Human Detection and Action Recognition in Video Sequences - Human Character Recognition in TV-Style Movies

Alexander Klaser

Human Detection and Action Recognition in Video Sequences

Human Character Recognition in TV-Style Movies

Alexander Kläser

A thesis submitted to the

Department of Computer Science

Bonn-Rhein-Sieg University of Applied Sciences

Sankt Augustin, Germany

in partial fulfillment of the requirements for the degree

Master of Science in Computer Science

Sankt Augustin, October 13, 2006

Prof. Dr. Rainer Herpers

Thesis Advisor, Bonn-Rhein-Sieg University of Applied Sciences

Dr. Cordelia Schmid

Thesis Advisor, INRIA Rhônes-Alpes

Declaration

I hereby declare, that the work presented in this thesis is solely my work and that to the best of my knowledge this work is original, except where indicated by references to other authors. No part of this work has been submitted for any other degree or diploma.

Alexander Kläser

Abstract

This master thesis describes a supervised approach to the detection and the identification of humans in TV-style video sequences. In still images and video sequences, humans appear in different poses and views, fully visible and partly occluded, with varying distances to the camera, at different places, under different illumination conditions, etc. This diversity in appearance makes the task of human detection and identification to a particularly challenging problem. A possible solution of this problem is interesting for a wide range of applications such as video surveillance and content-based image and video processing.

In order to detect humans in views ranging from full to close-up view and in the presence of clutter and occlusion, they are modeled by an assembly of several upper body parts. For each body part, a detector is trained based on a Support Vector Machine and on densely sampled, SIFT-like feature points in a detection window. For a more robust human detection, localized body parts are assembled using a learned model for geometric relations based on Gaussians.

For a flexible human identification, the outward appearance of humans is captured and learned using the Bag-of-Features approach and non-linear Support Vector Machines. Probabilistic votes for each body part are combined to improve classification results. The combined votes yield an identification accuracy of about 80% in our experiments on episodes of the TV series "Buffy the Vampire Slayer".

The Bag-of-Features approach has been used in previous work mainly for object classification tasks. Our results show that this approach can also be applied to the identification of humans in video sequences. Despite the difficulty of the given problem, the overall results are good and encourage future work in this direction.

Contents

List of Tables

List of Figures

1 Introduction

This master thesis work presents results on human detection and human character recognition in TV-style video sequences. Chapter 1 is organized as follows. It begins with a general description of the goal of this master thesis (Section 1.1) and an overview of related work (Section 1.2). Then, Section 1.3 gives a detailed overview of the approach and introduces the main terminology used throughout the work. Main contributions are summarized in Section 1.4, and Section 1.5 concludes with an outline of the master thesis.

1.1 Context and Goal

Computers are used to perform repetitive and data intensive tasks. For these tasks, they are efficient and even more accurate than humans. Yet, computers are still limited for more intelligent tasks such as the analysis of visual scenes, speech recognition, or logical inference and reasoning – in other words, high-level tasks which we humans perform with so much ease.

The context of this master thesis work is the automatic analysis of semantic content in video sequences. Its exact focus lies on the recognition of human characters in TV-style video sequences. In order to recognize the same person at different time instances in a video, the outward appearance of the person has to be captured and learned with a model. This model is then used to evaluate detections of humans and to predict their identity.

Figure 1.1 shows typical example images from a TV-style movie. The images show humans in different poses and views, fully visible and partly occluded, with varying distances to the camera, at different places, and under different illumination conditions. The diversity in which humans appear makes the task of human detection and character recognition to a particularly challenging problem. TV-style movies provide an uncontrolled, realistic working environment for human detection and character recognition.

For any semantic analysis of video content, human detection and character recognition play inevitably an important role. Therefore, this work can be seen as preliminary to any kind of more semantic video analysis.

1.2 Related Work

Everingham et al. focus in [ESZ06] on the recognition of human characters in TV-style video sequences. To achieve this, they fuse multiple data sources. Movie transcripts

(a) (b) (c) (d) (e) (f)

Figure 1.1: *TV-style example images from an episode of the series* Buffy the Vampire Slayer*; the images show (a,b) typical frontal/lateral poses of people, (c,d) distant/close-up views, and (e,f) different illumination conditions.*

and subtitles provide information about the characters' identity. A frontal face detector and tracker localizes the characters in the video frames. Face and clothing matching provide additional cues. They report good results on episodes of the TV-series *Buffy the Vampire Slayer*.

In most of the previous work, face detections provide the main information for any kind of character recognition or retrieval in TV-style movies. Fitzgibbon and Zisserman [FZ02] define an affine invariant distance measure on faces. Based on this distance measure, all detected faces in one movie are clustered. The resulting clusters represent the main characters in a movie. A similar face-based approach is introduced by Arandjelović and Zisserman in [AZ05]. They retrieve occurrences of a particular character in the movie based on user-specified query images.

Berg et al. [BBE+04] operate in a different context which is, however, still strongly related to our task. They aim at grouping images from web news pages according to the personality that is visible in the image. They perform a face-based matching and include additional information from extracted key words of the image caption.

1.3 Approach Overview and Terminology

We propose a supervised approach that can be split into two main parts and a pre-processing stage. In the first part, humans are detected at different time instances in a given movie. In the second part, the identity of each detected person is predicted based on learned models. In the following, this section gives a more detailed overview on the pre-processing stage and the two main parts and introduces the terminology that is used throughout this work.

Pre-processing. We work on extracted *full-frames* of video sequences. In order to structure the large amount of data, the extracted frames are grouped into *shots*. A shot is a continuous series of frames that runs for an uninterrupted period of time. *Cuts* mark the end of a shot in a video sequence and the beginning of the next one. They are detected by comparing color histograms of successive frames.

Human Detection. In TV-style video sequences, humans appear in different poses, perspectives, and with varying distances to the camera. In this work, we want to detect humans in frontal and side view (Figure 1.1(a) and (b)), and in resolutions where facial structures are clearly visible, i.e., from a person in upright pose with roughly 350 pixel in height (Figure 1.1(a) the person in the middle) up to *close-up views* where only the face is visible in the image (Figure 1.1(d)). Partly occluded people should also be detected. In order to detect humans within these constraints, our approach combines detectors of upper *body parts* which are *face*, *head*, and *head+shoulders*. For the *body part combination*, geometric relations between the body parts are learned by a model based on Gaussians. This approach is strongly inspired by the work of Mikolajczyk et al. [MSZ04].

The different body parts (face, head, and head+shoulders) are divided into *frontal* and *side view*. A detector is learned for each body part and each view (i.e., six detectors altogether). The *body part (view) detectors* are based on the detection system introduced by Dalal and Triggs [Dal06b, DT05]. Their system samples densely overlapping SIFT-like feature points which are trained with a linear Support Vector Machine. For the training of the body part detectors, we create a annotation data set.

Human Character Recognition. In order to recognize a particular *(human) character* in a given video sequence at different points in time, we propose a method that learns the character's appearance. This method takes into account that humans change in pose, perspective, and distance to the camera, that illumination conditions change, and the same character can wear different clothes.

The approach is based on the *Bag-of-Features* method [MS06, DWF+04, AR02] that has been used in previous work for object classification tasks. The Bag-of-Features method extracts *feature points* (i.e., image points that are described not necessarily by their color/intensity values, but by their local neighborhood based on, e.g., gradient information) from a set of training images. In their *feature space*, the feature points are grouped by a *clustering* algorithm. Based on the resulting *clusters* (all clusters together are referred to as *code book* and one cluster is referred to as *visual word*), *occurrence histograms* are generated for each body part image and a classifier is trained on these histograms. Occurrence histograms reflect how many feature points are assigned to each cluster.

For our implementation, we build SIFT- and CIE L*u*v* color-based code books by clustering with k-means. A *non-linear multi-class Support Vector Machine (SVM)* is learned on occurrence histograms of several main characters and on one category that consists of all other characters. The trained Support Vector Machines (or *SVM models*) are then used to predict the identity of a detected person. Probabilistic votes of *connected body parts* (i.e., body parts that belong to one and the same person) are combined for a more stable prediction. The training data is generated from our annotation data set in which the name of the corresponding character is noted for each body part.

1.4 Contributions

Our main contributions in this work include:

- Application of the human detector of Dalal and Triggs [DT05] to the detection of different body parts; performance study for different alignment methods, different view points, and different window sizes.
- Combination of the different body parts by modeling their geometric relations with Gaussians.
- Implementation and evaluation of the Bag-of-Features (BoF) approach for the recognition of human characters in a video sequence; study of the combination of a CIE L*u*v* color descriptor and a SIFT descriptor for the BoF-approach.

- Combination of probabilistic votes from connected body parts to improve the character recognition.

1.5 Outline of the Master Thesis

The master thesis is organized as follows. Chapter 2 focuses on the methodological background of this work. It introduces those methods that have been used in this work. The data sets for training and evaluation are presented in Chapter 3. The main approach that has been developed in this master thesis is described in two chapters: Chapter 4 introduces the human detection and Chapter 5 presents the human character recognition. The sixth chapter demonstrates final result of the entire system applied to an episode of the TV series "Buffy the Vampire Slayer". The thesis work is summarized in Chapter 7 by a conclusion and an outlook on possible future work.

2 Background

This chapter introduces methods which our approach builds on. Section 2.1 presents the general task of classification/detection and how the performance of classifiers/detectors is evaluated. Then, Support Vector Machines (Section 2.2) and clustering techniques (Section 2.3) are discussed. Finally, Section 2.4 and Section 2.5 introduce the Scale-Invariant Feature Transform (SIFT) and the CIE L*u*v* color space which are used in our approach.

2.1 Classification and Detection

In this work, machine learning methods are used to detect humans in images and to predict their identity (cf. Section 1.3). Therefore basic machine learning concepts and evaluation criterions are introduced in this section.

In image recognition, one can differentiate between *classification* and *detection*. A (binary) classifier assigns a given image to either a negative or a positive class. Such a classifier can predict, e.g., whether an image contains people or not. A *detector* not only classifies an image to contain a particular class of objects or not, it also *localizes* instances of the class in the image. For instance, a human detector predicts where exactly humans occur in an image. In the following, we discuss properties of classifiers. However, these properties apply usually also to detectors.

In this section, the terms "training data" (in Section 2.1.1) and "validation data" (in Section 2.1.2) are discussed first. Section 2.1.3 presents then various performance measures and discusses those in more detail that were used in this work .

2.1.1 Training Data

In Machine learning, a classifier is a more or less general system that is *trained* for a specific problem. For instance, a classifier can be trained to classify an image to contain particular objects such as buildings, cars, or humans. The *training data set* provides the *ground truth data* for the training. Instances of the training data set are referred to as *observations*, *patterns*, or *samples*. Since our data consists of images, we also use the term *training images* . In its training phase, the classifier seeks a hypothesis that explains the given observations. For example, a classifiers can be based on edge information. Characteristic edge structures in all training images (such as wheels for cars or eyes for humans) are part of a possible hypothesis that explains the data.

The quality of the training data plays an important role for the classifier's *quality* (or *reliability*, *performance*). This includes:

- How consistently the ground truth data is *annotated* (or *labeled*);
- How representative the data is;
- How the data is pre-processed before it is presented to the classifier.

The *size* of the training data set has also an influence on the final performance. Usually, the performance of a classifier increases with the size of the training data set. Yet, increasing the size of the training data set is always at a certain cost. A larger training set demands more computational resources (computational time and memory), and manually annotating data takes time. To a certain extend, it is possible to increase the size of the training data set in an unsupervised or semi-supervised fashion, e.g., by adding additionally scaled and rotated versions of original training images. How big the training data may be, the gain in performance saturates at a certain size. Therefore a trade-off between training size and performance has to be chosen.

2.1.2 Validation Data

The *performance* of a classifier refers to its *quality* or *reliability*. The performance can be thought of the classifiers' ability to generalize, i.e., to correctly classify new observations that are not included in the training set. In order to evaluate its performance, a *validation data set* (or *test data set*) with ground truth information is needed in addition to the training data set. This validation data set has to contain new observations that are not present in the training data set. Following [DHS00, BKL99], there are different types of validation strategies that split one set of ground truth data into a training and a validation set. These strategies are distinguished below.

Hold-Out Cross-Validation or Simple Validation. The ground truth data set is randomly split into one training set and one validation set. The classifier is trained on the training set, and its performance is evaluated on the validation set.

K-Fold Cross-Validation. From the ground truth data set, k (disjoint and equally sized) partitions are generated. The evaluation is carried out k times: Each partition is used as validation set for the classifier that has been trained on the remaining $k-1$ partitions. k performance results are obtained and combined by averaging. The k-fold cross-validation extents the concept of the holdout cross-validation. It can especially be applied to cases in which the number of observations in the ground truth data set is relatively small.

Leave-One-Out Cross-Validation, Jackknife, or Bootstrap Estimation. This is a special case of the k-fold cross-validation, where k is equal to the number of observations in the ground truth data set. One observation is used to evaluate, while the classifier is trained on all other remaining observations.

2.1.3 Performance Evaluation

Before presenting different measures that evaluate a classifiers's performance, certain variables are introduced in the following. The validation data set is denoted by $\mathbf{V}$. For the the case of a binary *classifier* $\mathcal{C}$, $\mathbf{V}$ can be divided into two subsets: The set of all positive observations, $\mathbf{P}$, and the set of all negative observations, $\mathbf{N}$, with $\mathbf{P} \cup \mathbf{N} \equiv \mathbf{V}$. For instance, in order to train a human classifier, all images containing humans are assigned to $\mathbf{P}$ and all images that do not contain any humans are assigned to $\mathbf{N}$.

$\mathcal{C}$ classifies an instance (or observation; or in our case image) of the validation dat set $\mathcal{I} \in \mathbf{V}$ as belonging to one of the two classes, i.e., being either positive or negative (e.g., containing humans or not). Its decision is based on certain properties of $\mathcal{I}$. $\hat{\mathbf{P}}$ is defined as the set that contains all predicted positive and $\hat{\mathbf{N}}$ as the set that contains all predicted negative instances. Since all instances of $\mathbf{V}$ are classified, the following equation holds

$$\hat{\mathbf{P}} \cup \hat{\mathbf{N}} \equiv \mathbf{P} \cup \mathbf{N} \equiv \mathbf{V}. \tag{2.1}$$

However, that does not necessarily mean that $\hat{\mathbf{P}}$ and $\hat{\mathbf{N}}$ are equal to $\mathbf{P}$ and $\mathbf{N}$, respectively. This would only be the case for a perfect classifier. For an instance $\mathcal{I} \in \mathbf{V}$ and a (binary) classifier $\mathcal{C}$, four classification results are possible [Faw03, Sec. 2]:

- *true positive*: $\mathcal{I}$ is correctly classified as positive (i.e., $\mathcal{I} \in \hat{\mathbf{P}} \cap \mathbf{P}$),
- *false positive*: $\mathcal{I}$ is classified as positive, although its correct label is negative (i.e., $\mathcal{I} \in \hat{\mathbf{P}} \cap \mathbf{N}$),
- *true negative*: $\mathcal{I}$ is correctly classified as negative (i.e., $\mathcal{I} \in \hat{\mathbf{N}} \cap \mathbf{N}$),
- *false negative*: $\mathcal{I}$ is classified as negative, although its correct label is positive (i.e., $\mathcal{I} \in \hat{\mathbf{N}} \cap \mathbf{P}$).

For $\mathcal{C}$ and $\mathbf{V}$, a two-by-two *confusion matrix* (or *contingency table*) can be constructed (see Figure 2.1). The classifications in the major diagonal of the confusion matrix (depicted in gray in Figure 2.1) represent the correct mappings, the classifications in the other fields represent the wrong mappings – the confusion – between the different classes. The sets for the true and false positives as well as the true and false negatives are defined as

$$\mathbf{TP} = \hat{\mathbf{P}} \cap \mathbf{P}, \qquad \textit{(true positives)} \tag{2.2}$$

$$\mathbf{FP} = \hat{\mathbf{P}} \cap \mathbf{N}, \qquad \textit{(false positives)} \tag{2.3}$$

$$\mathbf{TN} = \hat{\mathbf{N}} \cap \mathbf{N}, \qquad \textit{(true negatives)} \tag{2.4}$$

$$\mathbf{FN} = \hat{\mathbf{N}} \cap \mathbf{P}. \qquad \textit{(false negatives)} \tag{2.5}$$

From the confusion matrix (Figure 2.1), several measures can be derived for the performance evaluation[1]. The classifier's *true positive rate* (also known as *recall*, *detection*

[1] Often, the same measures are used in different contexts with different names. This can lead to some confusion.

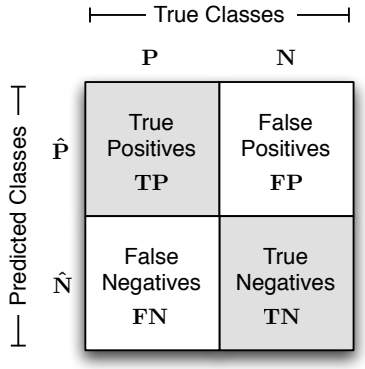


Figure 2.1: *An example of a confusion matrix (or contingency table);* $\mathbf{P}$ *and* $\mathbf{N}$ *denote the true classes of positives and negatives;* $\hat{\mathbf{P}}$ *and* $\hat{\mathbf{N}}$ *describe the classes as they are predicted by a classifier; the classifications in the major diagonal (depicted in gray) represent the correct mappings; the classifications in the other fields represent the wrong mappings (the confusion) between the different classes.*

rate, *sensitivity*, or *hit rate*) can be estimated as [Faw03, Sec. 2]

$$\textsc{tpRate} \approx \frac{|\mathbf{TP}|}{|\mathbf{TP} \cup \mathbf{FN}|} = \frac{|\mathbf{TP}|}{|\mathbf{P}|}. \tag{2.6}$$

The classifier's *false positive rate* (also known as *false alarm rate* or *fallout*) can be estimated as [Faw03, Sec. 2]

$$\textsc{fpRate} \approx \frac{|\mathbf{FP}|}{|\mathbf{TN} \cup \mathbf{FP}|} = \frac{|\mathbf{FP}|}{|\mathbf{N}|}. \tag{2.7}$$

Further measures are defined as follows [Faw03, Sec. 2]:

$$\begin{aligned}
\textsc{tpRate} &= \textsc{recall} = \textsc{sensitivity} = \textsc{hitRate}, \\
&= \textsc{detectionRate} = \textsc{classificationRate} && (2.8)\\
\textsc{fpRate} &= \textsc{falseAlarmRate} = \textsc{fallout}, && (2.9)\\
\textsc{specificity} &= \frac{|\mathbf{TN}|}{|\mathbf{FP} \cup \mathbf{TN}|} = \frac{|\mathbf{TN}|}{|\mathbf{N}|} = 1 - \textsc{fpRate}, && (2.10)\\
\textsc{missRate} &= 1 - \textsc{tpRate}, && (2.11)\\
\textsc{precision} &= \frac{|\mathbf{TP}|}{|\mathbf{TP} \cup \mathbf{FP}|} = \frac{|\mathbf{TP}|}{|\hat{\mathbf{P}}|}, && (2.12)\\
\textsc{accuracy} &= \frac{|\mathbf{TP} \cup \mathbf{TN}|}{|\mathbf{TP} \cup \mathbf{FP} \cup \mathbf{TN} \cup \mathbf{FN}|} = \frac{|\mathbf{TP} \cup \mathbf{TN}|}{|\mathbf{P} \cup \mathbf{N}|} = \frac{|\mathbf{TP} \cup \mathbf{TN}|}{|\hat{\mathbf{P}} \cup \hat{\mathbf{N}}|}, && (2.13)\\
\textsc{errorRate} &= 1 - \textsc{accuracy} && (2.14)
\end{aligned}$$

A *ranking* or *scoring classifier* returns a numeric value that measures the confidence with which a classified observation belongs to a certain class. These values can be

probabilities or uncalibrated scores (i.e., a likelihood). The performance of a classifier can then be evaluated for different threshold settings.

The following subsections discuss performance measures that were used in this master thesis.

Accuracy

Classification accuracy is a widely spread measure for the evaluation of machine learning systems (cf. [Alv02]). It relates the number of all true positives and false positives to the total size of the validation data set (i.e., number of observations in the validation data set, cf. Equation (2.13)). The accuracy thus gives equal weight to mislabeling of both types. Viewing the classifier as a system that labels observations as being either interesting or uninteresting, the accuracy represents an intuitive measure for the performance evaluation. Figure 2.2(b) illustrates the accuracy rate of an example distribution of positives and negatives. The accuracy rate is plotted for different threshold settings on classification scores.

Receiver Operating Characteristic (ROC) Curves

The *receiver operating characteristic* (ROC) curve plots the false positive rate (on the x-axis) versus the true positive rate (on the y-axis). An example ROC curve for a given distribution of positive and negative observations is shown in Figure 2.2(c). Figure 2.2(b) illustrates the corresponding true positive and false positive rate plotted against thresholds on classification scores. A very high true positive rate can be obtained at the cost of a very high false positive rate. On the other hand, a very low false positive rate can be obtained for a very low true positive rate. Generally, it can be said that the quality of a classifier increases with the separability of the distributions of positives and negatives in Figure 2.2(a).

An evaluation using ROC curves can be problematic especially for detection systems. A detector localizes instances of a class (e.g., humans for a human detector) in a given image. A possible detection scheme is a window of a fixed resolution which is shifted in the scale space of the image on a fixed grid – an illustration is given in Figure 4.3. At each position, the detection window is classified. The false positive rate includes the number of true negatives. For the mentioned detection scheme, the number of true negatives are the number of all inspected windows in all images of the validation set that do not contain positive instances. For a different resolution of the detection grid, the total number of negatives and consequently the false positive rate change. Since intrinsic parameters of the detection scheme (such as the resolution of the detection grid) have an influence on the false positive rate, it is somewhat difficult to interpret.

In this thesis work, we use either precision-recall curves (discussed in the following subsection) or we use ROC curves with the number of false positives (instead of the false positive rate). The number of false positives has the advantage that results of different approaches (and with different detection schemes) can directly be compared. The number of false positives can also be related to the number of images in the validation

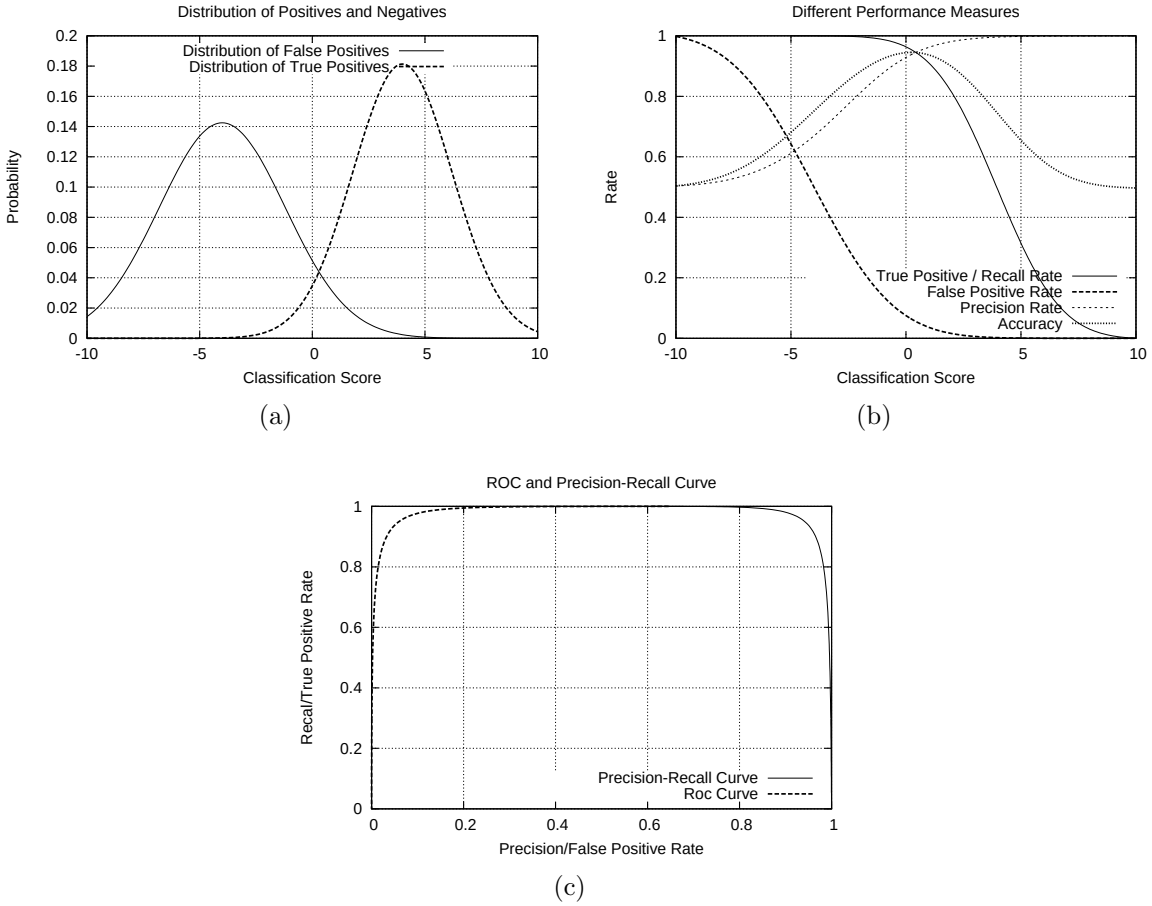


Figure 2.2: *Example distribution and its corresponding measures; (a) distribution of positive/negative observations for a scoring classifier and (b) corresponding performance measures; (c) corresponding ROC and precision-recall curve.*

data set. This yields a false positive rate per image which is also intuitive to evaluate.

Precision-Recall Curve

Precision-recall curves are used to evaluate the performance of information retrieval systems (cf. [VR79, Chap. 7]). An example curve is given in Figure 2.2(c). Precision-recall curves plot the recall rate (on the x-axis of the curve) against the precision rate (on the y-axis) . The recall rate is equal to the true positive rate. It describes the number of (validation) observations classified as positives with respect to the total number of positives. The recall rate penalizes false negatives, but not false positives. The precision rate, on the other hand, measures the number of true positives relative to the total number of (validation) observations classified as positives (i.e., the sum of true positives and false positives). This measure penalizes the retrieval of false positives, but not false negatives.

Precision-recall curves do not include the number of true negatives; they only use the

total number of positives, the number of classified positives, and the number of true positives. As discussed in the subsection before, they are therefore well-suited for the evaluation of detectors.

2.2 Support Vector Machines

We use *Support Vector Machines* (SVMs) as classifier for the human detection and for the human character recognition (see also in the overview in Section 1.3). In this section we introduce basic and some more advanced concepts of SVMs.

Given two data sets with *feature points* (or *features*, *observations*) distributed in a high-dimensional *feature space*. The basic idea of a *linear SVM* is to separate the feature points of the two data sets by a hyperplane. It is assumed that the optimal hyperplane is the one that maximizes the margin to the closest points of the data sets. An example for the two dimensional case is illustrated in Figure 2.3(a). In case the data sets cannot be separated by a hyperplane (see Figure 2.3(b)), the linear SVM can be extended to a *non-linear SVM*. Here, the basic idea is to project the feature points into a different feature space with more dimensions than the original one. Then, in this feature space, it is possible to find a separating hyperplane.

Even though work on *Support Vector Machines* (SVMs) started in the late seventies, they have gained increasing attention primarily in the late nineties as a very powerful classification and regression tool (cf. [Bur98, Sec. 1]). One very powerful property of SVMs is that during training they *always* find the optimal solution (cf. [Bur98, Sec. 4.4]) – the training problem can be formulated as a convex problem. SVMs usually exhibit good generalization capabilities (cf. [Bur98, Sec. 6]). Empirical experiments have shown that the performance of SVMs is either similar or better than competing methods (cf. [Bur98, Sec. 1]).

In the following, the concept of linear Support Vector Machines is formally introduced (Section 2.2.1). Section 2.2.2 discusses how linear SVMs can be extended to non-linear problems and Section 2.2.3 concludes with concepts for multi-class SVMs.

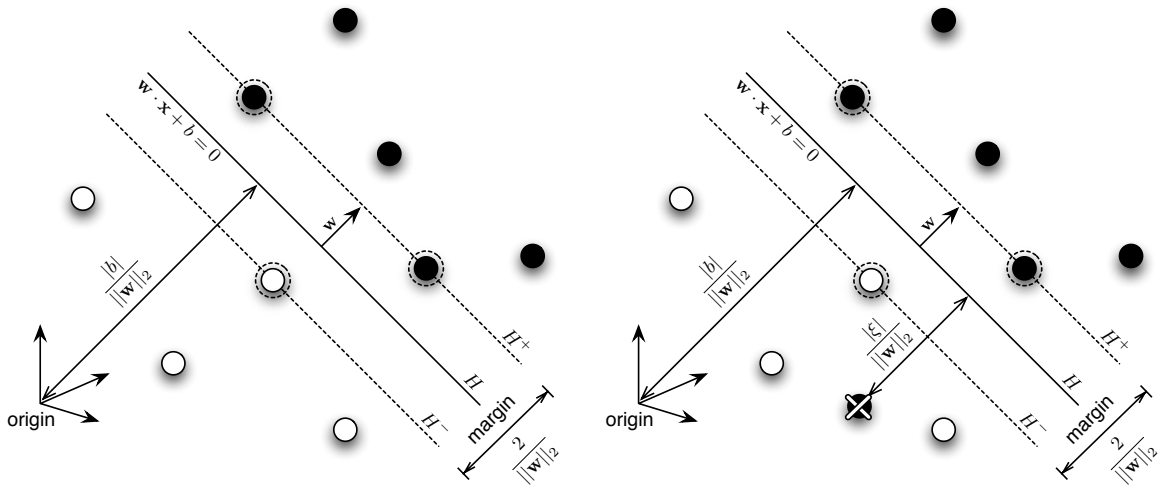
2.2.1 Linear SVM

Linearly Separable Data.

Given a binary classification task in an n-dimensional feature space $\mathcal{F} \equiv \mathbf{R}^n$, the i^{th} observation $\mathbf{x}_i$ in the feature space is labeled with $c_i \in \{+1, -1\}$. The label marks an observation either as belonging to a positive $(+1)$ or a negative (-1) class. It is assumed that the data points of the two classes are *linearly separable*, i.e., there exists a hyperplane H of the general form [Bur98, Sec. 3.1]

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2.15}$$

that divides the two classes perfectly. An example is given in Figure 2.3(a). $\mathbf{x}$ is a point in $\mathcal{F}$, $\mathbf{w}$ is the normal of the hyperplane, and the perpendicular distance from the



(a) Linearly separable. (b) Non-linearly separable.

Figure 2.3: *Examples for a (a) linearly separable and a (b) not linearly separable classification case; both figures illustrate how an SVM tries to separate the data by finding a hyperplane with maximal margin; the data points with an additional circle denote the support vector (this figure is similar to [Bur98, Fig. 5 and 6]).*

hyperplane to the origin is $\frac{|b|}{||\mathbf{w}||_2}$. A parallel hyperplane H^+ is defined by shifting H such that the point (possibly also several points) $\mathbf{x}_i$ with $c_i = +1$ that is the closest one to H lies exactly on H^+. Accordingly we define H^- as the hyperplane parallel to H such that the closest point $\mathbf{x}_j$ with $c_j = -1$ lies on H^-. H^+ and H^- can be described by the equations [Bur98, Sec. 3.1]

$$\mathbf{w} \cdot \mathbf{x} + b = +1 \tag{2.16}$$
$$\mathbf{w} \cdot \mathbf{x} + b = -1. \tag{2.17}$$

The points lying on H^+ and H^- are called the *supporting vectors*. They are depicted in Figure 2.3 with an additional circle. They are the most informative data points – in fact, training an SVM with only the support vectors will yield exactly the same result. The following constraints hold [Bur98, Sec. 3.1]:

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq +1, \qquad \forall \mathbf{x_i} : c_i = +1, \tag{2.18}$$
$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1, \qquad \forall \mathbf{x_i} : c_i = -1, \tag{2.19}$$

i.e., all observations lie on the correct side of H^- or H^+. The *margin* (see Figure 2.3) between the two hyperplanes is given by

$$\frac{|1-b|}{||\mathbf{w}||_2} - \frac{|-1-b|}{||\mathbf{w}||_2} = \frac{2}{||\mathbf{w}||_2}. \tag{2.20}$$

The concept of SVMs assumes that a new, unknown data point is most accurately predicted when the separation between the two classes is maximal. In order to maximize

the margin between the two hyperplanes (H^+ and H^-), $||\mathbf{w}||_2$ has to be minimized. This problem can formulated as convex maximization problem, i.e., the optimal solution is guaranteed to be found (cf. [Bur98, Sec. 3.1]). The solution of the maximization problem is referred to as *training*. A trained SVM classifier is also called a trained *SVM model*.

Non-Linearly Separable Data.

For a binary classification problem which is *non-linearly separable* (an example is given in Figure 2.3(b)), a hyperplane that separates both data sets in the feature space does not exist. Then, the constraints in Equations (2.18) and (2.19) can be relaxed by introducing for each observation $\mathbf{x}_i$ a slack variable ξ_i that penalizes a misclassification of $\mathbf{x}_i$. In this manner, misclassifications are possible, yet each misclassification is penalized by the slack variables. Equations (2.18) and (2.19) expand then to [Bur98, Eq. 40-42]

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq +1 - \xi_i, \qquad \forall \mathbf{x_i} : c_i = +1, \tag{2.21}$$

$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 + \xi_i, \qquad \forall \mathbf{x_i} : c_i = -1, \tag{2.22}$$

$$\xi_i \geq 0, \qquad \forall i. \tag{2.23}$$

An upper bound on the number of errors is given by $\sum_i \xi_i$. The optimization problem is equal to the one for the linearly separable case with an additional penalty term for misclassifications. Thus, object to minimization is [Bur98, Sec. 3.5]

$$\frac{||\mathbf{w}||_2}{2} + C \sum_i \xi_i, \tag{2.24}$$

with C as weight for misclassifications. A larger C results in higher penalties for errors.

2.2.2 Non-Linear SVM

For a non-linearly separable set of observations, a hyperplane to separate the observations does not exist in the feature space $\mathcal{F}$. However, if the observations are projected into a different feature space with a higher number of dimensions than $\mathcal{F}$, a hyperplane that splits the data sets perfectly can be found. For this, the linear SVM is extended to a non-linear SVM by applying what is known as the *kernel trick* which is explained in the following.

Kernel Trick and Kernel Function

The SVM-training is based on the minimization of Equation (2.24). This minimization problem can be formulated such that the dot product of two observations is the only operation that is applied to the data (cf. [Bur98, Eq. 43-45]). The dot product for two training observations is defined as:

$$\mathbf{x}_i \cdot \mathbf{x}_j = \begin{pmatrix} x_{i,1} \\ \vdots \\ x_{i,n} \end{pmatrix} \cdot \begin{pmatrix} x_{j,1} \\ \vdots \\ x_{j,n} \end{pmatrix} = x_{i,1}x_{j,1} + x_{i,2}x_{j,2} + \ldots + x_{i,n}x_{j,n}. \tag{2.25}$$

Given a function φ that maps the original feature space $\mathcal{F}$ to a different higher-dimensional space $\mathcal{H}$ (possibly also indefinite dimensional):

$$\varphi : \mathcal{F} \mapsto \mathcal{H}. \tag{2.26}$$

For the data projected into $\mathcal{H}$, the training algorithm only depends on the dot product of the data in $\mathcal{H}$, i.e.,

$$\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j). \tag{2.27}$$

Given a *kernel function* K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j), \tag{2.28}$$

only K needs to be known for the training. It is not necessary to know the explicit mapping φ. φ can actually be thought of as a (possibly very contorted) surface in $\mathcal{H}$, whose intrinsic dimension is that of $\mathcal{F}$ (cf. [Bur98, Sec. 4]).

The *kernel trick* replaces thus the dot product of two observations in the original feature space $\mathcal{F}$ by their dot product in a higher-dimensional space $\mathcal{H}$. The *kernel function* computes the dot product in $\mathcal{H}$, yet the actual mapping function from $\mathcal{F}$ to $\mathcal{H}$ for an observation can be unknown.

The RBF Kernel Function

A common kernel function is the *(Gaussian) Radial Basis Function* (RBF). It is defined as [Bur98, Eq. 60]

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|_2} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2}{2\sigma^2}}. \tag{2.29}$$

If the RBF kernel function is used, a new parameter σ (or γ) has to be chosen by the user. Its influence on the results can be described briefly as follows:

- *Small* σ: A smaller σ fits the observations more exactly; a σ which is too small overfits the data. Since the data is fitted more closely, more observations are included as support vectors.
- *Large* σ: A larger σ results in a model that fits the data less exact. Therefore single outliers have less influence on the separation of the data sets, i.e., the learned model is more robust. For a larger σ, less support vectors are included.

2.2.3 Extension to Multi-Class

Different strategies exist to extended the SVM to the classification of multiple classes (cf. [CL01, Sec. 6]). Strategies that are commonly used are the *one-against-all* (or *one-against-the-rest*, *one-versus-rest*) and the *one-against-one* (or *one-versus-one*).

The one-against-all strategy trains for k different classes k different models, i.e., one model for each class. For the model of class c, all observations that belong to class c are labeled as positive, and all observations of the remaining $k-1$ classes are labeled as negatives.

The one-against-one strategy trains for k classes $\frac{k(k-1)}{2}$ models, i.e., one model for any possible combination of two different classes. For the prediction, a testing point is designated to belong to the class with the maximum number of votes.

Hsu and Lin show in [HL01, Sec. 5B] that the performance of the one-against-one strategy performs better than one-against-all. They also report shorter training times for one-against-one. Although more models have to be trained for one-against-one ($\frac{k(k-1)}{2}$ vs. k), the size of the training sets is smaller since only samples of two different classes are used at one time.

2.3 Clustering

Clustering algorithms group together similar *feature points* (or *features*, *observations*, *patterns*) which lie in a common *feature space*. This *clustering* (i.e., grouping) is done on the basis of a homogeneity criterion. Points in one *cluster* (i.e., group) are more similar to each other than they are to points in other clusters [2]. In our work (see the overview in Section 1.3), clustering algorithms are used to generate visual code books, for instance, code books that consist of groups of similar structures in a set of images (example groups are shown in Figure 5.2). Generally, clustering methods can be useful for (cf. [JMF99, Sec. 1.1]): Exploratory pattern analysis, grouping, decision-making, and machine-learning situations (e.g., data mining, document retrieval, image segmentation, pattern classification). Since usually only little prior information is available on the data, only few assumptions on the data can be made.

This section is organized as follows. General steps that are part of any clustering method are presented in Section 2.3.1. Then, Section 2.3.2 overviews different clustering strategies. Since the k-means algorithm has been used in this work, it is introduced in detail in Section 2.3.3.

2.3.1 Typical Clustering Steps

Jan et al. [JMF99, Sec. 1.2] divide a typical feature clustering activity into the following abstract steps:

- The *feature representation* is the way in which features are encoded and presented to the clustering algorithm. This can involve *feature extraction*, i.e., transformations that are applied to the original features in order to obtain a different representation. Feature representation can also involve some kind of *feature selection*, i.e., the selection of a (more effective) subset of the original features. Since the feature representation is the basis for further steps, it can influence the quality of the final results significantly.

[2] The variability of points in one cluster (or class) is also referred to as *intra-class variability*, and the variability of points from different clusters is referred to as *inter-class variability*.

- The *feature proximity* is usually defined by some kind of distance function, i.e., metric for pairs of features. A common metric is the Euclidean distance between two features. However, there exist other metrics.

- Different methods exist to *cluster* (i.e., to group) a set features. A short overview over different methods is given in Section 2.3.2, and a detailed description of the k-means algorithm is given in Section 2.3.3.

- *Data abstraction* is an optional step. It extracts a compact representation of the clustering result. For instances, the result of a clustering algorithm can be represented by the centers of the clusters.

- The *assessment* of the obtained output is also optional. It analyzes clusters with respect to a particular quality criterion. For instance, a cluster can be deleted if it consists of a number of features below a given threshold.

2.3.2 Different Clustering Strategies

Combinatorially, clustering is a difficult problem. Therefore, different possible approaches to clustering have been proposed over the time. An extensive review of existing clustering techniques can be found in [JMF99]. Therein, Jan et al. distinguish two main clustering strategies: *hierarchical clustering* and *partitional clustering*. A combination of both strategies is referred to as *hybrid clustering* algorithms.

Hierarchical clustering methods yield a nested series of partitions. They try to find successive clusters based on the clusters that have been found in a previous step. By defining a threshold on the similarity between clusters, a particular number of clusters is obtained. Compared to partitioning algorithms, hierarchical algorithms are more versatile, yet at a higher computational cost (a time complexity of at least $O(n^2)$, with n being the size of the data set). The *single-link* and *complete-link* algorithm are two common hierarchical clustering algorithms.

Partitional clustering algorithms split the data set into a given number of output clusters. The difficulty is to determine the number of clusters which is in most cases unknown. Partitional techniques usually produce clusters by optimizing a *criterion function* defined either locally (on a subset of the patterns) or globally (on the whole data). Partitional clustering schemes tend to have a lower time complexity (common is $O(kn)$, with the total number of patterns n and the number of clusters k) and are therefore better suited for larger data sets. However, their clusters tend to be less versatile than those of hierarchical clustering methods. *k-means* (see Section 2.3.3) is a common example algorithm for partitional clustering algorithms.

Hybrid clustering schemes are interesting since they combine hierarchical and partitional clustering approaches and can yield more accurate clusters with a time complexity below $O(n^2)$. For instance, a possible hybrid clustering algorithm could first create very small clusters with k-means. In a second step this clusters could be combined by, e.g., the complete-link algorithm. The number of elements that the complete-link algorithm

needs to process would be significantly smaller than the number of all data points. Yet, the final clusters would be more versatile than clusters generated only with k-means.

Hierarchical, partitional, and hybrid describe one property of clustering approaches. In order to categorize and compare clustering algorithms, Jan et al. [JMF99, Sec. 5] mention further properties. We briefly introduce them in the following.

- *Agglomerative* vs. *Divisive*: An agglomerative algorithm starts with each feature as one cluster and then combines the clusters. A divisive method begins with one big cluster for all data points and splits it up into more clusters.

- *Monothetic* vs. *Polythetic*: Usually, all dimensions of a feature are used during the clustering process – this is referred to as polythetic. Monothetic methods perform the clustering on each dimension sequentially, i.e., they always work on one particular feature dimension at the same time.

- *Hard* vs. *Fuzzy*: A hard clustering algorithm assigns a feature to exactly one cluster. A fuzzy algorithm may assign a feature to several clusters. Therefore overlapping clusters are possible.

- *Deterministic* vs. *Stochastic*: This property is especially related to partitional approaches. The optimization of the criterion function can be carried out deterministically or in a random fashion (stochastic). Usually stochastic approaches are faster, yet results for one specific task vary in their quality.

- *Incremental* vs. *Non-Incremental*: Incremental algorithms minimize the number of scans through the data set, reduce the number of examined patterns, or reduce the size of data structures used in the algorithm. They are used in case the data set is too large such that memory and execution time constrain the clustering algorithm.

2.3.3 K-Means

The k-means clustering algorithm can be described as follows (cf. [JMF99, Sec. 5.2]). The number of clusters k has to be chosen in advance. In the beginning, k cluster center points are chosen, for instance, randomly. In one iteration, each feature is assigned to its closest cluster center. Then, the centers are recomputed based on all features that have been assigned to them. The procedure is repeated until convergence, i.e., until the assignment of features to clusters does not change anymore. Algorithm 1 gives an implementation of the k-means algorithm.

k-means is relatively fast ($O(kn)$ with n as number of features in the data set) and as clustering algorithm commonly used (cf. [JMF99, Sec. 5.2]). One of its limitations is that k-means is sensitive to the choice of the initial clusters. Additionally, k-means can only produce hyperspherical clusters (cf. [JMF99, Sec. 5.12]), i.e., information such as the variance of features in a cluster are not taken into account.

Algorithm 1 k-means

the data set of features is given as $F := \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}, \mathbf{f} \in \mathcal{F}$
choose (randomly) k cluster center points $C^{(0)} := \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}, \mathbf{c} \in \mathcal{F}$
$i := 0$
repeat
  $i := i + 1$
  reset the new cluster sets with $\hat{C}_j := \emptyset, 0 < j \leq k$
  for all $\mathbf{f} \in F$ **do**
    find the closest cluster $\mathbf{c}_j^{(i-1)}$ to $\mathbf{f}$, i.e., with
    $j := \arg\min_j \left\{ \left|\left|\mathbf{c}_j^{(i-1)} - \mathbf{f}\right|\right|_2 \mid \mathbf{c}_j^{(i-1)} \in C^{(i-1)} \right\}$
    add the pattern $\mathbf{f}$ to its corresponding cluster set by $\hat{C}_j := \hat{C}_j \cup \{\mathbf{f}\}$
  end for
  for all $\mathbf{c}_j \in C^{(i)}$ **do**
    the new cluster centers are given by the mean of the corresponding cluster set, i.e.,
    $\mathbf{c}_j := \frac{1}{|\hat{C}_j|} \sum_{\mathbf{f} \in \hat{C}_j} \mathbf{f}$
  end for
until $\mathcal{C}^{(i)} \equiv \mathcal{C}^{(i-1)}$

2.4 SIFT Descriptor

The SIFT (Scale Invariant Feature Transform) descriptor describes a point in an image by the structure of its neighborhood (i.e., local structure). The SIFT descriptor is based on local gradient information. Its idea is to define a representation that is well suited for the task of matching points in different images. In this thesis work (cf. the overview in Section 1.3), the concept of the SIFT descriptor has been used as well for the human detection (to learn characteristic structures/shapes of humans) as for the human character recognition (to find structures that are characteristic for particular humans). In the following, we briefly discuss the concept of descriptors and present the SIFT-descriptor in detail.

A *descriptor* can be defined as a technique that describes an image point by its local image neighborhood. Many different descriptors exist. For instance, descriptors can be based on color/gray level intensities, gray level/color histograms, gradient information (i.e., first order derivative), second or higher order derivatives, or frequency information. For instance, a vector of image pixels is a possible, basic descriptor. Descriptors of a local image region can be used for tasks such as image retrieval, object recognition, and texture recognition (cf. [MS05, Sec. 1]). Their advantage is that descriptors are distinctive, robust to occlusions, and they do not require any segmentation of the image (cf. [MS05, Sec. 1]), e.g., into object and background segments.

Different classes of descriptors can be summarized as (cf. [MS05, Sec. 2]):

- *Distribution Based Descriptors*: This class includes techniques that use histograms to represent different characteristics of appearance (e.g., gray and color values) and

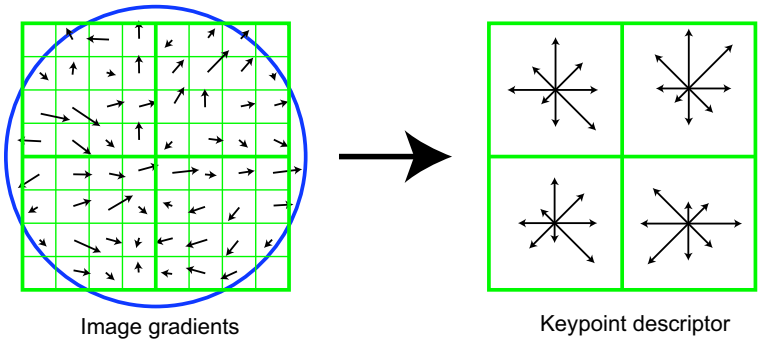


Figure 2.4: *The SIFT descriptor is computed by first computing the image gradient magnitude and orientation for each pixel (indicated by arrows of different length and orientation on the left side); the pixels are weighted by a Gaussian window (indicated by the circle); the weighted samples are accumulated into* 4×4 *pixel wide orientation histograms; the final descriptor (right part in the figure) consists of* 2×2 *orientation histograms with 8 orientation bins; the length of the arrows corresponds to the sum of gradient magnitudes (figure has been taken from [Low04, Fig. 7]).*

shape (e.g., gradient information).

- *Spatial-Frequency Descriptors*: Descriptors that use frequency information (e.g., Gabor filters and wavelets) belong to this class. They are frequently used in the context of texture classification.
- *Differential Descriptors*: They usually approximate the point neighborhood by a set of image derivatives computed up to a certain order.
- *Other Descriptors*: There exist also other descriptor types such as generalized moment invariants.

As descriptor, the *Scale Invariant Feature Transform* (SIFT) (cf. Section 2.4) has been developed recently for tasks such as image matching and object recognition (cf. [Low04, Abstract]). Mikolajczyk and Schmid evaluate in [MS05] the performance of local descriptors in the presence of real geometric and photometric transformations. They conclude that the SIFT descriptor and an extended version of the SIFT descriptor obtain best results in most tests. In the following, we give an overview of how the descriptor is computed.

Figure 2.4 illustrates the computation for a given image region. The computation for a $n \times n$ pixel image region can be summarized according to [Low04, Sec. 6.1] as follows:

(1) The image gradient magnitudes and orientations are computed for the given image region.

(2) A Gaussian with $\sigma = \frac{n}{2}$ weights each gradient magnitude in the region (illustrated by the circle in Figure 2.4). The Gaussian is center at the center point of the region. This is done in order to make the descriptor robust to small translational changes and to give less emphasis on gradients that are far away from the center.

(3) The region is divided into $c \times c$ cells, each cell has a resolution of $p \times p = \frac{n}{c} \times \frac{n}{c}$ pixels. The image content of a cell is represented as *Histogram of Oriented Gradients* (HOG). Each HOG divides the orientation, i.e., 360° into r histogram bins of $\frac{360°}{r}$ in width. All pixels in a cell contribute to the HOG of the cell. The gradient orientation α of a pixel determines the HOG bin into which the pixel votes. For smoother results, the pixel votes into its two closest HOG bins β_1, β_2 with $\beta_1 \leq \alpha < \beta_2$; its gradient magnitude g is assigned to the two bins proportionally to the gradient's distance to each bin orientation. That is, the gradient magnitude g_1 for the bin β_1 and the magnitude g_2 for the bin β_2 are

$$g_1 = g\left(1 - \frac{\alpha - \beta_1}{\beta_2 - \beta_1}\right) = g\left(\frac{\beta_2 - \alpha}{\beta_2 - \beta_1}\right) \tag{2.30}$$

$$g_2 = g\left(1 - \frac{\beta_2 - \alpha}{\beta_2 - \beta_1}\right) = g\left(\frac{\alpha - \beta_1}{\beta_2 - \beta_1}\right). \tag{2.31}$$

g_1 and g_2 are for each pixel additionally weighted by the value of the Gaussian at the pixel position. Figure 2.4 uses 2×2 cells with a size of 4×4 pixels and 8 orientation bins.

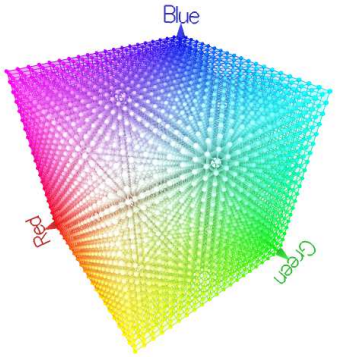
(4) The bin values of all HOGs in the given image region are collected in a final descriptor vector. Its dimension is rc^2, i.e., 32 for the example in Figure 2.4. In order to overcome non-linear illumination changes introduced by, e.g., camera saturation or 3D surfaces with differing orientations by different amounts, the following normalization procedure is applied. First, the descriptor vector $\mathbf{v}$ is normalized to unit length, i.e., divided by $||\mathbf{v}||_2$. Second, values larger than a given threshold t (e.g., $t = 0.2$) are clipped to t. Third, the clipped descriptor vector is normalized to unit length again.

Lowe carried out experimental evaluations of different descriptor parameters and reports that values of $c = 4, r = 8$ proofed to be a good choice [Low04, Sec. 6.2]. As application, he investigated an image retrieval system based on on the SIFT-descriptor.

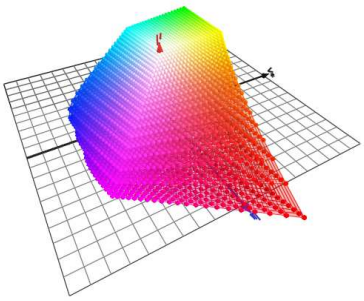
2.5 CIE L*u*v* Color Space

In this thesis work, color information is used explicitly (a) for the shot detection (see Section 3.2) in order to detect abrupt changes in the camera perspective as well as (b) for the human character recognition (see Chapter 5) in order to find characteristic colors for particular characters. For both tasks, CIE L*u*v* has been used as color model. In the following, we motivate this decision and give the transformation between RGB and CIE L*u*v* .

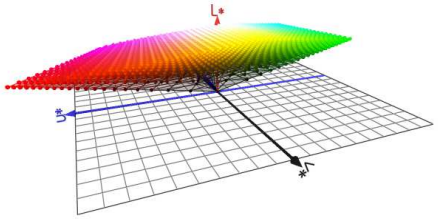
The CIE L*u*v* color space (developed by the International Commission on Illumination or Commission Internationale de l'Eclairage, CIE) is designed to be uniform to the human perception of color, i.e., it is *perceptually uniform*. A distance of 1 in this color space is the smallest difference which an average visual system of a human is able



(a) RGB color space.



(b) RGB cube transformed into the CIE L*u*v* color space.



(c) RGB cube transformed into the CIE L*u*v* color space.

Figure 2.5: *The (a) RGB color space and its transformation into the (b,c) CIE L*u*v* color space, which is shown from two different view points (generated by applying the software RGB Cube [Col06]).*

to distinguish. In the CIE L*u*v* color space, u^* and v^* describe the actual color, and L^* the luminance. Figure 2.5 illustrates that the dimensions u^*, v^*, L^* are weighted differently, i.e., the color space is elongated along u^* and v^*. Due to this elongation, the color space results in a stronger emphasis on the color information. Since the color and luminance information is separated and weighted according to a real model (the human visual perception), the CIE L*u*v* color space is interesting for tasks such as clustering (as it is used for the character recognition) and cut or shot detection based on color histograms.

Our implementation uses the *OpenCV Library* [ope06] to transform RGB values into the CIE L*u*v* color space. The color transformation is done as follows. First, the RGB values are translated into the CIE XYZ color space. This transformation is linear and can be done by a matrix multiplication [ope06, Func. cvCvtColor()]:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \qquad R, G, B \in [0; 1]. \tag{2.32}$$

Note that the exact conversion values depend on the underlying analogue television system (e.g., PAL, SECAM, NTSC) which defines the exact color encoding. The L^*, u^*, and u^* values are then computed as [FR98, Sec. 5.4], [ope06, Func. cvCvtColor()]

$$\begin{aligned} L^* &= \begin{cases} 903.3 \cdot Y/Y_n, & Y/Y_n \leq 0.008856 \\ 116 \sqrt[3]{Y/Y_n} - 16, & \text{else} \end{cases}, \qquad Y_n = 100 \\ u^* &= 13L^* \cdot (u' - u_n), \qquad u_n = 0.19793943 \\ u' &= \frac{4X}{X + 15Y + 3Z} \\ v^* &= 13L^* \cdot (v' - v_n), \qquad v_n = 0.46831096 \\ v' &= \frac{9Y}{X + 15Y + 3Z}. \end{aligned} \tag{2.33}$$

u_n and v_n refer to the reference white point of the light source.

3 Data Set and Video Processing

This chapter presents first the video data that has been used for our experiments and explains how the annotation data set has been created (Section 3.1). Then, methods that were used for the shot detection are discussed in Section 3.2.

3.1 Data Set

Since supervised classification methods have been used for this thesis work (cf. the overview in Section 1.3), a *ground truth data set* needs to be created in order to train and evaluate these methods. For the human detection, we propose a combination of detectors for different upper body parts (which are face, head, and head+shoulders) in frontal and side view. For the human character recognition, appearance models of the body parts are learned for the main characters. Therefore we annotate manually the different body parts and the corresponding character names in a set of full-frame images extracted from movies. In order to train the classifiers for the character recognition as well as the human body detectors, the annotated image regions need to be extracted from the full images and to be scaled to a common resolution.

This section first introduces the video material that has been used for the human detection and for the character recognition (Section 3.1.1). Section 3.1.2 explains how upper body parts are annotated in the ground truth data set and how the annotated image regions are extracted for training. The annotation tool that has been developed is briefly presented in Section 3.1.3, and Section 3.1.4 concludes with examples and statistics on our data set.

3.1.1 Video Material

Our annotation data set is based on manually extracted full-frame images (720×404 pixel in resolution) of the first two episode of *Buffy the Vampire Slayer Season 5*: "Buffy vs. Dracula" and "Real Me". Both episodes are 40 minutes long. The data from episode 2 of this season (we will refer to it as *Buffy2*) has been used for testing, whereas the data from episode 1 (which will refer to as *Buffy1*) has been primarily used for training (for the human body part detection), but also for testing (for the character recognition). Example images and statistics are presented in Section 3.1.4.

The videos contain a lot of very dark scenes as well as some running and fighting scenes. In fighting scenes, the human detection and character recognition is challenging due to occlusion and unusual poses. Dark scenes are challenging especially due to the

low contrast of image structures. The characters occur seldom in a full-view pose, much more frequent are views ranging from head and torso to close-up views.

One reason to chose *Buffy the Vampire Slayer Season 5* as data set was the fact that this data is used in the European CLASS project [cla06]. Using this data allows us to compare to other research projects carried out in this domain.

3.1.2 Data Annotation and Extraction

For the training of as well human body part detectors (for the human detection) as body part appearance models (for the character recognition), we need to provide body part images of a fixed width and height. In our image data, humans vary in scale, position, viewpoint, and pose. In order to obtain body part images of a common size, image regions that contain body parts have to be *annotated* (or *labeled*), extracted from the original images, and normalized in scale.

In our annotation data set, a body parts is marked by a *bounding box* (i.e., a rectangle) which holds information about scale and position of the body part. A set of *fix points* (i.e., single points in the image) on the body part gives additional information that is used to align the extracted images. The alignment keeps characteristic structures at a similar positions in all extracted images of a particular body part (cf. Section 4.3.2). Annotation examples with bounding boxes and fix points for different body parts are shown in Figure 3.3.

In the course of this section, the annotation strategy and the extraction of body part images from the annotation data is presented in more detail. At the end of this section, annotation examples are given and discussed.

Data Annotation

Given an annotated body part i, its bounding box is defined by its top-left and bottom-right corner

$$\mathbf{B}^{(i)} = \left(p^{(i)}_{min}, p^{(i)}_{max}\right), \quad p = (x, y), \quad x, y \in \mathbf{R}, \quad x_{min} \leq x_{max}, \quad y_{min} \leq y_{max}, \tag{3.1}$$

together with a set of $N^{(i)}$ fix points

$$\mathbf{F}^{(i)} = \left\{p^{(i)}_1, p^{(i)}_2, \ldots, p^{(i)}_{N^{(i)}}\right\}, \quad p = (x, y), \quad x, y \in \mathbf{R}. \tag{3.2}$$

For example, assuming an annotated body part i is given by:

- Top-left corner bounding box at $(50, 70)$, width and height of the bounding box are $(200, 100)$ and
- Two fix points with $(100, 100), (120, 150)$.

Then i is formally noted as

$$\mathbf{B}^{(j)} = ((50, 70), (250, 170)), \tag{3.3}$$

$$\mathbf{F}^{(j)} = \{(100, 100), (120, 150)\}. \tag{3.4}$$

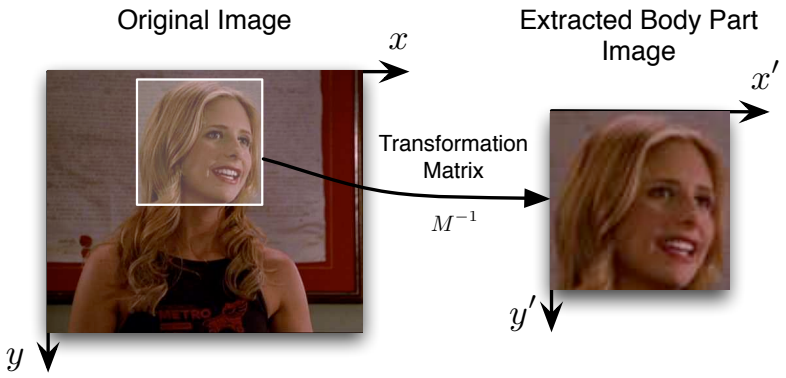


Figure 3.1: *Illustration of an annotated body part that is extracted from the original full-frame image into a new image; this transformation can be described by a* 3×3 *transformation matrix.*

Data Extraction

The extraction of a region that contains a body part in a full-frame image into a separate image (referred to as *body part image*) can be described by a transformation matrix. This is illustrated in Figure 3.1. The transformation matrix, denoted by M, maps a point from the body part image into the original image. In the following we show how M is computed.

First, all annotated bounding boxes of a particular body part (e.g., head) are normalized to a common scale. The normalization factor for a body part (annotation) is given by the height of its bounding box $\mathbf{B}^{(i)}$:

$$\alpha^{(i)} = y^{(i)}_{max} - y^{(i)}_{min}. \tag{3.5}$$

This choice is experimentally motivated (see Section 4.3.2).

In order to align all images, a *point of alignment* or *center point* is defined for each body part. This center point is given by the center of the annotated fix points. This choice is also experimentally motivated (see Section 4.3.2). The center point of the annotated body part i is defined as:

$$\bar{p}^{(i)} = \left(\bar{x}^{(i)}, \bar{y}^{(i)}\right), \quad \bar{x}^{(i)} = \frac{1}{N^{(i)}} \sum_{(x,y)\in\mathbf{F}^{(i)}} x, \quad \bar{y}^{(i)} = \frac{1}{N^{(i)}} \sum_{(x,y)\in\mathbf{F}^{(i)}} y. \tag{3.6}$$

Since the bounding boxes of a particular body part type (e.g., head) vary in their exact size from annotation to annotation, one common bounding box has to be defined. This is done by computing the smallest bounding box $\bar{\mathbf{B}}$ that covers the bounding boxes $\mathbf{B}^{(i)}$ of all annotated body parts that are extracted. $\bar{\mathbf{B}}$ is computed for all body parts

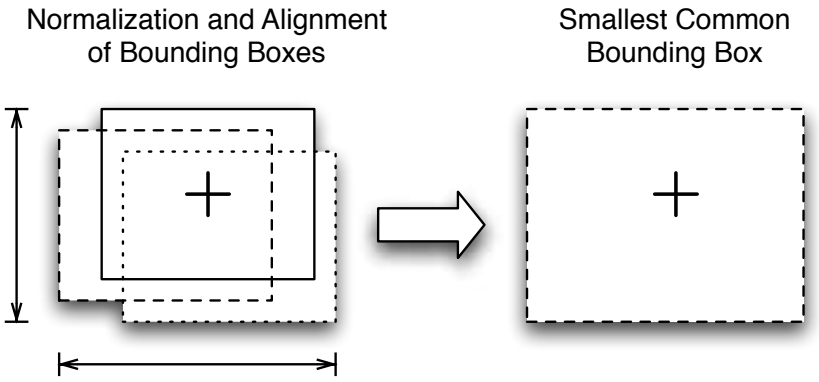


Figure 3.2: *Computation of a common bounding box; the bounding boxes of the extracted body parts are normalized in scale and aligned to their center point (indicated by the cross); the common bounding box is the smallest one that includes all normalized and aligned body part boxes.*

aligned at their center point $\bar{p}^{(i)}$ and normalized in scale by $\alpha^{(i)}$ (see also Figure 3.2):

$$\begin{aligned}
\bar{\mathbf{B}} &= ((\bar{x}_{min}, \bar{y}_{min}), (\bar{x}_{max}, \bar{y}_{max}))\,, \\
\bar{x}_{min} &= \min\left\{(x-\bar{x})/\alpha^{(i)} \middle| (x,y) = p^{(i)}_{min}, (\bar{x},\bar{y}) = \bar{p}^{(i)}\right\}, \\
\bar{y}_{min} &= \min\left\{(y-\bar{y})/\alpha^{(i)} \middle| (x,y) = p^{(i)}_{min}, (\bar{x},\bar{y}) = \bar{p}^{(i)}\right\}, \\
\bar{x}_{max} &= \max\left\{(x-\bar{x})/\alpha^{(i)} \middle| (x,y) = p^{(i)}_{max}, (\bar{x},\bar{y}) = \bar{p}^{(i)}\right\}, \\
\bar{y}_{max} &= \max\left\{(y-\bar{y})/\alpha^{(i)} \middle| (x,y) = p^{(i)}_{max}, (\bar{x},\bar{y}) = \bar{p}^{(i)}\right\}.
\end{aligned} \tag{3.7}$$

The body part image $\mathcal{O}^{(i)}$ is extracted from the original image $\mathcal{I}^{(i)}$ based on the annotation data as follows. A target height h is set for the body part image. The corresponding width w is computed from the bounding box $\bar{\mathbf{B}}$ as

$$w = \frac{\bar{x}_{max} - \bar{x}_{min}}{\bar{y}_{max} - \bar{y}_{min}} \cdot h. \tag{3.8}$$

For each extracted body part, the transformation matrix M can be computed. M maps a point $(x, y, 1)^\top$ from the body part image $\mathcal{O}^{(i)}$ into its corresponding point $(x', y', 1)^\top$ in the original image $\mathcal{I}^{(i)}$:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = M \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}. \tag{3.9}$$

For the annotated body part i, M is given by:

$$M = \text{scale}\left(\text{translate}\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \bar{x}^{(i)} + \alpha^{(i)} \cdot \bar{x}_{min}, \bar{y}^{(i)} + \alpha^{(i)} \cdot \bar{y}_{min}\right),\right.$$
$$\left.\min\left\{\frac{\hat{x}}{-\alpha^{(i)} \cdot \bar{x}_{min}}, \frac{w - \hat{x}}{\alpha^{(i)} \cdot \bar{x}_{max}}, \frac{\hat{y}}{\alpha^{(i)} \cdot -\bar{y}_{min}}, \frac{h - \hat{y}}{\alpha^{(i)} \cdot \bar{y}_{max}}\right\}\right), \quad (3.10)$$

with

$$\hat{x} = \frac{-w \cdot \bar{x}_{min}}{x_{max} - x_{min}}, \quad (3.11)$$

$$\hat{y} = \frac{-h \cdot \bar{y}_{min}}{y_{max} - y_{min}}. \quad (3.12)$$

The functions for translation and scaling are defined as

$$\text{translate}(M, x, y) = M + \begin{bmatrix}0 & 0 & x\\ 0 & 0 & y\\ 0 & 0 & 0\end{bmatrix}, \quad (3.13)$$

$$\text{scale}(M, \alpha) = \alpha \cdot M. \quad (3.14)$$

Using M, the pixel values for the object image $\mathcal{O}^{(i)}$ can be determined.

Annotation Examples

Figure 3.3 illustrates how different body parts are annotated for frontal and side views. As mentioned before, in order to keep characteristic structures (e.g., eyes, mouth, and nose) throughout all extracted body part images at a same position, a set of fix points is chosen. The center point of the fix points is the point of alignment. Also, multiple fix points define a more stable center point than, e.g., only one user-defined center point.

For the frontal head and face body parts (they are illustrated in yellow in Figure 3.3(a)), two fix points on the two eyes and one between mouth and nose are chosen. For the corresponding side views (in yellow, Figure 3.3(b)), only one fix point on the closest eye and one between nose and mouth are marked.

The head+shoulders frontal body part (illustrated in red in Figure 3.3(a)) is annotated with three fix points – two on each shoulder joint, the third below the larynx. The corresponding side view (in red, Figure 3.3(b)) has been annotated with three points as well: One on the shoulder joint, one on the middle of the neck, and the last one on the top of the head. In fact, head and shoulders are more difficult to align, since both parts are relatively independent in their movements. The head, on the other hand, is much more rigid. Due to a person's arms, the shape and structure of the upper torso is highly variable. Therefore not more information than head and shoulders down to the shoulder joints is included in the bounding box.

(a) Annotation example of face, head, head+shoulders object parts in frontal view.

(b) Annotation example of face, head, head+shoulders object parts in side view.

Figure 3.3: *Example annotations of the character Buffy; the different body parts (face, head, head+shoulders) are annotated with bounding boxes and fix points; the body parts face and head are depicted in yellow, head+shoulders in red; resolution of the images is* 400×350 *pixel.*

The annotation set consists mainly of views where facial structures have a resolution such that they are clearly visible in the image. The resolution for annotated humans vary from ca. 200 pixel height for person in upright pose up to close-up views where only the face of the person is visible in the image. The data set has mainly been annotated by one individual. Smaller deviations in the exact placement of the bounding boxes or in the exact positioning of the fix points are possible. However, since an exact alignment is generally not possible and since the final bounding box is computed over a set of body parts, these deviations do not have a great impact on the final results.

3.1.3 Image Annotation Tool

For the purpose of manual image annotation, we developed an image annotation tool. The most important features that were included in this tool are:

- Visually adjustable bounding boxes in order to improve the annotation data at a later date;
- Fix points for objects that can be added, deleted, and moved;
- String tags in order to describe body part properties more precisely (for annotating, e.g., character names or view direction);
- A flexible extraction of the annotated image regions combined with a filtering mechanism on the annotated string tags.

Figure 3.4 shows a snapshot of the main working window of the tool.

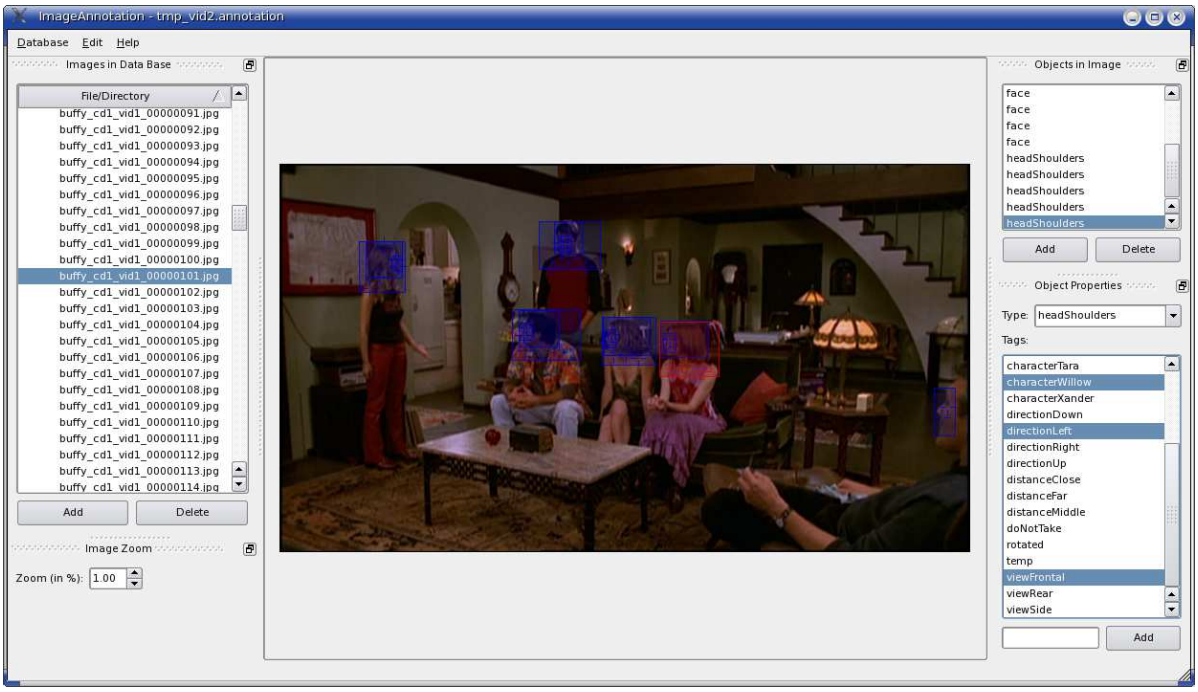


Figure 3.4: *A snapshot of the main window of our annotation tool with multiple annotated people in one image; the resolution of the annotation image showed in the application is* 720×404 *pixel.*

3.1.4 Example Images and Statistics

Tables 3.1, 3.2, and 3.3 present relevant statistics of our annotation data set. Table 3.1 gives the total number of images that were acquired from Buffy1 and Buffy2. Since the annotation data from Buffy2 is only used for testing purposes, it is significantly smaller. Table 3.2 gives the total number of body parts that were annotated. Therein, each body part (i.e., face, head, head+shoulders) is separated into two views: Frontal view and side view. Table 3.3 presents an overview on the number of body parts associated to different characters in the movie. These data has been used for the training and evaluation in Chapter 5. As can be seen in this figure, Buffy appears as main character most often. Willow, Riley, Xander, and Giles were the characters that appeared after Buffy the most frequently in our data set of Buffy1. The character class Others consists of all other characters that have been annotated in our data set. Figure 3.5 shows for each character extracted images of different body parts and views.

3.2 Shot Detection

A *shot* is a continuous series of frames that runs for an uninterrupted period of time. Since shots can be seen as elementary units in a video, they should be considered preliminary for any kind of high-level video processing or abstraction (e.g., scene detection; a scene consists of one or more shots). We use the shot segmentation in order to structure a large set of data by grouping single frames into shots.

A *hard cut* is a direct change between two shots, i.e., between the last frame of the one and the first frame of the next shot. In a hard cut, there are no additional edit frames and neither are transitional effects (e.g., fade-over) applied. Hard cuts are by far the most common type of transition effect (cf. [Lie01, Sec. 2]), also in the Buffy videos. Besides hard cuts, there are also *fade overs* to/from sequences of black frames in the videos Buffy1 and Buffy2.

Features that can be employed to detect transition effects are: Intensity/color histograms, edges/contours, and motion information. According to Lienhart [Lie01, Sec. 2.1.2, 2.1.3], color histograms are best suited for the detection of hard cuts. Therefore we implement a shot detection based on this approach. In the implementation, we use the OpenCV library [ope06] which provides a variety of image processing methods.

In the following, Section 3.2.1 introduces the hard cut detection in detail. The detection of fade-outs in video sequences is presented in Section 3.2.2. Section 3.2.3 concludes with evaluation results.

3.2.1 Hard Cut Detection

A video sequence is given by

$$\mathbf{i}(x, y, t) = \mathbf{i}^{(t)}(x, y). \tag{3.15}$$

For the hard cut detection, each RGB frame t is converted into the CIE L*u*v* color space (see also Section 2.5). The L^* values are discarded, and a u^* and v^* histogram

	Buffy1	Buffy2	Buffy1+2
Images	314	89	403

Table 3.1: *Total number of annotated images of our Buffy annotation data set; the images are full-frame images that have been manually extracted from two episodes of the series* Buffy the Vampire Slayer.

	Buffy1			Buffy2			Buffy1+2		
Body Parts	frontal	side	total	frontal	side	total	frontal	side	total
face	209	124	333	69	40	109	278	164	442
head	209	125	334	68	38	122	277	163	440
head+shoulders	173	150	323	63	45	108	236	195	431

Table 3.2: *Total number of annotated body parts of our Buffy annotation data set that have been annotated in all images in the set; some images contain annotations of several people.*

	Buffy1			
Characters	face	head	head+shoulders	total
Buffy	91	93	89	273
Willow	40	42	35	117
Riley	36	36	46	118
Xander	43	44	41	108
Giles	39	42	47	128
Others	84	90	102	276

Table 3.3: *Total number of annotated characters in our Buffy annotation data set; we only show the statistics for Buffy1 since Buffy2 has not been used for the character recognition part; note that we have sometimes more annotated heads than faces; in the annotation data set there are some cases where we annotated heads, but not the corresponding face since it was not visible (e.g., in Figure 3.5 last column, second row where the person's hair hides the face).*

Figure 3.5: *Examples of extracted body part images of our Buffy annotation data set; the images illustrate examples for body parts in different views (side and frontal) and for the character classes as they have been used in the human character recognition (cf. Chapter 5); the body part images have been normalized to a common height of 60 pixel.*

with 20 bins is built for each frame. The bins for the t^{th} frame are denoted as

$$\begin{aligned} h^{(t)} &= (u^{(t)} v^{(t)}) \\ u^{(t)} &= (u_1^{(t)} u_2^{(t)} \dots u_B^{(t)})^\top \\ v^{(t)} &= (v_1^{(t)} v_2^{(t)} \dots v_B^{(t)})^\top, \end{aligned} \tag{3.16}$$

with B as number of histogram bins ($B = 20$ for our case). In order to compare histograms of two consecutive frames, the correlation of the two histograms can be used. The correlation measures the linear relationship between the histograms. It is given by [ope06, Func. cvCompareHist()]:

$$\begin{aligned} d(h^{(t)}, h^{(t')}) &= \frac{\sum_{i=1}^{B} \sum_{j=1}^{C} \left(h_{i,j}^{(t)} - \bar{h}^{(t)}\right) \cdot \left(h_{i,j}^{(t')} - \bar{h}^{(t')}\right)}{\tilde{h}^{(t)} \cdot \tilde{h}^{(t')}}, \\ \bar{h}^{(t)} &= \frac{1}{B \cdot C} \sum_{i=1}^{B} \sum_{j=1}^{C} h_{i,j}^{(t)}, \\ \tilde{h}^{(t)} &= \sqrt{\sum_{i=1}^{B} \sum_{j=1}^{C} \left(h_{i,j}^{(t)} - \bar{h}^{(t)}\right)^2}, \end{aligned} \tag{3.17}$$

with C referring to the number of channels (which is here $C = 2$ since we only take the u^* and v^* channel).

Following [Lie01, Eq. 2], an adaptive ratio measure is introduced based on d, i.e., on the similarities of consecutive frames. The ratio measure is defined for a time window W (we use $W = 2$):

$$\text{RATIO}^{(t)} = \frac{d^{(t)} + c}{c + \sum_{-W \le i \le W, i \neq 0} d^{(t+i)}}, \qquad d^{(t)} = d(h^{(t)}, h^{(t-1)}). \tag{3.18}$$

c is a small constant (we use $c = 0.01$) to avoid numerical instabilities.

A hard cut detection before frame t (i.e., between frame $t-1$ and frame t) is detected when the two following criteria are fulfilled:

(1) $d^{(t)}$ is a maximum within the window W, i.e., within $d^{(t-W)}, \dots, d^{(t+W)}$, and

(2) $\text{RATIO}^{(t)} < \text{SHOTRATIOTHRESHOLD}$.

The threshold on the ratio measures is set to $\text{SHOTRATIOTHRESHOLD} = 0.96$ (cf. Section 3.2.3).

3.2.2 Fade-Out Detection

In order to detect black frames, a threshold (given by $\text{SHOTLTHRESHOLD} = 20$) is applied to the mean L^*-value (i.e., the luminance) of the current frame. In this way, fade-outs into a sequence of black frames can be detected.

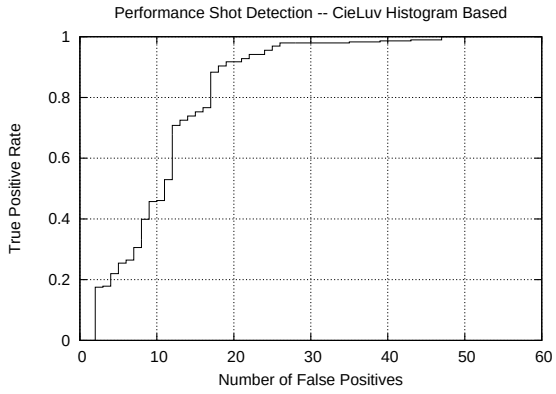


Figure 3.6: *Performance plot of our implemented CIE L*u*v* shot detection measured in a ROC curve; note that number of false positive detections is taken instead of the false-positive rate; the ground truth data consisted of 19000 frames including 291 shot transitions.*

3.2.3 Results and Conclusions

Our final shot detector combines the hard cut detection (Section 3.2.1) and the fade-out detection (Section 3.2.2). Since we use the shot detection only to structure our data, it has no influence on the final performance of our system. Therefore this basic shot detection suffices for our purpose.

In oder to evaluate the shot detection, we created a ground truth data set of manually annotated shots. The data set consists of 19000 frames (from Buffy1) with 291 annotated shots. Figure 3.6 depicts the performance of our shot detection. For the final detector, we use a threshold of SHOTRATIOTHRESHOLD = 0.96. This yields a true-positive rate of 96.9% and 25 false-positive detections on our evaluation set. This performance is reasonable for our application.

4 Human Detection

This chapter presents our approach to human detection. In order to detect humans in views ranging from full to close-up view and in the presence of clutter and occlusion, the approach is based on the assembly of several upper body part detectors. The body part detectors are trained using dense sampling of SIFT-like (see Section 2.4) feature points in a window as it is introduced by Dalal and Triggs in [DT05]. Detected body parts are combined with a model for geometric relations, similar to the approach presented by Mikolajczyk et al. in [MSZ04].

Our main contributions in this chapter include the application of the human detector from [DT05] to different body parts and the combination of the body parts using a geometric relations model. A further contribution is the performance study of different alignment methods and detectors that are trained for specific view points.

At the beginning of this chapter, related work and the state-of-the-art in human detection are reviewed (Section 4.1). Section 4.2 presents the architecture of the body part detector as is it is employed in our work. Last, Section 4.3 evaluates different parameter settings for the human detection and Section 4.4 presents final evaluation results of our detection approach. (Further results of the whole system are presented in Chapter 6.)

4.1 State of the Art

Since human detection is a challenging application with various potential applications (cf. [Dal06b, Sec. 1.1.1]), it has received a lot of attention in the computer vision and pattern recognition communities. A state-of-the-art human detector has been introduced by Dalal and Triggs [DT05] (for a detailed description see Section 4.2). They focus especially on the detection of people in upright pose that can be partly occluded. Their detector is based on the dense sampling of SIFT-like descriptors in a search window. After local normalization, the resulting feature vector is trained and classified with a linear SVM. Their approach outperforms previous methods. Dalal did also some work on part-based person detection [Dal06b, Sec. 7]. He reports a slight improvement.

Mikolajczyk et al. [MSZ04] model the full person by a probabilistic assembly of body part detectors. As body parts, they use face (frontal, profile), head (frontal, profile), head+shoulders (frontal, profile), and legs. The detectors for the body parts are based on groupings of simple orientation features which are combined and trained using AdaBoost. The assembly of body parts is done using Gaussians to model the geometric relationship. They report state-of-the-art results. An advantage of their model is the ability to deal with partly visible as well as fully-visible people.

Leibe et al. [LSS05] present an approach to the detection of pedestrians in crowded scenes. In a training stage, they generate a code book of local appearances and learn a spatial occurrence distribution for each code book entry. For the detection, image patches are extracted and matched to code book entries. In a Hough voting procedure, the matched entries vote for an initial object hypotheses. For each hypothesis, an object segmentation mask is computed per pixel. In a Chamfer matching stage, global cues from learned object silhouettes are combined with the information from the object segmentation. Finally, conflicts between overlapping hypotheses are resolved in an MDL-based (Minimal Description Length) verification step where the combination of hypotheses is taken that maximizes the savings – or in other words the combination that best explains the image content.

4.2 Detection Architecture

For the human detection, human body parts are trained and detected using the approach (and as well the implementation) of Dalal and Triggs presented in [Dal06b, DT05]. Their system can be divided into three main parts: Encoding, training, and detection. These parts are discussed in Sections 4.2.1-4.2.3. Then Section 4.2.4 introduces the different body parts and the model that combines the parts.

4.2.1 Encoding

The encoding part of the classifier architecture is described according to [Dal06b, Fig. 4.12] by the successive steps below (see also Figure 4.1). Dalal [Dal06a] suggested us in discussions parametric values that are well suited for the detection of our body parts. We briefly mention these parameters along with the description of each step.

(1) *Gamma/Color Normalization*: The image is pre-processed by applying a gamma correction. In our application we used the default-setting of log-compression on each individual color channel.

(2) *Gradient Computation*: The image gradients (magnitude and direction) are computed for each pixel by applying the 1D point derivative $[-1, 0, 1]$ in x- and y-direction. The gradients are computed for each color channel separately. For a particular pixel position, the value of the channel with the highest magnitude is chosen.

(3) *Spatial/Orientation Binning*: All pixels in a $\eta \times \eta$ pixel region, which is called a *cell*, vote into the same *Histogram of Oriented Gradients* (HOG, cf. Section 2.4). For our experiments we use the following values: As cell-size 4×4 pixels, as number of orientation bins $\beta = 8$ spaced over $[0°, 180°[$ (i.e., we ignore the 'sign' of the orientation).

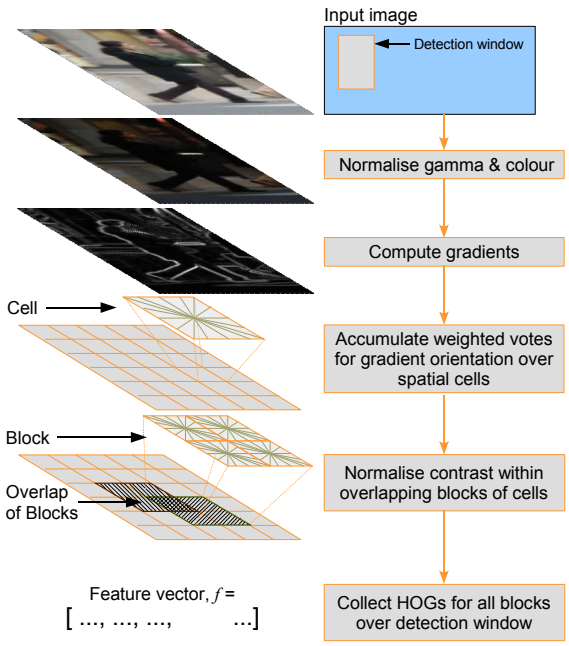


Figure 4.1: *An overview of the encoding scheme of the human classifier introduced by Dalal and Triggs [DT05] (figure has been taken from [Dal06b, Fig. 3.3]).*

(4) *Descriptor Blocks:* A *(descriptor) block* consists of $\varsigma \times \varsigma$ (HOG) cells. Before computing the HOG cells of one block, a Gaussian window weights all pixel gradient magnitudes in the block. The window is centered at the block center with $\sigma = 0.5\varsigma\eta$. In our experiments, we use a block size of 3×3 cells.

(5) *Normalization:* Since the gradients vary in their intensity depending on the illumination conditions as well as the background-foreground contrast, Dalal notes in [Dal06b, Sec. 4.3.4] that local contrast normalization is crucial for the performance. Within a block, the orientation bin values of the cells are normalized. The blocks are overlapping, thus each cell contributes to several blocks. According to [Dal06b, Sec. 4.3.4], different normalization strategies perform equally well for a person detector. We use the L2-norm normalization:

$$\mathbf{v} \leftarrow \frac{\mathbf{v}}{\sqrt{||\mathbf{v}||_2^2 + \varepsilon^2}}. \tag{4.1}$$

The normalization is followed by clipping (clipping the values above 0.2) and renormalization (this procedure is similar to SIFT, cf. Section 2.4). ε is a small normalization constant to avoid division by 0.

(6) *Descriptor Vector*: All normalized orientation bins of all blocks in the window are collected into one big descriptor vector.

4.2.2 Training

The training consists of two training stages which can be summarized by (cf. [Dal06b, Fig. 4.13]):

(1) *First Training Stage*: A positive data set is generated from our manually annotated data (see Chapter 3). All annotated image regions of a particular body part and a particular view point are aligned, scaled to the same resolution, and extracted from the full frame image (cf. Section 3.1.2). A second, negative data set consists of various images that do not contain any people. For the training, windows are sampled randomly at different scales and positions from this negative data set. These random windows have the same resolution as the extracted body part images in the positive data set. A first classifier is trained with a linear SVM (see also Section 2.2) based on these data sets of positive and negative samples.

(2) *Generating Hard Negative Examples:* A dense multi-scale scan (using the detection scheme described in the following section) on all original images in the negative data set is performed. All (by default false positive) detections are added to a third data set of hard negative examples.

(3) *Second Training Stage*: Using a linear SVM, the final classifier is trained with the additional hard negative examples.

4.2.3 Detection

A trained classifier (i.e., a trained SVM model) is only able to decide whether an image (or an image region) of a fixed resolution belongs to the class of interest or not. In our case, the class of interest is always a particular body part of a particular point of view. The following scheme (cf. [Dal06b, Fig. 5.5]) is used for a scale-independent detection in images (the scheme is illustrated in Figure 4.2):

(1) A scale pyramid (see Figure 4.3) of a given image is computed, i.e., the original image of width w and height h is scaled down by a factor $\gamma = 1.05$ (or 1.1). Iteratively, this process is repeated until a minimal size is reached. The width and height of the s^{th} scale level is w/γ^s and h/γ^s, respectively.

(2) For each generated scale, windows are extracted on a dense grid (one grid step is equal to the size of a cell) as it is shown in Figure 4.3. Each extracted window is classified by the trained SVM model and yields a classification score. This score expresses the likelihood that the investigated image region is an object of interest (i.e., a particular body part; resulting in a positive score) or not (resulting in a negative score).

(3) For a true detection, multiple positive detection scores are distributed in scale-space around the true location of the object (cf. Figure 4.2(a)). A non-maximum suppression (as described in [Dal06b, Sec. 5.2]) on all detections above a chosen

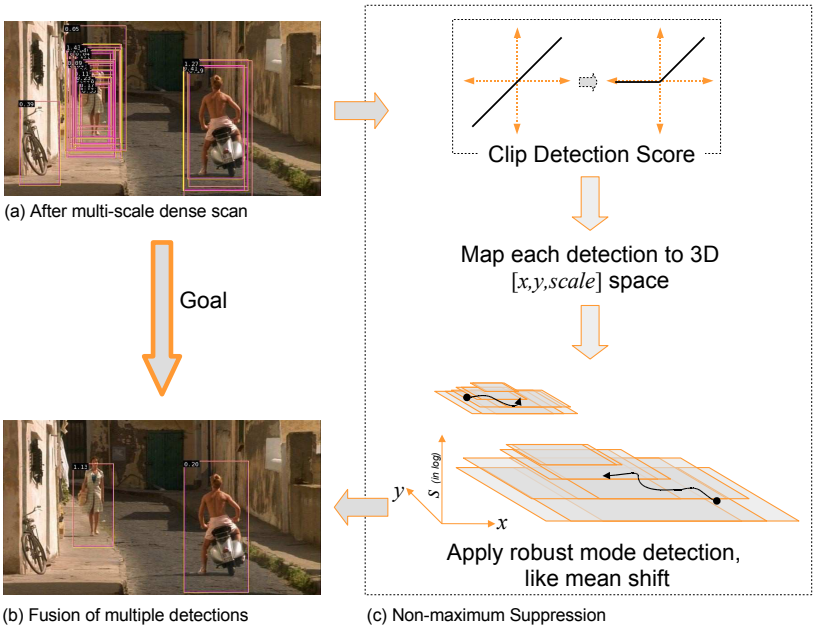


Figure 4.2: *An overview of the detection scheme and the non-maximum suppression of the human classifier introduced by Dalal and Triggs [DT05] (figure has been taken from [Dal06b, Fig. 5.1]).*

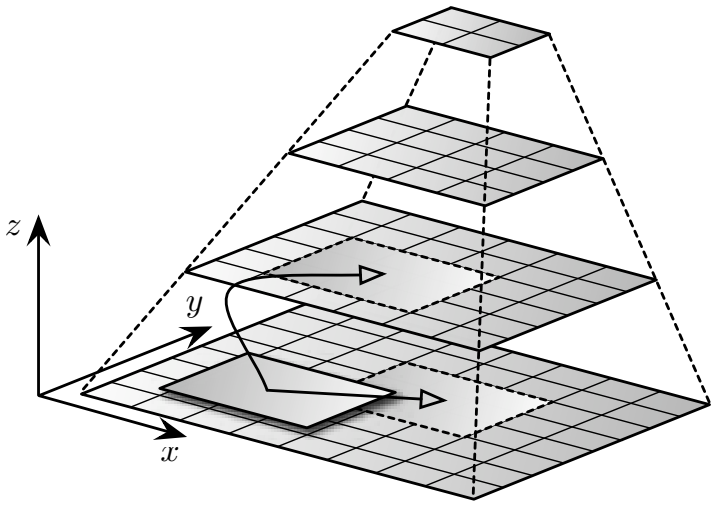


Figure 4.3: *In order to detect objects in a given image, a detection window is shifted through the image's scale-space; at each investigated position, the image content within the window is classified; the scale-space consists of the x- and y-dimensions in the image plane as well as the z-dimension representing the image-scale.*

SVM-score threshold determines the final detection results. For each maximum, the non-maximum suppression computes a confidence value and a refined detection position by fusing the detections in the local neighborhood of a maximum.

4.2.4 Body Part Detectors

Motivated by the approach of Mikolajczyk et al. [MSZ04], several body part detectors are trained and their detection results are combined in order to obtain more robust detections. For the combination of the different body parts, a model of geometric relations between the body parts is learned based on Gaussians. This model is similar to the one presented in [MSZ04].

We use the same body parts as Mikolajczyk et al. in [MSZ04]. Since cinema-style video sequences contain humans often in poses ranging from head+torso to close-up (cf. Section 3.1.1), legs are omitted as body part. Our body parts are:

- Face,
- Head, and
- Head+shoulders.

As in [MSZ04], each body part is split into two views: Frontal (or frontal+rear) view and side view (see Figure 3.5 for example body parts). The performance of a body part detector improves (see Section 4.3.3) by orienting its training images to a common direction (left or right). For this, a horizontal flip operation is applied to training images, if necessary. To detect body parts in both directions, the trained detector is applied to the original image as well as to its horizontally flipped version.

This section presents pre-/post-processing steps and the combination of single body part detections. In our notation, a detected body part i has several properties: A detection score ($\text{SCORE}_i \in \mathbf{R}^+$), its body part type ($\text{TYPE}_i \in \{\text{FACE}, \text{HEAD}, \text{HEAD-SHOULDERS}\}$), its view ($\text{VIEW}_i \in \{\text{FRONTAL}, \text{SIDE}\}$), its detection bounding box (BOX_i), the width and height of its bounding box ($\text{WIDTH}_i, \text{HEIGHT}_i \in \mathbf{R}$), as well as the coordinates of the center point of its bounding box ($\text{CENTERX}_i, \text{CENTERY}_i \in \mathbf{R}$). For the combination of a detection with other body part detections, a combined score ($\text{COMBSCORE}_i \in \mathbf{R}^+$) is used.

Merging Different Directions

Since all training images are oriented to a common direction, the body part detectors only find body parts of a particular direction (i.e., left or right). Therefore a detector is applied to the original image $\mathcal{I}_1$ and its horizontally flipped version $\mathcal{I}_2$. Body part detectors for the frontal view usually detect the same body part both images. To merge these detections, only the stronger detection of the two is taken.

Two detections in $\mathcal{I}_1$ and $\mathcal{I}_2$ are assumed to mark the one and same body part if they overlap strongly and if they exhibit same (body part) type . We employ as overlap

criterion the area of the intersection of their bounding boxes divided by the area of their union:

$$\frac{\text{area}(\text{BOX}_1 \cap \text{BOX}_2)}{\text{area}(\text{BOX}_1 \cup \text{BOX}_2)} > \text{MERGERATIO}. \tag{4.2}$$

We set a rather strict threshold of $\text{MERGERATIO} = 0.75$. The image (either $\mathcal{I}_1$ or $\mathcal{I}_2$) in which a body part has been detected determines the direction to which the body part is oriented (left or right).

Deleting Overlapping Detections

It may happen that body part detections of the same type overlap (e.g., an additional detected face within a face detection). In these cases we apply the winner-takes-it-all strategy and keep only the detection with the highest score. The overlap criterion for this case is slightly different to Equation (4.2). Instead of relating the area of the intersecting box to the area of the union of the boxes, it is related to the area of the smaller box of the two detections:

$$\frac{\text{area}(\text{BOX}_1 \cap \text{BOX}_2)}{\min(\text{area}(\text{BOX}_1), \text{area}(\text{BOX}_2))} > \text{DELETERATIO}. \tag{4.3}$$

We use different thresholds for different body parts. For faces, we employ a strict threshold of $\text{DELETERATIO} = 0.15$ since even a small overlap is not expected. For head and head+shoulder detections, a larger overlap is possible, e.g., when two persons stand close together. Therefore we relax the threshold for heads to $\text{DELETERATIO} = 0.5$ and for head+shoulders to $\text{DELETERATIO} = 0.75$.

Normalizing Detection Scores

Before the different detections are combined, their detection scores need to be normalized such that they lie within a similar range. This is done by taking for each body part type b and for each view v the N detections with the highest scores:

$$\mathcal{N}_{b,v}^{N} = \{i|\ \text{TYPE}_i = b, \text{VIEW}_i = v\} : \\ |\{j|\text{TYPE}_j = b, \text{VIEW}_j = v, \text{SCORE}_j > \text{SCORE}_i\}| < N, \forall i \in \mathcal{N}_{b,v}^{N}. \tag{4.4}$$

The mean of the N detections with the highest scores is

$$\bar{\mathcal{N}}_{b,v}^{N} = \frac{1}{N} \sum_{i \in \mathcal{N}_{b,v}^{N}} \text{SCORE}_i. \tag{4.5}$$

It is used along with a constant factor SCOREFACTOR to normalize the score of a detection i:

$$\text{SCORE}_i := \frac{\text{SCOREFACTOR} \cdot \text{SCORE}_i}{\bar{\mathcal{N}}_{\text{TYPE}_i, \text{VIEW}_i}^{N}}. \tag{4.6}$$

We set $N = 10$. During our experiments, we experienced that higher values for SCOREFACTOR privilege combined body parts, whereas lower values put less emphasis on the combination of the parts. Experimentally we found $\text{SCOREFACTOR} = 5$ to be a good trade-off.

Model for Geometric Relations of Body Parts

The different body parts are geometrically related to one another. This information introduces additional constraints. By combining detections of different body parts, more robust overall results can be obtained. Mikolajczyk et al. model in [MSZ04] the geometric relations of different body parts by Gaussians that depend on the relative position and the relative scale of the parts. We model these relations in the same manner. In the following, the computation of the relative position and the relative scale of two body parts is explained first. Then the learning of the model from the annotation data is discussed.

Given two detected body parts, i, j, of different types, i.e., $\text{TYPE}_i \neq \text{TYPE}_j$, the relative position of j with respect to i is computed as

$$x_{i,j} = \frac{\text{CENTERX}_j - \text{CENTERX}_i}{\text{HEIGHT}_i}, \tag{4.7}$$

$$y_{i,j} = \frac{\text{CENTERY}_j - \text{CENTERY}_i}{\text{HEIGHT}_i}. \tag{4.8}$$

In order to be invariant to scale, the coordinates are normalized by the height of i (cf. Section 3.1.2). The relative scale of the objects is given by the ratio of the area of their bounding boxes:

$$\theta_{i,j} = \left(\frac{\text{WIDTH}_j \cdot \text{HEIGHT}_j}{\text{WIDTH}_i \cdot \text{HEIGHT}_i} \right). \tag{4.9}$$

The geometric body part relations (i.e., relative position and relative scale) are modeled by three independent Gaussian functions: One for relative x position (Equation (4.7)), one for relative y position (Equation (4.8)), and one for relative scale (Equation (4.9)). The mean and variance parameters for theses Gaussians are estimated from our annotation data base (cf. Section 3.1). Before these parameters are computed, the individual bounding boxes of the different body parts are replaced by the corresponding smallest common bounding box (Equation (3.7)). This represents the data that is used for the training of the body part detectors. The means and variances are computed by

$$\bar{x}_{b,c} = \frac{1}{|\mathcal{S}_{b,c}|} \sum_{(i,j) \in \mathcal{S}_{b,c}} x_{i,j}, \tag{4.10}$$

$$\bar{y}_{b,c} = \frac{1}{|\mathcal{S}_{b,c}|} \sum_{(i,j) \in \mathcal{S}_{b,c}} y_{i,j}, \tag{4.11}$$

$$\bar{\theta}_{b,c} = \frac{1}{|\mathcal{S}_{b,c}|} \sum_{(i,j) \in \mathcal{S}_{b,c}} \theta_{i,j}, \tag{4.12}$$

and

$$\tilde{x}_{b,c} = \frac{1}{|\mathcal{S}_{b,c}|} \sum_{(i,j)\in\mathcal{S}_{b,c}} \left(\bar{x}_{b,c} - x_{i,j}\right)^2, \tag{4.13}$$

$$\tilde{y}_{b,c} = \frac{1}{|\mathcal{S}_{b,c}|} \sum_{(i,j)\in\mathcal{S}_{b,c}} \left(\bar{y}_{b,c} - y_{i,j}\right)^2, \tag{4.14}$$

$$\tilde{\theta}_{b,c} = \frac{1}{|\mathcal{S}_{b,c}|} \sum_{(i,j)\in\mathcal{S}_{b,c}} \left(\bar{\theta}_{b,c} - \theta_{i,j}\right)^2, \tag{4.15}$$

respectively. $\mathcal{S}_{b,c}$ is a set of all existing body part combinations in our annotation data base with the types b and c, i.e.,

$$\mathcal{S}_{b,c} = \{(i,j) | \text{TYPE}_i = b, \text{TYPE}_j = c\}. \tag{4.16}$$

Note that the actual view or direction of body parts are ignored. Otherwise the model would be too specific and would perform worse.

Body Part Assembly

After all body parts have been detected in a given image separately, the model for geometric relations that has been learned from the annotation data is used for assembling the detections. The assembly is done by seeking plausible connected body parts for each detection i. A valid connected body part has to be of a different type than i and it has to have sufficient overlap with i. A combined score is computed for the combination of i with other detections. The combination that maximizes the combined score is chosen in case there are several valid candidates that could be combined with i. After all body part detections have been processed, the detection score value is recomputed by summing the original detection score value and the combined score value. This approach can be seen as a kind of score boosting: Other connected body parts can boost the score of a single detection. Detections that do not have support by other body parts, yet that have a sufficiently high detection score are not ignored. The body part assembly is described in detail in Algorithm 2.

The combined score of two detections i, j with $\text{TYPE}_i \neq \text{TYPE}_j$ is the product of their detection scores weighted by the product of the Gaussians for the relative x and y position and scale:

$$\begin{aligned}\text{combScore}(i,j) = \text{SCORE}_i \cdot \text{SCORE}_j \cdot \exp\Bigg(&- \frac{\left(\bar{x}_{\text{TYPE}_i,\text{TYPE}_j} - x_{i,j}\right)^2}{2 \cdot \tilde{x}_{\text{TYPE}_i,\text{TYPE}_j}} \\ &- \frac{\left(\bar{y}_{\text{TYPE}_i,\text{TYPE}_j} - y_{i,j}\right)^2}{2 \cdot \tilde{y}_{\text{TYPE}_i,\text{TYPE}_j}} - \frac{\left(\bar{\theta}_{\text{TYPE}_i,\text{TYPE}_j} - \theta_{i,j}\right)^2}{2 \cdot \tilde{\theta}_{\text{TYPE}_i,\text{TYPE}_j}} \Bigg).\end{aligned} \tag{4.17}$$

The mean and variance parameters are learned from the annotation data set (cf. Equations (4.10)-(4.15)). In our final experiments we set $\text{COMBINERATIO} = 0.5$ (used in Algorithm 2).

Algorithm 2 Combination of Body Part Detectors

Given an image $\mathcal{I}$ on which we run our body part detectors
$\mathbf{D} := \{\text{all body part detections in } \mathcal{I}\}$
// compute the combined score for each detection
for all $i \in \mathbf{D}$ **do**
  // find all body parts with a different type and with sufficient overlap
  $\mathbf{C}_i := \emptyset$
  for all $j \in \mathbf{D}$ with $\text{TYPE}_i \neq \text{TYPE}_j$ **do**
    if $\frac{\text{area}(\text{BOX}_i \cap \text{BOX}_j)}{\min(\text{area}(\text{BOX}_i), \text{area}(\text{BOX}_j))} > \text{COMBINERATIO}$ **then**
      $\mathbf{C}_i := \mathbf{C}_i \cup \{j\}$
    end if
  end for
  // select only those body parts of each type that yield the highest combined scores
  $\mathbf{C}_i^* := \emptyset$
  for all $t \in \{\text{FACE}, \text{HEAD}, \text{HEADSHOULDERS}\}$ **do**
    $j^* = \arg\max_j \{\text{combScore}(i,j) | j \in \mathbf{C}_i, \text{TYPE}_j = t\}$
    $\mathbf{C}_i^* := \mathbf{C}_i^* \cup \{j^*\}$
  end for
  // compute the combined score
  $\text{COMBSCORE}_i := 0$
  for all $j \in \mathbf{C}_i^*$ **do**
    $\text{COMBSCORE}_i := \text{COMBSCORE}_i + \text{combScore}(i,j)$
  end for
end for
// recompute the final score for all detections
for all $i \in \mathbf{D}$ **do**
  $\text{SCORE}_i := (\text{SCORE}_i + 1) \cdot (\text{COMBSCORE}_i + 1)^2 - 1$
end for

Deleting Unconnected Body Parts

In a final step, unconnected body part detections are deleted since the side as well as head+shoulders detectors are not very reliable (cf. Figure 4.11). However, frontal face detections are not deleted since in close-up views head or head+shoulders might not be detected. Also, the frontal face detector proofed to be the most reliable one together with the detector for the frontal head. Unconnected frontal head detections are deleted nevertheless since we expect them to be connected either with a face or a head+shoulders detection. Thus, a detection i is deleted if it fulfills the condition:

$$\text{COMBSCORE}_i < \varepsilon \wedge \neg(\text{TYPE}_i = \text{FACE} \wedge \text{VIEW}_i = \text{FRONTAL}) \tag{4.18}$$

Detection with a combined score below ε are assumed to be unconnected. We set a low threshold $\varepsilon = 0.01$.

4.3 Parameter Evaluation

In this section, different parameter settings for the body part detectors are evaluated: Different body part alignment and normalization strategies (Section 4.3.2), view specific detectors (Section 4.3.3), different sizes for the training images (Section 4.3.4), and different methods for the augmentation of the training data set (Section 4.3.5). Before evaluating, Section 4.3.1 gives more information on the data set and the evaluation methods that are used.

4.3.1 Data Set and Evaluation

The experiments for the parameter evaluation of the body part detectors are based on the data set as described in Chapter 3. The full annotated data set of Buffy1 is used to train the detectors. The evaluation is carried out on the annotated data from Buffy2. Body parts are extracted, aligned, and scaled to a common image size (cf. Section 3.1.2). Various possibilities of how this is done exactly are evaluated and discussed in Section 4.3.2. As negative data set we use a large set of images that do not contain any humans. How the negative data set is exactly included during the training is described in Section 4.2.2.

The performance of different parameter settings is evaluated on the classifier itself. The classifier operates only on extracted positive and negative training images. Therefore the evaluation results do not include the localization of body parts in images. For the evaluation, we use precision-recall curves (see Section 2.1.3 for a discussion of precision-recall curves and performance measures).

4.3.2 Different Alignment/Scale Normalization Strategies

The goal of the normalization and alignment step is ideally to keep characteristic structures at the same positions in all training images. Yet owing to general variation in

appearance, changes in view point, and rotation, an ideal alignment does not exist. For our body part detectors, we evaluate experimentally in the following several alignment/scale normalization strategies to determine the one that is best suited for our case.

Figure 4.4 shows several experimental results with different alignment/normalization approaches. Four different strategies are compare in this experiment:

- The center point of the bounding box for the alignment, the height of the bounding box as normalization factor (denoted as "center+scale box");
- The center point of the bounding box for the alignment, the standard deviation of the distance between all fix points and their common center point as normalization factor (denoted as "center box/scale fix points");
- The center point of the fix points for the alignment, the height of the bounding box as normalization factor (denoted as "center fix points/scale box");
- The center point of the fix points for the alignment, the standard deviation of the distance between all fix points and their common center point as normalization factor (denoted as "center+scale fix points").

All heads are scaled to a common height of 60 pixel and all head+shoulders to a common width of 72 pixel. Note that the scale ratio changes due to the different alignment/normalization strategies (e.g., for heads between 60×60 and 48×60).

Figure 4.4 shows that the performance improves significantly when fix points are used to align body part images. A further improvement can be achieved by taking the height of the bounding box as normalization factor. Based on these results, we choose to align body parts at the fix point center and use the bounding box height for scale normalization as they are introduced in Section 3.1.2.

4.3.3 View Specific Detectors

To investigate performance of view specific detectors, two experiments are carried out in this section. In the first experiment, a head detector that has been trained on frontal *and* side view head images is compared with two detectors that are trained separately on frontal and side view images. The second experiment compares two frontal head detectors: One detector trained on the original head images as well as their horizontally flipped versions, the other trained on head images oriented to a common direction (see also Section 4.2.4).

The results of the first experiment (Figure 4.5(a)) suggest that a detector trained on view specific image data (either frontal or side) performs better than one that is trained on more general data. In comparison with the frontal+side detector, the performance of the side head detector drops slightly, yet the performance of the frontal head detector increases significantly. One has to keep in mind that we use more frontal view training images than images for the side view (63% vs. 37% in Buffy1 and Buffy2, cf. Table 3.2).

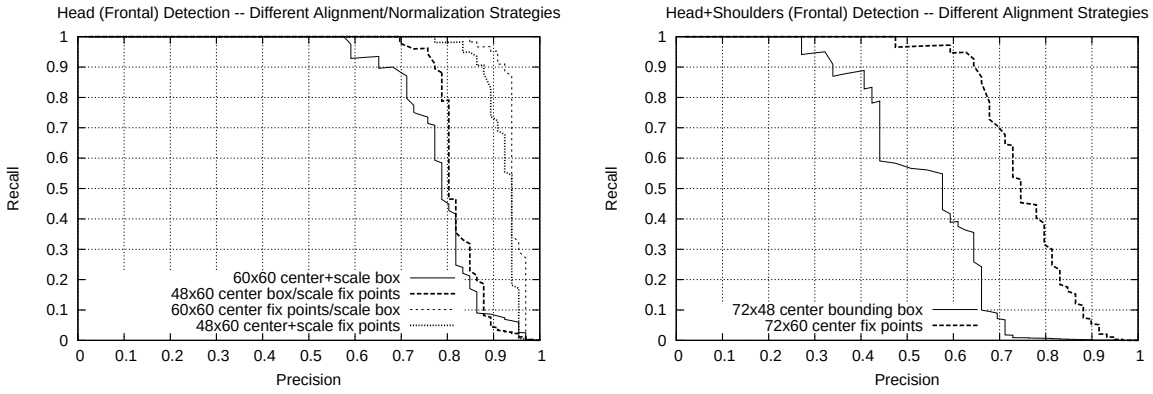
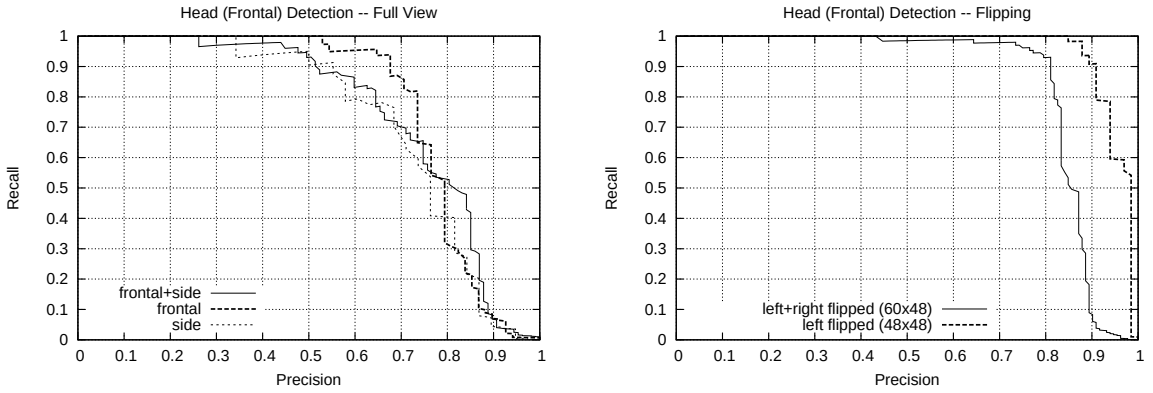


Figure 4.4: *Performance comparison of several detectors trained with different alignment and normalization strategies; for the alignment we use the center of the bounding box ("center box") and the center of the fix points ("center fix points"); as scale normalization factor we use the height of the bounding box ("scale box") and the standard deviation of all fix points to their common center ("scale fix points").*



(a) (b)

Figure 4.5: *(a) performance of a head detector trained on frontal and side images in comparison to detectors that are separately trained on frontal and side images (all detectors used the bounding box center as alignment point); (b) performance of a frontal head detector using the original training images* and *their flipped versions ("left+right flipped") compared to a detector that has been trained on a training set where all objects have been flipped to a common direction ("left flipped", we used a common direction the left side).*

However, the changes are significant enough such that they do not solely stem from the different number of samples.

In our second experiment, the first head detector is trained and tested on the original training images *and* their flipped versions. For the second head detector, all training head images are oriented to a common direction (here the left side). To orient the training images, a horizontal flip operation is applied to the training image if necessary. Figure 4.5(b) shows that the latter detector (with training images oriented to one direction) clearly outperforms the first one. Note that this experiment has been carried out on frontal head images. We expect an even stronger difference in performance for side view detectors. Orienting training images to a common side seems to improve the data representation such that the performance of the SVM increases.

Based on these results, we train detectors for specific views (i.e., frontal and side) and we orient training images to a common side (i.e., left or right). In addition to the gain in performance, the detectors provide information about the orientation of a body part (frontal/side as well as left/right).

4.3.4 Training Image Size

Figure 4.6 illustrates the performance of different body part detectors trained with training images that vary in resolution. Since fixed settings are used for the internal parameters of the detectors (see Section 4.2.1), the resolution of the training images influences the performance. It controls the degree of finer structures that are incorporated in a detector. The results show that the optimal training image resolution varies for different body parts. For our work, we use the following resolutions for the body part detectors:

	face	head	head+shoulders
frontal	36×36	60×60	72×60
side	36×36	60×60	60×60

Although the 48×48 pixel resolution for the face detector seems to perform better, we choose to use 36×36 pixel. This allows us to find faces with a lower resolution in an image.

4.3.5 Training Data Augmentation

To increase the size of our training sets and thus to improve the performance of the body part detectors, we study two data augmentation strategies. The first strategy scales training images additionally up and down (the center point for the scaling is the point of alignment). The second strategy rotates a training image additionally rotation (in positive and negative direction) around its center point. Figure 4.7 illustrates the outcome of an additionally scaled and rotated head image.

The results for the augmentation strategies applied to the body part detectors are shown in Figure 4.8. Since the number of training images increases, the resolution of the precision-recall curves increases as well. The different strategies as they are denoted in Figure 4.8 extend the original training data set as follows:

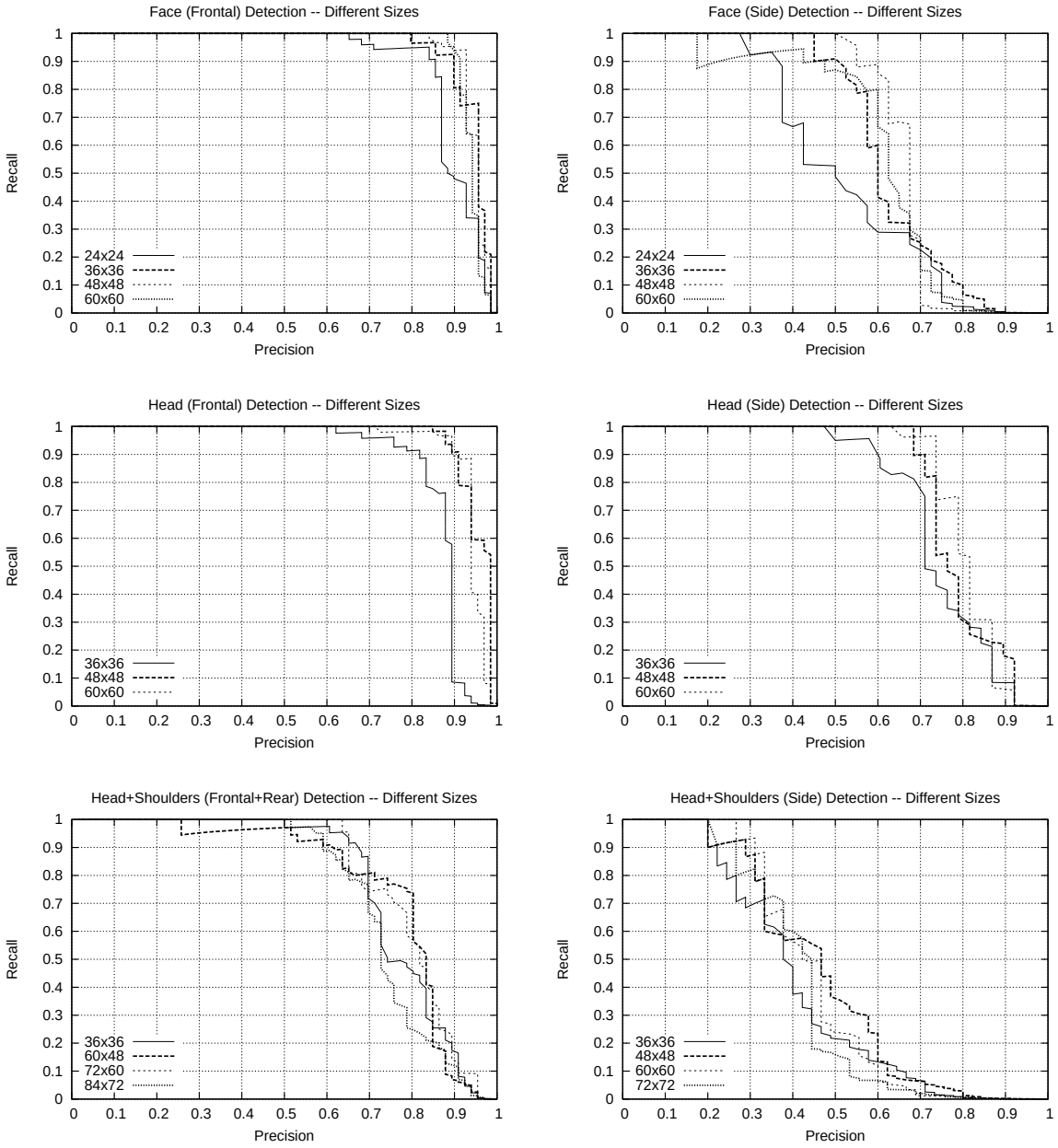


Figure 4.6: *Performance comparison of several body part detectors trained on different training image sizes; all images have been aligned to the center point of their fix points; the height has been taken as normalization factor.*

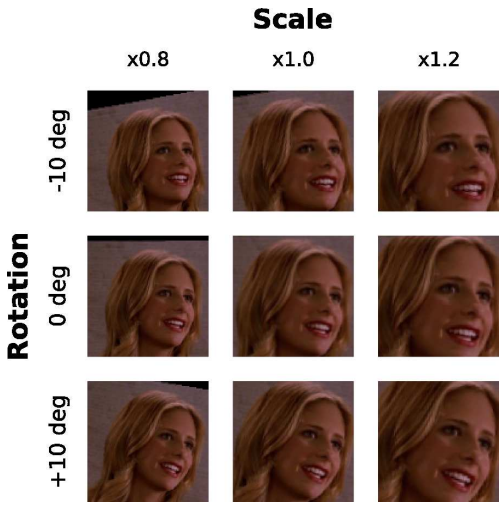


Figure 4.7: *Example for scaling and rotation of an extracted object; here we scale the object image by the factor 1.2 and 0.8 and rotate it by* $+10°$ *and* $-10°$*; the original image lies in the center.*

- *Normal*: This denotes the body part detector trained on the original training image set.

- *Rotation*: The training data set consists of the original training images rotated by $5°, 2.5°, 0°, -2.5°, -5°$.

- *Scale*: For this training data set, the original training images are scaled by 1.05, 1.025, 1.0, 0.975, 0.95.

- *Scale+Rotation*: This data set includes the original training images scaled by 1.05, 1.0, 0.95 as well as rotated by $5°, 2.5°, 0°, -2.5°, -5°$.

The additional scaling and rotation turns out to improve especially the performance of face and head detectors significantly. Therefore we use for the training of detectors in the remainder of this work training data sets that have been augmented by scale and rotation.

4.4 Detection Results

This section presents overall results of the entire detection system, i.e., the evaluation includes the classification *and* localization of body parts. The evaluation is based on different data sets. Section 4.4.1 gives more information on the data sets and the evaluation methods that are used.

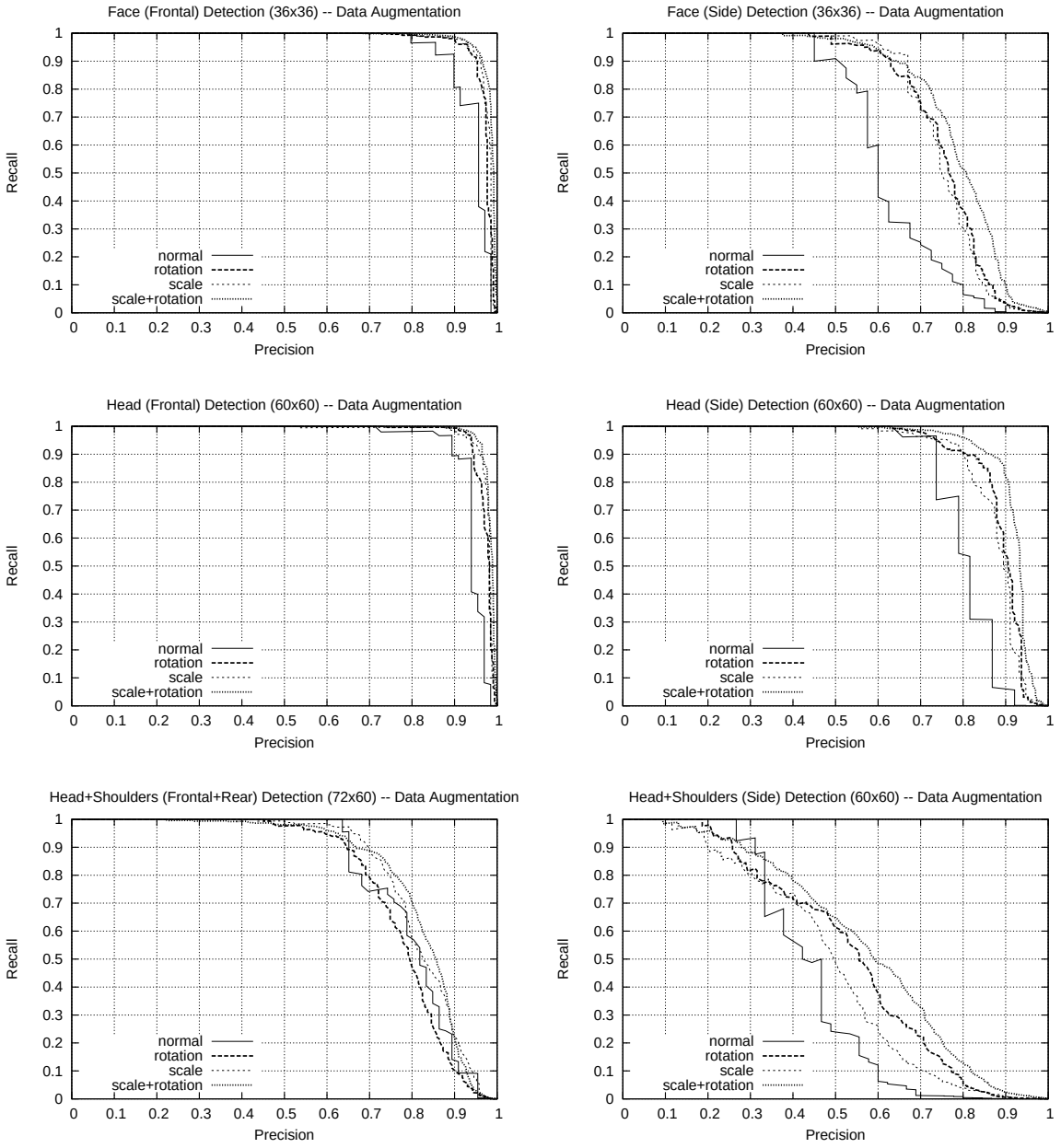


Figure 4.8: *Performance comparison of several body part detectors using different data augmentation strategies: No augmentation ("normal"), generating rotated versions (by* $5°, 2.5°, 0°, -2.5°, -5°$*) of the training images ("rotation"), generating scaled versions (by* $1.05, 1.025, 1.0, 0.975, 0.95$*) of the training images, generating scaled (by* $1.05, 1.0, 0.95$*) and rotated versions (by* $5°, 2.5°, 0°, -2.5°, -5°$*) of the training data; in all cases the augmentation of the data set leads to a gain in performance.*

4.4.1 Data Sets and Evaluation

For detection results in the following sub-sections, three different evaluation data sets are used. Results in Section 4.4.2 are based on the Buffy data set as it is described in Chapter 3. The body part detectors are trained on the data of Buffy1 and evaluated on Buffy2. Section 4.4.3 uses two external data sets, the *CMU-MIT* and the *CMU-Profile* set. The two external data sets are used to evaluate and, as before, Buffy1 is used to train the detectors.

The CMU-MIT set (we evaluate on the sub-sets A, B, and C) contains altogether 511 faces in 117 images (see Figure 4.9 for example images). It is used to evaluate our frontal face detector. The CMU-Profile data set contains altogether 364 profile and 95 frontal faces in 208 images (Figure 4.10 shows example images). Based on this data set, we evaluate our side detectors. Frontal faces in the CMU-Profile are ignored for the evaluation. In both data sets, faces are ignored that are smaller than our detector window size (altogether 193 frontal and 126 side faces).

For the evaluation, a detection is considered to be correct (true positive) if the area of overlap a_o between the predicted bounding box BOX_p and the ground truth bounding box BOX_{gt} exceeds 50% by the formula [EZW+06, Eq. 2]

$$a_o = \frac{\text{area}(\text{BOX}_p \cap \text{BOX}_{gt})}{\text{area}(\text{BOX}_p \cup \text{BOX}_{gt})}. \tag{4.19}$$

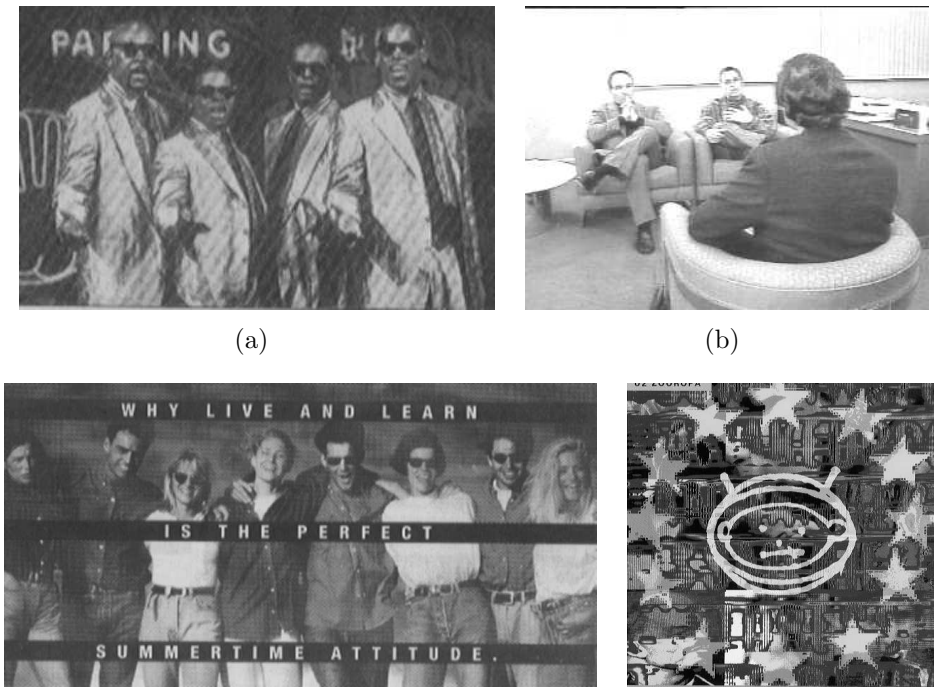
Once a body part has been detected, any other additional detection is considered as false positive. Since the CMU-MIT and CMU-Profile data set included only annotations of points on eyes, nose, and mouth, we created automatically bounding boxes based on the given points.

We present results using ROC curves. Note that the ROC curves are based on the number of false positives and not on the false positive rate. The number of false positives is more intuitive to evaluate since it gives direct information on how many false-positive detections occurred (see Section 2.1.3 for a further discussion on evaluation measures).

4.4.2 Results on Buffy2 Video

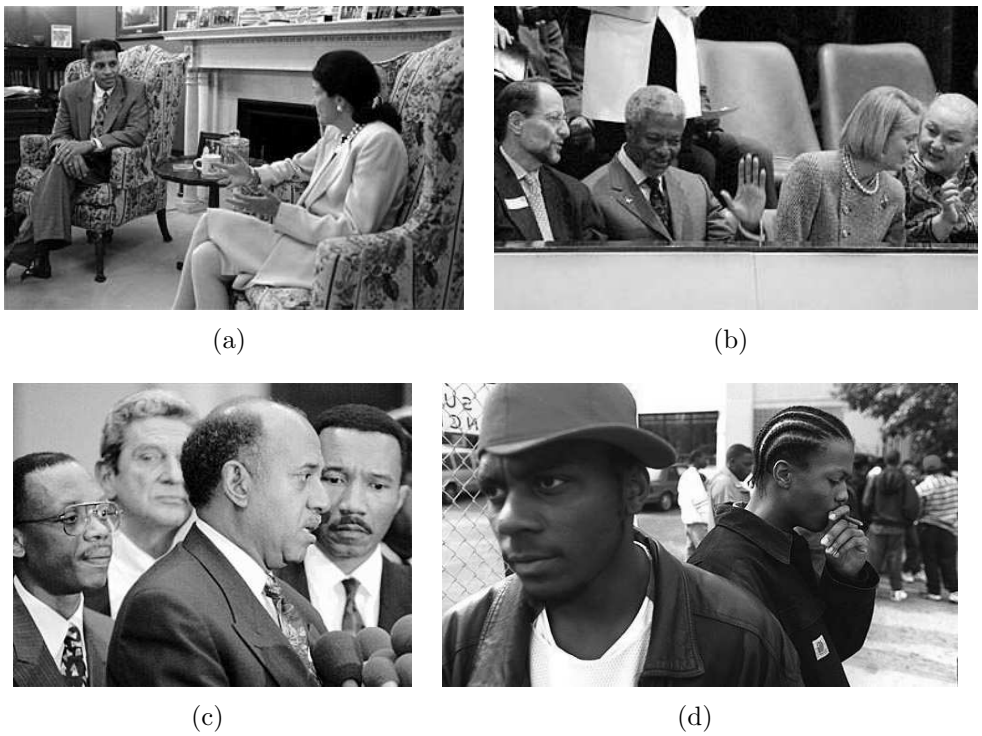
Figure 4.11 illustrates body part detection results on the Buffy2 data set. As in the parameter evaluation (cf. Section 4.3), the frontal face and the frontal head detector perform best. The side detectors perform in all cases worse than the frontal ones. This is due to the fact that frontal face/head images show more characteristic structures (two eyes, mouth, and nose; the oval shape of heads and of shoulders) than the lateral face/head ones. Additionally, the structure of a face is more rigid than the shape of heads (owing to, e.g., different hair cuts) or head+shoulders. Rigid structures are more constrained in their appearance and are therefore easier to detect.

The combination of a detected body part with other connected parts improves detection results. The improvement is especially significant for few false positives. As the number of false positive rises, the performance difference between the single detector and its combined version decreases. After a certain point, the performance for the combined



(a) (b) (c) (d)

Figure 4.9: *Example images taken from the CMU-MIT face data set.*



(a) (b) (c) (d)

Figure 4.10: *Example images taken from the CMU-Profile face data set.*

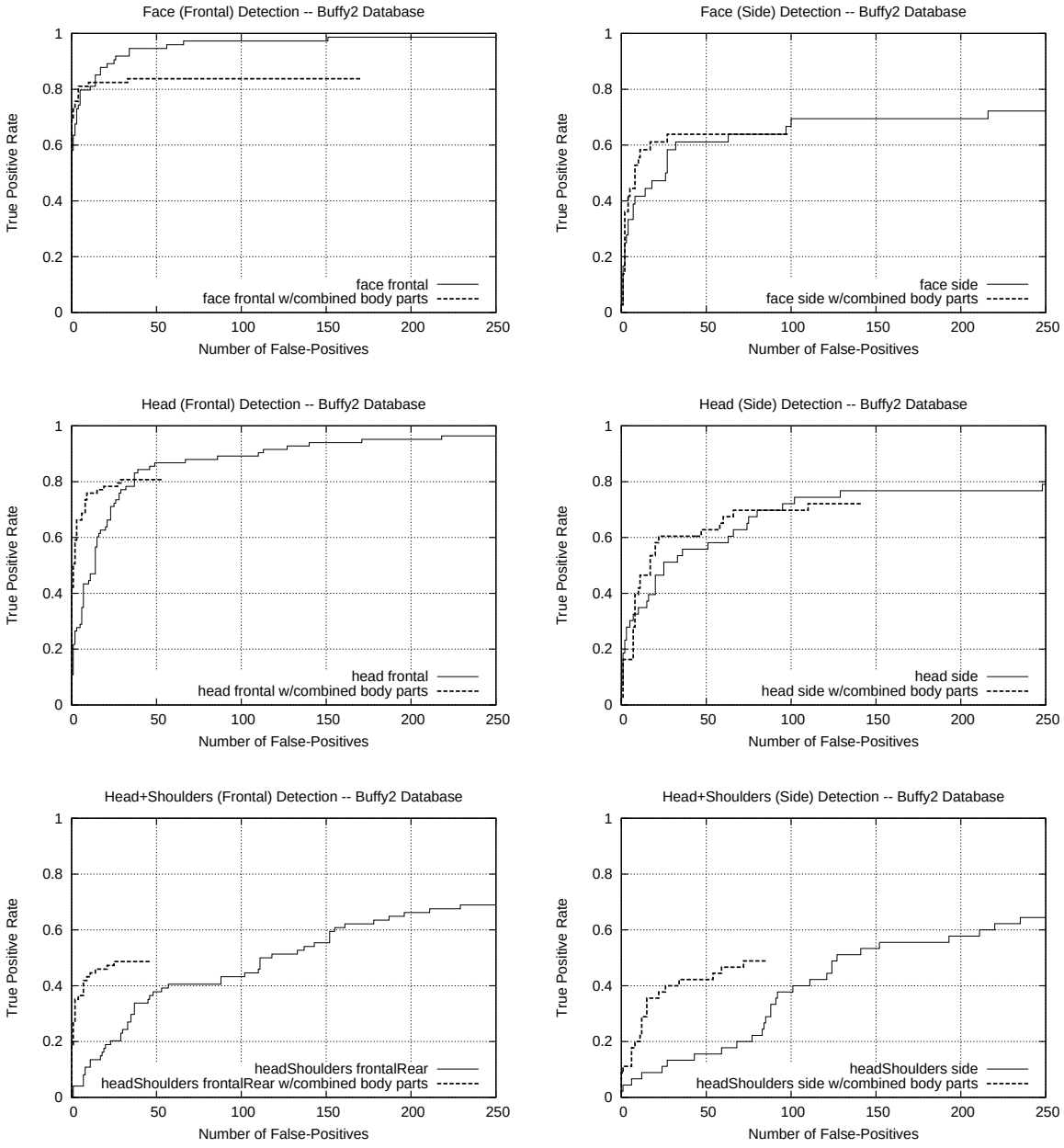


Figure 4.11: *Performance of our different body part detectors on the Buffy2 annotation data set; we note results for the single detectors as well as their combination.*

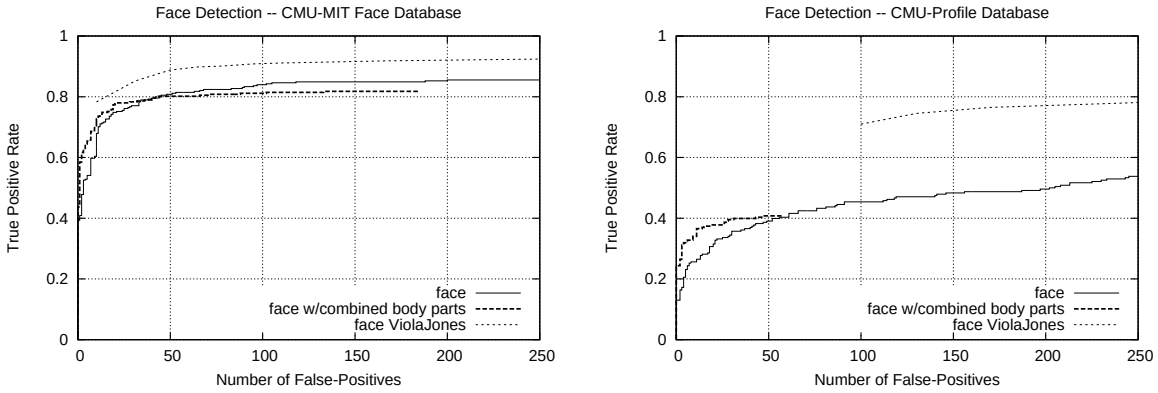


Figure 4.12: *Performance of the frontal and side face detectors on (left) the CMU-MIT and (right) the CMU-Profile data set; we denote results for only the face detection as well as results using our body part combination; in order to compare we denote the results of the face detector introduced by Viola and Jones [VJ03, VJ02].*

detector is worse than for the single detector. Since our frontal face detector is the most reliable one, its combination with other detections seems to be contra-productive. However, even the frontal face gain in performance for a low number of false positives. The performance of other body part detectors increases significantly when they are combined.

4.4.3 Results on Other Data Sets

Figure 4.12 illustrates the performance of our face (frontal and side) detectors applied to the CMU-MIT and CMU-Profile data set. The detectors perform worse than the popular Viola-Jones face detector [VJ03, VJ02]. This is certainly due to the different nature of our training data set (Buffy1) which is based on full-frames of a TV-style video sequence. The CMU-MIT data set contains a.o. scanned newspaper images as well as some hand-drawn faces (e.g., Figure 4.9(d)). This kind of image data differs significantly from our training data. The combination of the body parts slightly improves face detection results for lower numbers of false positives. Again, it is due to the different kind of image data that the gain in performance is relatively small.

Note that we mark the performance of the Viola-Jones face detector as it is given in [VJ03, VJ02]. In their work, Viola and Jones did not mention which exact evaluation criterions they used.

5 Character Recognition

This chapter presents and discusses our approach to learn identities of particular characters in a video sequence based on detected body parts. The method is based on the recent *Bag-of-Features* (BoF) approach [MS06, DWF+04, AR02].

Our contribution presented in this chapter is threefold. First, we implement and evaluate the BoF approach for learning human identities; second we combine different feature descriptors (a CIE L*u*v* color descriptor and a SIFT feature descriptor) in our BoF implementation; and third, we combine probabilistic votes of connected body parts.

The outline of this chapter is the following. Section 5.1 gives a review of related work in this area. Then, the architecture of our recognition system is introduced (Section 5.2) and parameter evaluation results are presented (Section 5.3).

5.1 State of the Art

Most relevant to our work is the approach of Everingham et al. [ESZ06]. They present an approach for automatic character labeling in TV or film material. They fuse multiple data sources. By aligning subtitles and a transcript of the movie, they know which character speaks at what time in the movie. A frontal face detector localizes the characters in the image material. By detecting lip movements, the character name from the aligned subtitles and the transcript is associated with a detected face of a person that is speaking. Additional matching of faces and clothing build the last source of information. They applied their system, as we do, to the TV series "Buffy the Vampire Slayer" and report good results.

Other works rely especially on face detection and face matching. Fitzgibbon and Zisserman [FZ02] define for faces an affine invariant distance measure with a prior. They cluster detected faces from the movie "Groundhog Day" by using the k-medoids algorithm. Their approach outputs the main characters in a movie.

Berg et al. [BBE+04] consider the problem of clustering detected frontal faces in images from web news pages. For that they also fuse different data sources: Detected faces in images are labeled with extracted key words from the image caption. Since the image labels are not exact, their data set contains false labels as well as some ambiguities. Based on facial features, face images are rectified to a standard pose. On the rectified face images, a distance measure is defined for clustering. The kernel principal component analysis (Kernel PCA) reduces the dimensionality of the data and the linear discriminant analysis (LDA) projects the data into a better suited feature space for the discrimination task. For the clustering, they employ a modified version of k-means. After pruning and

merging, they obtain final clusters for various personalities.

Arandjelović and Zisserman present in [AZ05] an automatic face recognition for film characters. Their goal is to find corresponding face images of a character for a given query face image. They base the face detection on facial feature detectors (eyes and mouth). Given the feature positions, an affine translation warps the face image into a canonical position. The face region is segmented out by a face boundary detection. After a further pose refinement step, the distance of two faces is given by the sum of square errors of the two aligned face images.

The *Bag-of-Features* (BoF) approach is the current state-of-the-art method for the classification of images containing objects of different categories (see results from [EZW+06]). Research has focused mainly on the classification of different object categories (e.g., [MS06, DWF+04, AR02]) and therefore results in this field are not really comparable to our work. However, a somewhat related work has been published by Nilsback and Zisserman [NZ06]. They create and combine different visual vocabularies (color, shape, and texture) for the classification of different species of flowers based on the BoF approach.

5.2 Recognition Architecture

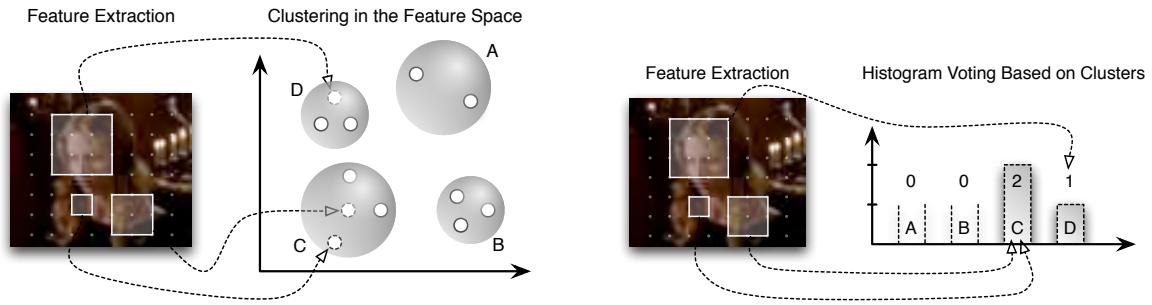
Our recognition system for human characters in video sequences is based on the *Bag-of-Features* (BoF) approach. This approach has been successfully applied to image classification tasks (cf. [EZW+06]). Due to its very general nature, it can be used for any kind of classification task. As far as we know, the BoF approach has not been applied so far to human character recognition. Therefore we were motivated to explore BoF in this context. Other workings use for character recognition especially methods that are based on distance measures for faces (cf. Section 5.1).

The BoF approach has been inspired by the *Bag-of-Words* (BoW) approach [Joa98] which has been developed for text classification. Extending the BoW to *visual words*, it is usually referred to as Bag-of-Features (BoF). The different processing steps for the BoF approach can be summarized as follows:

(1) *Feature Extraction*: Features are extracted from a (representative) set of training images. The features may be obtained by *dense sampling* (i.e., either on a fixed grid or by randomly sampling in the image scale space) or by *sparse sampling* using interest point detectors (e.g., Harris Corner detector, Harris-Laplace Detector, Difference of Gaussian [DoG], cf. [MS04]).

(2) *Codebook Generation*: The sampled feature points from all training images can be seen as a distribution in a high dimensional feature space. A clustering algorithm (cf. Section 2.3) is applied to the feature space. It groups ideally similar feature points together into clusters (see Figure 5.1(a)). These clusters are referred to as visual words, and they form the *code book* or *visual vocabulary*.

(3) *Classification*: In the same manner as before, features are extracted from a particular image. Each feature is assigned to its closest (or most likely) cluster or visual



(a) Code book generation. (b) Code book histogram voting.

Figure 5.1: *Illustration of our implementation of the Bag-of-Features approach; (a) a so-called code book is created by extracting features from a training set of images; in the feature space, the features are grouped together to clusters (we refer to them as* visual words*); (b) for the classification of an image, an histogram of visual words is created; again, image features are extracted and assigned to the closest (or most probable) visual word (i.e., cluster); each feature votes into the histogram for its assigned visual word.*

word from the code book. All extracted features of the image build together a histogram of visual words (see Figure 5.1(b)). This histogram stores how often a visual word appeared in the image. A classifier (e.g., k-Nearest Neighbors, SVM [see Section 2.2], etc.) is trained on the histograms of a set of training images. Some visual words may be less discriminative than others, but the combination of all visual words holds enough information to successfully predict the class membership of a new, unknown image.

In our implementation of this approach, we use an internal library of the LEAR research group [LEA] mainly for feature extraction, the libSVM [CL01] as Support Vector Machine, and the OpenCV library [ope06] for color conversion.

In the following, Sections 5.2.1-5.2.3 present our implementation for each processing step in detail.

5.2.1 Feature Extraction

For the feature extraction, all (either training or test) body part images are extracted from the annotation data set at a fixed height. This is done in order to constrain the number of extracted features that need to be clustered. We choose a height of 60 pixel as trade-off for our application. Two different kind of code books a generated: One based on the SIFT descriptor and the other one based on a CIE L*u*v* color descriptor. Both descriptors are sampled densely from a body part image.

The code book based on the *Scale Invariant Feature Transform* (SIFT) descriptor (cf. Section 2.4) captures the structure of body parts belonging to a particular character. Feature points are sampled from the image on a grid of 4×4 pixel in x- and y-dimension and with a scaling factor of 1.2 in scale (starting with the scaling factor 1 and ending with the highest scaling factor ≤ 6). Note that the 4×4 pixel grid in x- and y-dimension

is relative to the scale. For the descriptor, the standard settings of 8 orientation bins and 4×4 cells are used. In order to be more sensitive to smaller changes, 3×3 pixel are used as cell resolution (instead a standard value of 4×4 pixel). Due to the grid size, the sampled feature points have overlap with their neighbor points.

The CIE L*u*v* color feature descriptor is composed by the L^*, u^*, and v^* values of a particular point in scale. Feature points are sample on only one scale level (with the factor 2) and in that scale on a 1×1 pixel grid. This is done in order to limit the number of points that are clustered.

5.2.2 Code Book Generation

With the two descriptor types, two different code books (one SIFT code book and one color code book) are generated for each body part (face, head, and head+shoulders; see also Section 5.3.2), i.e., six different code books altogether. k-means (cf. Section 2.3) is applied to the feature space in order to generate the code books. The visual words are then presented by the cluster centers. In order to generate a code book, all sampled features are used from all extracted images of the annotation data of Buffy1.

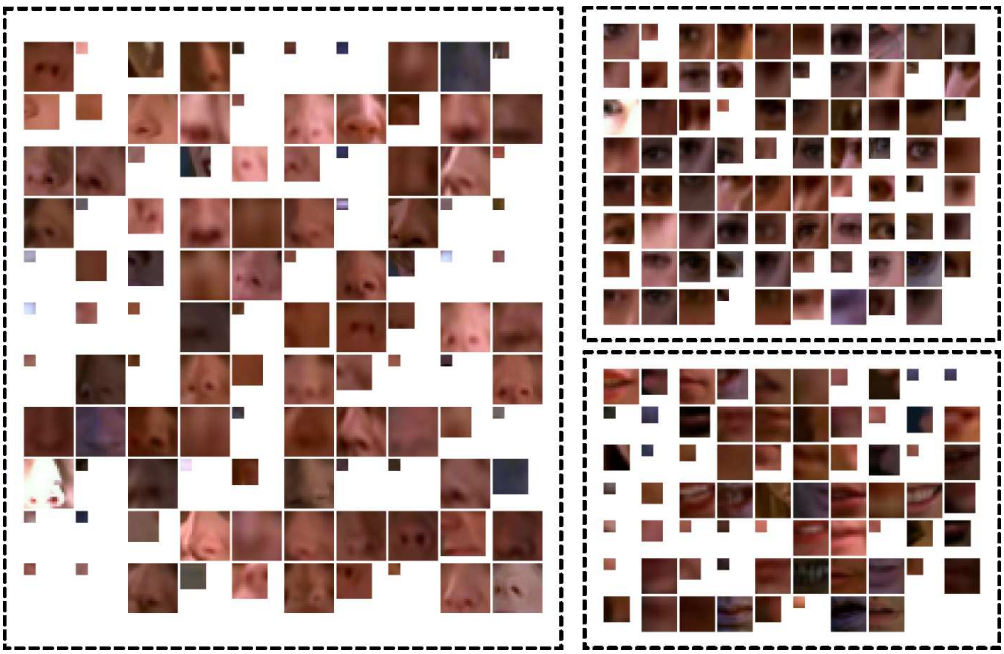
Figure 5.2 shows example image patches that were clustered into visual words. Note that the visual words of the SIFT code book (Figure 5.2(a)) contain image patches regardless of scale. Interestingly, a particular topic (such as nose, eye, mouth) can be found within the same cluster. However, this is not true for all clusters. In visual words of the color code book, rather similar colors are grouped together – as expected. But also here, there are patches that could be considered as outliers. This is assumably due to the less versatile k-means clustering.

5.2.3 Classification

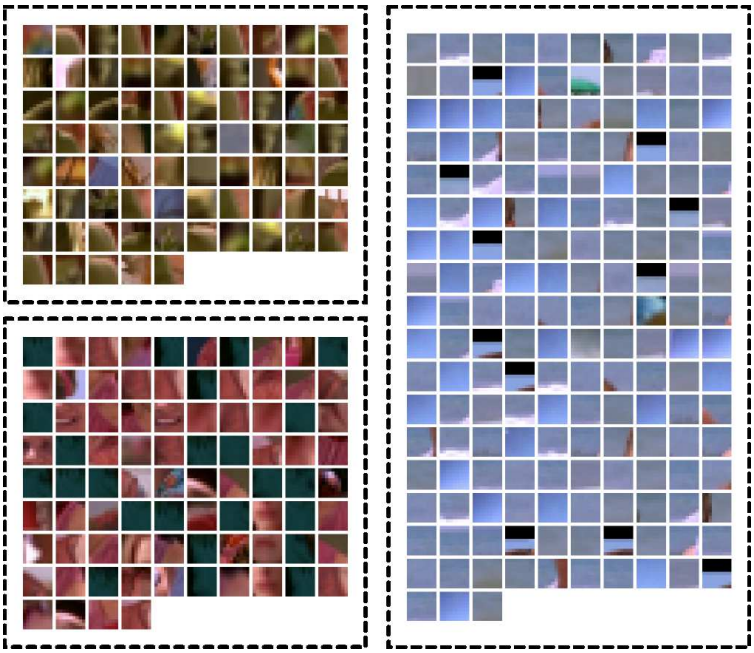
To train a classifier for character classes of a particular body part, occurrence histograms of visual words are computed for all body part images of the Buffy1 training set. To build such a histogram, feature points are extracted from one body part image (cf. Section 5.2.1) in the same manner as for the code book generation. Each extracted feature point is assigned to the closest cluster center (visual word) of the corresponding code book. The histogram stores (for one body part image) how often a feature point has been assigned to each visual word. The histogram entries are binarized to either 0 (no occurrences) or 1 (one or more occurrences). This is done since descriptive visual words are not expected to occur more than once in an image. A word that occurs several times is not expected to be very informative. The binarization step truncates this erroneous information.

Based on the occurrence histograms, a non-linear multi-class SVM (cf. Section 2.2) is trained with labels for different character classes. We build classes for each main character and group all other characters into a class “others” (resulting in six different classes altogether, see also Section 3.1.4).

In our work, we use the libSVM library [CL01] as SVM implementation. We follow the suggestion of Hsu et al. [HCL03, Sec. 3.1] and work with the RBF kernel function.



(a) Example visual words from a SIFT code book.



(b) Example visual words from a color code book.

Figure 5.2: *Illustration of image patches that were (based on the descriptor type) clustered into visual words; (a) three example visual words taken from a SIFT code book; note that the patches have been clustered regardless their scale; (b) three example visual words taken from a color code book; note that for these visual words the surrounding of the actual image patch has been included as well.*

They give three main reasons for this choice. First, the linear kernel is a special case of the RBF kernel. Second, the number of parameters is lower for the RBF kernel than, e.g., for the polynomial kernel. Last, the RBF kernel has less numerical difficulties than, e.g., the polynomial kernel (which can yield infinity as result) or the sigmoid kernel (which is not defined for all parameters). Since the one-against-one strategy is computationally more efficient (for our number of classes) and since its performance is better than one-against-all (cf. Section 2.2.3), we use a one-against-one implementation for the multi-class SVM.

To balance the training set, the weight for misclassifications (i.e., C in Equation (2.24)) of each character class is based on its size as follows. Let n_c be the number of samples for class c and let N be the number of classes. The average number of instances per class is defined as

$$\bar{n}_c = \frac{\sum_c n_c}{N}. \tag{5.1}$$

The weight w_c for class c is then given by

$$w_c = \frac{\bar{n}_c}{n_c}. \tag{5.2}$$

The character class prediction of an unknown body part image (e.g., a new detection in an image) is done similarly to the training part described before: The detected image is extracted, scaled to the fixed height, feature points are extracted by dense sampling, and a histogram of visual word occurrences is computed using the code books. The code book is chosen according to the body part type and the sampled feature type (SIFT or color). The histogram is classified with the corresponding SVM model that has been trained for the particular body part and code book type.

Classification with Combined Code Books

In order to combine the SIFT and color code books, the histograms of visual word occurrences are concatenated. An SVM model is learned on the combined code book histograms. For the character class prediction of an unknown body part image, histograms of the SIFT and color code book are concatenated and classified with the learned SVM model.

By combining different code books, a higher overall performance is obtained. In Section 5.3.4 different weights for the histograms are investigated.

Classification with Connected Body Parts

Our human detection is based on the assembly of different body parts. In order to improve the performance of the character recognition, the prediction results of connected body parts can be combined to a more robust prediction. This is done as follows.

The classification (as explained before) of a detected body part i with a trained SVM model yields a probabilistic vote for the detection. This vote consists of a probability

value for each character class c. The probability that detection i belongs to the character class c is denoted as

$$p_c^{(i)} \in \mathbf{R}^+, \qquad \sum_{j=1}^{N_C} p_j^{(i)} = 1, \forall i. \tag{5.3}$$

N_C represents the number of character classes and is $N_C = 6$ in our case (cf. Section 3.1.4). Connected body parts of i are found using the same heuristics as in Section 4.2.4. All connected body parts of i are described by the set $\mathbf{C}^{(i)}$. The final vote $c^{(i)}$ for a character class of detection i is then given by the class with the highest summed probability over all connected body parts including i itself, i.e.:

$$c^{(i)} = \arg\max_c \left\{ \sum_{j \in \mathbf{C}^{(i)} \cup \{i\}} p_c^{(j)} \right\}. \tag{5.4}$$

5.3 Parameter Evaluation

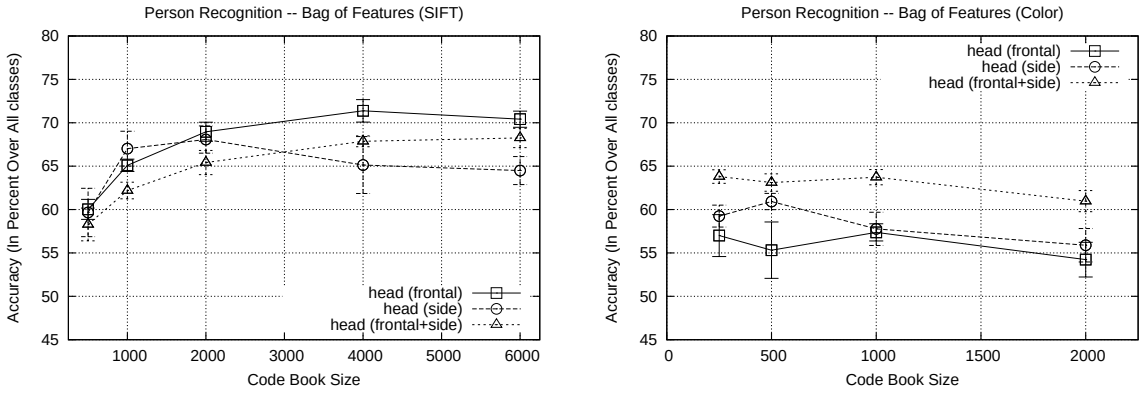
In this section, various parameter settings for the character recognition are evaluated. Section 5.3.1 gives information on the data set and the evaluation methods that are used. The classification performance based on code books that combine frontal and side views are studied in Section 5.3.2. Then, Sections 5.3.3 and 5.3.4 evaluate different code book sizes and the combination of code books, respectively. The chapter concludes with an analysis of combined character classification based on connected body parts in Section 5.3.5. (Further results of the whole system are presented in Chapter 6.)

5.3.1 Data Set and Evaluation

The experiments for our character recognition are based on the data set described in Chapter 3. Since the focus lies on finding characters in one particular video sequence, only the annotation data of Buffy1 is used. The appearance (especially clothing) of characters in different video sequences can change drastically. All body part images are extracted at a common height (cf. Section 5.2.1). As for the human detection (cf. Section 4.2.4), all body parts are also oriented to a common side (which is left).

For the Buffy1 data set, we create 6 different categories: Buffy, Willow, Xander, Riley, Giles, and Others (cf. Section 3.1.4). The five characters are the main characters and occurred the most often in our annotation data. The category Others contains all other annotated characters that did not belong to one of the five main classes.

For the performance evaluation, we use accuracy (see also Section 2.1.3) over all classes, i.e., the total number of correctly classified positives (true positives and true negatives of all character classes) divided by the total number of samples (number of positive and negative samples of all classes). The accuracy is obtained by applying a five-fold cross validation (see also Section 2.1.2) to the computed occurrence histograms. Altogether, four independent runs are performed on the data set. Their mean accuracy is noted with error bars that indicate the standard deviation. Our k-means implementation



(a) (b)

Figure 5.3: *Performance of the character recognition based on the frontal head, side head, and the combined front+side head body part data; (a) the performance based on the SIFT code book; (b) the performance based on the color code book (we plot the mean value of multiple runs, the error bars denote the standard deviation).*

is similar to Algorithm 1, it does not perform multiple runs with different initial cluster centers.

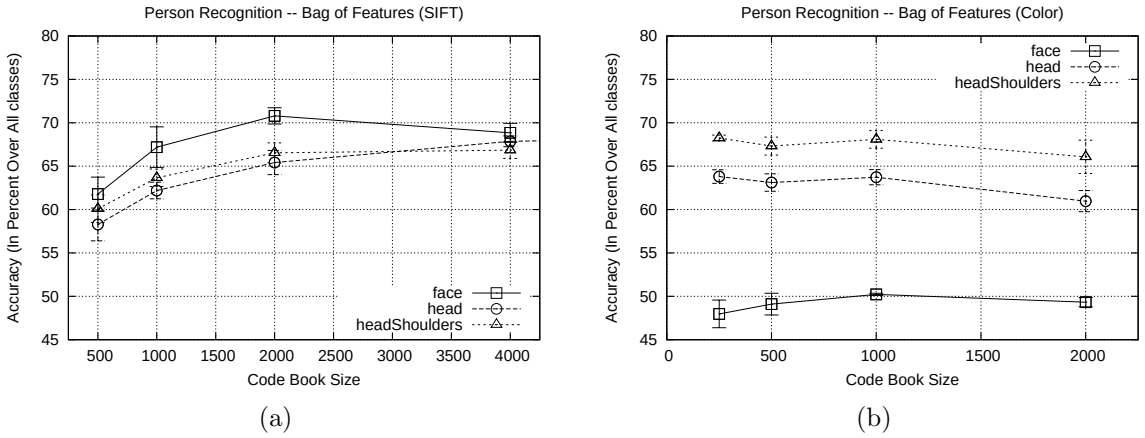
5.3.2 Combining Code Books of Different Views

Before evaluating different parameter settings, we are interested in knowing how the performance changes with a combined code book for frontal and side detectors of a particular body part. Combining side and frontal code books is practicable since it allows us to save computational costs. Figure 5.3 illustrates the performance of an SVM classifier based on different code books. We investigate the performance on code books based on the frontal head, the side head, and the combined frontal+side head body part data. We generate code books of different sizes for the different head views and code book types (SIFT [Figure 5.3(a)] and color code book [Figure 5.3(b)]).

The SVM on the SIFT code book performs worse for the side view than for the frontal view (see Figure 5.3(a)). This is assumably due to more discriminant structures in the frontal view. A code book size of 2000 words performs best for the side view. For the frontal view, the performance rises up to a code book size of 4000 words. The combination of both views results in a worse performance than the frontal view.

Interestingly the combination of the color code books yields a better performance than the classification based solely on one of the code books (see Figure 5.3(b)). It seems that the feature space of the combination can be better clustered since it consists of more feature points.

Since the combination of the code books of both views is practicable, we accept the loss in performance for the SIFT-based code books. This decision simplifies our approach and saves computational cost. In the remainder of our work, we combine frontal and



(a) (b)

Figure 5.4: *Performance of the character recognition based on the face, head, and head+shoulders body part data; (a) the performance based on the SIFT code book; (b) the performance based on the color code book (we plot the mean value of multiple runs, the error bars denote the standard deviation).*

side views of a body part into one code book.

5.3.3 Different Code Book Sizes

Figure 5.4 shows the performance of SVM models that have been learned on code books of different sizes. Among the SIFT code books (Figure 5.4(a)), the face code books yield the best results. The structure in face images is more rigid than in images of other body part types. Therefore characteristic structures can be more reliably recognized. This is an explanation why the face code book performs best. Also, there is less background information in face images than in, e.g., head+shoulders images. A code book size of 2000 seems a sensible trade-off between computational cost and accuracy.

Figure 5.4(b) shows performance results for color code books. SVM models that are only based on face color yield the worst results among all SIFT and color code books. This is understandable since face regions do not contain a lot of discriminative color information. Hair color may play a role here since it is sometimes visible in face images. The best performance among the color code books yields the head+shoulders code book. Head+shoulder body parts have the highest amount of discriminative color information. Especially the color of clothing can give a strong hint which character is present in the image. We choose a code book size of 1000 words as trade-off for all color code books.

5.3.4 Code Book Combination

By combining SIFT and color code books, a gain in performance is achieved. Figure 5.5 shows results for the combination of histograms of both code book types. Different weights are applied to the color code book by multiplying its binarized histograms with

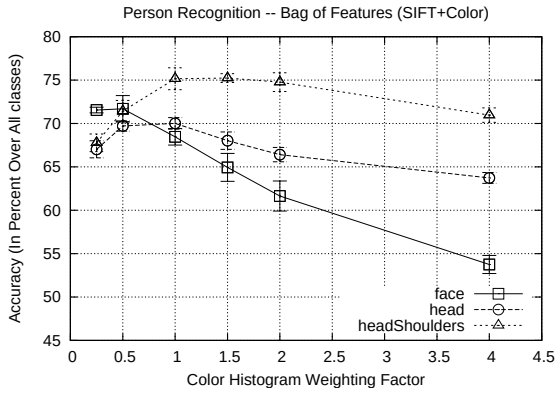


Figure 5.5: *Performance of the character recognition based on combined SIFT+color code books for the face, head, and head+shoulders body parts; for the SIFT code book we use a size of 2000 words and for the color code book 1000 words (we plot the mean value of multiple runs, the error bars denote the standard deviation).*

a weighting factor. The SIFT code book histograms are left unchanged (i.e., with a weight of 1). The code book size is 2000 for the SIFT descriptor and 1000 for the color descriptor. One can see that more significant weights on the color code book make a difference. With exception of the face recognition, a weight of 1 always yields best results. This is probably due to the significantly worse results of the color code book. For the sake of simplicity, we decide to use a weighting factor of 1 for both code books. The following table summarizes the accuracy for the single code books (with SIFT code book size 2000, color code book size 1000) and the final combined one:

	color	sift	color+sift
face	50.2	70.8	68.5
head	63.7	65.4	70.0
headShoulders	68.1	66.5	75.2

5.3.5 Body Part Combination

A significant gain in performance is achieved by combining probabilistic votes of connected body parts (cf. Section 5.2.3). Tables 5.1 and 5.2 show confusion matrices for single body parts and connected body parts, respectively. The accuracy over all classes (and over all body parts) increases from 74.4% to 81.3% for the combination of votes from connected body parts. Note that the leave-one-out cross validation (cf. Section 2.1.2) has been applied here in order to compute the confusion matrices and the accuracy rates. Recall and precision rates vary since the confusion matrices are not symmetric. One can notice a general confusion of all classes with the classes Buffy and Others. The most probable explanation is the general nature of the class Others as well as the larger set of training data for Buffy and Others (cf. Table 3.3). Although weights have been assigned to the classes in order to balance the data set (cf. Section 5.2.3), the larger classes seem to dominate the classification slightly. A larger data set as well as different

sampling strategies for the feature extraction (e.g., based on interest point detectors) could help to decrease the confusion between the classes.

True Classes \ Predicted Classes	Others	Buffy	Riley	Willow	Giles	Xander	Precision	Recall
Others	761	158	39	32	20	18	70%	74%
Buffy	115	864	9	33	14	13	75%	82%
Riley	69	23	296	5	24	23	73%	67%
Willow	46	54	8	302	13	29	75%	67%
Giles	46	40	30	3	326	7	79%	72%
Xander	48	17	25	27	15	356	80%	73%

Table 5.1: *Confusion matrix for single body parts; accuracy over all classes and body parts is* 74.4%*; note that the confusion matrices of 4 independent runs and of all different body parts (face, head, head+shoulders) have been accumulated; confusion matrix and precision/recall rates are based on combined SIFT-color code books; the size of the SIFT code book is 2000 and the color code book is 1000.*

True Classes \ Predicted Classes	Others	Buffy	Riley	Willow	Giles	Xander	Precision	Recall
Others	828	143	10	23	16	8	76%	81%
Buffy	81	949	1	12	1	4	78%	91%
Riley	54	10	347	0	11	18	89%	79%
Willow	43	50	13	319	12	15	84%	71%
Giles	49	35	5	2	358	3	88%	79%
Xander	40	24	15	24	11	374	89%	77%

Table 5.2: *Confusion matrix for the combination of connected body parts; accuracy over all classes and body parts is* 81.3%*; note that the confusion matrices of 4 independent runs and of all different body parts (face, head, head+shoulders) have been accumulated; confusion matrix and precision/recall rates are based on combined SIFT-color code books; the size of the SIFT code book is 2000 and the color code book is 1000.*

6 Experimental Results for Video Annotation

This chapter presents results of our entire system (i.e., human detection and character recognition). First, an overview of the system is given in Section 6.1. Section 6.2 presents and discusses then experimental results for the detection of humans and for the character identification. The system to the entire Buffy1 video sequence.

6.1 Overall Architecture and Training Data

The system executes the following steps:

(1) A given video is segmented into shots (cf. Section 3.2). For each shot, every n^{th} frame is extracted from the video for further processing (we set $n = 10$). Shots which contain only one single frame are deleted.

(2) The trained human body part detectors are applied to each extracted frame (cf. Section 4.2). Before detected body parts are combined, a low threshold on the normalized score values suppresses weak detections. For all body part detections in a frame, plausible body part combinations are identified. The final detection results are again thresholded.

(3) The character identity of each detected body part is predicted based on the created code books and the corresponding SVM models that have been trained on the code book histograms (cf. Section 5.2). The votes of connected body parts are combined in order to determine the character's identity.

As example video sequence, we use the entire Buffy1 video (cf. Section 3.1.1). The body part detectors are those that have been trained in Chapter 4. The generated code books and trained BoF-SVM models for the character recognition system are those from Chapter 5. The detectors as well as the code books and BoF-SVMs are trained/generated on our annotation data set from the video Buffy1. The training set for Buffy1 consists of images that are randomly taken from the whole video sequence, and our system is applied on, roughly speaking, every 10^{th} frame of the Buffy1 sequence. Therefore it is possible that some frames that are part of the training data set are also among the test frames that are automatically extracted by the system. Assuming that every 10^{th} frame is investigated, i.e., $\sim 10\%$ of all frames of the video sequence, we can estimate that $\sim 10\%$ of the training data, i.e., ~ 31 frames are present in both data sets. 31 frames

Others	Buffy	Riley	Willow	Giles	Xander
3146	3062	1125	1750	1440	1534

Table 6.1: *Total numbers of automatically assigned characters classes for all body parts.*

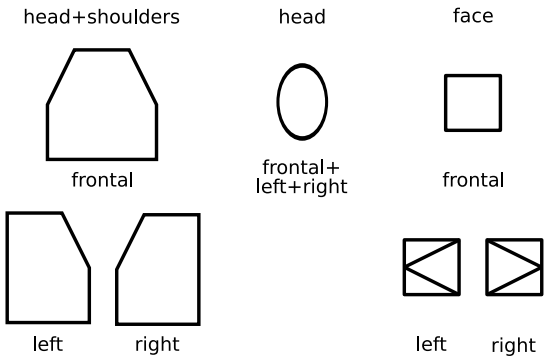


Figure 6.1: *Illustration of how the different body parts are visualized in the result images; note that for the head detection, we denote frontal and side detections both with an ellipse in order to avoid an overload of visual information in the images.*

out of a total of 6291 test images, i.e., $\sim 0.5\%$ of the test data is assumably already known to the system.

6.2 Results

In the full Buffy1 video, the system detects 829 different shots. In 6291 extracted frames, it detects 12057 different body parts. Table 6.1 shows for each character class the number of body part classifications. In the following, detected body parts are visualized as shown in Figure 6.1. The identified character class (cf. Sections 5.3.1 and 3.1.4) of a group of connected body parts is written below the detections. All presented detections have been obtain by applying one common threshold.

Figure 6.2 gives some examples of typical correct detections and character labeling. As can be seen, different orientations and combinations of the body parts are detected successfully. Note that the size of humans that can possibly be detected is due to our detection scheme (cf. Section 4.2) generally constrained to the window size of the body part detectors (ca. 300 pixel height of person in upright pose). If the size of a person in an image is too small, his faces cannot be detected. However, in order to detect the person, the detection of head and head+shoulders suffices. Note also that partly occluded and overlapping people are detected correctly.

In Figure 6.3, examples for false (or missing) detections as well as incorrect character identifications are presented. Especially during dark scenes with blurred lights in the background, the head detector fired on blob like structures since they resemble the shape of a head (Figures 6.3(a) and (b)). If a head+shoulders-like structure was detected as well, both false-positive detections would be combined. However, most false-positive

Figure 6.2: *Typical examples for correct final detections and character identifications.*

Figure 6.3: *Examples for false (or missing) detections and character identifications; (a,b) light blobs can be detected as head-like structure; (c,d) distant people are detected less well than closer ones; (e,f) mislabeling, especially due to similar appearance (e.g., false Buffy label); (g) the profile detection is more difficult; (h) less usual view point are not always detected (e.g., downwards looking and rotated face).*

detections are labeled with the class OTHERS. That shows that this character class serves well as container for everybody (and everything) else.

Figures 6.3(c) and (d) illustrate the limitation of the system with regard to the size of humans. Due to the fixed window size of our body part detectors, humans that appear too small in images cannot be detected.

We experienced that the character class labeling works quite well, somewhat similar to the results reported in Chapter 5. We found that Buffy, Willow, and Xander were the characters that were most reliably identified. However, sometimes the identification failed as shown in Figures 6.3(e) and (f). More common were mislabeling of other characters as belonging to one of the main character classes than vice versa. Characters with for example blond hairs were several times classified as Buffy, whereas Buffy herself would be identified correctly in most cases. This is certainly due to the general nature of the class OTHERS and to the relatively small data set that was used for the training of the BoF-SVMs (cf. Section 3.1.4). With a larger number of training examples, we expect that the confusion between the classes decreases.

We note that the detection of profile faces seems to be more difficult (i.e., less reliable) than the detection of frontal faces (see Figure 6.3(g)). We also experienced difficulties with detecting faces and other body parts from unusual (with respect to the training data) point of views (see Figure 6.3(g)). These problems are more related to the training of the detectors and can (at least partially) be overcome with a larger training data set. For our detectors, we used a medium sized data set. In other works, detectors are trained on data sets that are one order of magnitude larger, e.g., in [DT05].

Concluding, we can state that we are able to obtain good detection and recognition results with a system that combines different body part detectors and a Bag-of-Features based character recognition.

7 Conclusion and Future Work

7.1 Conclusion

In this master thesis, we proposed and evaluated an approach to the detection and identification of humans in TV-style video sequences. Our approach is supervised and builds on two main techniques:

- A body-part-based human detector and
- A Bag-of-Features-based character identification system.

To segment a full video sequence into smaller coherent sub-sequences, a basic video shot detection is introduced based on color histograms.

For the human detection, we employ the detector introduced by Dalal and Triggs [DT05]. In order to deal with humans in full and close-up view and in order to improve performance, separate body part detectors (face, head, head+shoulders) are trained. Following the approach of Mikolajczyk et al. [MSZ04], body part detections are combined using a Gaussian based model for the geometric relations of the parts. We investigated different training data alignment, scale normalization, and augmentation strategies. We also evaluated view-dependent detectors. Concluding, we found for the training images that an alignment based on fix points and a scale normalization based on the height of annotated bounding boxes worked best for our case. The performance improved significantly when the training data set was augmented by additionally scaled and rotated versions of the training images. A further gain in performance is achieved by view specific detectors, i.e., frontal/side and especially left/right. Similarly to results reported by Mikolajczyk et al. in [MSZ04], the performance of our body part detectors increases with the model for geometric relations.

We base the character identification on the Bag-of-Features approach. SIFT and color code books are generated from our training data by using k-means. On the same training data, occurrence histograms of visual words are computed and learned with a non-linear Support Vector Machine. We evaluated different code book sizes, the combination of code books, and a combined voting scheme based on multiple detected body parts. We conclude that the Bag-of-Features approach has been applied successfully to the task of human identity recognition. The combination of different types of code books and a combined voting scheme of different connected body parts help to improve the classification results.

Applied to the full video "Buffy vs. Dracula" from *Buffy the Vampire Slayer Season 5*, we obtain good results for both, the human detection as well as the character identification. Especially the combination of the different body parts helps to improve the overall

human detection. False-positive detections can occur at, e.g., blob like structures that are detected as head- and shoulder-like structures. Due to fixed detector window sizes, the size of people in images that can be detected is constrained. The detection works most reliably for people whose upper body part is fully visible and with in resolution such that the head+shoulders body part is about 150 pixel in height. Then all body parts can be detected and combined. The character identification works well on test sequence, although we employ less sophisticated methods for the clustering (k-means) and the feature extraction (dense sampling). As with the human detection, the combination of the votes from the different body parts helps to improve the final results. Sometimes we obtain wrong predictions due to similar appearances of certain characters and due to a relatively small training data set (for the code books as well as for the SVM training).

7.2 Future Work

Our initial results are very promising. In the following, we present possible improvements for future work. We group these ideas into three categories: Human detection, human character recognition, and scene understanding.

Human Detection. It would be interesting to know wether a higher detection performance can be achieved with the body part detectors. For this, a more careful investigation of their intrinsic parameter settings would be necessary. The detection system by Dalal and Triggs [DT05] learns characteristic structures and shapes for a class of objects. Therefore faces and heads – as they are rather rigid objects – yield a good detection performance with this approach. However, other detection approaches (e.g., based on contour parts) could be more effective for non-rigid-body parts such as head+shoulders or also legs.

Temporal information from the video sequence is presently not used. Constraints on temporal consistence would certainly improve detection (and identification) results. Further clues could be incorporated by, e.g., additional template based tracking of detected body parts throughout a shot sequence. In a video sequence, we could also speed up the body part detection by limiting the exhaustive search in scale space.

Human Character Recognition. For the Bag-of-Features based character recognition, we use k-means clustering along with a binarized histogram of visual word occurrences. More sophisticated clustering methods could be hierarchical ones, e.g., a hierarchical k-means or mixture of Gaussians. Works on kernel k-means and Support Vector Clustering could also be interesting. To improve the identification accuracy, occurrence histograms based on soft clustering methods (e.g., based on the distance of a feature to the cluster center) could be promising.

We extract features points on a dense grid whereas other approaches sample in content-driven fashion, e.g., with interest point detectors. Since our images are quite small, interest point detectors do not yield enough points. Instead, a random sampling based on some kind of continuous cornerness-function could be more appropriate.

Other directions for future work include experiments with different descriptors, especially color descriptors. We expect more sophisticated color histogram descriptors to enhance results. Then, instead of combining the code book histograms, the color and SIFT descriptors could be combined directly into one single descriptor. Since this combination augments the feature space, more specific visual clusters would be generated. Additionally, the scale level of sampled feature points could be included into the feature space.

Currently, we learn character classes in a supervised manner. Much more appealing would be an unsupervised approach. This could be done by applying clustering methods to code book histogram space.

Scene Understanding. The shot detection is preliminary for scene detection since a scene consists of several shots. A scene could be detected based on background models that are learned for each shot (e.g., based on the Bag-of-Features approach). Hypotheses for character identities could then be strengthened throughout one scene. If it is possible to detect scene changes, reasoning on a more semantical level could be possible (e.g., reasoning about the interaction between people throughout a scene).

Bibliography

[Alv02] Sergio A. Alvarez. An exact analytical relation among recall, precision, and classification accuracy in information retrieval. (BCCS-02-01), June 2002.

[AR02] Shivani Agarwal and Dan Roth. Learning a sparse representation for object detection. In *Proceedings of the 7th European Conference on Computer Vision*, volume 4, pages 113–130, 2002.

[AZ05] O. Arandjelović and A. Zisserman. Automatic face recognition for film character retrieval in feature-length films. *In Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1:860–867, June 2005.

[BBE+04] Tamara L. Berg, Alexander C. Berg, Jaety Edwards, Michael Maire, Ryan White, Yee-Whye Teh, Erik Learned-Miller, and D. A. Forsyth. Names and faces in the news. volume 02, pages 848–854, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[BKL99] Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Computational Learing Theory*, pages 203–208, 1999.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[cla06] Class – cognitive-level annotation using latent statistical structure. http://class.inrialpes.fr/, 2006.

[Col06] Philippe Colantoni. Rgb cube – software to display the rgb cube in different color spaces, 2006.

[Dal06a] Navneet Dalal. Personal communication, April 2006.

[Dal06b] Navneet Dalal. *Finding People in Images and Videos*. PhD thesis, Institut National Polytechnique de Grenoble, 07 2006.

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000. http://www.cs.brown.edu/research/vis/results/bibtex/Duda-2000-PCL.bib(bibtex: Duda-2000-PCL).

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[DWF+04] Chris Dance, Jutta Willamowski, Lixin Fan, Cedric Bray, and Gabriela Csurka. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

[ESZ06] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy – automatic naming of characters in tv video. In *Proceedings of the British Machine Vision Conference*, 2006. to appear.

[EZW+06] Mark Everingham, Andrew Zisserman, Christopher K. I. Williams, Luc van Gool, Moray Allan, Christopher M. Bishop, Olivier Chapelle, Navneet Dalal, Thomas Deselaers, Gyuri Dorko, Stefan Duffner, Jan Eichhorn, Jason D. R. Farquhar, Mario Fritz, Christophe Garcia, Tom Griffiths, Frederic Jurie, Daniel Keysers, Markus Koskela, Jorma Laaksonen, Diane Larlus, Bastian Leibe, Hongying Meng, Hermann Ney, Bernt Schiele, Cordelia Schmid, Edgar Seemann, John Shawe-Taylor, Amos Storkey, Sandor Szedmak, Bill Triggs, Ilkay Ulusoy, Ville Viitaniemi, and Jianguo Zhang. The 2005 pascal visual object classes challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment (PASCAL Workshop 05)*, number 3944 in Lecture Notes in Artificial Intelligence, pages 117–176, Southampton, UK, 2006.

[Faw03] Tom Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, Intelligent Enterprise Technologies Laboratory, HP Laboratories Palo Alto, January 2003.

[FR98] Adrian Ford and Alan Roberts. Color space conversions. Technical report, Westminster University, 1998.

[FZ02] A. W. Fitzgibbon and A. Zisserman. On affine invariant clustering and automatic cast listing in movies. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 3, pages 304–320, 2002.

[HCL03] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

[HL01] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines, 2001.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[LEA] LEAR. Research group at inria rhône-alpes. http://lear.inralpes.fr.

[Lie01] Rainer Lienhart. Reliable transition detection in videos: A survey and practitioner's guide. *International Journal of Image and Graphics*, 1(3):469–486, 2001.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LSS05] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 878–885, Washington, DC, USA, 2005. IEEE Computer Society.

[MS04] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[MS05] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[MS06] Marcin Marszałek and Cordelia Schmid. Spatial weighting for bag-of-features. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2006.

[MSZ04] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision*, volume I, pages 69–81, 2004.

[NZ06] M-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006. To appear.

[ope06] *Intel Open Source Computer Vision Library (OpenCV)*, 07 2006.

[VJ02] Paul Viola and Michael Jones. Robust real-time object detection. Technical report, 2002.

[VJ03] Paul Viola and Michael J. Jones. Fast multi-view face detection. Technical report, Mitsubishi Electric Research Labs, June 2003. Demonstration at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03).

[VR79] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.