approximation to the scale-normalized Laplacian of Gaussian, which is truly scale invariant as shown by Lindeberg in [12]. The difference of Gaussian pyramid can be computed efficiently as shown in figure 3.5. The inital image is repeatedly blurred by convolution with the Gaussian kernel producing a set of scale space images according to equation 3.6. To compute the difference of Gaussian images, adjacent scale space images are simly substracted. After each octave, i.e. that the scaling factor σ has doubled its value, the Gaussian image is downsampled by a factor of 2, and so on. The difference of gaussian pyramid (DOG) then contains images, that represent band-pass filtered versions of our original image.

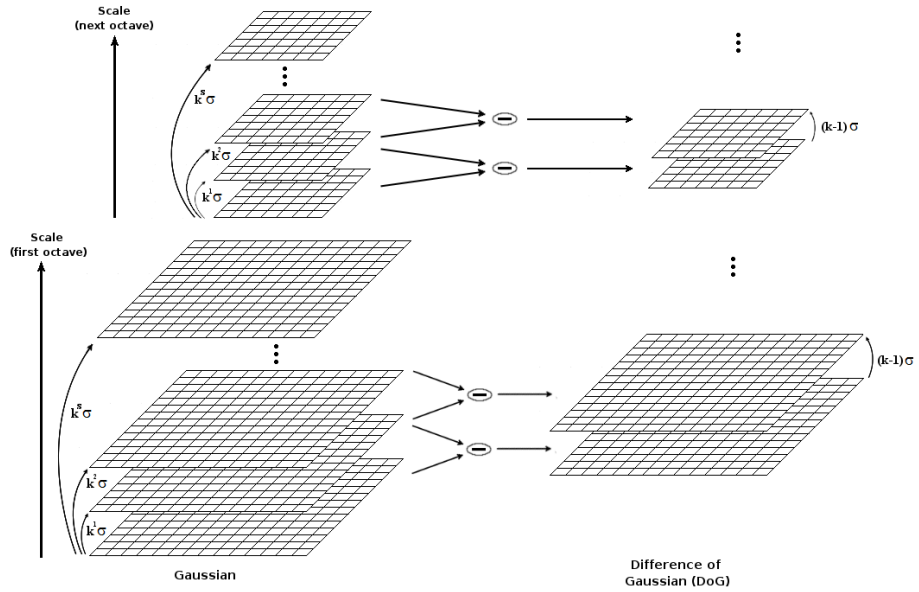


Figure 3.5: Gaussian image pyramid and the associated DoG pyramid, created by the subsequent subtraction of two neigbouring images. $k = 2^{\frac{1}{s}}$ and s denotes the amount of intervals for each octave.

As already mentioned, we can find points in scale and space repeatably by detecting peaks in the DOG. Our first step in feature extraction is the detction of local extrema, i.e. local minima and local maxima, in the DOG pyramid. This is done as shown in figure 3.6 by camparing every point of a DOG image to its 26 neighbours in a 3x3 region around the point in the current, and the two adjacent scales. A candidate location for a keypoint is found, if its value is bigger or smaller than the values of all its neighbours. We defined the scale space as a continous function. Thus, local extrema can be arbitrarily close together. The discretization with the factor $(k - 1)$ in the DOG pyramid might therefore lead

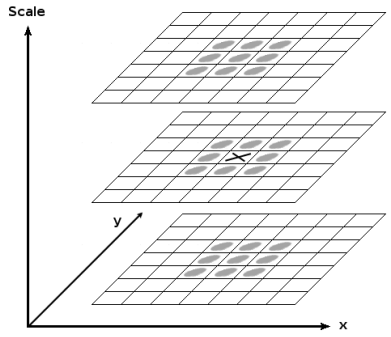


Figure 3.6: Feature Detection via local maxima/minima in the DoG pyramid

to the omission of candidates. But, as experiments showed, such candidate locations turned out to be unstable and are therefore not considered any further. In order to eliminate the remaining unstable candidates, two postprocessing steps are performed.
The first step is to eliminate points with low contrast and to increase the accuracy of our localization. This is performed by fitting a 3D quadratic function to the neighbouring sample points. The accurate candidate location is determined by setting the derivative of the function to 0. If the final candidate location is closer to another sample point than to the original one, the function is fitted again around the closest sample. A threshold is applied to the final candidate to discard keypoints with low contrast.
In the second postprocessing step, candidate locations, that are poorly localized along edges are eliminated. This is necessary, since their location is sensitive to small amounts of noise. Such keypoints can be determined by measuring the principal curvatures in two orthogonal directions. If the ratio between the principal curvatures is greater than a predefined constant, the associated keypoint candidate should be eliminated. This constant is usually chosen to be 10.
Figure 3.7 shows a video frame including the positions and scales of the keypoints, that lie in the frame's foreground. They were detected by the SIFT algorithm.

The last step of the SIFT detector is to ensure rotation invariance, i.e. invariance to in-plane orientations. This can be achieved by calculating a main orientation for every keypoint and expressing its description relative to the assigned orientation. Therefore, gradients at sample points, that surround the keypoint on the same scale are used as a basis of computation. They are weighted by applying a Gaussian window around the keypoint to prevent single values with a big distance from having a dominating effect on the main orientation, thereby ensuring its locality. Finally, a histogram is formed from the weighted gradients and the bin with the largest value taken as the main orientation. Furthermore, for each bin with a value larger than 80% of the maximum, a new keypoint is created and with that value assigned as its orientation.

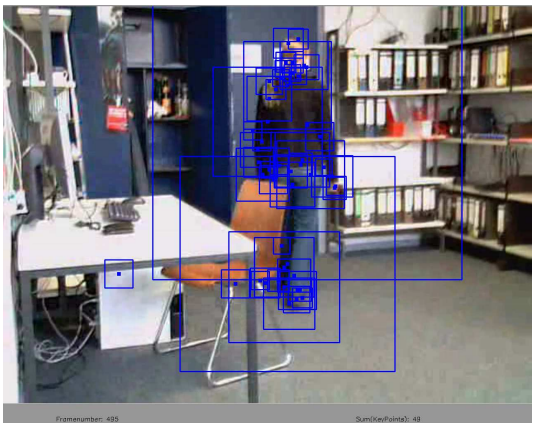


Figure 3.7: Feature point locations scales of s foreground segmented video frame.

3.3 Feature Description

The second step of feature extraction is the description of a previously detected keypoint location. Our approach uses two local basic feature descriptors: the SIFT descriptor and local color histograms. These basic features are used to create higher level features, that rely on histograms of feature clusters and will serve as training examples for our learning algorithm.

3.3.1 Basic Features

The term "basic" refers to the fact, that only raw image data and keypoint locations are required as input values. By describing keypoints relative to the output parameters of our feature detector, scale, shift and orientation invariance are achieved. Apart from these attributes, robustness to out-of-plane rotation and illumination changes are important for the performance of our classification module. Furthermore, as we will cluster our basic feature descriptions in order to create a higher level feature, the distinctiveness of a descriptor has to be balanced to prevent over-specification as well as the opposite. In figure 3.8, two feature locations are depicted with their associated scale. The keypoint in the left image will further on be regarded as "Feature 1", the keypoint on the right as "Feature 2". They will serve as examples to illustrate the structure of our basic feature descriptors.

The SIFT Descriptor

Our first descriptor is part of the SIFT algorithm proposed by D. Lowe (e.g. [13]). It is highly distinctive and partly robust to out-of-plane rotation and illumination changes. The descriptor relies on the orientations of the gradients surrounding a keypoint at a certain scale. The gradients of the sample points that lie within

Figure 3.8: Two examples (left: "Feature 1", right: "Feature 2") of keypoints with their according positions and scales depicted in the video frame. The scale of each keypoint is indicated by the size of its associated rectangle.

a 16x16 array around the keypoint are calculated in the image corresponding to the appropriately collected scale and expressed relative to its main orientation. The orientation has been chosen as described in section 3.2 . After that, they are weighted with a Gaussian, that is centered around the keypoint location to give less weight to gradients, that lie further away from the description center. The next step is to split up the array in 16 square shaped regions, each containing 4x4 elements. An 8-bin histogram, consisting of the gradient's directions, is calculated for every array. The parameters have been chosen to ensure good recognition performance and result in a vector with 128 entries for each feature description. The bin entries are then multiplied with a factor of $(1 - d)$ with d being the relative distance to the bin's center. This ensures a smooth transition from one bin to another. The last step is the normalization of the resulting feature vector to ensure linear illumination invariance. Non-linear illumination changes are reduced by thresholding the largest gradient magnitudes. The description process is visualized in figure 3.9 for an 8x8 vector in order to preserve clarity. The circle represents the convolution with the Gaussian function.

To give an impression of the descriptor, figure 3.10 displays the SIFT features of the keypoint locations depicted in 3.8.

Color Histograms

As described in the previous chapter, SIFT features have been extracted from input images. They are invariant to multiple forms of transformation, such as rotation, translation and scaling, which makes them a valuable source of information for the upcoming recognition task. SIFT features are expressed as the orientations of the gradients, that surround a keypoint at a certain scale. The

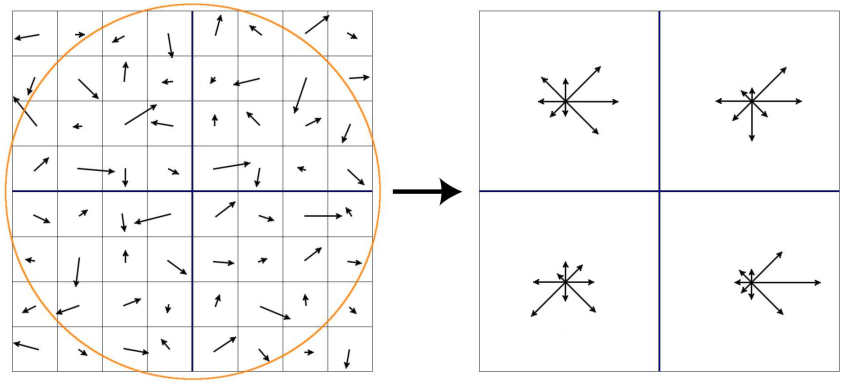


Figure 3.9: Visualization of the SIFT feature description process. It is computed by determining the gradient magnitude for a local image region around the keypoint location. This is shown on the left. In order to give less emphasis to gradients, whose distance to the center is large, a Gaussian window is applied. This is indicated by the circle around the left image. Orientation histograms are then formed from subregions with a size of 4×4. This process is depicted on the right. The arrow lengths correspond to the sum of the gradient magnitudes in the associated image regions.

data basis for their computation is the grayscale information of an image. Unfortunately, the overall appearance of a person significantly depends on colors (e.g. color of clothes or skin). In order to obtain color information from the input images, our approach also uses local color histograms as features.
In contrast to SIFT features, histograms are by definition orientation and shift invariant. Scale invariance is achieved by using the position and scale of the keypoints detected by the SIFT algorithm. These values define the location and amount of pixels for the calculation of a histogram. One disadvantage of using color values as a descriptor is its sensitivity to out-of-plane rotation and illumination changes. We minimize these problems by using an appropriate color space as a computation basis, by normalizing our histograms and by gathering our training data from multiple perspectives. The size of the window we use to gather our input data is square shaped with its size depending on the keypoint's scale factor. The amount of histogram bins are set to 42 for each color channel. The bins are distributed uniformly among the range of input values. After the histogram has been built, it is normalized. Otherwise two histograms with a different amount of input elements would always have a different result. We normalize the histogram bins by scaling them, such that the sum of the bins becomes equal to the minimal amount of input values per histogram.
Color information may define different aspects of a color: for example, colors can be expressed as a compound of the colors red, green and blue with varying

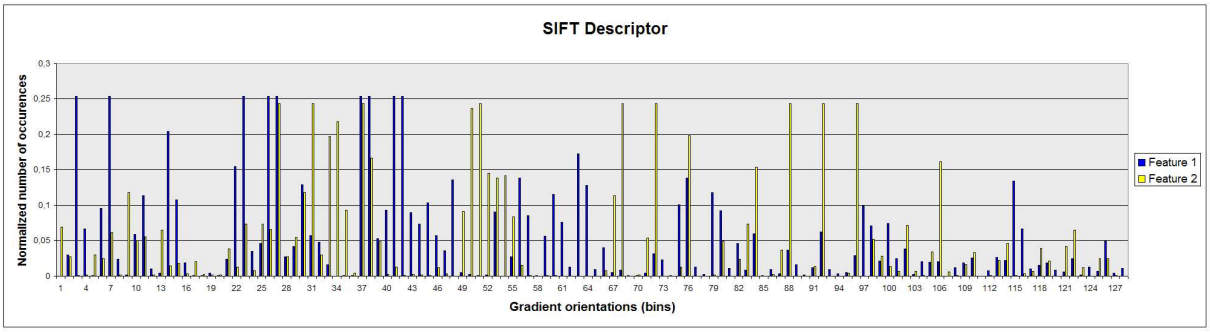


Figure 3.10: SIFT feature descriptions for the keypoints shown in figure 3.8. The feature values are normailzed to 1.

intensities or as a combination of a color's type, brightness and saturation.
The following paragraphs will introduce two different color spaces, which we use for feature extraction.

RGB color space The RGB color space is the standard model for every electronical display, that is based on the addition of various amounts of red (R), green (G) and blue (B) light to produce color. Therefore it is called an *additional color space*. The concept is illustrated in figure 3.11. RGB's use of light sources for color creation and the resulting ease of implementation for display manufacturing has promoted it to be the standard color space for computer imaging tasks. There is also a biological reason for the popularity of the RGB model. The human color perception system is based on three types of photoreceptor cells or *cones*, that measure the intensities of various ranges of wavelengths. The range of wavelenghts each cone perceives can be identified as a color, the so called *primary color*. In the RGB system, red, green and blue are called primary colors, which expresses, that they cannot be created by the mixture of other colors in the RGB color space.

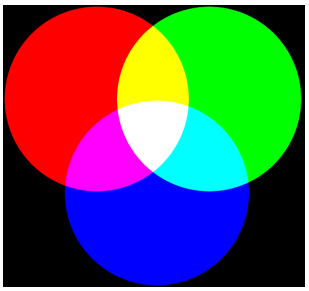


Figure 3.11: Additive color mixing

HSV color space The Hue-Saturation-Value (HSV) color space was invented in 1978 by Alvey Ray Smith. He was inspired by the way artists compound paint and created a model, that shows similarities to human color perception. Hence, it is called a perceptual color space. Today HSV is typically used in computer graphics applications and defined in terms of three individual components:

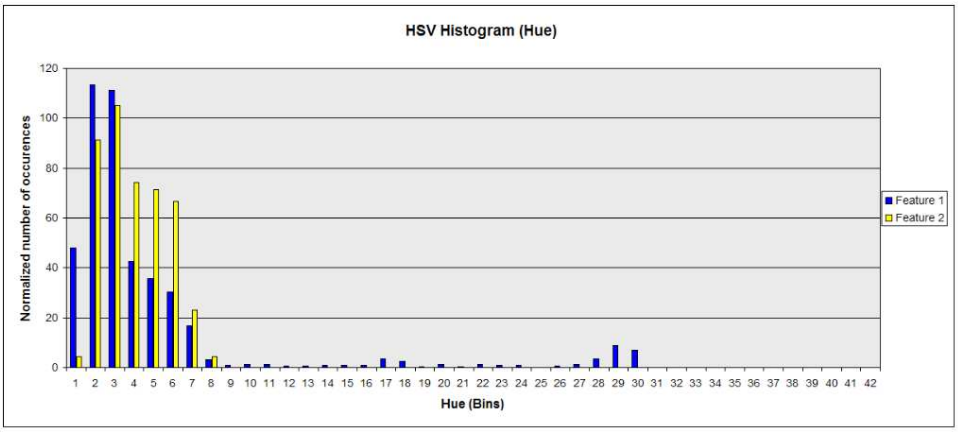
- **H**ue: specifies the color tone (such as yellow or blue) and can be understood as the color's dominant wavelength. If a color is divided in its spectral components, H represents the spectrum's extremum and has the strongest impression on the human eye. It is typically expressed as an angle between 0 and 360 degrees, but normalized to fit $[0, \ldots, 1]$ range in our approach.

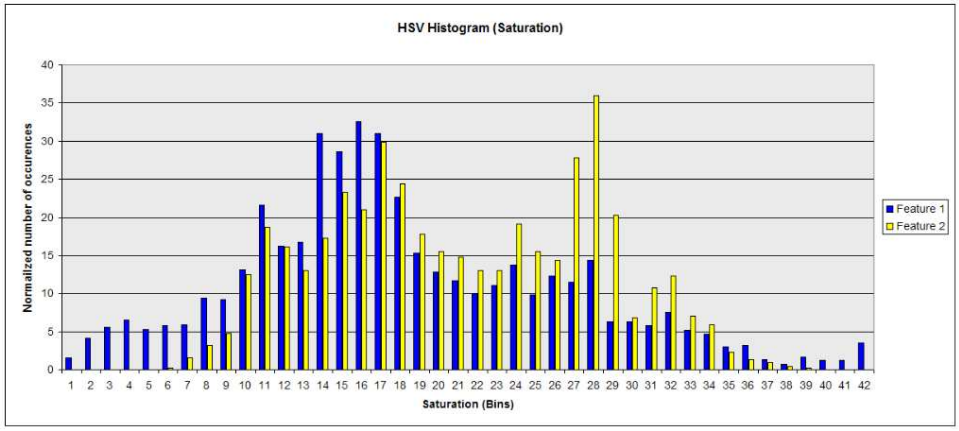
- **S**aturation: this property defines, how faded a color appears. If it is high, the color seems pure and vibrant. As saturation decreases, the color will appear more faded. In physical terms, saturation can be described as the width of the color's wavelength spectrum. A color is entirely desaturated, if the spectrum is distributed equally and fully saturated in the case of an impulse. Its value typically ranges from $[0, \ldots, 1]$.

- **V**alue: describes the lightness of a color. Physically, V corresponds to the integral of the color's wavelength spectrum and can be seen as the amount of light, that hits the eye. It ranges from 0 to 1.

For our approach it is important to note, that the value property (in contrast to hue and saturation) contains no information on the color itself. Hence, it is possible to omit the lightness component when calculating color histograms, thereby achieving robustness to illumination changes. We will furthermore refer to color histograms, that are based on the HSV color space as HS(V) color histograms, with the V being present depending on the use of color lightness information.

In our approach, a histogram is created for each color component seperately. This proceeding trades the disadvantage of the information loss by breaking up the color information for the advantage of higher accuracy, better generalization and, as a result, smaller histograms. The examples shown in figure 3.12 and figure 3.13 substantiate the extraction of color features and refer to the keypoints depicted in figure 3.8. A comparison of the two color histograms shows, that they reflect the visual similarities of the regions, the histograms are based on. To be able to use color component histograms as a local feature descriptor, we merge them in a single vector containing all histogram entries consecutively. Having three color component histograms with 42 bins each, each feature vector consists of 126 entries.





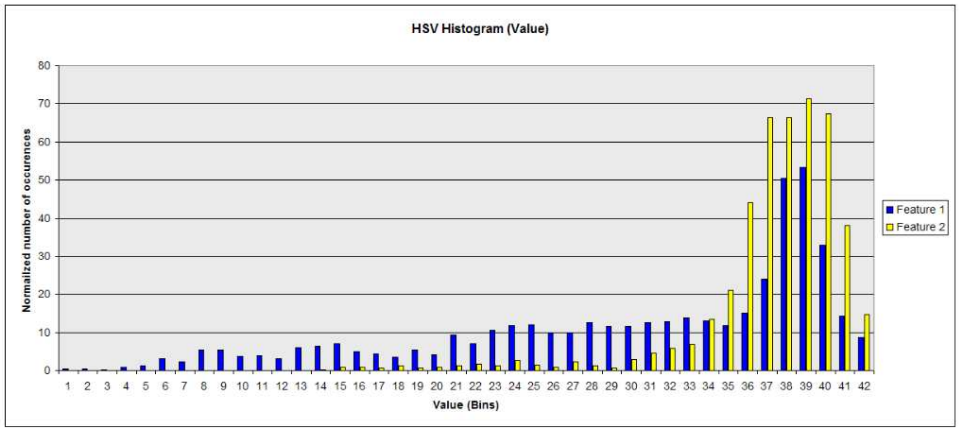


Figure 3.12: Comparison of two local HSV color features describing the keypoints shown in figure 3.8

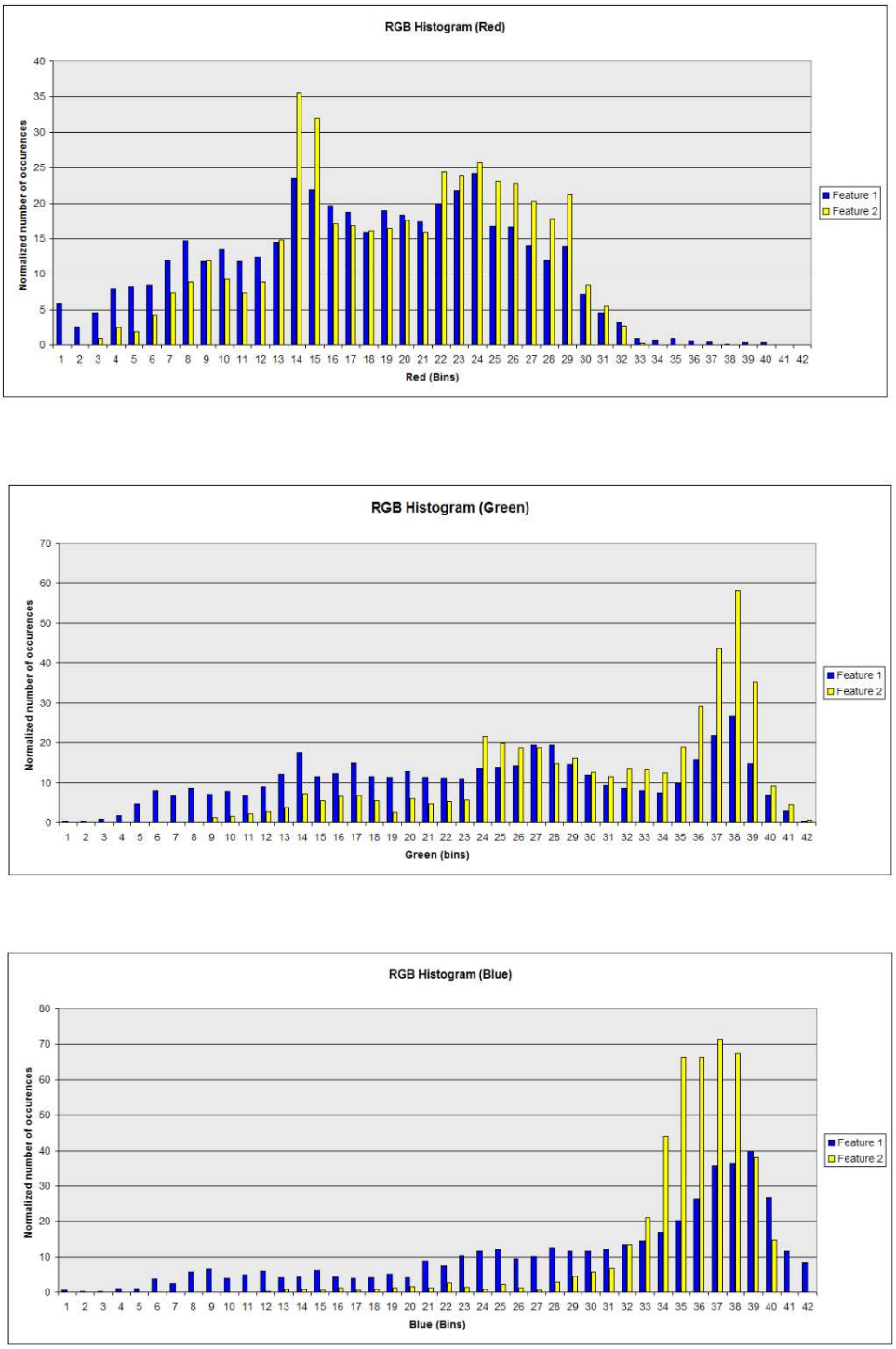


Figure 3.13: Comparison of two local RGB color features describing the keypoints shown in figure 3.8

3.3.2 Feature Clustering

The features we have extracted so far are limited to local regions of the input images. We also need to take advantage of the fact, that the general appearance of a person is defined by a combination of local features. Thus, the intention is to express the presence and correlation of local features in an image as a single vector. To achieve this, a histogram containing the relative frequency of local features is formed for each image and will be used as training data for our classification module.
A core aspect when building a histogram is the choice of bins, by which the input data is partitioned. Hence, it is reasonable to identify further similarities in our data and abstract the orignial feature set. That process is called generalization and can be achieved by data *clustering*.
Clustering is defined as the partitioning of a data set into different classes or "natural groups". Elements, that have been assigned to the same group are called a *cluster*. More precisely, clusters are defined as a set of objects, that bear resemblance to one another and are dissimilar from objects, that are not elements of the set. To measure the total quality of a particular partitioning, a weighting function is used. It typically includes a combination of two measures: the first one is the *"within cluster variation"*, which is a measure for the scatter of all elements within a cluster and should naturally be small. Second, the *"between cluster variation"* quantifies the scatter of all clusters among each other and is supposed to be large.

The K-Means Clustering Algorithm

Our recognition system uses a *"k-means"* algorithm. It was proposed by J.A. Hartigan and M.A. Wong [5] and is a fast (the computational complexity lies in $O(|iterations| \cdot |clusters| \cdot n$, $n = |input\ data|)$ and intuitive algorithm, that outputs disjoint, round clusters. K-means is an iterative method, that is based on *partitional clustering*. The idea of partitional clustering is, to divide the input data in a previously determined amount of clusters. This restriction is reasonable, because any weigthing function would automatically favour higher cluster numbers. As the "within cluster variation" decreases monotonically with the amount of elements in the cluster while maintaining the "between cluster variation", a partitioning with one cluster per element would always be accounted optimal.
Cluster centroids are typically initialized randomly and rearranged, until a quality criterion is met. The random selection of centroids in the beginning of the algorithm may lead to varying results with identical input parameters and data. One possible procedure is then to execute the algorithm multiple times (under the same conditions) and choosing the best result. Figure 3.14 depicts a short description of the algotithm.

In our case, the algorithm's input data consists of all basic features that are used to classify a particular person (both positive and negative samples).

Input F = set of local feature vectors; K = total amount of clusters

Initialization Assign a random point m_k to each cluster C_k, $k = 1, \ldots, K$

While Partitioning quality criterion is not satisfied

1. **For** $k = 1, \ldots, K$ (Every element of the input data is assigned its nearest cluster)

   (a) $C_k = \{x \in F | d(m_k, x) \leq d(m_i, x)\ \forall i = 1, \ldots, K; i \neq k\}$

2. **For** $k = 1, \ldots, K$ (recalculate cluster centroids)

   (a) $m_k = \left\{ \frac{\sum_{j=1}^{J} r_j}{J} | r_j \in C_k, J = |C_k| \right\}$

Output Mapping of F on C_k

Figure 3.14: Pseudocode of the K-Means clustering algorithm

Different kinds of basic features are clustered seperately. The resulting clusters form the bins of our higher order feature histogram.

Feature Cluster Histograms

After the bins of our histogram are determined, the final step of feature extraction is to create a higher level feature type, that focuses on the description of a person's overall appearance, the so called *feature cluster histogram (FCH)*. Feature cluster histograms are built in the following way: Let $F = \{i_1, \ldots, i_n, i_{n+1}, \ldots, i_{n+m}\}$ be the set of features, that is used to learn a person's appearance and builds the data basis of our histogram bins, n denote the amount of video frames, that represent the person to be learnt and m stand for the number of images, that are used as negative training samples. Finally, i_k is the set of basic features, that represent one image. Then, a histogram H_i is built for each input set i_k and normalized to abstract from the size of n, m and to emphasize the differences of entries instead of their absolute values. The labeled $m + n$ feature cluster histograms $H_1, \cdots, H_{m+n}$ represent the training set for the creation of a person's statistical model.
An important aspect of feature cluster histograms is the fact, that similarities between a set of features and one of its subsets are reflected in their associated feature cluster histograms. This is, because our classification module benefits from being able to classify parts of the person as the person, that shall be recognized.

3.4 Storage

The extraction of features is a computationally expensive task. It is therefore reasonable to calculate feature values prior to the classification process and store them persistently.
The specific choice of storage technique strongly depends on the recognition system's intended purpose. For example, a prototype for evaluation and testing of a recognition system is supposed to be more flexible and expandable than a business application with a strict functional outline, which will mainly focus on performance and stability. Our implementation features a binary file based approach, that is encapsulated in an object-oriented persistency layer. The basic features, that represent a video, are expressed as a set of three files:

- A binary coded file containing the feature data. The data consists of the individual feature vectors being stored one after another without separations.
- An ASCII coded index file to grant quick access to features of a specified video frame. It contains the amount of feature vectors for each frame and an offset in bytes to where the first keypoint of each frame is localized in the binary file.
- A file used for data summary and the storage of global variables. It stores the size of each feature vector in bytes as well as the overall amount of frames and features, that were extracted from the video.

This file set is generated for each feature type and for every performed preprocessing step per video, i.e. the reduction of features to a subset, that lies in the foreground of each video frame or the reduction of features to a subset, that can reliably be tracked for a predefined time period in the video. The described file structure is encapsuled by an objet oriented layer, that is able to read features from videos in every preprocessing stage, thereby abstracting from specific file sets, as well as to create feature file sets and store extracted features.
The following paragraph outlines the reasons for the implementation of a file based storage approach: The frame wise data indexing and the storage of features at different preprocessing levels provide high-performance data access, because calculations are typically performed sequentially on entire video frames and on one level of preprocessing. As the physical data storage is encapsulated, the system is provides flexibility and expandability. It is, for example simple to add a new feature type or a new preprocessing step. It shall be noted, that expandability in this context is refered to the system's structure. Every step, that follows the newly introduced preprocessing step, has to be recalulated, nevertheless. Because no additional software (like a Data Base Management System) is needed to operate the system, it can easily be integrated in an existing environment and

is robust to hardware or software changes. Finally, data backups can easily be achieved by copying the desired files, which is important due to the enormous amount of computing time necessary for feature extraction.

Chapter 4

Learning and Classification

Now that we have extracted features from our video data, that describe a person, we use these features to learn a statistical model for a person's outer appearance. Therefore, an appropriate machine learning algorithm is applied in order to build a model, which serves as a classifier for persons in images.
Discrete AdaBoost has been chosen for learning this model as it is known to be robust to noisy training data and capable of learning interdependencies in the training data.
Once we have computed a model for every person we want to recognize, we can use the models to solve our recognition task in images. In order to achieve that, a similarity index between every model and the test image is calculated. The model with the highest similarity is considered to represent the person on the image.

4.1 Discrete AdaBoost

The discrete AdaBoost ("Adaptive Boosting") algorithm was proposed by Freund and Schapire in [20]. This statistical learning method has mainly been chosen, because of its good recognition results in related research fields. A second reason is its ability to learn interdependancies in every element of the training data. This is important, as the presence of specific features of a person's appearance, like a characteristic color pattern on the clothes, is correlated and helpful for determining a correct classification result.
The AdaBoost algorithm is an iterative method to train a set of multiple classifiers together with a function, through which the predictions generated by the models can be combined. It is based on the idea, that a complex classification task can be accomplished by the individual training and subsequent combination of multiple weak classifiers. A weak classifier's only prerequisite is, that has to be able to classify the training set better than chance. Initially, a value is assigned to every element in the training set, that determines the importance of a training sample for the classification process. This value will be refered to as a training

sample's "weight". Those weights are used for the calculation of the error, that each classifier produces when evaluating the training set. If an element is misclassified, its weight is increased. This ensures the algorithm's concentration on difficult examples, thereby selecting a classifier in the next iteration, that leads to a better identification of the examples, that were previously misclassified. A brief summary of the algorithm is depicted in figure 4.1.
In our training phase, the input of the AdaBoost algorithm consists of a training set $(x_1, y_1), \ldots, (x_m, y_m)$, with x_k being a feature cluster histogram as described in section 3.3.2 and an associated label y_k of some label set $Y = \{negative, positive\}$. As we use the discrete version of AdaBoost, the label set is assumed to contain only two elements. In our case, the boolean value of an assigned label states, whether or not the person we want to learn or classify is present on the image. The algorithm calls a base learning function in a series of $t = 1, \ldots, T$ iterations, generating a weak classifier $h_t(x_i)$ in each of them. h_t can be understood as a discrete function, that maps instances of the training or test set to an element of the label set Y. Furthermore, a discrete function $D_t(k)$ with $k = 1, \ldots, m$ is defined, containing the weight of every training sample k in iteration t. Initially, all weights are set equally, but will be updated individually every round, depending on the classification error. If the classification error of a training sample is high, its weight increase will be high as well in order to emphasize the sample's importance compared to other elements of the training set. The error of a single classifier h_t can be calculated as follows:

$$e_t = \frac{\sum_{k=1, h_t(x_k) \neq y_k}^{m} D_t(k)}{\sum_{k=1}^{m} D_t(k)} \tag{4.1}$$

The precondition, that every classifier has to be able to perform better than chance can now be formulated as $e_t < \frac{1}{2}$ or by using the weight function D_t:

$$\sum_{k=1, h(x_k)=y_k}^{m} D_t(k) > \sum_{k=1, h(x_k) \neq y_k}^{m} D_t(k) \tag{4.2}$$

After the classification error has been determined, a parameter α_t is calculated for every classifier h_t and can intuitively be looked upon as the importance of h_t regarding the final classification result:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - e_t}{e_t} \right) \tag{4.3}$$

Note, that $\alpha_t \geq 0$ and that it gets larger as e_t decreases. In the algorithm's next step, α_t is used for an update of D_t in order to increase the weight of

misclassified examples (N_t is a normalization factor to ensure D_{t+1} remaining a distribution):

$$D_{t+1}(k) = \begin{cases} \frac{D_t(k) \cdot e^{-\alpha_t}}{N_t}, \text{ if } h_t(x_k)=y_k \\ \frac{D_t(k) \cdot e^{\alpha_t}}{N_t}, \text{ if } h_t(x_k)\neq y_k \end{cases} \tag{4.4}$$

The final classifier H is based on a weighted majority vote of the T weak classifiers h_t and defined as follows:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right). \tag{4.5}$$

In our approach, every weak classifier $\{h_t | t = 1, \ldots, T\}$ is based on a CART classifier. CART stands for "Classification And Regression Trees" and is a learning method proposed by L. Breiman et al. ([8]). It uses a tree structure as a data model, that encodes the distribution of our class labels (in our case: being a particular person or not) in terms of the data set's attributes (in our case: each entry of a feature cluster histogram vector).
During its learning process, the training data set is subsequently split in parts, that are as homogenous as possible. Each branch of the tree represents a partitioning step according to split rules, that are based on the attributes of the data samples. Every node contains the probabilities for a data sample to be element of a particular class. A node is considered a leaf, if every data sample in the node is classified identically. The process terminates, if every node at the bottom of the tree is a leaf.
In order to classify a test instance, it is sorted down the tree from the root to some leaf, depending on the results of the split rules, that are applied at every node. The leaf node provides the classification of the test sample.

In our approach, feature cluster histograms we extracted from videos, that display the person to be learnt, form our positive training set along with an associated positive label. Negative training samples are acquired from non-person images, such as aeroplanes, motorcycles, landscapes, offices and so on. Feature cluster histograms are created from the entire image and labeled negatively, before passing them on to our learning algorithm.
A CART decision tree is then built in every iteration and used to split the histogram set (our training set) into subsets, that are as homogenous as possible. The separation is done by using the amount of features in a histogram bin as a threshold.

Input Set of feature cluster histograms and labels $(x_1, y_1), \ldots, (x_m, y_m)$

Initialization $D_1(k) = \frac{1}{m}$, where $k = 1, \ldots, m$

For $t = 1, \ldots, T$

1. Generate a weak classifier h_t
2. Calculate the classification error e_t using equation 4.1
3. Determine the "importance" factor α_t using equation 4.3
4. Update the weight distribution D_t as described in equation 4.4

Output Classifier $H(x)$ as defined in equation 4.5

Figure 4.1: Pseudocode of the AdaBoost learning algorithm

4.2 Classification

To accomplish the task of recognizing a person in a test image, a statistical model of every recognizable person is learnt, using the AdaBoost algorithm described in the previous section. As the background of a static image cannot be subtracted without a priori information about the image's contents, classification cannot be performed in one step. Therefore, different parts of the image are classified subsequently using every person's model in order to achieve a correct classification result. A sliding window technique is applied to traverse the test image. The classification results for all image parts are summed up in a set of classification matrices. These matrices are used to determine the overall classification result.

The Sliding Window Technique

To be able to traverse an image, we put a grid on our test image. Each grid element has a rectangular shape of 20×20 pixels and sets the minimum distance of two consecutive classification operations.
Next, a region of interest (the *"window"*) is defined, which determines the image area, that is currently processed. It is expressed as a number of grid elements and aligned with the grid. The size of every window depends on two attributes, that change their values during the classification procedure, namely its scale and aspect ratio. The scale of a window is a factor, by which the basic window height and width are multiplied, whereas the aspect ratio defines the fraction of the window height divided by its width. Examples of a grid and of windows having various sizes and scales (as we used them for our approach) are depicted in figure 4.2, 4.3 and 4.4. The grid is illustrated by parallel horizontal and vertical lines. Every image shows four scales of sliding windows having a particular aspect ratio. The

aspect ratio is varied between the images, but remains constant for every window within an image, whereas the scales do not change between images, but vary for every window, that is shown within an image. The color of each window indicates its scale.

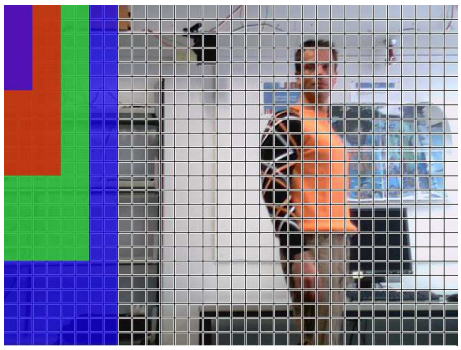


Figure 4.2: Classification grid with sample windows at various scales having an aspect ratio of 3:1

To traverse an image, the window moves from left to right and from top to bottom with a distance of one grid element between two consecutive steps. In every step, a feature cluster histogram (see also section 3.3.2) is created using the features, that lie within the window's borders, as input data. The histogram bins are identical to the clusters, that were used to create the training data of the current learning model. Finally, we classify this histogram using the AdaBoost classifier created from our training samples. Each classifier returns a boolean value, that states, whether or not a person is assumed to be present in the classified image part. These classification results are summed up in matrices for each window scale and each person seperately (see also section 4.2). After the image has been traversed entirely, the procedure is iterated with varying window sizes.

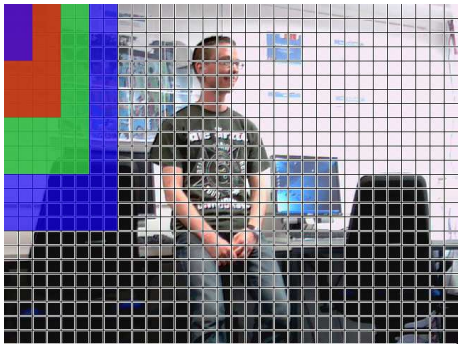


Figure 4.3: Classification grid with sample windows at various scales having an aspect ratio of 2:1

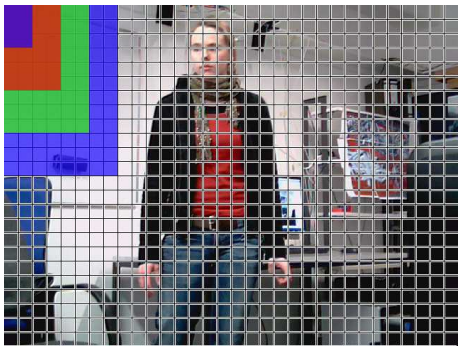


Figure 4.4: Classification grid with sample windows at various scales having an aspect ratio of 3:2

The similarity between the training data and the histogram we created from image parts will reach its maximum, when the size of the person and the size of the sliding window are identical. If the window is smaller than the person, it is still likely to obtain correct positive classifications, because similarities between parts of a person and the whole person are reflected in their associated feature cluster histograms. As we traverse the entire test image with a small window, we can still acquire good classification results, depending on the image data. If the sliding window is larger than the person, the amount of noise in the histograms will rise, therefore leading to misclassifications. Another reason, why window sizes, that are smaller than the person on the test image, may lead to correct classification results lies in the training set. Due to the huge amount of video data, that is used for the creation of a training set, there is a natural variability regarding the size and aspect ratios of a person or the part of the person, that is accounted as a positive training sample.

Classification Matrix

The classification results we obtained from the parts of our test image, that were created by the sliding window, are stored in a set of matrices. The size of each matrix is 32×24, which means it is identical to the size of the grid, we used in the last section. Every matrix element represents the overall amount of positive or negative classifications for the grid entry, where its position within the grid is identical to the position of the matrix element within the matrix (the *"associated"* element). The matrix set can be divided in a subset of *positive classification matrices*, which contain the amount of positive classifications for each grid element, and a subset of *negative classification matrices* accordingly. These subsets consist of one classification matrix for each scale we used to define the size of our sliding window.
A matrix set is constructed for each person, that has been learnt. Initially, every matrix element is set to zero. As a next step, the sliding window algorithm is per-

formed for every person's model, every window scale and aspect ratio. Whenever a feature cluster histogram of an image part is classified (positively or negatively), the associated (positive or negative) classification matrix element of each grid entry, that lies within the borders of the current window, is incremented. The process is illustrated in figure 4.5.

As described above, the choice of the specific matrix for incrementation depends on the classification result, the window scale and the person's model. The aspect ratio of a window, however, does not induce its own subset of classification matrices. Instead, every classification result obtained from sliding windows with different aspect ratios, is summed up in the classification matrix with the according scale. The seperation of results from different scales is a reasonable proceeding, because the person we want to recognize is depicted in a distinct size.

The next section will describe a classification index, that reduces the impact of false positive classification results. It weighs the amount of positive classifications with a value, that is determined by their distribution in the positive classification matrix. Figure 4.6 shows three grayscale pictures of positive, normalized classification matrices at different scales. They are displayed on the left hand side. The learning model our classification matrices are based on, represents the same person, that is depicted on the test image to the right.

The next step, after classification matrices have been computed for every person in our model set, is to normalize the matrices before we can use them to classify the test image as a whole. This is done to reduce the influence of the amount of classifications computed for each grid element. The different amounts occur due to different window sizes as well as the fact, that the sliding window's borders are used to determine the beginning and the end of a row or column (i.e. the algorithm starts at a window position, where the upper and the left window borders are aligned with the corresponding image borders and ends, when the lower and the right window borders are in the corresponding position). Therefore, grid elements with a position, that is closer to the image border, are classified less often than those in the image center. Thus, every matrix element is divided by the amount of classifications performed on the associated grid element. Furthermore, as the total sum of classifications depends on the scale of the sliding window, the distribution for the amount of performed classifications varies between two classification matrices at different scales. Figure 4.7, 4.8 and 4.9 depict examples of such distributions at three commonly used scales. The x and y direction represent the classification matrice's row and column, whereas the z-direction shows the amount of positive classification for each grid element. The figures show, that without normalization, the center of each classification matrix becomes emphasized. As we do not make assumptions to a person's position in the image, it is reasonable to counteract this effect by normalizing each classification matrix.

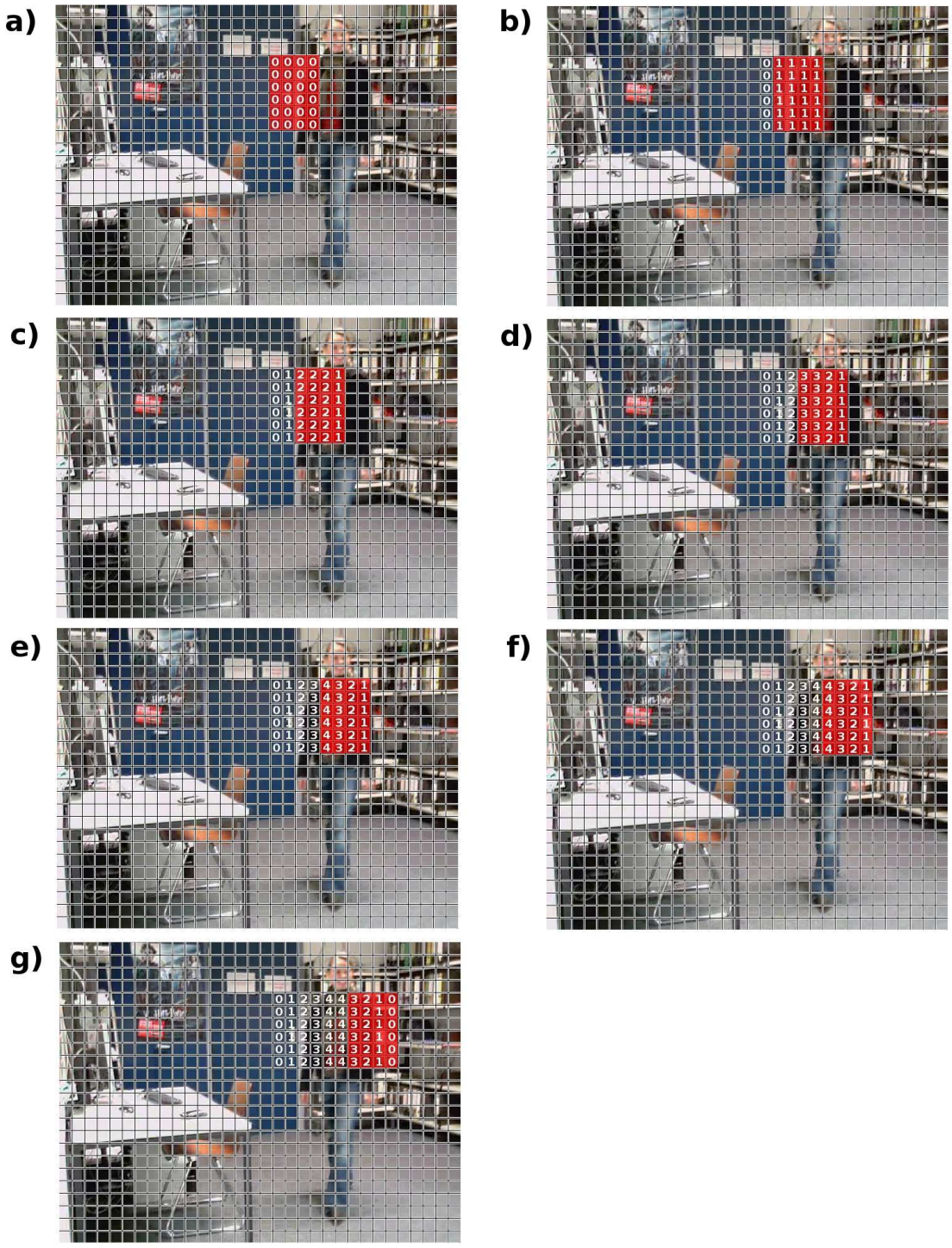


Figure 4.5: Step-by-step example of a positive classification matrix being built using the sliding window algorithm. The red rectangle represents the current window position, the numbers on the grid elements stand for the amount of positive classifications and are identical to the corresponding positive classification matrix values.

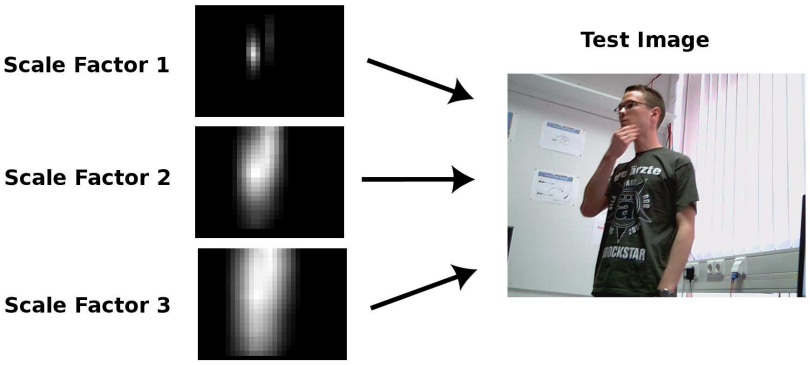


Figure 4.6: Positive Classification matrices. On the left, three normalized classification matrices, representing different scale factors, are displayed as grayscale images. The underlying model represents the person, that is shown on the test image to the right.

Classification Index

After the classification matrices $M_{c,p,s}$ have been created and normalized for every person p, scale s and classification result $c \in \{positive,\ negative\}$, we are able to classify the test image. Each classification matrix is of size $m \times n$ and m, n denote the number of grid elements in x, y-direction, respectively. In the following we denote the number of persons with P and the number of scales used with S. The first step is to create an index $L(M_{p,s}) \widehat{=} L(M_{\{positive\},p,s}, M_{\{negative\},p,s})$, which measures the likelihood of a particular positive classification matrix to represent the person, whose model it is based on. An important aspect of the index is its ability to express the classification result as a real number, instead of simply assigning a boolean value, as we want to compare classification results from different models and select the person, whose classification index has the highest value. The index is defined as

$$L(M_{p,s}) \quad = \quad \frac{\sigma^2(M_{\{positive\},p,s}) \sum_{i=1}^{m} \sum_{j=1}^{n} M_{\{positive\},p,s}(i,j)}{\sum_{i=1}^{m} \sum_{j=1}^{n} M_{\{positive\},p,s}(i,j) + M_{\{negative\},p,s}(i,j)} \tag{4.6}$$

with $\sigma^2(M)$ being the variance of a matrix M. The variance of the current classification matrix is used as a measure for the "compactness" of its value distribution and is used as a weight, which is multiplied with the relative amount of positve classifications in the matrix. We assume here, that the positive classifications of a correctly classified image are not spread over the entire image but localised at a small number of positions in the image.
In order to obtain the final classification result, we determine the maximum of $L(M_{p,s})$ for each person's classification matrix set. The person, whose model was

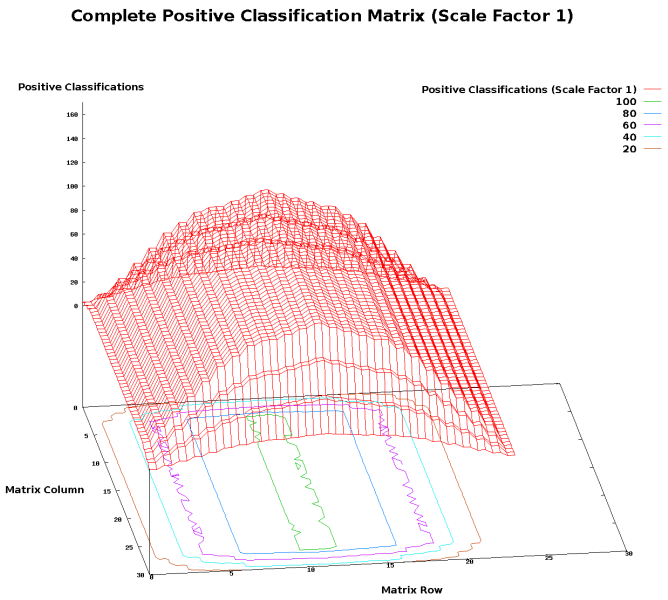


Figure 4.7: Example of a positive classification matrix with a window scale factor of 1 without normalization, that has been classified entirely positive. The x and y direction represent the classification matrice's row and column, respectively. The z-direction shows the amount of positive classification for each grid element.

used to create the set with the highest value is our final classification result. In order to obtain the final classification result, the maximum $L_{max}(M_{p_m,s_m})$ of all indices $L(M_{p_i,s_j})$, with $i = 1, \cdots, |P|$ and with $j = 1, \cdots, |S|$ is defined as

$$L_{max}(M_{p_m,s_n}) \quad = \quad \max_{p_i,s_j} L(M_{p_i,s_j}) \tag{4.7}$$

where p_m is the person classified and s_n denotes the scale, where the person was detected.

4.3 Position Estimation

The positive classification matrix with the highest classification index, can be used to estimate the position of the recognized person in the image. Since every element of the classification matrix contains the probability each grid element to be a part of the person to recognize, the task of position estimation can be reduced to picking the grid element in the matrix, where the center of the person is assumed to be and thresholding the surrounding matrix elements in a reasonable way.
We determine the center of the person by calculating the centroid of the positive classification matrix. Therefore, we calculate the image moment of our matrix,

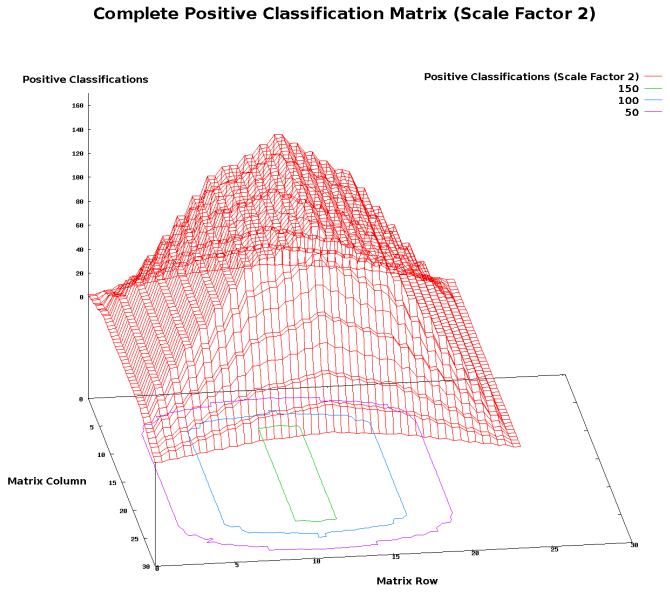


Figure 4.8: Example of a positive classification matrix with a window scale factor of 2 without normalization, that has been classified entirely positive. The x and y direction represent the classification matrice's row and column, respectively. The z-direction shows the amount of positive classification for each grid element.

as the classification matrix can be understood as a grayscale picture with the grey levels indicating the likelihood for grid elements to contain a part of the person. We start the computation for the classification matrix $M_{\{positive\},p_m,s_n}$, that successfully classified a person, by determining the following values:

$$V_{00} = \sum_i \sum_j M_{\{positive\},p_m,s_n}(i,j) \tag{4.8}$$

$$V_{10} = \sum_i \sum_j i\, M_{\{positive\},p_m,s_n}(i,j) \tag{4.9}$$

$$V_{01} = \sum_i \sum_j j\, M_{\{positive\},p_m,s_n}(i,j) \tag{4.10}$$

$$\tag{4.11}$$

The centroid location i_c, j_c in the matrix are then determined according to:

$$i_c = \frac{V_{10}}{V_{00}} \tag{4.12}$$

$$j_c = \frac{V_{01}}{V_{00}} \tag{4.13}$$

$$\tag{4.14}$$

We take it as a prerequisite, that the classification matrix will describe one

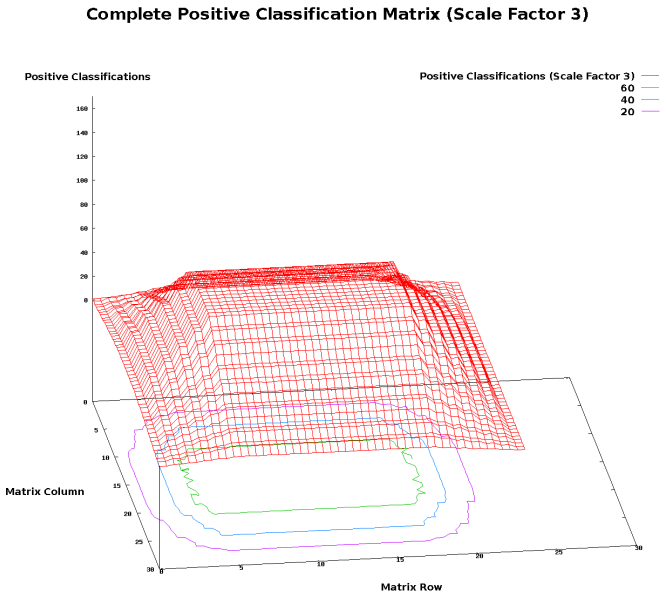


Figure 4.9: Example of a positive classification matrix with a window scale factor of 3 without normalization, that has been classified entirely positive. The x and y direction represent the classification matrice's row and column, respectively. The z-direction shows the amount of positive classification for each grid element.

connected region, if the person was recognized correctly. Hence, every centroid is assumed to be located on the person, we want to recognise. In order to detect the circumference of the person, we use a flood fill algorithm with the centroid being the starting point. A threshold is applied in the algorithm, describing the minimal ratio of an element to the maximum of the matrix. The threshold is necessary, as usually an area around a person is classified positively, although the person is not displayed. This is a side effect of the person model's ability to classify feature cluster histograms positively, even if only a part of the person is displayed. The value of the threshold depends on the scale of the sliding window as well as on the average amount of similarity, that is necessary for a model to identify parts of the person as being the person. An example of a position estimation is shown in figure 4.10. It shows three grayscale pictures of positive, normalized classification matrices at different scales. They are displayed on the left hand side. The matrix, that represents a scale factor of 3, has the highest classification index and is therefore used to determine the person's location. The learning model our classification matrices are based on, represents the same person, that is depicted on the test image to the right. To indicate a person's position, a rectangle is drawn around the outline of the person recognized.

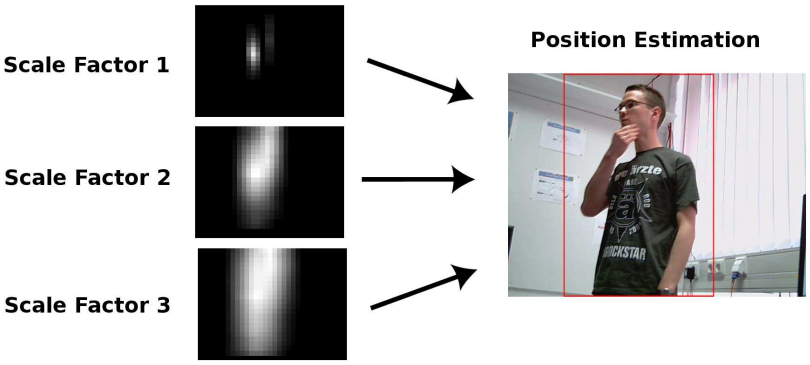


Figure 4.10: Position estimation of a person in the test image. On the left, three grayscale images of positive classification matrices are shown. The matrix with the highest classification index (Scale factor 3) is used to determine the person's location. The final position is illustrated by drawing a rectangle around the area.

Chapter 5

Experimental Evaluation

This chapter describes the test design we used to perform our experiments and shows the recognition results we obtained by experimental evaluation.

5.1 Test Design

The next section describes the experimental setup in detail. It mentions fixed values in our implementation and lists parameters, that are varied throughout experimental evaluation.

5.1.1 Training Sample Set

The training set is divided in positive and negative training samples. We create a training set for every person to be recognized. Our experiments feature four different persons.
Figure 5.1 shows sample frames of several training videos, that display our test persons and are used to construct positive samples for our learning algorithm. We used between one and four videos to create the positive training sample set of each person.
The videos are captured by six webcams of the type "Philips SPC 600NC". They are installed in a room at various positions and heights (see section 2.3). Figure 2.1 displays the camera perspectives we used for data acquisition. Every video consists of 1000 to 3000 frames, each of them having a resolution of 640×480 pixels. The video cameras are set on automatic light control to compensate gradual illumination changes. As the cameras need a certain time period to adjust to the current lighting conditions, the first 40 frames of each video are dropped, in order to increase the performance of our foreground segmentation algorithm. Furthermore, a period of absent foreground is available in every training video. After the segmentation has been performed and basic features have been extracted (see section 3), we empirically evaluated several videos, in order to find similarities between poorly segmented video frames. We discovered, that the ten percent

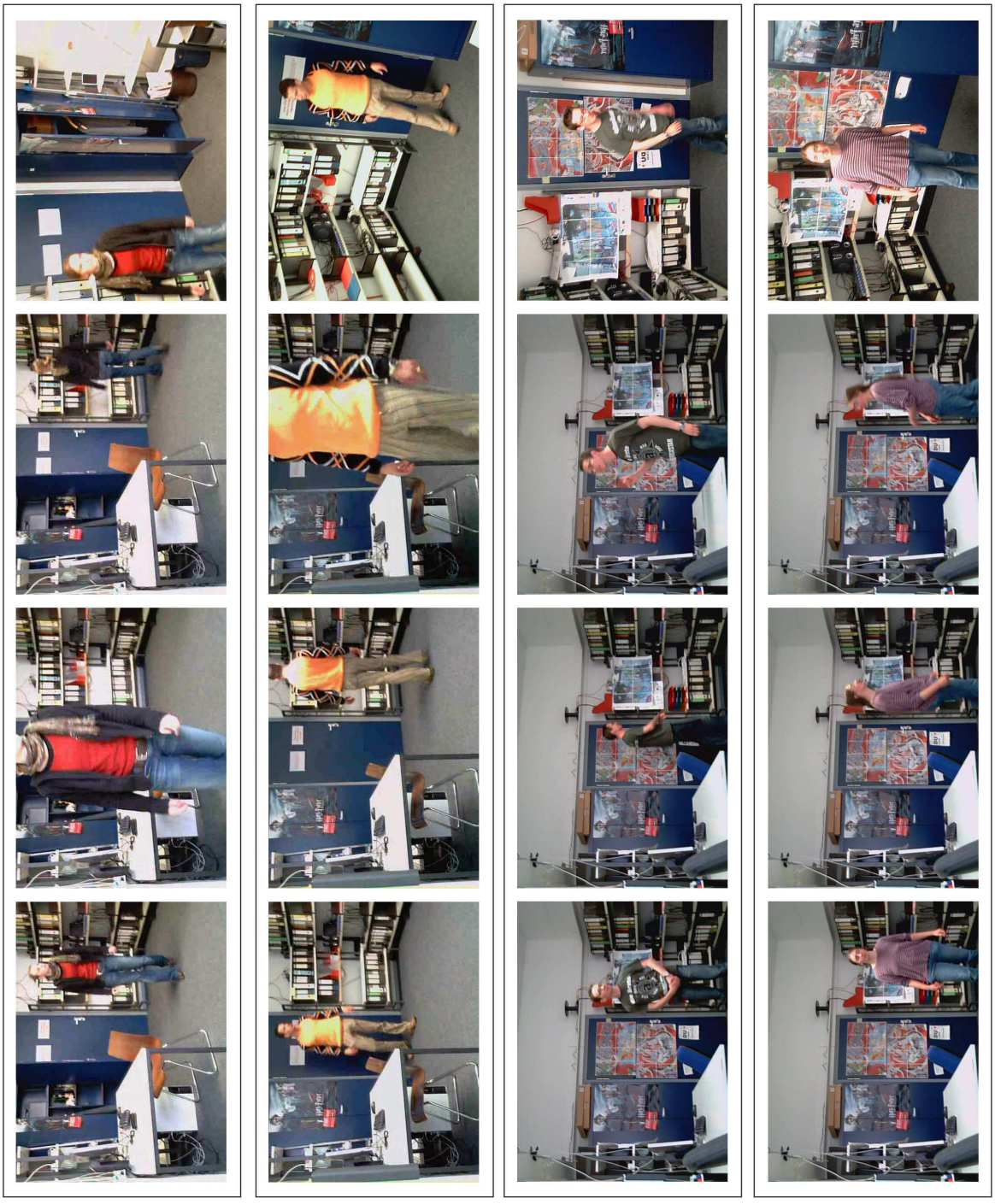


Figure 5.1: Examples of the training video set. The test persons are labeled A,B,C and D from right to left.

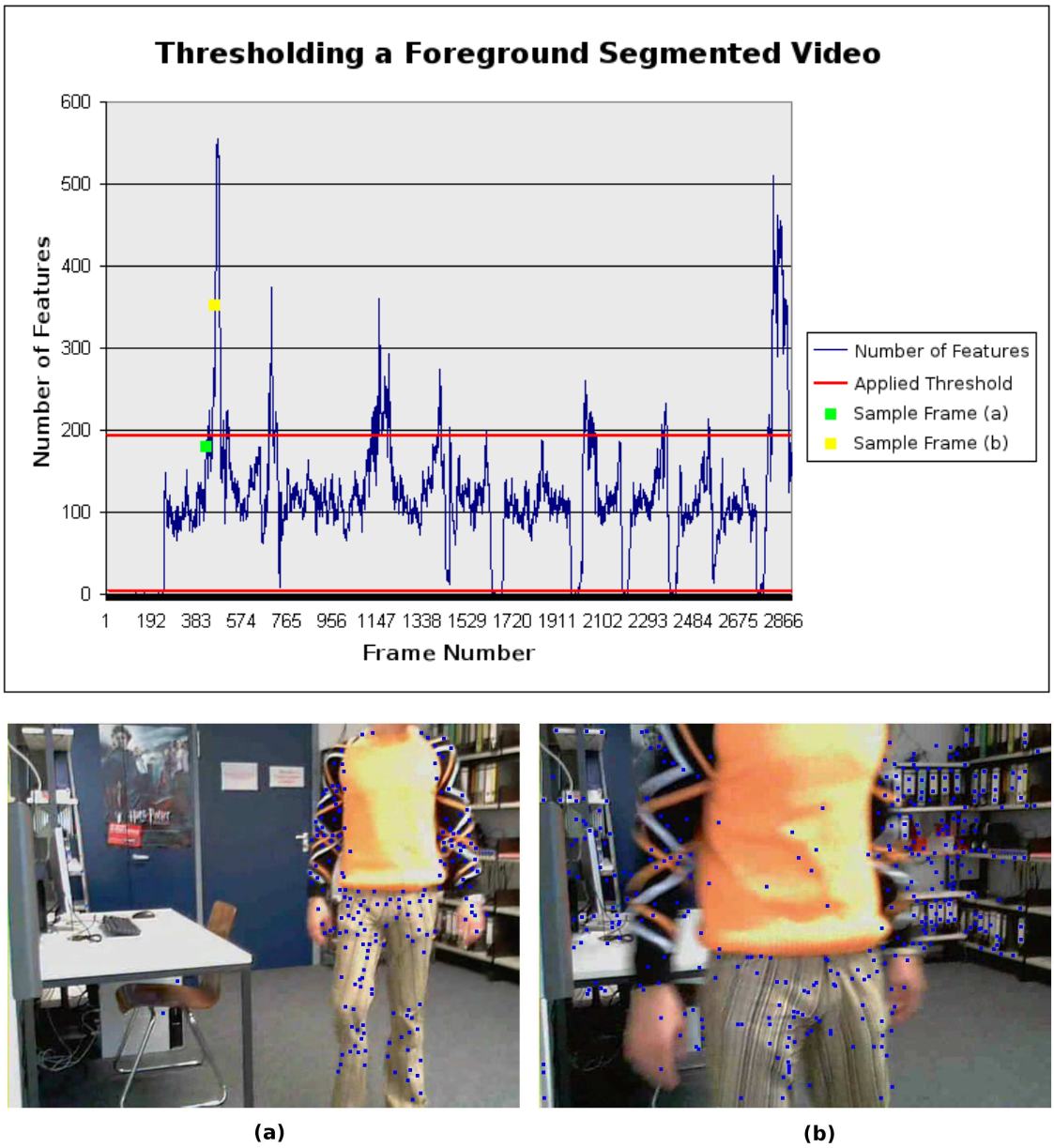


Figure 5.2: Top: A diagram of a foreground segmented video. Two thresholds are displayed, that are used to reduce the amount of poorly segmented video frames in our training data. Position of two sample frames are marked by colored rectangles. Bottom: Two sample frames, marked in the top diagram. Blue dots mark foreground feature locations.

of foreground segmented frames with the largest amount of features have a high probability of the background being considered as foreground. An explanation for these misclassifications is, that the camera's automatic light control adjusts illumination parameters too sudden for the background subtraction algorithm to adapt or other sudden illumination changes occur. Furthermore, the ten percent of foreground segmented frames with the least amount of features typically contain clutter without a person being present. In order to avoid frames with a poor foreground segmentation becoming part of the training data set, we applied a threshold at the previously stated percentage levels.
Figure 5.2 (top) shows an example of a video with the threshold values and the positions of two sample frames being plotted in the diagram. The frames are depicted in figure 5.2 (bottom) and illustrate the difference between a frame, that is unaffected by the applied threshold (a) and a frame, that would not be used for the creation of a training sample (b). The applied threshold is obviously working well, as in the right frame, a lot of features are describing the background. Finally, we omit every frame with less than 25 features before constructing our training set. This measure is implemented to ensure a frame being representative for the person, it displays.
Our negative training data is created from color images of differing sizes, that display various non-person objects, such as aeroplanes, landscapes, cars, motorcycles, offices, plants, buildings etc. Corresponding to the creation of our positive training samples, images with less than 25 features are not used for negative training set construction.
After extracting basic features from the video(s) and the image data, we define the number of training samples N. In our test environment, the number of positive and negative samples is set equally. A classifier is created for each person, using an equal amount of positive and negative samples as its training set. N is varied throughout the experiments. Positive training samples are picked randomly from the person's video(s), the data for negative samples from the negative image data set.
Initially, all basic features of the selected images/frames are read. Tests are performed using RGB color histograms, HSV color histograms, HS color histograms and Lowe's SIFT descriptor. Color histograms are computed by using a histogram with 42 bins for each color channel. The region, that is described around the detected keypoint location is square shaped with an initial size of 21×21 pixels, multiplied by the scale factor σ. The histograms of all channels are serialized to a vector, that contains 126 entries. The computation of the SIFT descriptor is based on 16 histograms with 8 bins, that represent the gradient orientations of the local image region around a keypoint location. These histograms are also serialized to a vector of 128 values.
Next, the overall amount of basic features F, that are used for building the training set, is defined. We sample the acquired basic features randomly, so that $\frac{F}{2}$ elements remain within the set of positive and negative features. The parameter

F is varied for the purpose of testing. It shall be noted, that different classification outcomes can occur while using the same parameter values, due to the stochastic nature of the sampling process. Hence, all tests are run multiple times and the mean of their results is used as the final outcome.

After two equally sized positive and negative feature sets have been obtained, the k-means clustering algorithm described in section 3.3.2 is applied to determine the bins for our feature cluster histograms. The input values for k-means are all basic features of the positive and negative feature set. The algorithm terminates after a maximum of 30 iterations or if all cluster centroid positions are below 0.01, compared to their positions during the last iteration. In our experiments, we vary the amount of clusters k.

The next step after determining our histogram bins is to create a histogram of the feature clusters for every frame/image of our positive/negative basic feature set. In our experiments, the distance measure for calculating cluster histograms is varied between the L1 and L2 norm. The resulting feature cluster histograms are labelled according to their corresponding basic feature set and serve as training data of our AdaBoost learning algorithm.

5.1.2 Test Images

Our test set consists of 35 images for each of our four test persons. The time period between video recording and image capturing reaches from two weeks to six months. There are no restrictions to the images, except that the person's clothes must have been learnt in order to achieve good classification results. Two of our test persons are wearing brightly colored tops, one of them being sparsely texturized in the chest region. The other two test persons are wearing mutedly colored, but highly texturized tops. In some images, one of our test persons is shown in front of the same background we used for video capturing. Apart from these images, special care was taken to show backgrounds that differ from those contained in the videos. In our approach, the video cameras used for image capturing and video recording are identical. To compensate illumination changes, the cameras have been set on automatic control.

Figure 5.3 displays a subset of the test images we used for evaluation and testing purposes.

5.1.3 Learning and Classification

Our learning approach is based on the construction of a person's model for each person, we want to recognize. AdaBoost has been chosen as the underlying learning algorithm. AdaBoost is an iterative algorithm, that uses a weighted majority voting system to create a statistical model. The voting committee consists of a number of CART decision trees with two node splits each. In every iteration, the weights are reevaluated to classify the training data as correctly as possible.

Figure 5.3: Examples of the test image set. The test persons are labeled A,B,C and D from right to left.

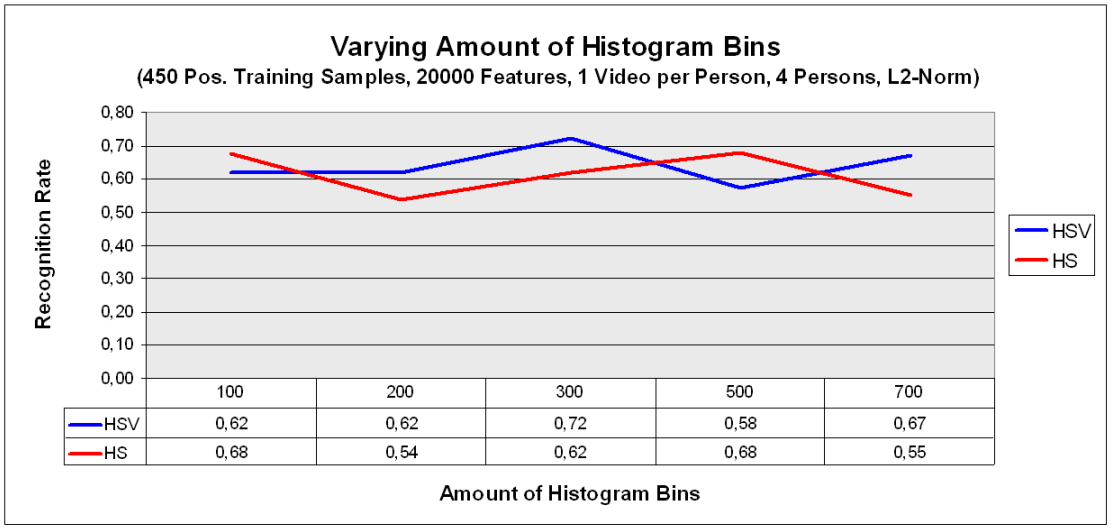
We set the number of boosting iterations, that will be performed during model creation, to 1000.
In our experiments, we evaluated two different classification mechanisms. One classification method is to use all four classifiers simultaneously on each test image and recognize the person by selecting the model, whose classification index has the highest value. This approach does not consider the absence of persons or persons being displayed, that have not been learnt. Each one of our 140 test images is classified by using every person's model. Finally, the recognition rate is determined by dividing the number of correctly classified images with the overall amount of test images.
The second classification method we evaluated, is based on the use of a global fixed threshold for each person's classifier. The advantage of the second approach is, that unknown and absent persons can be handled, since the classification result is a boolean value for each classifier.
In order to classify an image, a grid with a patch size of 20×20 pixels is applied. A sliding window technique (as described in section 4.2) is used to traverse the test image. The minimum window size is set to 40×40 pixels. In our experiments, we use different combinations of window scales and window aspect ratios. Typical combinations for window scales are $\{2, 3, 4\}$ and $\{4, 6, 8\}$. The window aspect ratios are subsets of $\{2/1, 3/2, 3/1\}$ (window height / window width).

5.2 Results

We executed various tests to determine the performance of our person recognition approach. The tests are basically separated in the evaluation of a classification procedure without unknown persons, and experiments for a classification method, that uses an absolute threshold and is therefore able to handle the absence of persons in test images or persons, which have not been learnt prior to the classification procedure.

5.2.1 Classification without Unknown Persons

The first classification method we will evaluate, is based on building a statistical model for every person, that shall be recognized, and classifying every image of our test set with each person's classifier. The result of each classifier is called a classification index, and describes the similarity between the person and the image. The model, that classified a test image with the highest value, is assumed to represent the person on the image. We will vary several of the parameters we mentioned in section 5.1 and use the resulting recognition rate to evaluate the efficiency of various parameter settings. These settings are not limited to the classification method without unknown persons, because they only affect each person's classifier seperately and can therefore be used to optimize the second



	100	200	300	500	700
HSV	0,62	0,62	0,72	0,58	0,67
HS	0,68	0,54	0,62	0,68	0,55

Figure 5.4: Recogniton Results for Various Amounts of Feature Cluster Histogram Bins. 450 positive and 450 negative training samples are used. Feature cluster histogram bins were created by using 20000 features as input for the k-means algorithm. One video was used for each person. Models of four persons have been learnt. Feature cluster histogram computation is performed using the L2-norm as a distance measure. HSV and HS color histograms are used as basic features. Sliding window scales were set to $\{2, 3, 4\}$, the window's aspect ratios were $\{3/1, 2/1, 3/2\}$.

classification method as well.
The first parameter, that will be varied, is the amount of bins for every feature cluster histogram. Since every bin is determined by the application of the k-means clustering algorithm on all features, that will be used for the creation of the training set, we simply adjust the amount of clusters "k". The test results are depicted in figure 5.4 and show recognition rates between 60% and 70% with the HSV descriptor performing slightly better than the HS color histograms. The illustration indicates the algorithm's flexibility regarding the choice of histogram bin numbers. Next, the impact of the distance measure on recognition performance is evaluated. Our tests focus on a comparison between the L1 and the L2 norm. Results indicate, that both norms perform almost equally well, with a small advantage for the L2 norm. This can be explained by the fact, that the L2 norm uses the square of the input values for distance computation, thereby avoiding the possibility of vector entries annihilating one another, if their algebraic signs are reversed. The experiment's results are shown in figure 5.5. As we have stated in section 5.1.3, different window aspect ratios are used to create a classification matrix. We decided to evaluate, which ratio performs best. Dia-

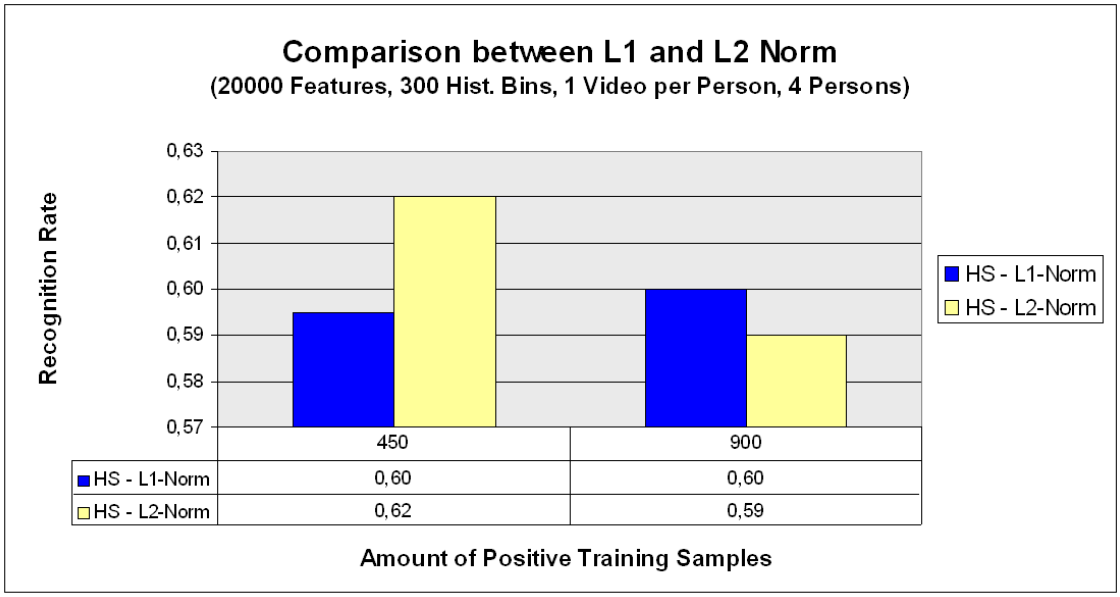


Figure 5.5: Comparison of Distance Measures for Feature Cluster Histogram Generation. The amount of training samples is varied. Feature cluster histogram bins were created by using 20000 features as input for the k-means algorithm. The amount of feature cluster histogram bins has been set to 300. One video was used for each person. Models of four persons have been learnt. HS color histograms are used as basic features. Sliding window scales were set to $\{2, 3, 4\}$, the window's aspect ratios were $\{3/1, 2/1, 3/2\}$.

gram 5.6 illustrates the results. Typical humans have an aspect ratio of 4/1 or 5/1. On the first glance, it is therefore surprising, that an aspect ratio of 2/1 performs best, as one might surmise, that a window aspect ratio, that matches a human being's shape, will turn out to have the best test results. The actual result can be explained by the fact, that due to the similarity between a feature cluster histogram and its subsets, most correct classifications in the classification matrix are the result of recognizing parts, instead of the whole person. In our next experiment, we will compare all basic feature descriptors while varying the amount of samples, the algorithm is given as training data. Figure 5.7 shows the experiment's results. The system is not able to generalize well, if the learning algorithm is given 200 positive training samples or less, independent of the decision, which basic feature descriptor type is used. The SIFT descriptor shows the lowest recognition rates. An explanation for this may be, that the gradient orientations of the descriptor are too specific for our learning approach. RGB color histograms perform significantly better, but are still second to HS and HSV histograms. This result indicates, that color parameters, like illumination, saturation and color type generalize better than its primary color components. The idea behind ignoring the value (V) parameter of our HSV color histograms was,

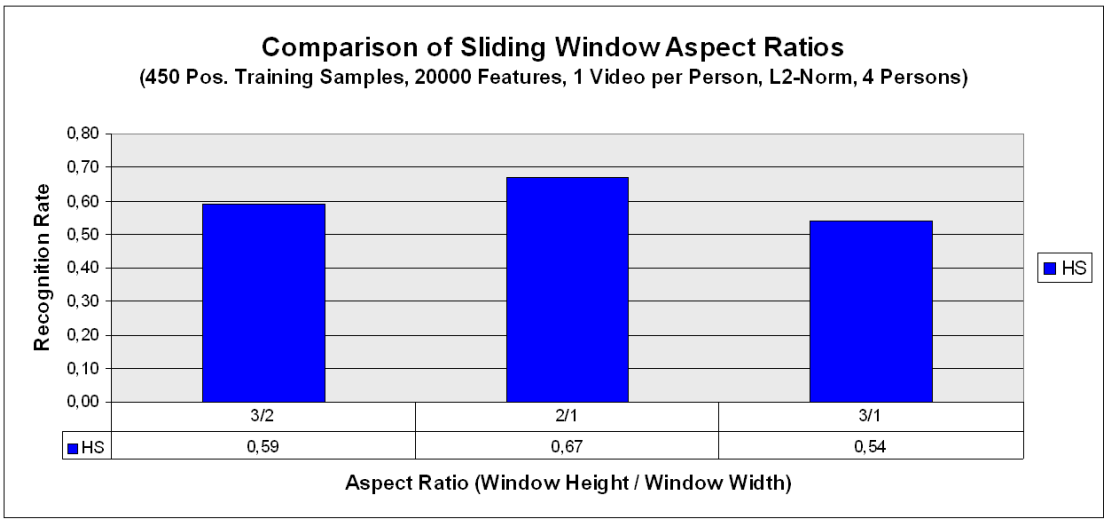


Figure 5.6: Evaluation of Different Sliding Window Aspect Ratios. 450 positive and 450 negative training samples have been used. Feature cluster histogram bins were created by using 20000 features as input for the k-means algorithm. The amount of feature cluster histogram bins has been set to 300. One video was used for each person. Models of four persons have been learnt. Feature cluster histogram computation is performed using the L2-norm as a distance measure. HS color histograms are used as basic features. Sliding window scales were set to {2,3,4}.

to achieve robustness against illumination changes. Since the HSV descriptor outperforms HS throughout the entire experiment as well as it has the highest peak performance, the omission of color information seems to have more impact on recognition performance, than the achievement of robustness to illumination changes. It shall be noted, that reducing a basic feature vector's size can affect the optimal amount of feature cluster histogram bins. This issue might explain the recognition results, we acquired in experiment 5.4, where different numbers of histogram bins caused different descriptor types to perform best.

The following test evaluates the connection between the total amount of features, that are used for the calculation of the feature cluster histogram bins, and the overall recognition rate. The experiment's outcome is illustrated in figure 5.8. Classification results, that were acquired by using HS color histograms as our basic feature descriptor, are almost unaffected by changes of the total feature number. Models, that are based on HSV histograms perform best, when using approximately 20000 features.

Until now, all tests have been performed with training samples, that are based on one video for each person. As we have gathered our training data from a sensor array consisting of six webcams, we will now evaluate the incorporation of multiple perspectives into our recognition system. Figure 5.9 and 5.10 show the

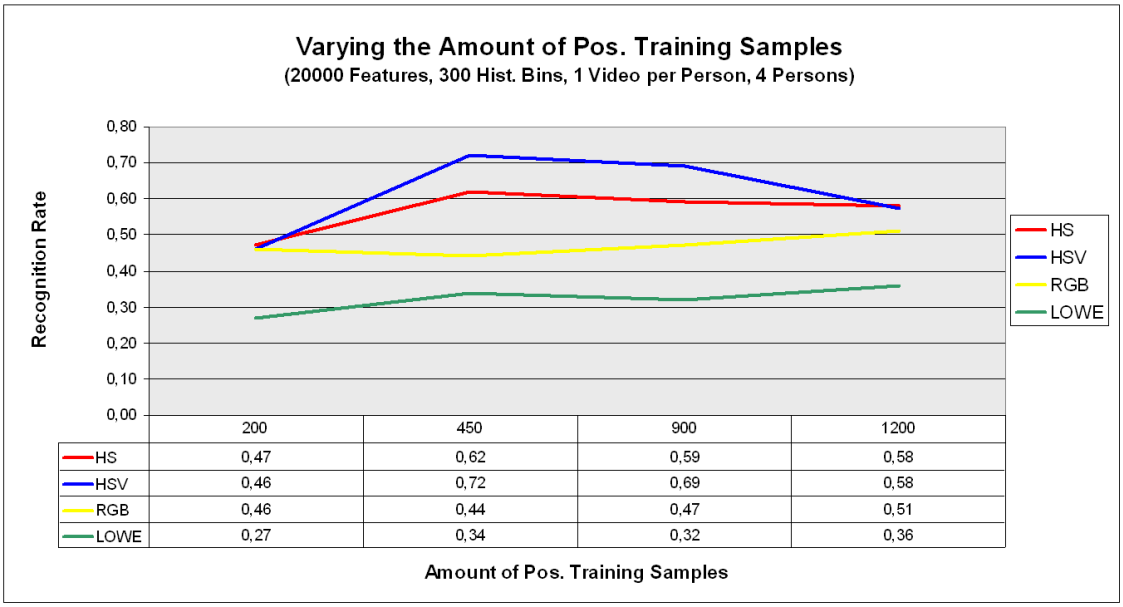


Figure 5.7: Recogniton Results for Various Amounts of Training Samples. Feature cluster histogram bins were created by using 20000 features as input for the k-means algorithm. The amount of feature cluster histogram bins has been set to 300. One video was used for each person. Models of four persons have been learnt. Feature cluster histogram computation is performed using the L2-norm as a distance measure. HS, HSV and RGB color histograms as well as the SIFT descriptor are used as basic features. Sliding window scales were set to $\{2, 3, 4\}$, the window's aspect ratios were $\{3/1, 2/1, 3/2\}$.

obtained classification results for one, two, or four perspectives of each person.

After having introduced multiple perspectives, we will take a closer look on the classifiers of each person. The average recognition rate for the classifiers of each test person in figure 5.9 and 5.10 is listed in table 5.1.

	Test Person A	Person B	Person C	Person D
Recognition Rate	28%	88%	30%	85%

Table 5.1: Average classification results for each person's model (data taken from figure 5.10 and 5.9)

Test person A and C seem to have significantly lower classification results, and are therefore performing poorly. A plausible reason for this behaviour lies in the data itself. We described in section 5.1.2, that one test person (C) is wearing a colorful, but untextured top. After an inspection of the associated learning videos, it became clear, that our basic feature detector is not able to detect keypoints in the test person's chest region. This effect is illustrated in figure 5.2

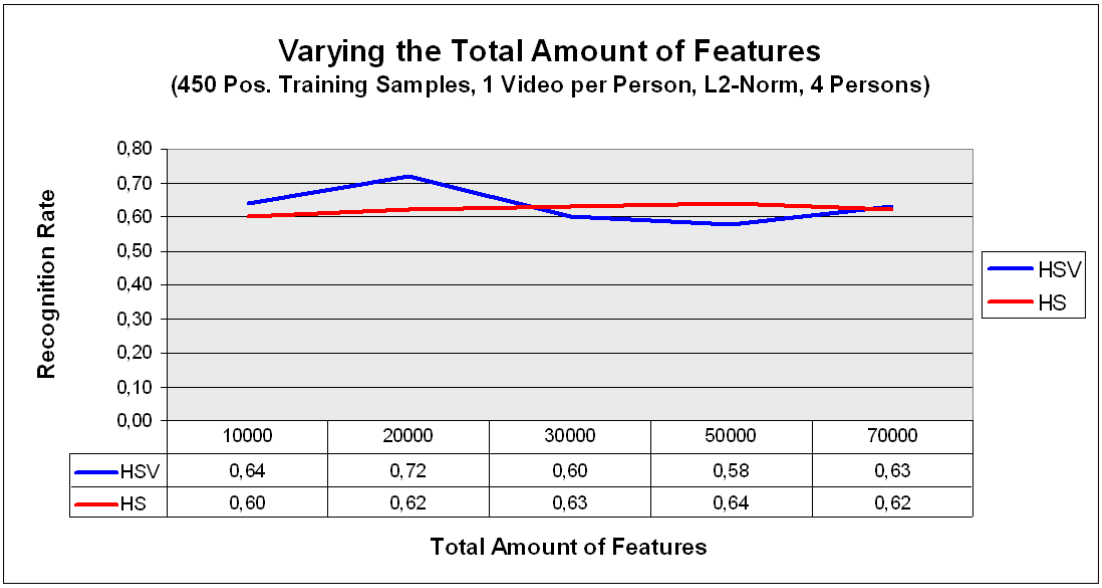


Figure 5.8: Varying the Total Amount of Features for K-Means Clustering. 450 positive and 450 negative training samples are used. One video was used for each person. Models of four persons have been learnt. Feature cluster histogram computation is performed using the L2-norm as a distance measure. Sliding window scales were set to $\{2, 3, 4\}$, the window's aspect ratios were $\{3/1, 2/1, 3/2\}$.

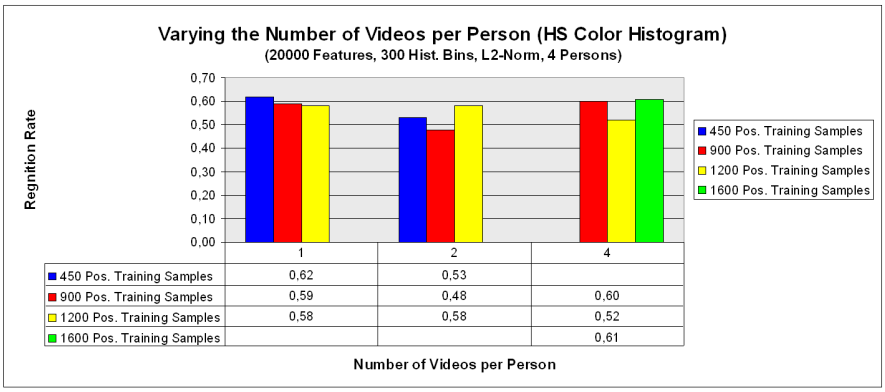


Figure 5.9: Varying the amount of videos per person as well as the amount of positive and negative training samples. Feature cluster histogram bins were created by using 20000 features as input for the k-means algorithm. Models of four persons have been learnt. Feature cluster histogram computation is performed using the L2-norm as a distance measure. HS color histograms are used as basic features. Sliding window scales were set to $\{2, 3, 4\}$, the window's aspect ratios were $\{3/1, 2/1, 3/2\}$.

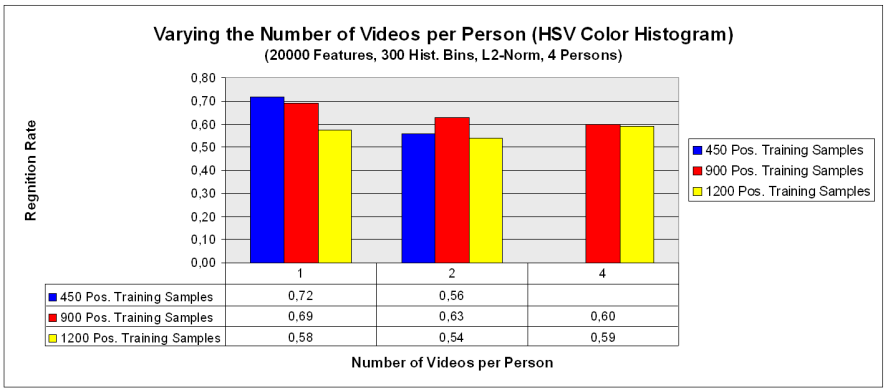


Figure 5.10: Varying the amount of videos per person as well as the amount of positive and negative training samples. Feature cluster histogram bins were created by using 20000 features as input for the k-means algorithm. Models of four persons have been learnt. Feature cluster histogram computation is performed using the L2-norm as a distance measure. HSV color histograms are used as basic features. Sliding window scales were set to $\{2, 3, 4\}$, the window's aspect ratios were $\{3/1, 2/1, 3/2\}$.

(a). The images of test person A, on the other hand, have partly been captured with the same background as is contained in the learning videos. Although the result of person A's classifier would indicate the person's presence on the image, other classifiers tend to have higher classification indices. In order to test our recognition approach without being influenced by these issues, we evaluated the performance of our recognition approach for two persons. The results are depicted in figure 5.11 and 5.12.

Finally, table 5.2 summarizes well performing parameter combinations.

5.2.2 Classification by using an Absolute Threshold

One disadvantage of our above presented classification method is, that it cannot handle images containing no, unknown or multiple persons. We can address this issue by introducing a global threshold t. This threshold defines, if a person is displayed in the respective test image or not. In detail, this means, we will compute the classification index for each learnt person model. If the computed classification index is larger than the global threshold t, then the associated person is assumed to be pictured.
Due to time limitations we were not able to evaluate this approach, thus this will be one topic of future work.

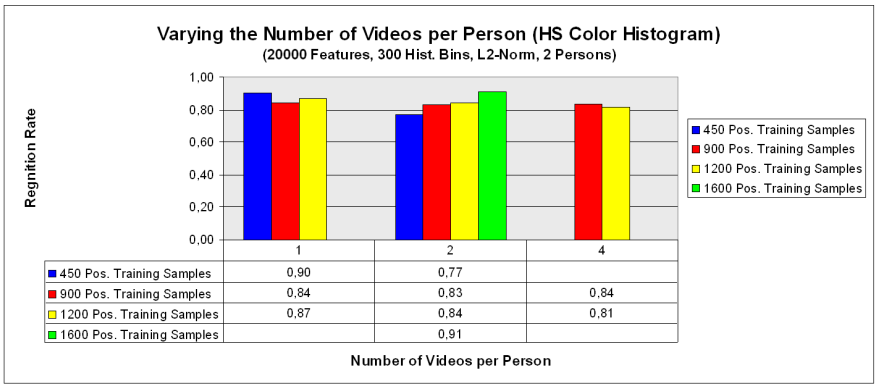


Figure 5.11: Varying the Amount of Videos per Person as well as the amount of positive and negative training samples. Feature cluster histogram bins were created by using 20000 features as input for the k-means algorithm. Models of two persons have been learnt. Feature cluster histogram computation is performed using the L2-norm as a distance measure. HS color histograms are used as basic features. Sliding window scales were set to {2,3,4}, the window's aspect ratios were {3/1, 2/1, 3/2}

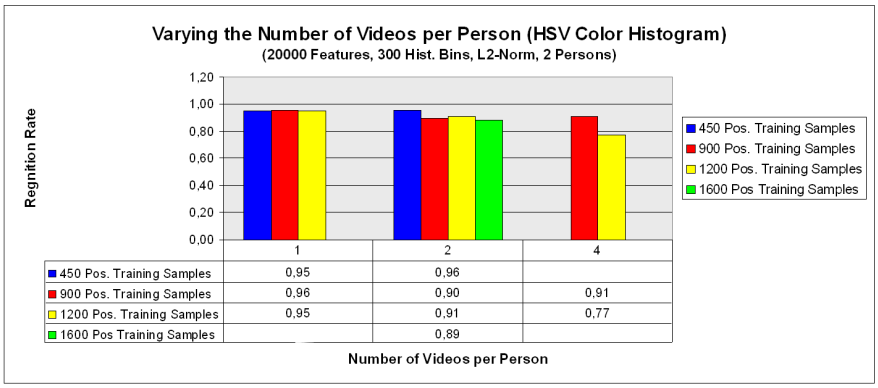


Figure 5.12: Varying the Amount of Videos per Person as well as the amount of positive and negative training samples. Feature cluster histogram bins were created by using 20000 features as input for the k-means algorithm. Models of two persons have been learnt. Feature cluster histogram computation is performed using the L2-norm as a distance measure. HS color histograms are used as basic features. Sliding window scales were set to {2,3,4}, the window's aspect ratios were {3/1, 2/1, 3/2}

# Persons	# Hist. Bins	# Videos/Person	# Pos. Tr. Samples	Feat. Type	Rec. Rate
4	300	1	450	HSV	72%
4	100	1	450	HS	68%
4	300	1	1200	RGB	51%
4	300	1	1200	LOWE	36%
4	300	2	1200	HS	58%
4	300	4	1600	HS	61%
4	300	2	900	HSV	63%
4	300	4	900	HSV	60%
2	300	1	450	HS	90%
2	300	2	1600	HS	91%
2	300	4	900	HS	84%
2	300	1	900	HSV	96%
2	300	2	450	HSV	96%
2	300	4	1200	HSV	91%

Table 5.2: Summary of Classification Results (without Unknown Persons)

Chapter 6

Conclusion

In this document, an approach for recognizing people in images has been described. The method is based on learning the outer appearance of persons from videos. Positive training data has been acquired by using a distributed camera network. Video data is labeled automatically in order to minimize user interaction. The videos are foreground segmented to detect the person of interest. Various features are extracted and combined to form a training data set for each person to be recognized. A statistical model of every person is created by using the Discrete AdaBoost algorithm. Persons are classified in test images by traversing the picture with a sliding window algorithm and classifying the bounded image region, using every person's model. These classification results are stored seperately in a classification matrix. The overall classification result is determined by calculating an index for every person's classification results. The person with the highest classification index is considered to be the person on the image. After the final result is determined, a position estimation of the person on the image is performed. Experimental results indicate the proposed approach to perform well.

Our algorithms can be improved in several ways. For future work, it is intended to use keypoints only, when at least a certain percentage of the area, that is described by each keypoint, is covered by the segmentation mask.

Histogram creation can be improved without increasing the data volume by trading the least significant bits of every entry for more histogram bins. Smooth transitions between similar histograms can be incorporated for color histograms and feature cluster histograms as described for the LOWE descriptor in section 3.3.1.

Different machine learning algorithms can be tested to reduce the impact of noisy training data (e.g. due to a badly segmented foreground) on classification results. In order to implement the rejectance of images of persons, that were not learnt, the final result can be verified by a nearest neighbour distance based threshold. If the classification index of another person is too close to the index of the recognized person, the test image will be classified "unknown". Finally, computation

time can be reduced significantly by dropping window scales and aspect ratios, that do not improve classification performance.

Bibliography

[1] Bernd Heisele Chikahito Nakajima, Massimiliano Pontil and Tomaso Poggio. People recognition in image sequences by supervised learning. Technical Report CBCL-188, MIT Artificial Intelligence Laboratory, June 7 2000.

[2] Intel Corp. *Open Computer Vision Library*, 2006. http://www.sourceforge.net/projects/opencvlibrary.

[3] Philips Electronics. *Technical Specifications SPC 600NC/00*, 2006. http://www.p4c.philips.com/files/s/spc600nc_00/spc600nc_00_pss_eng.pdf.

[4] UPnP Forum. *UPnP Device Architecture*, 2000. http://www.upnp.org/download/UPnPDA10_20000613.htm.

[5] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[6] X. Miralles J. Lluis and O. Bastidas. Reliable real-time foreground detection for video surveillance applications. Technical report, 2005. In VSSN '05: Proceedings of the third ACM international workshop on video surveillance & sensor networks.

[7] A.K.C. Wong J.N. Kapur, P.K. Sahoo. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics and Image Processing*, 29:273–285, 1985.

[8] R. A. Olshen und C. J. Stone L. Breiman, J. H. Friedman. *Classification and regression trees*. Wadsworth International Group, Belmont, CA, 1984.

[9] N. Krüger C. von der Malsburg L. Wiskott, J.-M. Fellous. *Face Recognition by Elastic Bunch Graph Matching*. CRC Press, 1999.

[10] Liyuan Li, Weimin Huang, Irene Y. H. Gu, and Qi Tian. Foreground object detection from videos containing complex background. In Lawrence A. Rowe, Harrick M. Vin, Thomas Plagemann, Prashant J. Shenoy, and John R. Smith, editors, *ACM Multimedia*, pages 2–10. ACM, 2003.

[11] R. Lienhart. Vssn 2005 open source algorithm competition. In *VSSN 2005*, 2005. In VSSN '05: Proceedings of the third ACM international workshop on video surveillance & sensor networks.

[12] T. Lindeberg. *Discrete Scale-Space Theory and the Scale-Space Primal Sketch*. PhD thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, 1991.

[13] David G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, Corfu Greece, 9 1999.

[14] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[15] K.-F. Kraiss M. Hähnel, D. Klünder. Color and texture features for person recognition. In *International Joint Conference on Neural Networks IJCNN 2004*, 2004.

[16] Microsoft. *Universal Plug and Play in Windows XP*, 2001. http://www.microsoft.com/technet/prodtechnol/winxppro/evaluate/upnpxp.mspx.

[17] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition (CVPR-99)*, pages 246–252, Los Alamitos, June 23–25 1999. IEEE.

[18] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[19] www.wikipedia.com. *Universal Plug and Play*. http://en.wikipedia.org/wiki/Upnp.

[20] R. E. Schapire Y. Freund. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.