Looking at People

Detecting People

&

Estimating their Facing Direction

Martijn J. Brekhof

University of Amsterdam

Looking at People

Detecting People

&

Estimating their Facing Direction

Martijn J. Brekhof

Advisor: Prof. D. M. Gavrila

IAS
intelligent autonomous systems

University of Amsterdam

Martijn J. Brekhof

Looking at PeoplE

Detecting people & Estimating their Facing Direction

Master's thesis in Computer Science,
Intelligent Automous Systems
University of Amsterdam
Supervisor: Prof. Dariu M. Gavrila

April 27, 2005

Looking at people logo design: M. J. Brekhof and M. Spekle

Acknowledgements

I would like to thank my advisor Prof. Dariu M. Gavrila for his patience and guidance through the whole of this project. Also many thanks to Christian Wöhler and Stefan Munder from DaimlerChrysler research in Germany, for their assistance in using texture classification techniques.

I especially would like to thank Marius Olsthoorn for his detailed peer reviews and insightful discussions. Thanks to Rinaldo Sikking for the in-depth discussions and Marcel Toele for his comments on camera calibration.

Table of Contents

List of Figures

The ability to recognize humans and their activities by vision is key for a machine to interact intelligently and effortlessly with a human-inhabited environment

D. M. Gavrila

1 Introduction

Arguably, the most interesting object in a human-inhabited environment is the human itself. Much research has been done to use imagery of humans to detect and analyze its behavior. In this thesis we address the problem of estimating a person's facing direction in low resolution gray scaled imagery.

In this chapter we describe why people are interesting objects to analyze by vision. We describe the most common methods developed over the last few years and present the scope and overall structure of this thesis.

1.1 Vision

According to Zeki [37] and Russ [25], we humans rely mostly on our vision to provide us with information about the environment. As Zeki [37] states

> Vision is the most developed sense in man and much of our knowledge of the external world comes through it.

Russ [25] goes a little bit further and states that

> Humans are primarily visual creatures. Not all animals depend on their eyes, as we do, for 99% or more of the information received about their surroundings.

This means that a human-inhabited environment is especially organized to be perceived by visual means. Traffic signs are placed at clearly visible places and made in such a way that they are easily spotted by vision. We use sign language or printed text to communicate through visual means. When a computer system needs to provide information to a human this is mainly done using a monitor. Although auditory signals are not to be neglected, visual information is for humans the most important. Hence, vision is a good candidate to provide any system with the information needed to intelligently interact with a human-inhabited environment.

Practical research areas using vision

We humans use computers, for example, entertainment, to manage large amounts of data or information retrieval. We delegate simple and annoying tasks we do not want to be burdened with to machines, which means that most have to act in a human-inhabited environment. Currently the interfaces for humans to interact with machines are not at all similar in how humans interact with each other. Although mouse devices have been accommodated to the form of our hands and we can use light pens to make drawing on a computer screen more realistic, speech and gesture interfaces are still mainly in a research phase [10].

An advantage of using vision is that it is non-intrusive. This is especially useful in surveillance tasks where the system must observe people without requiring them to provide information actively. For example, to protect the pedestrian, the European Union and several car manufacturers have combined their efforts in developing a car-driver assistant system. This system should detect pedestrians and take precautious measures when a collision is unavoidable to limit the amount of injury to the pedestrian [11]. Another area of interest is surveillance where public areas are analyzed and unusual usage of these areas should be detected. Observing areas is mainly done by using cameras, however the analyzing process is still done by humans. Observing areas where most of the time nothing unusual happens, we humans easily overlook situations which are suspicious. The University of Amsterdam in collaboration with TNO research are conducting research on how ”intelligent” sensors might aid human observers, making them more efficient by only considering observations that are of interest.

1.2 Scope of this thesis

The examples in the previous section require more than just detecting people. For a system to successfully use gesture interfaces, it must have a clue when

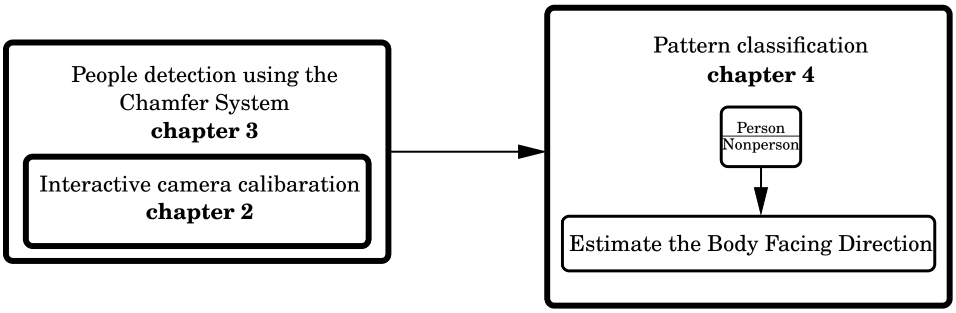

People detection using the
Chamfer System
chapter 3
Interactive camera calibaration
chapter 2
Pattern classification
chapter 4
Person
Nonperson
Estimate the Body Facing Direction

Figure 1.1: *Main system blocks*

it is addressed to and when not; to avoid a possible collision, it must know the trajectory of a walking pedestrian; to detect suspicious interactions (e. g. fighting, picking someone's pocket) between two or more people the system must be able to tell who are interacting and who are not. For this we address in this thesis the problem of estimating the *Body Facing Direction* (BFD) of people from imagery.

The BFD is defined as the direction the body is pointing, we take the chest as the front and the back as the rear. In this thesis the BFD is restricted to four main directions: **front, back, left** and **right**. Hence a front view is a human with his or her chest facing the camera, a back view the opposite, a left view when his or her chest is facing left and a right view when facing right.

The subject of this thesis falls within the area of *image processing* and *computer vision*. Image processing is concerned with data acquisition, processing and visualization [25]. Computer vision is mainly concerned with interpreting images making explicit the geometric and dynamic properties of the 3-D world.

The methods used to process and analyze imagery are dependent on the acquisition method used and target application domain. We have taken surveillance as our application domain and therefore only consider methods using single view gray valued images most common to camera surveillance systems. The viewpoint in a typical camera surveillance system is shown in figure 1.2. As can be seen people partly occluded are mostly visible by their heads and shoulders. Hence we concentrate on image features most apparent in these body parts.

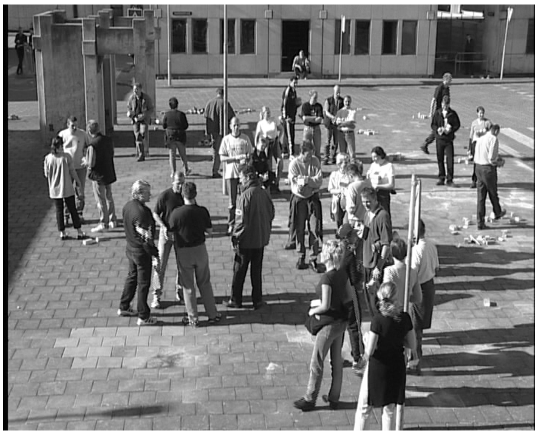


Figure 1.2: *Example of an image taken from a typical surveillance camera viewpoint*

The proposed system consists of two main components which are shown in figure 1.1. The idea is that using the BFD and the distance between people facing each other, people can be classified as having some kind of interaction. The distance between people in the 3-D world can be estimated using the camera parameters. To estimate the camera parameters we use a calibration method based on vanishing points. However, we only use the camera parameters to limit the search space for the detection method. To detect people in imagery the **Chamfer System** [12] is used. This method uses shapes to locate similar shapes in imagery. Estimating the BFD of detected people is done using texture classification.

1.3 Related work

Camera calibration

Classical camera calibration techniques uses the geometry of known objects to estimate the camera parameters. These originated from photogrammetry, aimed at extremely accurate camera calibration [17]. The most well known calibration methods in computer vision are those of Tsai [29] and Caprile and Torre [3]. Tsai [29] uses a planar calibration grid of known geometry, Caprile and Torre [3] use the properties of vanishing points of straight lines for orthogonal directions.

The methods by Tsai [29] and Caprile and Torre [3] have been used by many others to develop auto-calibration or intuitive interactive calibration techniques. Cipolla et al. [4] use vanishing points to calibrate imagery of architectural scenes from multiple viewpoints. This can be used to create 3-D models and renderings of architectural buildings from photographs. Knight et al. [16] use prior knowledge of the motion pattern of a ground plane from a rotating camera. This may be used by non-stationary surveillance cameras or mobile robots. Lv et al. [19] use a sequence of a walking human to estimate the horizon line and vertical vanishing point. This is typically applicable in surveillance scenarios using stationary cameras.

Detecting people

Methods for detecting objects in an image make use of low-level features such as optical flow, color cues, textures or shapes. According to Gavrila [10], the methods to analyze people in imagery can be divided into using 2-D or 3-D models. Where 2-D models can be further divided into two groups making use of either explicit or implicit shape models. The difference between explicit and implicit models is that explicit shape models contain labeled body parts. Methods using 3-D modeling generally require detection of limbs and their joints to estimate the body pose (e. g. [21]). As we are primarily interested in the upper parts of a human body, 3-D modeling techniques will not be considered.

Detection of people in imagery is a complicated task as a human body shows a significant variability in pattern and color. This makes it hard to create a general model of a human being. The problem is made more complicated as humans are generally observed in complex dynamic scenes. These are scenes that change over time in a complex manner, making it hard to predict the changes. Most people detection or analysis methods therefore restrict to only a few features found characteristic for the task at hand. An often used feature to detect people in complex dynamic scenes is shape [13, 10, 33, 12].

The shape of a human being is characteristic and mostly consistent for every human being. One problem is the non-rigidity of a human being, which makes its shape change over time. To overcome this problem Haritaoglu et al. [13] uses shape models which are adjusted during matching to accommodate to small changes. Oren et al. [22] use wavelet coefficients as low-level intensity features to represent an object of interest. Of the many wavelet coefficients they select a small subset to represent the object of interest, after which they trained a support vector machine to classify image regions to detect frontal and rear views of pedestrians. Gavrila and Philomin [12] uses the chamfer matching algorithm based on template matching in distance-transformed binarized edge images to detect pedestrians. The computational complexity is kept moderate by generating a shape model hierarchy during a training stage and by a coarse-to-fine detection approach.

One of the main advantages of the methods developed by Oren et al. [22] and Gavrila and Philomin [12] is that they are somewhat tolerant to partial occlusions.

Estimating the facing direction

Much research has been done in estimating the head pose of humans from images. Two main methods have been employed over the years, which are geometric or texture based. The former exploits the geometric properties of a human's face (e. g. distance and location of the eyes, mouth and nose) [26, 15, 14] the latter builds for different poses a model and estimates which model matches best on the current input pattern [36, 27].

According to Stiefelhagen et al. [27], the main advantage of using neural networks (i.e. texture based) to estimate head pose as compared to using a model based approach is its robustness. The geometric based approach relies on tracking a minimum number of facial landmark points in the image correctly, which is a difficult task and is likely to fail. The neural network based approach does not require tracking detailed facial features because the whole facial region is used for estimating the user's head pose.

1.4 Facing direction features

We have restricted our problem domain to single view gray scaled imagery taken from an elevated viewpoint using low cost hardware. Hence, we need to focus on features that are visible within low resolution imagery.

According to Wu and Toyama [36] the information needed to reliably detect facial landmarks (e.g. an eye) is not available in low-resolution imagery. Considering the application area as shown in figure 1.2, we cannot assume that the resolution of the images (i. e. 720×576 pixels) will contain enough detailed information to perform gaze-direction estimation. Therefore, we consider the human upper body as well.

However little research has been performed on the subject of human-body facing direction features. Analyzing the people as shown in figure 1.2 we can distinguish three main features to determine the facing direction

- shoulder width
- facial skin and hair contrast
- head location relative to the body

The shoulder width feature can be used as people's shoulder width differs when they are standing sideways to or towards/away from the camera; People tend to keep their face clear from hair and thus there is usually a clear intensity difference between a person's facial skin and hair. Most people naturally tend to lean their head forward relative to their body, this is especially apparent when a person is viewed from a side.

Combining the three features is not straightforward as people can move their head independently from their body to some extend. To limit the amount of variation between the three BFD features we only consider situations where people are looking in the same direction their chest is facing.

1.5 Thesis overview

Chapter 2 describes a method to interactively calibrate a camera using limited knowledge about the scene and camera. The method is based on estimating the location of three vanishing points. Using prior knowledge of the camera used, the camera can be partially calibrated. By estimating the length of a vertical object (i.e. perpendicular to the ground plane) the camera height can be calculated and all camera parameters are obtained to estimate a detected person's location in the real world.

Chapter 3 describes and discusses the Chamfer System for detecting people in images. It uses a Distance Transform to efficiently search for possible locations an object of interest may be located and makes use of a hierarchy of shape models to exploit the similarities between different shape models.

Chapter 4 describes the classification method used based on textures. This is used to reduce the amount of false detections (false positives) and estimate the facing direction of people in the four main BFDs: left, right, front and back. For this a neural network (NN) is used based on local processing (i.e. receptive fields).

Chapter 5 presents and discusses the performed experiments using the methods described in chapters 2, 3 and 4.

Chapter 6 discusses the obtained results in chapter 5 and describes some additional extensions that might be implemented in the future, making the system more robust and less reliable on human input.

The camera makes everyone a tourist in other people's reality, and eventually in one's own.

Susan Sontag

2 Camera Calibration

For any system using a camera to observe its environment, camera calibration is a necessity when one wants to relate objects based on distances. Cameras come in many sizes and types which have different camera parameters. This means that for a system to robustly cope with different cameras or changing camera parameters (due to wear or change of focal length) it needs to adjust its camera model using as less prior knowledge about the camera as possible.

In this chapter we describe a calibration method applicable on most up to date consumer cameras. The method is based on a method developed by Lv et al. [19]. It uses the calibration method developed by Caprile and Torre [3] using the properties of vanishing points to calibrate a camera. Lv et al. [19] uses a walking human to locate the vanishing points, we however manually pinpoint straight parallel lines for each of the three vanishing points.

cal-i-bra-tion

A set of gradations that show positions or values.

The American Heritage® Dictionary of the English Language: Fourth Edition. 2000.

2.1 Camera calibration for off-the-shelf cameras

Off-the-shelf cameras are cameras which are low-cost and widely available. Each camera has differing camera parameters and therefore each requires calibration. Although the parameters might be identical for cameras of the same model, one cannot rely on this as the manufacturing process may not be truly identical. Also, during usage the parameters might change, for instance when zooming in or using different lenses. Hence, a method to automatically or at least easily calibrate a camera is needed.

Our test sequence has been taken by a stationary off-the-shelf camcorder with a CCD imaging plane. Hence, calibration techniques requiring camera motion are not applicable. As we have only access to the images and not the camera itself, calibration techniques based on the one introduced by Tsai [29] are also not suitable. We therefore use the method developed by Lv et al. [19], which uses video of a walking human to calibrate a camera. Their method assumes the camera has square pixels and zero skew, which is most common for a CCD camcorder. They also assume no lens distortion and the walking human's height is known. By detecting a human at different locations in the scene the horizon line and the vertical vanishing point are estimated. This requires detection of the same individual in different frames which is a difficult task, especially in complex dynamic scenes. We therefore manually select straight parallel lines and use these to estimate the horizon line and the vertical vanishing point.

2.2 Modeling cameras

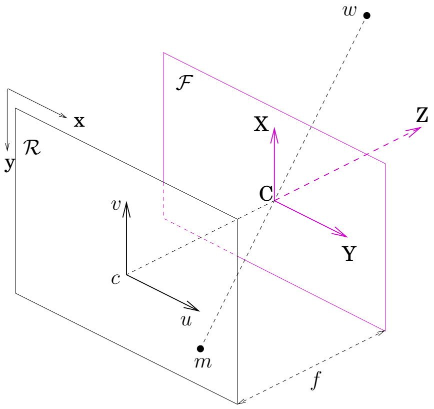


Figure 2.1: *Pinhole camera model*

In each camera, a 3-D point in world coordinates is projected onto a 2-D point in camera coordinates. One possible model for a camera is illustrated in figure 2.1, where $\mathcal{F}$ can be seen as the lens and $\mathcal{R}$ as the light sensitive plane of the camera. A point w in world coordinates is projected onto the *retinal plane* ($\mathcal{R}$) by a straight line through the *optical center* (C) intersecting $\mathcal{R}$ at m. This simple model is called the *pinhole camera model*. According to Faugeras [8], the pinhole camera model can accurately model the geometry and optics of most Vidicon, CCD and CID cameras. Thus we assume that it is a suitable model to estimate the camera parameters of the camera used in the test sequence.

The camera model can be conveniently written in matrix form, describing the projective relation between world and camera coordinates (for details see appendix C). In the rest of this chapter we will refer to this matrix as $\mathbf{P}$, which is a 4×3 matrix with 11 unknown parameters. These parameters are divided into *intrinsic* and *extrinsic* parameters.

Intrinsic and Extrinsic parameters

The *intrinsic* parameters define the perspective projection in camera coordinates (i. e. u and v axis in figure 2.1). The size and shape of the light sensitive cells on the CCD chip (i. e. the *image plane* or *retinal plane*) in the camera may vary. Therefore, the units on the u and v axis may differ. This is referred to as the *aspect ratio* and is needed to map the camera coordinates onto the image coordinates (i. e. x and y axis in figure 2.1).

The *extrinsic* parameters define the relationship between a world (i. e. X, Y and Z axis in figure 2.1) and a camera coordinate system.

There are four intrinsic and six extrinsic parameters:

Intrinsic		**Extrinsic**	
Aspect ratio	λ		α = pan angle
Principal point	u_0 and v_0	Rotation factors	β = tilt angle
focal length	f		γ = yaw angle
skew	s	Translation factors	$\mathbf{t}_x, \mathbf{t}_y$ and $\mathbf{t}_z$

The principal point (u_0, v_0) is where the principal axis (indicated by $< c, Z >$ in figure 2.1) intersects the imaging plane. The pinhole camera model does not compensate for radial distortion effects, which are typically caused by the use of a wide-angle lens. In our case the radial distortion is minimal and can therefore be neglected.

The process of estimating the *intrinsic* and *extrinsic* parameters is called *calibration* and can be thought of as a two-stage process:

1. Estimating the projection matrix **P**

2. Estimating the intrinsic and extrinsic parameters from **P**

The Projection Matrix

P is determined by eleven parameters, however Lv et al. [19] use a simpler model by assuming that the camera has zero skew (i. e. the u and v axis are orthogonal) and unit aspect ratio ($\lambda = 1$), which is most common for CCD cameras [18].

We need three coordinate systems to relate real-world objects to their imaged counterparts; a *World Coordinate System* (WCS), a *Camera Coordinate System* (CCS) and an *Image Coordinate System* (ICS). The WCS and CCS are both 3-D coordinate systems, the ICS is a 2-D coordinate system. The CCS is taken as shown in figure 2.1 by the u, v and Z axis. In figure 2.1 the WCS is shown with its origin at the optical center. In our case the WCS is located on the ground plane, with its origin vertically below the CCS origin (i. e. the X axis intersects c). Therefore the distance between the CCS and WCS is equal to the camera height (H_c). The ICS is related to the aspect ratio and the CCS. We take an aspect ratio of 1 (i. e. square pixels), therefore the camera coordinates map one on one on the image coordinates.

Taken from Lv et al. [19], the rotation matrix (**R**), translation vector (**t**) and **P** are as follows

$$
\begin{aligned}
\mathbf{P} &= \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0_3^\top & 1 \end{bmatrix} \\
\mathbf{R} &= \begin{pmatrix} \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & -\cos\beta\sin\gamma & -\sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma \\ \cos\alpha\sin\gamma - \sin\alpha\sin\beta\cos\gamma & \cos\beta\cos\gamma & -\sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma \\ \sin\alpha\cos\beta & \sin\beta & \cos\alpha\cos\beta \end{pmatrix} \\
\mathbf{t} &= (t_1, t_2, t_3)^\top = \mathbf{R}(0, H_c, 0)^\top
\end{aligned}
\tag{2.1}
$$

where β is the tilt angle, α is the pan angle and γ is the yaw angle of the camera.

2.3 Using vanishing points for camera calibration

Caprile and Torre [3] have shown that using three vanishing points for orthogonal directions allows for partial calibration of a camera from a single view.

Vanishing points

Vanishing points are defined as the image points at which the projection of parallel lines intersect (see figure 2.2). The following is taken from Caprile and Torre [3], which states a few basic properties of vanishing points.

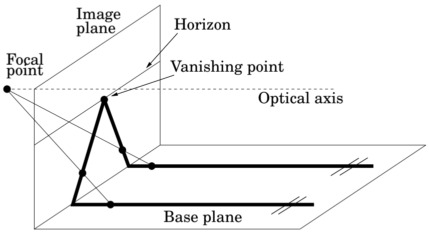


Figure 2.2: *Perspective projection of parallel lines*

Property 1 *Let $A \equiv S_i$ be a set of straight lines (not parallel to the image plane ($\mathcal{R}$)[1]). The set $\alpha \equiv v_{S_i}$ of the vanishing points associated with the elements of A lies on a straight line if and only if the straight lines in A are parallel to the same plane $\prod_A$.*

Property 2 *Let Q, R and S be three mutually orthogonal straight lines in space and $v_Q \equiv (x_Q, y_Q, f)$, $v_S \equiv (x_S, y_S, f)$, $v_R \equiv (x_R, y_R, f)$ the three projected vanishing points associated with them. If we know the coordinates of one of these points, say v_Q, and the direction of a straight line on the image plane which passes through a second one, say v_R, we can compute the coordinates of both v_R and v_S.*

Property 3 *Let Q, R, S be three mutually orthogonal straight lines in space (i. e. real world), and let $v_Q \equiv (x_Q, y_Q, f)$, $v_S \equiv (x_S, y_S, f)$, $v_R \equiv (x_R, y_R, f)$ be the three vanishing points associated with them. The orthocenter of the triangle with vertices in the three vanishing points is the intersection of the optical axis and the image plane.*

[1] The projected counterparts of parallel straight lines in the real-world that are parallel to the image plane, are also parallel in the image

From Lv et al. [19] we add two additional properties

Property 4 *Images of a family of parallel lines pass through a common point in the image plane called their vanishing point. The image coordinates of the vanishing point can be derived as:*

$$\left(f\frac{r_{11}a + r_{12}b + r_{13}c}{r_{31}a + r_{32}b + r_{33}c} + u_0, f\frac{r_{21}a + r_{22}b + r_{23}c}{r_{31}a + r_{32}b + r_{33}c} + v_0 \right)$$

Where (a, b, c) is the direction vector of the parallel lines and r_{nm} are the elements of the rotation matrix from equation 2.1. If we set the world coordinate system as the direction vectors of three families of parallel lines mutually orthogonal to each other, the vanishing points(V_x, V_y, V_z) can be obtained by setting (a, b, c) to be $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$ respectively,

$$\begin{array}{ll} (f\frac{r_{11}}{r_{31}} + u_0, & f\frac{r_{21}}{r_{31}} + v_0) \\ (f\frac{r_{12}}{r_{32}} + u_0, & f\frac{r_{22}}{r_{32}} + v_0) \\ (f\frac{r_{13}}{r_{33}} + u_0, & f\frac{r_{23}}{r_{33}} + v_0) \end{array} \tag{2.2}$$

Property 5 *The angle from a horizontal line in the image to the horizon line is equal to the yaw angle γ.*

These properties together with some assumptions about the camera used, allow us to estimate the camera parameters.

Estimating the camera parameters

Using the matrix defined in 2.1 and the properties of the vanishing points, Lv et al. [19] estimates the camera parameters in 4 steps:

Step 1 *By taking three orthogonal families of parallel lines in the image V_x, V_y and V_z) (i. e. the three vanishing points associated with three mutually orthogonal straight lines) can be determined by computing the intersection of parallel lines in the family. It is taken that V_x and V_z are on the horizon line and V_y is the vertical vanishing point.*

Step 2 *Using Property 3 the principal point (u_0, v_0) can be computed.*

Step 3 *Yaw angle γ is computed using Property 5.*

Step 4 *If $\gamma = 0$ then from equations 2.2 and 2.1 the location of V_x and V_y in image coordinates are*

$$u_{V_x} = f \cdot \cot(\alpha)/\cos(\beta) + u_0 \tag{2.3}$$

$$v_{V_x} = -f \cdot \tan(\beta) + v_0 \tag{2.4}$$

$$u_{V_y} = u_0 \tag{2.5}$$

$$v_{V_y} = f \cdot \cot(\beta) + v_0 \tag{2.6}$$

$$u_{V_x} = -f \cdot \tan(\alpha)/\cos(\beta) + u_0 \tag{2.7}$$

$$v_{V_x} = -f \cdot \tan(\beta) + v_0 \tag{2.8}$$

From equations 2.4 and 2.6 f can be derived

$$f = \sqrt{-(v_{V_x} - v_0)(v_{V_y} - v_0)} \tag{2.9}$$

by using equation 2.4 again, β is derived as

$$\beta = \tan^{-1}(\frac{v_0 - v_{V_x}}{f}) \tag{2.10}$$

and by using equation 2.3 and the previous two results, α can be obtained

$$\alpha = \cot^{-1}(\frac{(u_{V_x} - u_0)\cos(\beta)}{f}) \tag{2.11}$$

To calculate the camera height H_c the cross ratio invariance from Criminisi et al. [5] is used

$$\frac{H}{H_c} = 1 - \frac{d(t,i)d(b,v)}{d(b,i)d(t,i)} \tag{2.12}$$

If H, the height of the reference object is known (indicated in figure 2.4 by $< b, t >$), the height of the camera (H_c) can be calculated using equation 2.12.

If $\gamma \neq 0$, the image needs to be rotated around the image origin by γ, such that the horizon is aligned with the horizontal line $< Vx, Vy >$. Now $\gamma = 0$ and the above steps can be applied. Suppose the result is $(f', u_0', v_0', \alpha', \beta', \gamma' = 0, H_c')$, than the following are the parameters for the original image: $(f = f', u_0 = u_0' \cos(\gamma) - v_0' \sin(\gamma), v_0 = u_0' \sin(\gamma) + v_0' \cos(\gamma), \alpha = \alpha', \beta = \beta', \gamma, H_c = H_c')$.

The above algorithm gives the height of the camera in the unit chosen for the reference object, the pan, tilt and yaw angle in radians and the focal length in pixels.

2.4 Computing the calibration parameters

Estimating the location of the vanishing points requires detecting straight lines. These can be detected by using Hough transforms [2], edge detection techniques [20] or pinpointed by a user [4]. As automatic detection of straight lines is not very robust we select the straight lines manually.

We select pairs of points belonging to parallel straight lines in one of the three *families*[2] of parallel lines (V_x, V_y or V_z). From these points the corresponding line equations are calculated. These are used to calculate all pairwise intersections of the lines belonging to the same family. The location of the vanishing point is then estimated by taking a weighted average of all pairwise intersections in the corresponding family.

Estimating a vanishing point

As the straight lines are estimated manually and are taken from a discrete image the estimate is bound to contain some error. Considering an identical error for different lines, lines parallel propagate the error more than lines that are perpendicular or less parallel. To limit the error propagation Caprile and

[2] When we speak of a family of lines or lines in a family we refer to the set of parallel lines defining the location of one of the three vanishing points.

Torre [3] uses the cosine of the angle between two intersecting lines, to give more weight to lines that are less parallel

$$\bar{x} = \sum_{i=1}^{n(n-1)/2} w_i x_i \tag{2.13}$$

where n is the number of lines in a family and

$$w_i = \frac{\cos\theta_i}{\sum_{i=1}^{n(n-1)/2} \cos\theta_i}$$
$$\sum_{i=1}^{n(n-1)/2} w_i = 1$$

where θ_i is the angle between the two lines intersecting at x_i.

As many parallel lines are used some might result in erroneous intersections and should not be considered when determining the vanishing point. To eliminate these points the standard deviation of all intersections is used

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{2.14}$$

where $\bar{x}$ is the weighted mean from equation 2.13, x_i is the intersection point i and N is the total number of intersections. As we only wish to eliminate intersection points that are obviously wrong we only discard points whose distance to the mean is greater than 2σ. Hence we have a confidence interval of about 95%, meaning we expect 95% of all intersection points to be correct.

When intersections are discarded the value of $\bar{x}$ changes and thus σ changes. Hence the procedure is repeated and the amount of intersections reduced until all fall within the 2σ range.

Calculating the principal point

When the three vanishing points have been estimated we can use Property 3 to calculate the principal point as illustrated in figure 2.3. The three vanishing points V_x, V_y and V_z form a triangle with edges $< V_x, V_y >, < V_x, V_z >$ and $< V_y, V_z >$. For two of the edges the perpendicular lines are calculated which intersect the vanishing point not part of the edge. The intersection of these two lines is the location of (u_0, v_0) in image coordinates.

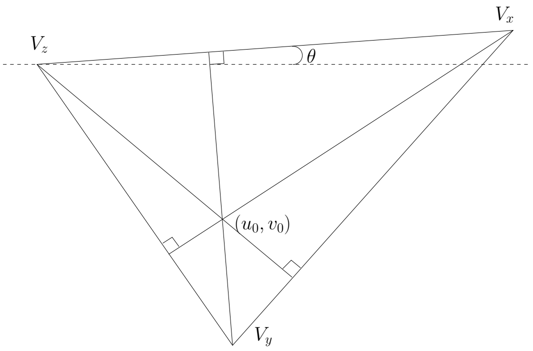


Figure 2.3: *Determining the principle point*

Calculating the camera height

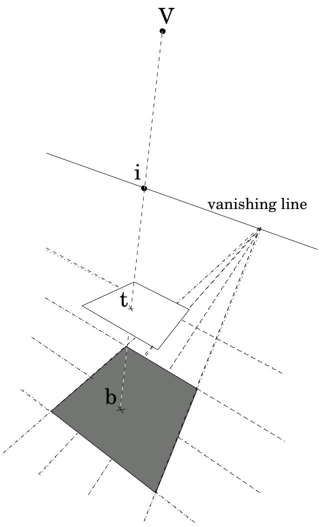


Figure 2.4: *Cross-ratio*

In figure 2.4 the method to calculate the camera height is illustrated. Through the vanishing line goes a pencil of planes all parallel to each other of which two are shown. One parallel plane goes through the camera center and is visible as the vanishing line. These planes together with the vertical vanishing point define a *cross ratio* which determines a ratio of distances between planes in the world [5]. Using the cross ratio and a reference object with known length, the lengths of objects parallel to the reference object can be calculated. Therefore, as the height of the camera is measured along a line perpendicular to the ground plane, we can calculate its height using a vertical standing reference object (see equation 2.12).

The camera height is interactively estimated by estimating a person's height and calculating the camera parameters. First a suitable person should be located in the image and his or her principal axis determined. **b** and **t** (as in figure 2.4) are respectively his or her feet and head located on the principal axis. Taking an initial guess of the person's length the camera height is calculated. Afterwards the camera parameters are used to calculate the person's height, which we will refer to as the person's ”calibrated” height. If the estimated height and the calibrated height differ too much, the person's length should be re-estimated and the process repeated. When the difference between the estimated and calibrated height is small enough, the process can be stopped.

2.5 Robustness of the method

In the previous section we have described how we reduce the error by considering the angle of the straight lines and their distance to the mean. Here we introduce a measure called the condition number which states under which conditions best robustness to noise is achieved.

Besides acknowledging certain conditions as being good or bad, prior knowledge of certain camera parameters can be used to constrain the parameters. This particularly reduces the error propagation as a result of a poorly constraint vanishing point estimate.

Condition number

As shown by Caprile and Torre [3], the condition number c_l for two straight lines and $c_{\mathbf{R}}$ for the rotation matrix **R** is

$$c_l = \frac{1 + |\cos(\theta)|}{1 - |\cos(\theta)|} \tag{2.15}$$

where θ is the angle between the two straight lines.

The rotation matrix **R** is estimated by using the three unit vectors associated with the three vanishing points

$$\mathbf{V'} = \mathbf{RV} \tag{2.16}$$

where $\mathbf{V}$ is a 3×3 matrix whose columns are the three unit vectors. Rewriting equation 2.16 we get

$$\mathbf{R} = \mathbf{V'V}^{-1} \tag{2.17}$$

and

$$c_{\mathbf{R}} = \frac{1 + \sqrt{\cos^2\theta_{lm} + \cos^2\theta_{ln} + \cos^2\theta_{mn}}}{1 - \sqrt{\cos^2\theta_{lm} + \cos^2\theta_{ln} + \cos^2\theta_{mn}}} \tag{2.18}$$

where $\mathbf{l}, \mathbf{m}$ and $\mathbf{n}$ are the columns of $\mathbf{V}$ and $\cos\theta_{lm} = \mathbf{l} \cdot \mathbf{m}, \cos\theta_{ln} = \mathbf{l} \cdot \mathbf{n}$ and $\cos\theta_{mn} = \mathbf{m} \cdot \mathbf{n}$.

In order to have best robustness against noise the condition number must be close to 1, therefore θ should be close to $90°$. Thus the best case, in which three families of parallel lines are chosen, is when three orthogonal planes (in the real world) each makes an angle of $45°$ with the image plane. This is similar to placing a cube with one corner pointing at the camera where each face makes an angle of $45°$ with the image plane.

The translation vector is estimated using triangulation (see equation 2.12 and 2.1). Therefore its accuracy depends on the exact measure of the length of the reference object and the localization of its endpoints in the image plane. According to Caprile and Torre [3], the relative error could be less than 1%, depending on the image resolution and object measurements. However as we do not use a nicely positioned calibration object (e. g. a cube) like Caprile and Torre [3] uses, we expect our estimate to be less accurate.

Constraining the principal point

Vanishing points imaged at infinity (i. e. parallel lines in the real world also parallel in the image) may degenerate the estimate of the camera parameters significantly. Using prior knowledge about the location of the principal point its location in the image can be constrained. Most CCD cameras have the principal point near the image center. Lv et al. [19] uses this to calibrate a camera when only the vertical vanishing point and horizon line locations are known. Their method has the principal point at half the image height on the line perpendicular to the horizon line going through the vertical vanishing point.

Generally, computers tend to be better than humans at classification because they are not distracted by random variations in noncritical parameters.

John C. Russ

3 The Chamfer System

Detecting people in complex dynamic scenes is a difficult task. People are non-rigid objects and may show a variety of shapes and textures. This makes it hard to create a general model for a human being, especially when considering that people might be partly occluded. In complex dynamic scenarios, background modeling techniques are likely to fail as the background changes in a complex and difficult to predict manner. Therefore, many people detection systems use shape-features as these are generally the same for different people and visible in low resolution imagery.

In this chapter we describe a method called the Chamfer System developed by Gavrila and Philomin [12] to detect the location of people in imagery using shape-features. It uses a shape model hierarchy representing the variety of people's shapes. Matching is done using a Distance Transform (DT) which allows for a coarse-to-fine search. The system has been successfully used to detect pedestrians and traffic signs [11].

cham-fer

A flat surface made by cutting off the edge or corner of a block of wood or other material.

The American Heritage® Dictionary of the English Language: Fourth Edition. 2000.

3.1 Detecting people in complex dynamic scenes

An important step in any system trying to efficiently search for objects in imagery, is reducing the search space to only those parts of the image an object of interest is likely to occur. This is usually accomplished by segmentating foreground from background regions. A widely used segmentation method is background subtraction, where a model of the background is compared with an input image. Image parts not part of the background model are considered foreground and usually further analyzed.

We are interested in dealing with complex and dynamic surveillance scenarios (i. e. multiple moving objects possibly partially occluding each other and a non-stationary background). When considering complex dynamic scenes the background model should be kept up to date to incorporate any changes to the background, e. g. changing lighting conditions or objects removed from or added to the background. Keeping the background mode up-to-date, the segmentation depends on moving objects. Whenever objects are stationary for some time, these will be added to the background model. In a camera surveillance scenario people may be standing still or hardly move, hence people detection systems based on background modeling are not well suited for detecting people in complex dynamic scenes.

To cope with complex dynamic scenes, we use a method developed by Gavrila and Philomin [12], called the *Chamfer System*. It uses shape features (i. e. edges) to detect objects of interest. This has the advantage that it does not require background modeling, as the segmentation of irrelevant and relevant image parts is done by transforming the input image to an edge image. A coarse to fine search is achieved by transforming the edge image to a distance image, where the pixels intensity represent the distance to the nearest edge. To cope with the large variety in people's shape, the method exploits the similarities between shape models to reduce the number of models to be processed during matching.

3.2 Efficient shape matching

Matching using a Distance Transform

Simply correlating the shape models with the edge image requires the models to perfectly resemble the object of interest. As humans come in a large variety of forms (e. g. different postures and clothing) this requires an excessive amount of models to robustly detect humans in images. By transforming the edge image into a Distance Transform (DT) image, where the pixel values represent the distance to the nearest edge, the similarity measure between a shape model and an object of interest in the input image becomes more smooth. This means that the matching procedure can be more tolerant towards dissimilarities between a shape model and the shape of a person in the input image.

A matching scheme using a DT, allows for a coarse to fine search using different levels of tolerance and step sizes. At high tolerance levels the image is more or less skimmed using a large step size. Image locations where the correlation value is below the tolerance threshold are further analyzed using a smaller step size and lower tolerance level.

Shape model hierarchy

The chamfer distance [1] can also be used as a measure of similarity between the shape models. Grouping shape models based on their similarity and choosing one or more shape models to represent the mean of the group, one can build a *hierarchy of shape models*.

In a shape model hierarchy, each sub tree is represented by one or more *prototypes*. During matching, the hierarchy is traversed from top to bottom. When a prototype matches, the shape models in the underlying sub tree are also processed. When a prototype does not match the underlying shape models do not have to be considered. This has the advantage that when the distribution of shape models show some structure, the amount of shape models to be processed at an image location during matching can be significantly reduced. Resulting in a speedup gain during matching.

Ground plane constraint

Using a coarse to fine search and limiting the amount of shape models to be processed during matching, can improve the detection speed of the system significantly. Reducing the search space by using prior knowledge of where humans might occur, can improve the detection speed even more.

Due to gravitational forces, people are most likely to be standing on the ground. By calculating the location of the *ground plane* (e. g. pavement), the expected person's height at a certain distance can be used to limit the search space for each size of shape models. This has the advantage that detection speed is improved and that the amount of possible false positives is reduced.

3.3 Creating a Distance Transform image

The process of creating a *DT image* is shown in figure 3.1. The first step is creating an *edge image*, where object edges are highlighted and others are suppressed. Retrieving object edges is done by looking at the contrast differences in the image using a *Sobel* filter. The Sobel filter consists of two kernels which detect horizontal and vertical intensity changes in an image. If both are applied to an image, the results can by used to compute the magnitude and direction of the edges in the image (see appendix B). The magnitude image is converted into an edge image by applying a threshold. This edge image cleaned where edges containing less than a specified amount of pixels are discarded.

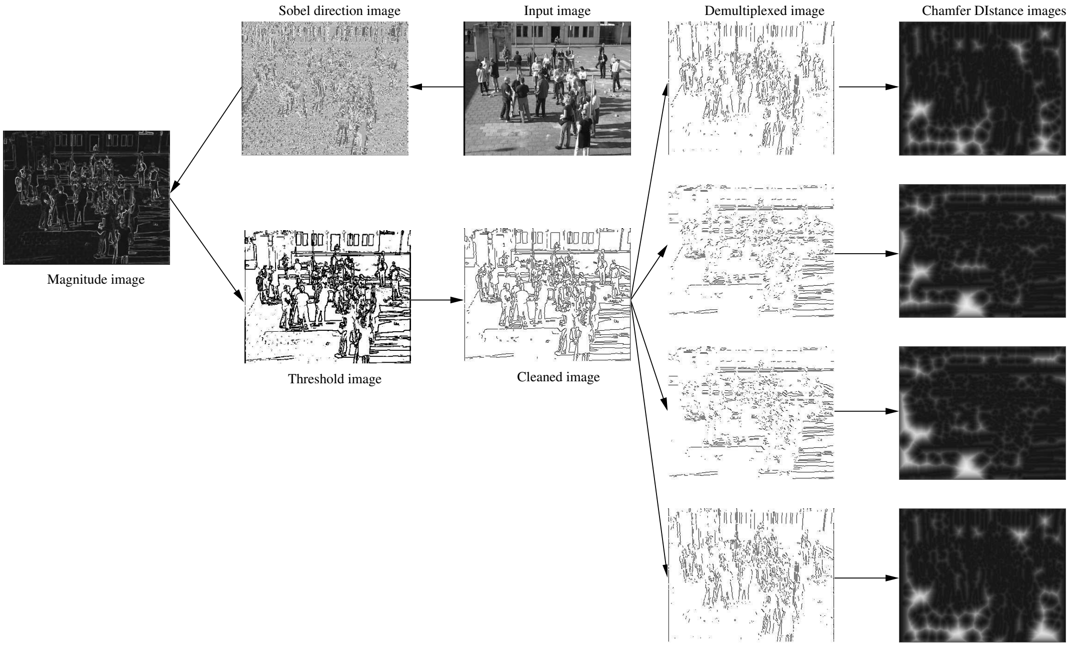


Figure 3.1: *Creating a Distance Transform (DT) image*

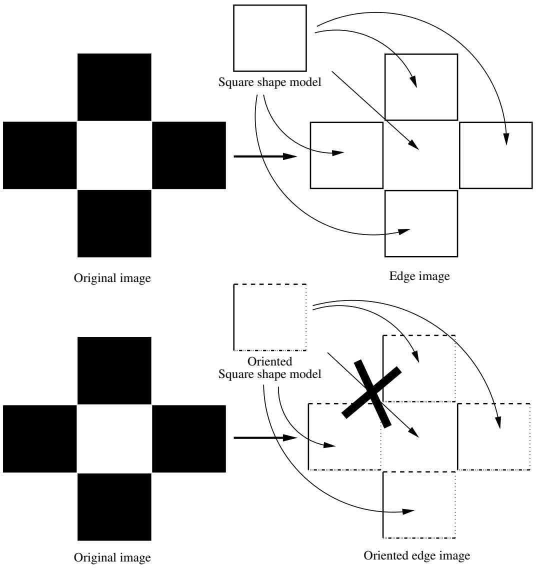


Figure 3.2: *Example which benefits from using feature typed matching*

The edge image is divided into four separate images using the *Sobel* direction as divider. These images are referred to as edge oriented images, which are either up, bottom, left or right oriented. The shape models are also divided into four separate shape models, for the same orientations as the edge images. This has the advantage that if a model contains more than one orientation, these cannot match on different oriented edges in the image.

To illustrate the advantage of using edge orientations, consider the scenario of detecting squares as shown in figure 3.2 using a square shape model. The arrows point to locations where the shape model matches. When we do not consider edge orientations (top example in figure 3.2), the shape model also matches at the center of the four squares. When matching with edge oriented images (bottom example in figure 3.2), each edge of the squares will only be matched with the same oriented shape model's edge. Thus only the four squares will match and not the square in the center formed by the four squares.

For each of the four edge images a DT image is created using the *chamfer-2-3* metric. The chamfer-2-3 metric [1] approximates a Euclidean pixel distance of 1 and $\sqrt{2}$ respectively. This is accomplished by a convolution using a 3×3 mask which is split in two, as shown in figure 3.3. One part is applied from top to bottom and left to right and the second part from bottom to top and right to left. The advantage of splitting the mask in two is that both can be applied in parallel (see appendix B).

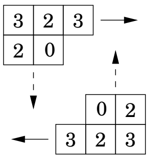


Figure 3.3: *Chamfer-2-3 metric masks*

3.4 Building a shape model hierarchy

In figure 3.4 an example of a shape model hierarchy is shown. Building the hierarchy is done automatically offline using a "K-means"-like clustering technique. Starting with an initial (random) partition at leaf level, shape models are moved between groups while the following function E is minimized

$$E = \sum_{k=1}^{K} \max_{s_i \in S_k} D(s_i, p_k^*) \tag{3.1}$$

Here, $D(s_i, p_k^*)$ is the *Chamfer* distance between the i-th element s_i of group k and the prototype p_k^* for that group at the current iteration. A *prototype* is selected by taking the smallest maximum distance to the other shape models. The amount of shape models at each level is specified by the user. To minimize E, *simulated annealing* is used.

By recursively applying this grouping one obtains a hierarchy, where at each *prototype* level (i. e. not at leaf level) a set of prototypes represents the mean of the shape models at lower levels. As only leaves are given as a matching result, all prototypes must also occur in the underlying sets. Hence at the lowest level all shape models occur.

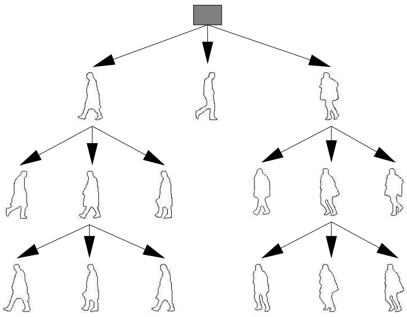


Figure 3.4: *Shape model hierarchy*

To represent the objects at different sizes the models are scaled. For each size-set of shape models a hierarchy is build which are combined afterwards through a dummy root node. This root node does not contain a prototype but is simply needed to connect the different size-sets.

3.5 Matching using a shape model hierarchy

Matching is done by traversing the hierarchy, correlating a prototype with the DT image. A prototype matches when its distance at the location is below a user supplied threshold. At high levels of the hierarchy a large step-size is used to more or less skim the image for ROIs. At this point the threshold for determining a match or not can be set relatively high. If a prototype matches, the process is repeated for the matched location with the underlying shape models (children nodes) and a smaller step-size. When leaf level is reached all shape models that fall below the threshold are taken as positive matches.

Traversing the tree can be done in a breadth-first or depth-first manner. Here a depth-first traversal is used, as this has the advantage that the process only needs to keep a set of $L - 1$ ROIs, where L is the depth of the hierarchy.

As matching is done by correlating a shape model T over the DT image, the matching measure $D(T, I)$ is determined by the pixel values of the DT image that lie under the ”on” pixels of the shape model (i. e. feature points). As distance measure the average chamfer distance is used

$$D_{chamfer}(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t) \tag{3.2}$$

where $|T|$ denotes the number of features in template T and $d_I(t)$ denotes the chamfer distance between pixel t in T and the closest edge pixel in image I.

The brain can only acquire knowledge about the invariant properties of the objects and surfaces if it is able to discard the continually changing information reaching it from the visual environment.

Semir Zeki

4 Texture Classification

In this chapter we describe the method used to identify a person's main facing direction and reduce the number of false positives we get from using a detection method like the Chamfer System.

The method is based on the work done by Wöhler and Anlauf [33], who obtained good results detecting the gait pattern of a walking human. We describe how local processing solves the issue of using a limited set of training patterns and computational resources. We explain the architecture of the neural network (NN) and how multiple NNs can be used to identify a person's main facing direction.

tex-ture

The distinctive physical composition or structure of something, especially with respect to the size, shape, and arrangement of its parts

The American Heritage® Dictionary of the English Language: Fourth Edition. 2000.

4.1 Neural Network using local receptive fields

Texture classification is often used in the domain of image analysis to classify images, or parts of it, as either containing an object of interest or not. Instead of using a single object feature, for instance shape as used by the Chamfer System, statistical methods can be used to discriminate between multiple groups of image features. For instance, areas of pixel intensities which define a certain structure (e. g. a brick wall). In gray scale images the image features are usually a combination of areas of identical pixel intensities and edges. This usually allows for more complex combinations of image features defining the characteristic properties of the object of interest. For the purpose of determining a proper discriminating function, neural networks are well suited for the task as these can approximate any smooth function.

Determining a discriminating function using a Neural Network (NN) to classify high dimensional input patterns, requires a large amount of examples to successfully learn the appropriate weight configurations. According to Wöhler et al. [34] neural network architectures like the fully connected[1] multi layer perceptron (MLP) do not seem to be suitable to classify high dimensional input patterns, as these involve a number of weights parameters that typically exceeds the number of training samples. This will lead to long training cycles and overtraining effects[2].

To limit the amount of training time and overtraining effects the amount of weight parameters must be brought to a minimum. However taking too few weight parameters results in a network unable to create a general model of the input patterns. Searching for the optimal number of weights is often referred to as dimensionality reduction of the input patterns. This can be seen as discarding the irrelevant and retaining the relevant properties of the input patterns for the task at hand.

Considering the limited processing time and amount of samples (i.e. 2633 images of pedestrians and 11651 images of scenery) available, dimensionality reduction is a necessity.

Principal Component Analysis

A classical approach to dimensionality reduction is the well-known *Principal Component Analysis* (PCA)[3]. An advantage of the PCA algorithm is that it does not require prior grouping of the different patterns one wishes to classify. However because it takes no account for groups within the data it may obscure the existence of separate groups [31]. Another disadvantage is that the processing

[1] Fully connected means that each neuron in one layer is connected to each other neuron in a higher or lower layer of the network

[2] An overtrained neural network learns a small data set by heart. Thereby learning the noise in the data instead of generalizing the functional relationship of the data.

[3] PCA can be thought of as rotating the axes of the original coordinate system to a new set of orthogonal axes that minimize the amount of variation of the original data.

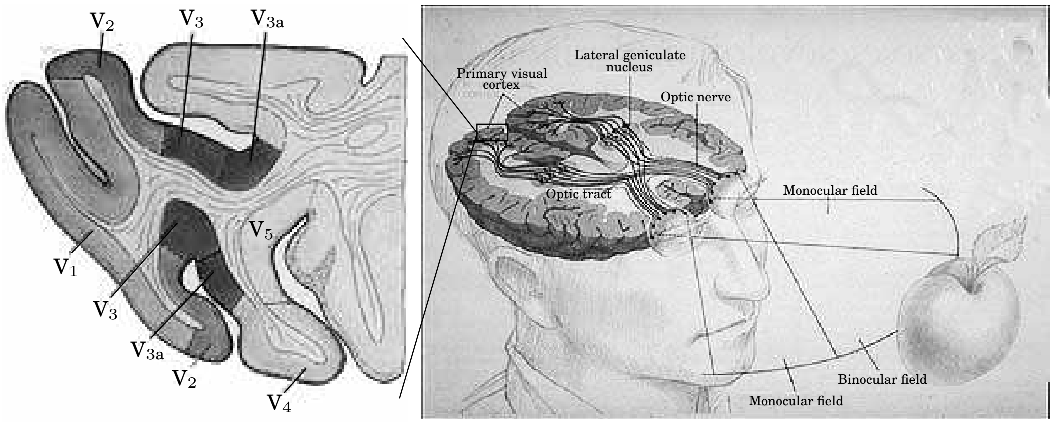


Figure 4.1: *The visual cortex in a human's brain*

time needed for global dimensionality reduction might be too costly with respect to real-time constraints [32].

Receptive fields

Instead of using a global dimensionality reduction method Wöhler et al. [35] use local processing, which instead of connecting each neuron in one layer to every other neuron in a higher or lower layer it connects to only a subset of the available neurons in a higher or lower layer.

This concept is based on how the visual system of organisms like ourselves works. David H. Hubel and Torsten N. Wiesel were the first to discover that cells in the visual cortex of cats are arranged in a precise and orderly manner [37].

The visual cortex (a layer of brain tissue receiving and processing impulses from the optic nerves, see figure 4.1) is the first area within the cerebral cortex[4] that processes neural signals from the eye. It performs the basic tasks of recognizing the geometric features of a scene before relaying that information to higher brain regions. In these higher brain regions the basic visual data is transformed into the conscious perception of the visual world [37]. The entire area an eye can see is called its visual field. The part of the visual field a single neuron monitors is called its *receptive field* [9, 7, 37].

Although it is not completely clear what the function of each area in the visual cortex is, experiments on monkeys [37], cats and flies [28] have shown that with differing receptive field sizes comes specific functionality specialization. Thomas L. Adelman and Olberg [28] found that the information content found in the receptive fields of the target-selective descending neurons of dragonflies, which are relatively large, contains more information about target velocity than about position. According to Zeki [37] cells with large receptive fields are not able to signal the position of stimuli in the world at the same precision that cells with small receptive fields can. His findings are based on tests on monkeys, where the topography of the brain shows an increase in receptive field sizes in the visual cortex the more further away these are from the retina.

[4] This is the part of the brain largely responsible for higher brain functions, both sensory and motoric (i.e. muscle movement, hearing, vision etc.)

4.2 Network architecture

An example structure of the network we used is shown in figure 4.2 (more details are given in appendix A). It is based on a *Time Delay Neural Network* (TDNN) as described by Wöhler and Anlauf [33]. They use it to detect the gait pattern of human beings. Although the time delay concept improves the overall detection scheme we are primarily interested in spatial features and therefore neglect the temporal aspect of the TDNN.

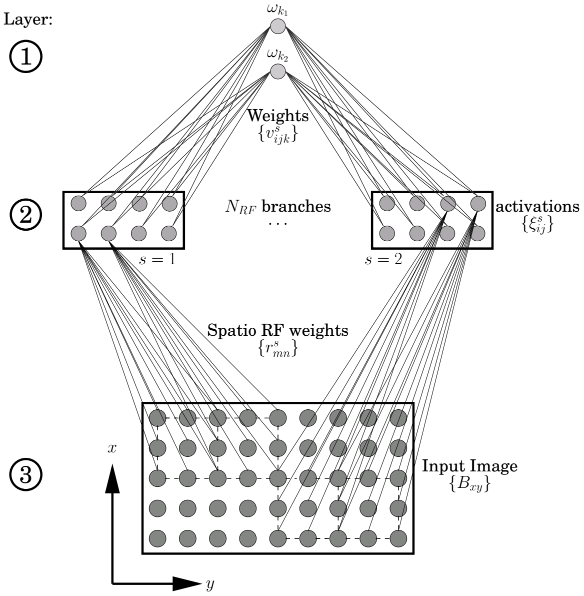


Figure 4.2: *Architecture of a shared weight neural network with local receptive fields(from Wöhler et al. [35])*

The second layer of the network contains one or more branches. Each branch has a set of neurons, which have a sigmoidal activation function and are connected with the input pattern through their receptive field. In each branch the neurons use a common set of weights for the receptive fields, which is commonly referred to as the *shared weights* principle.

Wöhler and Anlauf [32] found that using the concept of shared weights still allowed for extractions of a wide variety of features. This reduces memory demands and the amount of samples needed to train the classifier. Taking the network as shown in figure 4.2 as an example, we need to maintain and learn $2 \times 3 \times 3 = 18$ weights instead of 45 as is the case when using a fully connected MLP network.

During training the receptive field weights are updated to detect the features being regarded instead of being imposed a priori. Wöhler et al. [35] used PCA and an online gradient descent method[5] to adapt the weights of the receptive fields. They found that an online gradient descent method results in a network with a slightly higher recognition rate.

In each branch a different filtered version of the input pattern is formed in layer two. The filtered versions can be seen as a generalization of the classes of input patterns being regarded. Therefore the optimal amount of branches is dependent on the amount of classes of input patterns one wishes to discriminate. For example when one wants to classify three different types of input patterns, one needs at least three different branches to model each class.

One might consider using more than three layers and imposing the concept of receptive fields in higher layers. However considering the low resolution of the input patterns and the amount of branches in layer 2 (i. e. at least two), the amount of neurons in layer 2 will be relatively small. Thus further imposing the concept of receptive fields in higher layers will probably not improve the network performance. Therefore, a three layer network will be sufficient and the output neurons in layer 1 are fully connected to the neurons in layer 2.

[5] An online gradient descent method adjusts the weight parameters with a small amount during training in the direction that reduces the error the most.

Receptive field sizes

As found by Zeki [37] and Thomas L. Adelman and Olberg [28] the best configuration of a receptive field depends on its given task. Wöhler and Anlauf [32] found that using a receptive field of size 9×9 pixels with a distance of 5 pixels between adjacent receptive fields centers gives the best result to detect the gait pattern of walking humans. Estimating a person's BFD requires sensitivity to spatial features (e. g. object width or location) which probably results in smaller receptive field sizes and spatial distances between adjacent receptive fields centers.

Generalization and robustness of the network

Wöhler and Anlauf [33] tested the robustness and generalization of a network using local receptive fields by considering random variations of an object's position and shape in the input pattern. They found that the network is able to adapt and that performance barely decreases using a test set containing object position variations up to the same amount as in the training set. They also found that considering strongly variating object positions the best performing network configuration is that with small receptive field sizes. Training the network using two distinctive shapes and test it using intermediate shapes they found that the activation of a output neuron shows a clear interpolation behavior corresponding to the shape deformation.

4.3 Visualizing a classifier's performance

To test a classifier's performance a separate set of labeled images is used, called the test set. Using a trained classifier on the test set we can determine its performance by calculating the amount of correctly and incorrectly classified test patterns. To visualize the performance of a classifier a Receiver Operating Characteristic (ROC) curve is often used.

A ROC curve provides a good means of visualizing a classifier's performance in order to select a suitable decision threshold [31]. In a ROC curve the True Positive Rate (TPR) is set out against the False Positive Rate (FPR). The TPR represents the percentage of correctly classified images belonging to class 1. The FPR represents the percentage of images classified as class 1, however belonging to class 2.

Two other values TNR and FNR (i. e. True Negative Rate and False Negative Rate, respectively) can be derived from the TPR and the FPR as follows

$$\begin{aligned}\mathrm{TNR} &= 1 - \mathrm{FPR}\\ \mathrm{FNR} &= 1 - \mathrm{TPR}\end{aligned}$$

The terminology *positive* and *negative* has been taken from radiology studies where the ROC curve was introduced to analyze diagnostic tests [24].

In our experiments we talk about $\boldsymbol{direction}_1$ ***versus*** $\boldsymbol{direction}_2$ or $\boldsymbol{direction}_1$ ***versus*** $\boldsymbol{direction}_{2,3,4}$ classifiers, where $\mathbf{direction}_n$ can be left, right, front or back. Thus, positive means $\mathbf{direction}_1$ and negative $\mathbf{direction}_2$. In other words, the TPR is the amount of images classified as $\mathbf{direction}_1$ which are also labeled as $\mathbf{direction}_1$, the FPR is the amount of images classified as $\mathbf{direction}_1$ which are labeled as $\mathbf{direction}_2$.

Note that we performed experiments with both ***1 versus 1*** and ***1 versus Rest*** classifiers. For the ***1 versus Rest*** classifiers the FPR means all other directions than **direction**$_1$ classified as **direction**$_1$.

Creating a ROC curve

We only consider binary classifiers[6] and therefore the NN has two output values: ω_1 and ω_2. An input pattern is classified to class 1 if

$$\frac{\omega_1}{\omega_2} > m \tag{4.1}$$

where m is usually referred to as the decision threshold. By varying m and plotting for each m the corresponding TPR on the y-axis and the FPR on the x-axis we obtain a ROC curve which visualizes the trade off between the TPR and the FPR for different thresholds.

The area under the curve

The *area under the curve* (AUC) can be used as a performance measure for a classifier. The greater this area the better the classifier has been able to separate the two distributions of relevant features. A low FPR and a high TPR indicates that the classifier was able to model class 1 and class 2, respectively, well. This is usually referred to as the goodness of generalization of the input patterns. This gives an indication on how well the classifier will perform when confronted with new input patterns, not seen during training.

4.4 Estimating the Body Facing Direction

We pursue texture classification in our system in two steps:

- Classify an input pattern as containing a person or not
- Determine a person's main body facing direction: **front, back, left** and **right**

The person/non-person classifier is used as a verification step for the regions of interest detected by the Chamfer System. The Chamfer System has been optimized for speed and uses only shape features to detect people in images (see chapter 3). This may result in detecting objects that have a similar shape as people do. Training a classifier to distinguish people from scenery (also referred to as background) relevant features other than shape will also be regarded. Thus the person/non-person classifier complements a detection method like the Chamfer System.

Estimating a person's BFD using NNs can be done by either building a hierarchy of multiple networks or train one network classifying an input pattern in one of the four BFDs. Training a multi-class neural network requires much more training samples to achieve a similar performance as a binary classifier. Thus considering the limited amount of training samples we only consider binary classifiers.

[6] A binary classifier is trained to classify input patterns in two classes, a ternary classifier in three classes, etc. etc.

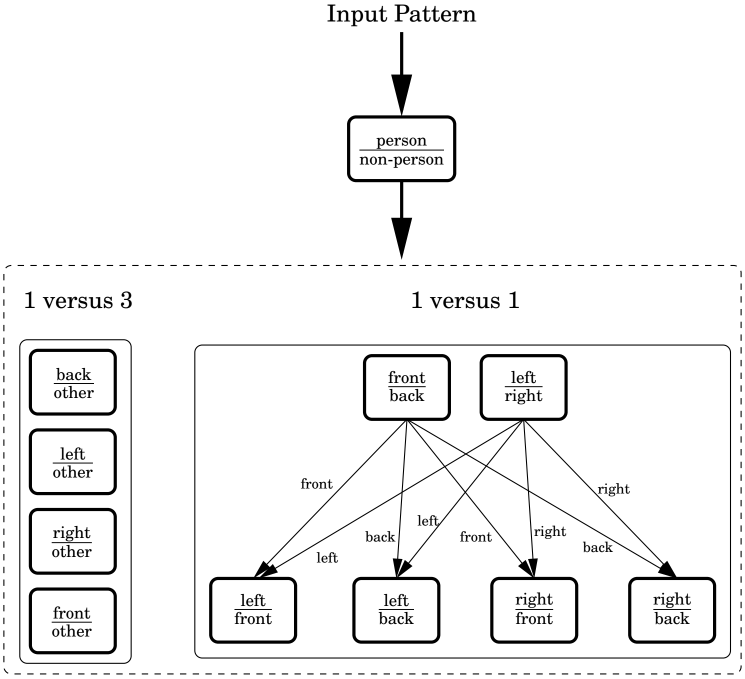


Figure 4.3: *Hierarchy of networks*

We consider two different classification tasks, which are ***1 versus 1*** and ***1 versus All***. ***1 versus 1*** classifiers discriminate between a single BFD and a single other BFD and the ***1 versus Rest*** (also referred to as ***1 versus 3***) classifiers discriminate between a single BFD and all other BFDs

A hierarchy of networks

To estimate the BFD using binary classifiers we need to combine the results of the different classifiers. Figure 4.3 shows two possible hierarchies of classifiers. The thick outlined boxes represent the different classifiers. The amount of *1 versus 1* classifiers needed is six and the amount of *1 versus All* classifiers is four to determine the BFD in four or eight directions respectively.

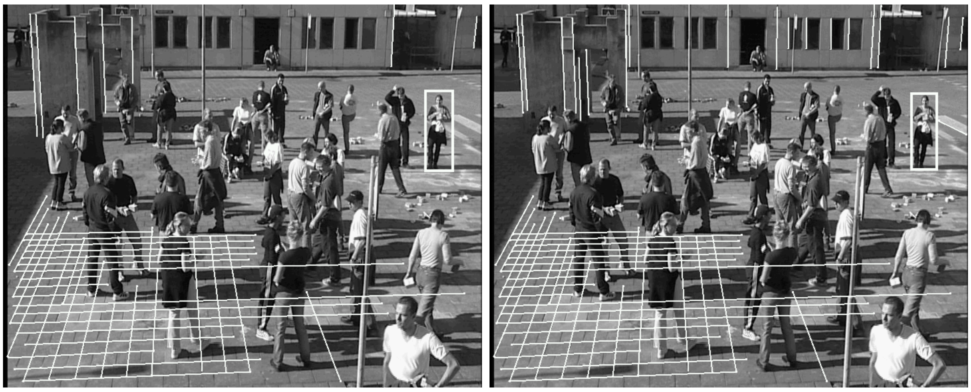
Although we only wish to discriminate between four different facing directions, using *1 versus All* classifiers may result in classifications like **front/left** or **back/right**. A combination like for instance **front/back** leads to a contradiction and can therefore not be considered. This is a disadvantage as this may lead to unclassified image patterns. Using the *1 versus 1* classifiers the input patterns are always classified in one of the four BFDs.

Just because something doesn't do what you planned it to do doesn't mean it's useless.

Thomas A. Edison

5 Experiments

Here we present the experiments done using the techniques described in the previous chapters. First we show the results for the calibration method from chapter 2. Second we describe the experiments performed using the Neural Network as described in chapter 4 to reduce the number of false detected people and estimate a person's facing direction. Last we present some results of the overall system, combining camera calibration, shape detection and texture classification.



(a) *Limited set of lines used for the vertical vanishing point*

(b) *All selected straight lines*

Figure 5.1: *Example of parallel lines used to estimate V_x(right), V_y(vertical) and V_z(left).*

5.1 Camera Calibration

In figure 5.1 one frame taken from the test sequence is shown together with the parallel lines used to estimate the vanishing points. We used the original image size, which is 720×576 pixels, to calibrate the camera. Although we do not have specific information about the camera used, we do know that it was a digital camcorder with a CCD imaging plane. Hence, according to Faugeras [8] we can assume that the pinhole camera model is a suitable model for the camera used for the test sequence.

As already mentioned in chapter 2 the best robustness to noise is when the 3D lines (in world coordinates) have an angle of $45°$ to the imaging plane. When the lines are parallel to the image plane they will also appear parallel in the image. Liebowitz [17] refers to this case as the ideal imaged vanishing point(i. e. imaged at infinity). To compensate for such ”ideal” situations he constrains the location of the principal point near the center of the image. We only constrain the y coordinate as used by Lv et al. [19] (see section 2.5 on page 17). This is a lot easier to implement and, as we will see later on in this section, is sufficient to calibrate the camera.

We first analyzed the image, pinpointing suitable straight lines and picking out the person to be used to calculate the camera height. We then calculated the vanishing points locations in image coordinates together with their standard deviations to indicate their accuracy. Instead of just using all suitable straight lines we also used a limited set of lines for the vertical vanishing point to show the effect of the lines chosen on the calibration results (see figure 5.1).

Analyzing the scene

To estimate the three vanishing points we need three families of straight lines of which we know they are parallel in the real world and each family of parallel lines is orthogonal to the other families. The scene contains three structures fulfilling these constraints: the building in the back, the pavement and the art work in the top-left corner.

We only use the pavement to estimate V_x[1] and V_z[2], as we are not convinced

[1] V_x is the vanishing point on the horizon right from the camera

[2] V_z is the vanishing point on the horizon left from the camera

Vanishing point	Amount of lines used	Amount of intersections	
		before elimination	after elimination
V_x	18	153	55
V_y	14	91	40
V_z	15	105	63

Table 5.1: *Amount of lines and intersections used to estimate the vanishing points*

that the pavement is aligned with the building [3]. The advantage of using the pavement is that it happens to have a grid-like structure which contains many parallel lines, hence we have two families of parallel lines of which we are confident they are orthogonal.

To estimate V_y the possible structures containing vertical lines are: the art work at the top left, the building in the back and the signposts. These three structures are not connected and therefore one cannot be sure these are mutually parallel. The signposts are all independent, hence their slopes are not likely to all be the same and therefore will only degenerate the estimate of V_y. The building contains some straight lines (i. e. the windows left to the door) that are parallel in the image. Considering robustness to noise these lines were not used to estimate V_y. The slopes of the lines are estimated by pinpointing the endpoints of the lines in the image. This contains a fixed source of error which propagates more when considering short lines. Hence we only used straight lines at the left and right border of the image from the building in the back and the art work to estimate V_y.

To estimate the camera height we need a reference object with known height and standing vertical. For this we use a person for whom we can determine his or her feet location. Looking at figure 5.1, the woman standing in the top right of the image (indicated by a rectangle) satisfies these conditions.

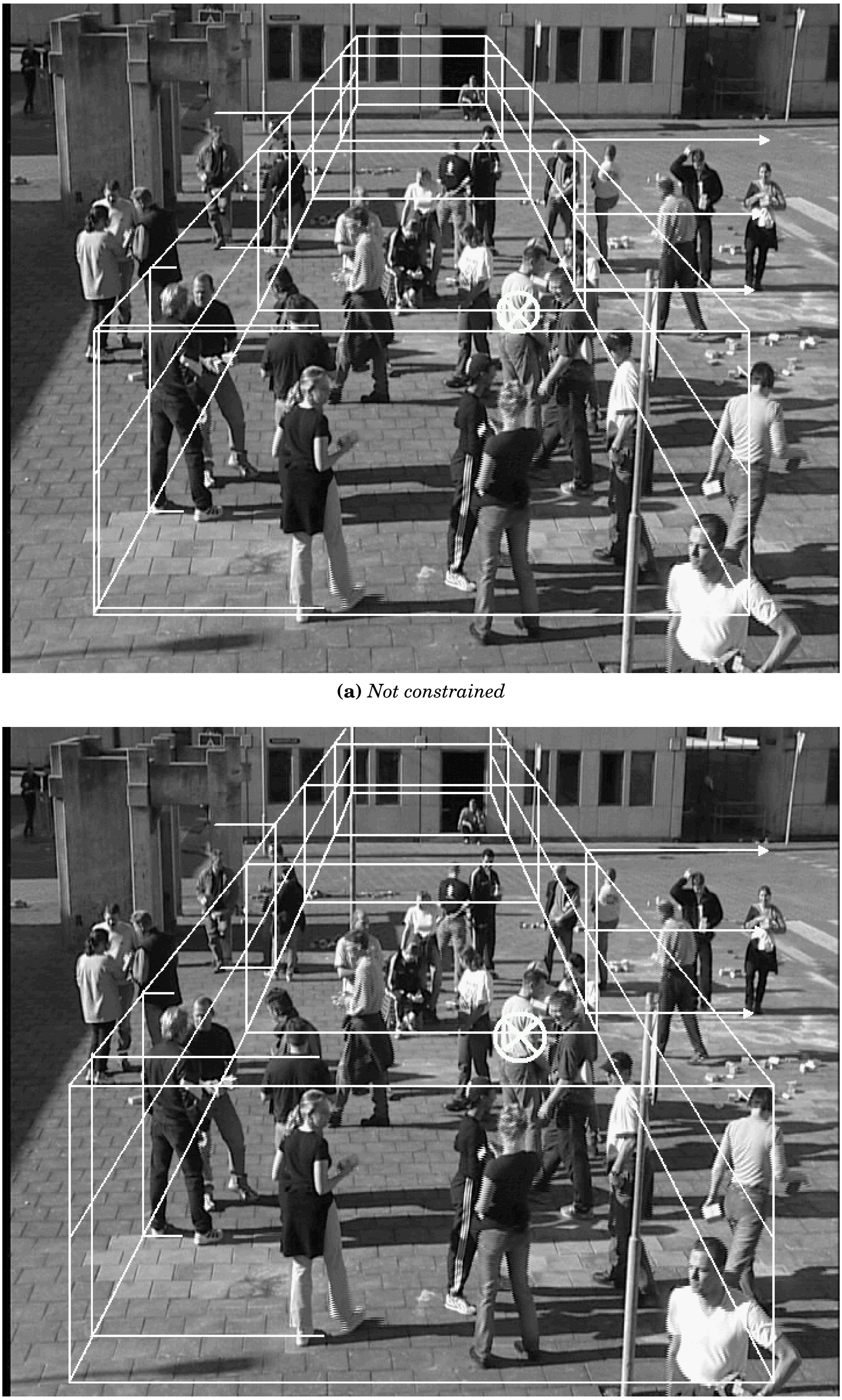
Estimating the vanishing points locations

In table 5.1 the amount of lines used to estimate the vanishing points are given. The first column shows the number of lines used, the second the total amount of intersections and the last column the actual amount of intersections used to estimate the vanishing point's location. In figure 5.1b the lines are visualized as straight white lines.

The estimated vanishing points locations in pixel coordinates are shown in table 5.2. The first row is the estimated vanishing point's location, the second row contains the standard deviations after eliminating all outliers. The last column shows the results for V_y when only 7 lines are used, 5 taken from the architectural structure and 2 taken from the right-side of the building in the back as shown in figure 5.1a. The standard deviation gives an indication on how well the chosen family of parallel lines correspond (i. e. intersect in the same vanishing point).

One might argue that the estimates of V_x and V_y are very inaccurate and therefore not suitable to use in the calibration process. However, the horizon line is practically horizontal. When considering an identical change in the x and y coordinate, the error in the y coordinate has a much bigger influence on the slope of the horizon line than the error in the x coordinate has. Hence, as

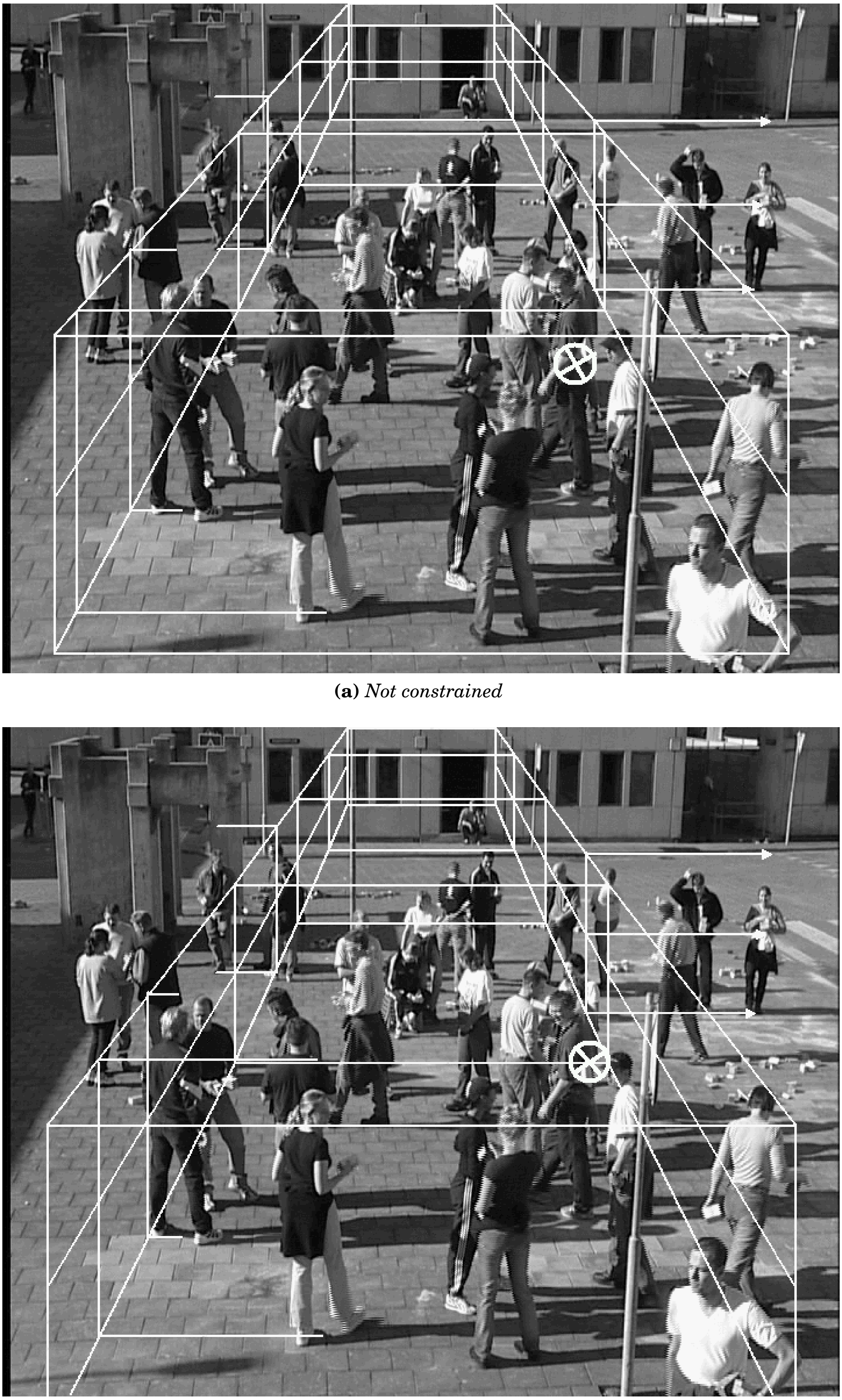
[3] The straight line of the curb in front of the building intersects at the left of the image while the straight lines from the pavement intersect at the right.



(a) *Not constrained*

(b) *Constrained*

Figure 5.2: *Calibration results for the lines as specified in figure 5.1a. The circle with a cross in each picture represents the principal point location and the arrows indicate the person used as reference object. The lower line is supposed to be on the ground, the middle line is at height* $1m$ *above the ground and the upper line is supposed to be* $2m$ *above the ground.*



(a) *Not constrained*

(b) *Constrained*

Figure 5.3: *Calibration results for the lines as specified in figure 5.1b. The circle with a cross in each picture represents the principal point location and the arrows indicate the person used as reference object. The lower line is supposed to be on the ground, the middle line is at height* 1m *above the ground and the upper line is supposed to be* 2m *above the ground.*

V_x	V_y	V_z	$V_{y\text{limited}}$
$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 7542.8 \\ 229.4 \end{bmatrix}$	$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 313.1 \\ 4657.5 \end{bmatrix}$	$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 261.2 \\ -70.0 \end{bmatrix}$	$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 283.7 \\ 4148.3 \end{bmatrix}$
$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} 1115.6 \\ 37.5 \end{bmatrix}$	$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} 17.6 \\ 499.3 \end{bmatrix}$	$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} 1.9 \\ 5.7 \end{bmatrix}$	$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} 13.3 \\ 340.0 \end{bmatrix}$

Table 5.2: *Vanishing points locations and their standard deviations*

S_y for V_x is small we can assume the estimate is accurate enough to estimate the horizon line. The same goes for V_y except now the x coordinate is of greater importance for estimating the principal point.

V_z and V_x are both estimated using straight lines taken from the pavement. Here we clearly see the effect that the angle of the parallel lines with the image plane has on the accuracy of estimating the vanishing point. The lines used to estimate V_x are practically parallel to the image plane, while those used to estimate V_z are not. If we look at the standard deviations of both vanishing points, we see that the intersections to estimate V_z lie closer to each other than those used to estimate V_x.

Estimating the reference object's height

The height of the person used as a reference object is indicated by the arrows in figures 5.2 and 5.3. Our initial guess was that she would be about $1.70m$ tall, however this did not correspond well with the calculated height by back projecting the image points. We lowered our estimate to $1.60m$, now the woman's height calculation corresponds well with our estimate. However calculating the height of the men standing close to the camera, they should be about $2m$ tall. Although $2m$ is not unrealistic it is not very common and thus we reconsidered our estimate. We lowered our estimate again with $10cm$ to $1.50m$ and this resulted in the calibration parameters as shown in figures 5.2 and 5.3 and in table 5.3.

Calibration results

The calibration results are shown in table 5.3 and figures 5.2 and 5.3 for both the constrained and unconstrained version.

Figure 5.2a shows the calibration result when using the lines as drawn in figure 5.1a without constraining the principal point. On first sight this is a decent estimate, it has the principal point near the image center, as expected, and people's estimated heights are fair. However the the woman used as a reference object should be $1.70m$ tall, which is not in agreement with our estimate of $1.50m$. If we constrain the principal point as shown in figure 5.2b, we see we can do better. Again overall people's estimated heights are fair, however now the woman should be $1.56m$ tall, which is in better agreement with our estimate.

Figure 5.3 shows the calibration results when using the lines as specified in figure 5.1b. Here we see that constraining the principal point's location has less impact on the final calibration result. In figure 5.3a the result is shown when not constraining the principal point. The woman should be about $1.48m$ tall. In

	Using the lines as specified in figure 5.1b		**Using the lines as specified in figure 5.1a**	
	constrained	*unconstrained*	*constrained*	*unconstrained*
Focal length	1291.564323	1264.260174	1213.923657	1131.530763
Principal point	(504.2, 288.0)	(492.0, 306.8)	(454.0, 288.0)	(443.3, 267.4)
Pan angle	0.256423	0.194820	0.270674	0.164678
Tilt angle	0.288679	0.282572	0.306042	0.283468
Yaw angle	-0.041104	-0.041104	-0.041104	-0.041104
Camera height	475.1	429.6	468.3	426.8

Table 5.3: *Calibration results for the TNO sequence, the focal length is given in pixels, the principal point in image coordinates, the angles in radians and the camera height in centimeters*

figure 5.3b the result is shown when the principal point is constraint, here the woman should be about $1.52m$ tall.

Considering the overall height estimates of people in the four different tests and that best robustness to noise is obtained using as many parallel lines as possible and constraining the principal point, we take the calibration result obtained using the lines as specified in 5.1b (i. e. all selected straight lines) and constraining the principal point location as the most accurate.

5.2 Texture classification

In figure 5.4 some example images used to train the classifiers are shown. We have performed a total of eighteen experiments, three person/non-person and fifteen BFD classification experiments. The person/non-person experiments were done for different parts of the human body:

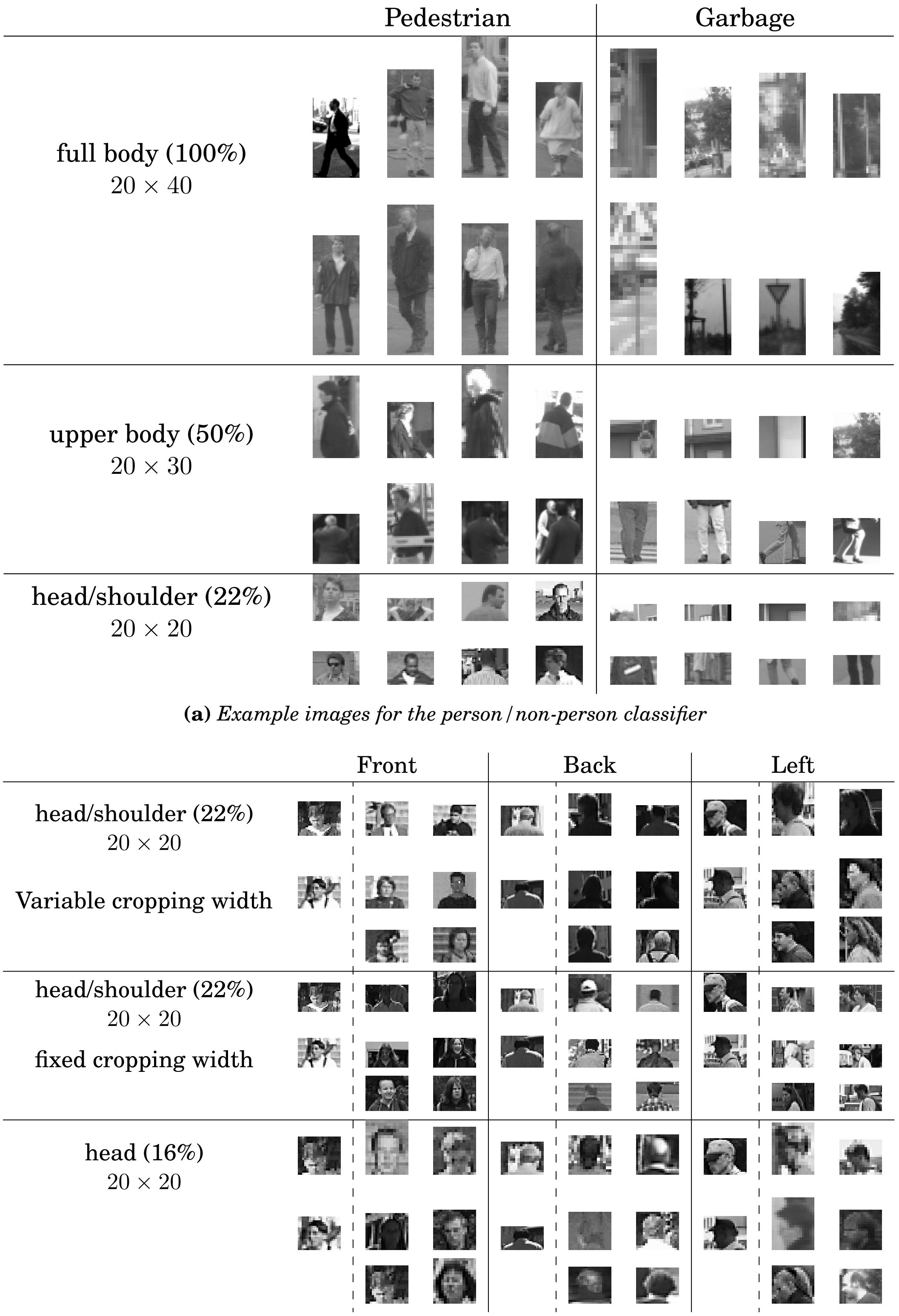
- full body (100%) normalized to 20×40 pixels
- upper body (50%) normalized to 20×30 pixels
- head/shoulder (22%) normalized to 20×20 pixels

Normalization was needed, as the classifiers need input patterns of the same dimensions.

The BFD experiments were done for

- head/shoulder (variable cropping width)
- head/shoulder (fixed cropping width)
- head (16%) normalized to 20×20 pixels

Depending on the shape model that matches the input pattern given to the classifiers differs in size. However, for the classifiers the input patterns all need to be of the same size and therefore require scaling. To minimize distortion from scaling, the images used to train and test the *head*/*shoulder* classifier were scaled to 20×20 as this is by average the size of the ROI given by the Chamfer System for the *head*/*shoulder* configuration and image resolution of



(a) *Example images for the person / non-person classifier*

(b) *Example images for the BFD classifiers. The column separated by a dashed line contains the same images for different cropping settings*

Figure 5.4: *Example images used to train and test the different classifiers*

360×288 pixels[4]. For the *full body* and *upper body* classifiers the images were scaled to 20×40 and 20×30 pixels, respectively.

In the variable cropping width experiments, the cropping width is determined by the person's width. Hence, the images of people standing sideways are typically smaller than those of people standing towards to or away from the camera. To illustrate the effect of scaling with respect to the shoulder width feature, we also performed experiments using a fixed cropping width. The fixed cropping width was determined by taking the average width of all people in the training set. The experiments using the *head* configuration were done to show the effect of the facial skin and hair ratio feature (see chapter 1).

Determining the best network configuration

We manually determined the best network configurations by training the network for different learning rates using an initial ”random” configuration for the receptive fields[5]. The best learning rate was then used to train the network for different receptive field configurations. After which the best receptive field configuration was used to once again calculate the best learning rate. This process was repeated until the best learning rate and receptive field configuration no longer changed.

The classifiers used in all tests have two branches in layer 2 and two output neurons. These were trained for differing receptive field sizes (R×R pixels) ranging from R = 1 to R = 10 and spatial offsets of D = 1 to D = R. Larger values for R are possible but require too much time to train the network. We believe, however, that the best performing receptive field configurations will be found for sizes smaller than 10 (see chapter 4).

Creating the train and test images

The original database of images contained 2633 images of pedestrians and 11651 images of scenery (i. e. images not containing humans, also referred to as garbage). Each of the pedestrian images in the database has two versions, one with labeling and one without. In figure 5.5 one image from the database is shown with labeling. This labeling consists of the contours of the two people outlined using different colors for each different person. We used this contour labeling to determine the bounding box for each person for a specific cropping percentage.

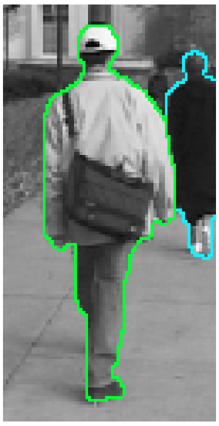


Figure 5.5: *Example image of labeled pedestrians*

People in the database do not always look in the same direction their body is pointing. These were limited to those where the person is looking in approximately the same direction his or her body is facing. This resulted in 1628 images found suitable to train the BFD classifiers.

Each of the pedestrian images were flipped and shifted which increased the amount of images by a factor of 10, as discussed later in the text. The actual amount of train and test images used for each classifier is shown in table 5.4. The difference in amount of samples used for the different cropping percentages is due to the fact that some images contain more than one

[4] People in the center of the image standing sideways have an average ROI of 10×20 and people facing the camera 30×20

[5] We always took a receptive field size of 9×9 (i. e. R=9) and spatial offset of 5 (i. e. D=5) as initial configuration as used by Wöhler and Anlauf [32]

	full body		upper body		head shoulder	
	person	garbage	person	garbage	person	garbage
train	24600	11651	24700	10892	24910	12004
test	16400	9623	16825	8287	17060	8838

(a) *Person / non-person classifier*

	cropping 22% head/shoulder configuration			cropping 16% head configuration		
	front	back	side	front	back	side
train	4695	5025	3275	4715	5040	3275
test	1530	3720	2380	1570	3765	2385

(b) *BFD classifier*

Table 5.4: *Amount of samples used to train and test the neural networks*

labeled person. Some of these, as can be seen in figure 5.5, are not completely visible. A specific cropping is only included when completely visible. Hence the person in the middle in figure 5.5 is added in all sets, however the person on the right is only added when considering a cropping percentage of 16%.

Calculating the cropping area

The cropping area is calculated by first finding the minimum and maximum y coordinate of points on the labeled contour (i. e. y_{max} and y_{min}). Then y_{crop} is calculated as follows

$$y_{crop} = (y_{max} - y_{min}) \times c + y_{min} \tag{5.1}$$

where c is the percentage of cropping.

The maximum width is obtained by finding the maximum and minimum x coordinate of points on the contour (i. e. x_{max} and x_{min}) within the cropping boundary y_{min} and y_{crop}. The cropping area then becomes

$$\big((x_1, y_1), (x_2, y_2)\big) = \big((x_{min} - 1, y_{min} - 1), (x_{max} + 1, y_{crop} + 1)\big) \tag{5.2}$$

The cropping area is made 2 pixels larger to make sure that any occlusion of a person's contour due to labeling would not discard any pixels belonging to the person.

Using a detection method like the Chamfer System we may expect the input patterns to contain some spatial displacements. To represent this in the training and test set the cropping area was shifted over the images as

$$\begin{aligned} &(a, b) \in \big\{(0,0), (-2,-2), (-2,2), (2,-2), (2,2)\big\} \\ &\text{cropping area:} \quad \big((x_1, y_1) + (a, b), (x_2, y_2) + (a, b)\big) \end{aligned}$$

Some cropping offsets result in cropping areas outside the image boundaries. When this happened the corresponding shift was not included in the test or training set.

Each of the included images were mirrored around the vertical center axis. Hence we increased the used train and test images by a factor of $2 \times 5 = 10$.

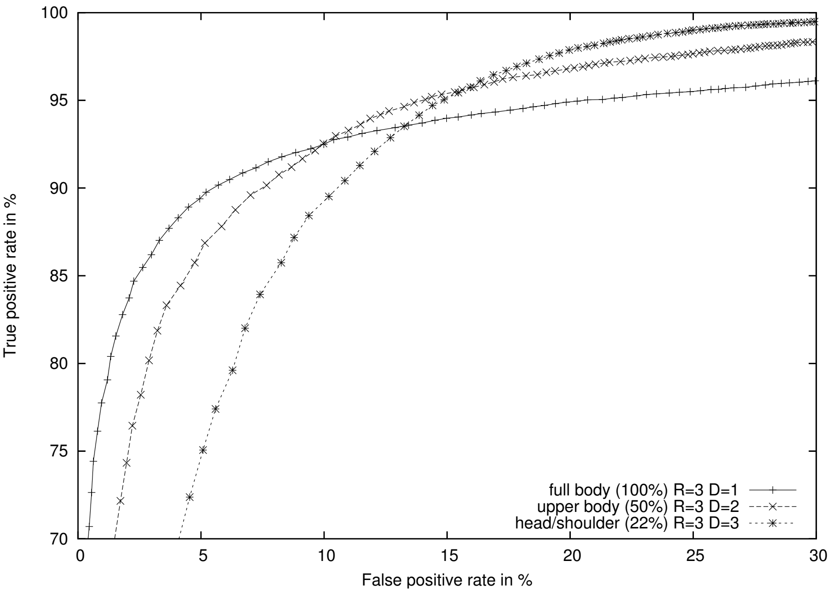


Figure 5.6: *ROC curves for person / non-person classification*

Classifier	TPR	FPR
head / shoulder	95%	15%
upper body	91%	8%
full body	90%	5%

Table 5.5: *Person / non-person classifiers performance*

Person/non-person classifier

In figure 5.6 the ROC curves are shown for the person/non-person classifier. Note that for clarity only the results for the best performing network configurations are shown.

For low FPR we see that the *full body* performs better than the others. This shows that the additional information (i. e. legs) has a positive effect even though the dimensionality of the input patterns is increased, whereas the amount of training samples is approximately the same. For TPR $> 95\%$ we see that the effect is reversed, where the *head / shoulder* classifier performs better than the others.

The amount of characteristic properties is of course crucial in separating the two classes. This amount is greatest when considering *full body* images and decreases for lower cropping percentages. However, if the variation of these properties in the input patterns is large, the classifier needs an increasing amount of samples to create a general model. This is typically the case for the *full body* classifier as it has to cope with the criss-cross movement of the legs and flexibility of the arms. Hence, for the *full body* patterns the variations in position and shapes of pedestrians is much greater then it is considering the *head / shoulder* or *upper body* patterns. Therefore, the *full body* classifier performs best for lower FPR, as its input patterns contain the most characteristic features and the *head / shoulder* classifier performs best for higher FPR, as its input patterns are most consistent (i. e. least variation in the input patterns).

Depending on the specific application area one might accept a lower TPR when this results in a lower FPR, or vice versa (i. e. giving more or less weight to one of the two classes). Here we simply take the performance when using an equal weight for both classes.[6] The performance for the three classifiers is shown in table 5.5.

Body facing direction classifiers

We manually sub-divided the images of pedestrians in the database used by the person/non-person classifier into four sets: **front**, **back**, **right** and **left**. The left and right facing direction images are actually the same where opposite directions are flipped to the direction of the corresponding set.

We have trained the NN for two differing classification tasks, ***1 versus 1*** and ***1 versus 3***. Six ***1 versus 1*** classifiers were trained

> ***back versus front, left versus right, left versus back***
> ***left versus front, right versus back*** and ***right versus front***

and four ***1 versus 3*** classifiers were trained

> ***front versus other***, ***back versus other***, ***left versus other*** and ***right versus other***.

1 versus 1 classifiers

back versus front	***left versus right***
•facial skin and hair ratio	•facial skin and hair ratio •head position relative to the body

left versus front	***left versus back***
•facial skin and hair ratio •head position relative to the body •shoulder width	•facial skin and hair ratio •head position relative to the body •shoulder width

Table 5.6: *Facing direction features that are apparent for a specific classification task*

In figure 5.7a the ROC curves are shown for the ***1 versus 1*** classifiers using the *head/shoulder* patterns. Taking an equal weighting for both classes we get a TPR and FPR as shown in table 5.7. In chapter 1 we have described three features we find important for estimating a person's BFD. The features that can be used for a particular classification task are given in table 5.6.

The best performing classifier is the ***back versus front*** classifier followed by the ***left versus right***. Interestingly the other classifiers perform far less, although these should be able to use all three features found relevant to make a distinction between the two BFDs. The ***right versus*** {***front or back***} classifiers have been left out as these are the same as the ***left versus*** {***front or back***} classifiers. The *shoulder width* feature can be removed from the list of useful features as due to scaling the difference in shoulder width for a side or frontal/rear view is nihil. The performance of the ***back versus front*** classifier indicates that the *facial skin and hair ratio* is an important feature.

To confirm our assumption that the *facial skin and hair ratio* is an important feature, we have trained four classifiers using input patterns of people's heads (see figure 5.4 row 16%). The results are shown in figure 5.7b. We have left

[6] This can be visualized in a ROC figure by the line l going through (TPR, FPR)= $(0, 0)$ and $(100, 100)$. The corresponding threshold for each classifier is the point where the tangent line parallel to l touches the ROC curve.

out the results for the ***right versus*** {***back, front***} classifiers as performance is practically the same as for the ***left versus*** {***back, front***} classifiers.

The results show that the performance of the ***back versus front*** is significantly improved, which we believe is mainly due to the improving ratio of background pixels in relation to foreground pixels. The performance of the other three classifiers ***left versus*** {***back, front, right***} all decreased. This indicates that the *head location relative to the body* feature is most apparent for the ***left versus right*** classifier and for the {***left or right***} ***versus*** {***back or front***} classifiers (i. e. a side view versus a frontal/rear view).

To analyze the effect of scaling the input patterns with respect to the shoulder width feature, we have trained two classifiers ***left versus front*** and ***left versus back*** using the fixed width cropping data set as shown in figure 5.4 row 'fixed cropping width'. The results are shown in figure 5.8. Again, the ***right versus*** {***back, front***} classifiers were left out as performance is similar as the ***left versus*** {***back, front***} classifiers. The ***left versus right*** and ***back versus front*** classification tasks are not considered as the shoulder width should be practically the same in both classes.

We used a cropping width of 35 pixels for all images in the train and test set. This was determined by taking the average shoulder width of people in the data set. The maximum shoulder width was found to be 57 pixels and the minimum 16 pixels. This means that for some images the people are not completely visible and some where the amount of background pixels is about twice as large as foreground pixels.

Comparing the results with those for the ***1 versus 1*** classifiers as shown in figure 5.7a, we see that performance is decreased considerably. The only difference between the two tests is the amount of horizontal cropping of the train and test images. Hence, we conjecture that the decrease in performance is mainly caused by the increasing amount of background pixels and therefore these tests, unfortunately do not tell us anything about the effect of scaling with respect to the *shoulder width* feature.

1 versus 3 classifiers

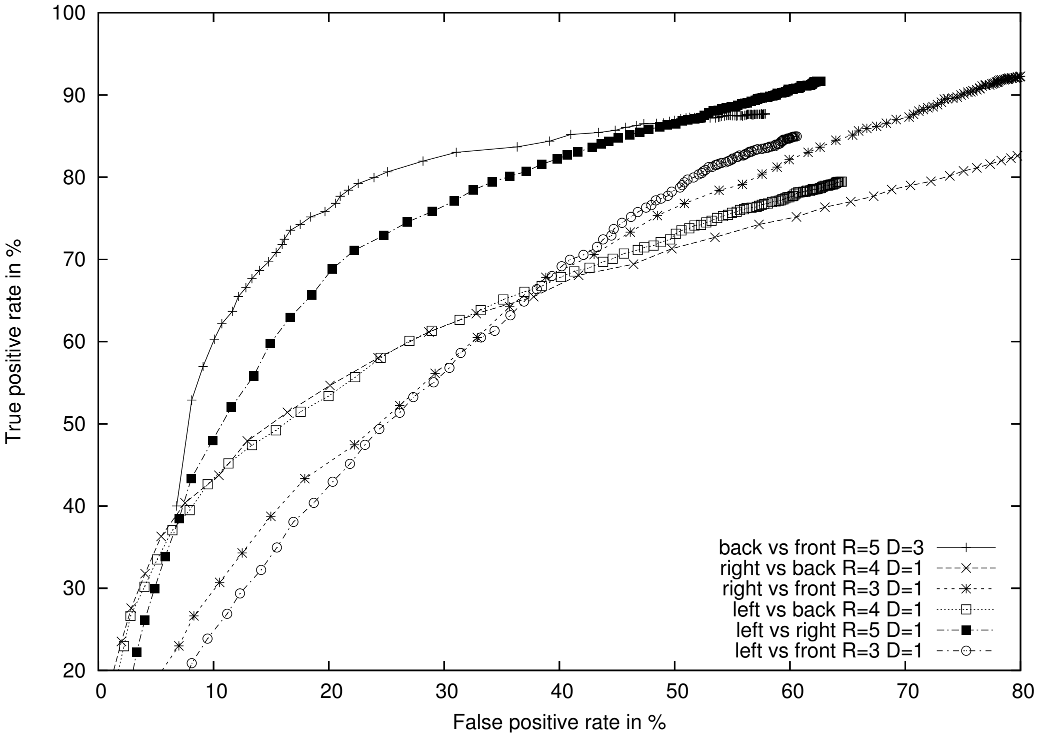
In chapter 4 we mentioned two possible hierarchies of networks to estimate the BFD. One uses ***1 versus 1*** classifiers, the other ***1 versus Rest*** (i. e. ***1 versus 3*** in our case). In figure 5.9 the ROC curves are shown for the ***1 versus 3*** classifiers. We see that performance of all classifiers is worse then the ***1 versus 1*** case.

The ***1 versus 3*** classifiers have to generalize three classes in one branch (layer 2) and one class in the other, instead of just one for both branches as is the case when considering a ***1 versus 1*** classification task. The increase in variation of the input patterns makes it more difficult to learn a general model for the corresponding input class. This is probably why the ***1 versus 3*** classifiers perform less than the ***1 versus 1*** classifiers.

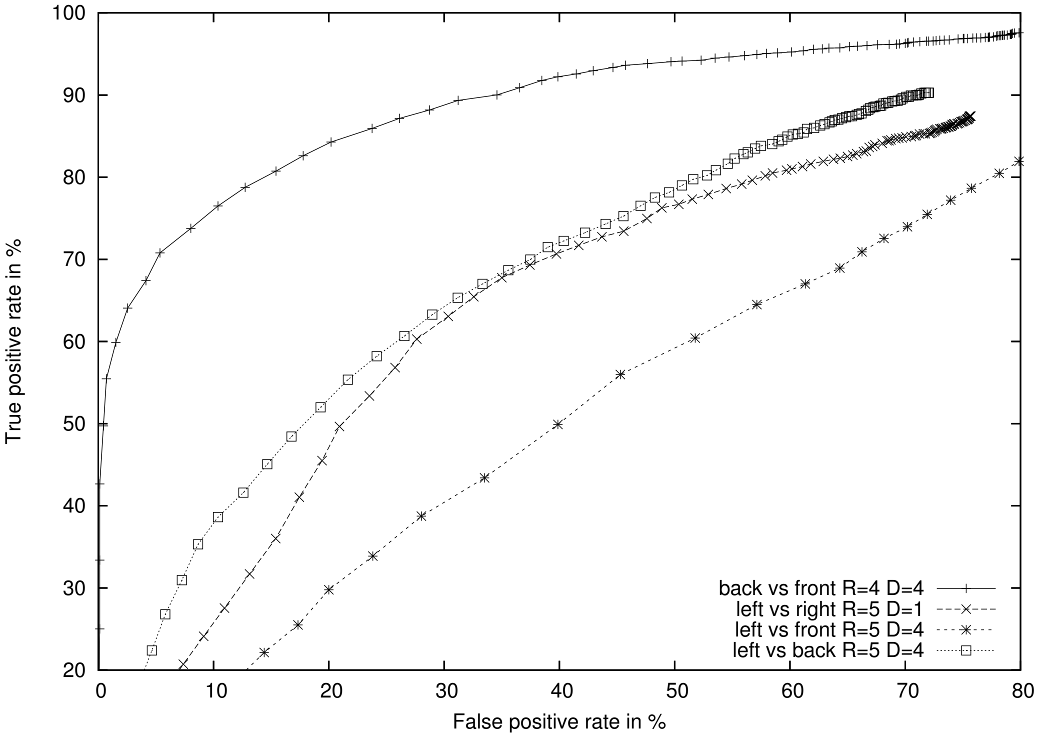
Estimating the BFD using a hierarchy of classifiers

Using the ***1 versus 1*** classifiers we have tested the BFD estimation using a hierarchy of networks as shown in figure 4.3. In table 5.7 the TPR and FPR are shown for each of the ***1 versus 1*** classifiers we have chosen as best using the ROC curves from figure 5.7a.

To test the BFD estimation we have used the images of the test set which were also used to test the ***1 versus 1*** classifiers after training. The results are



(a) ***1 versus 1*** *classifiers using the head/shoulder data set*



(b) ***1 versus 1*** *classifiers using head data set*

Figure 5.7: *ROC curves for the* ***1 versus 1*** *classifiers using the head/shoulder and the head data sets*

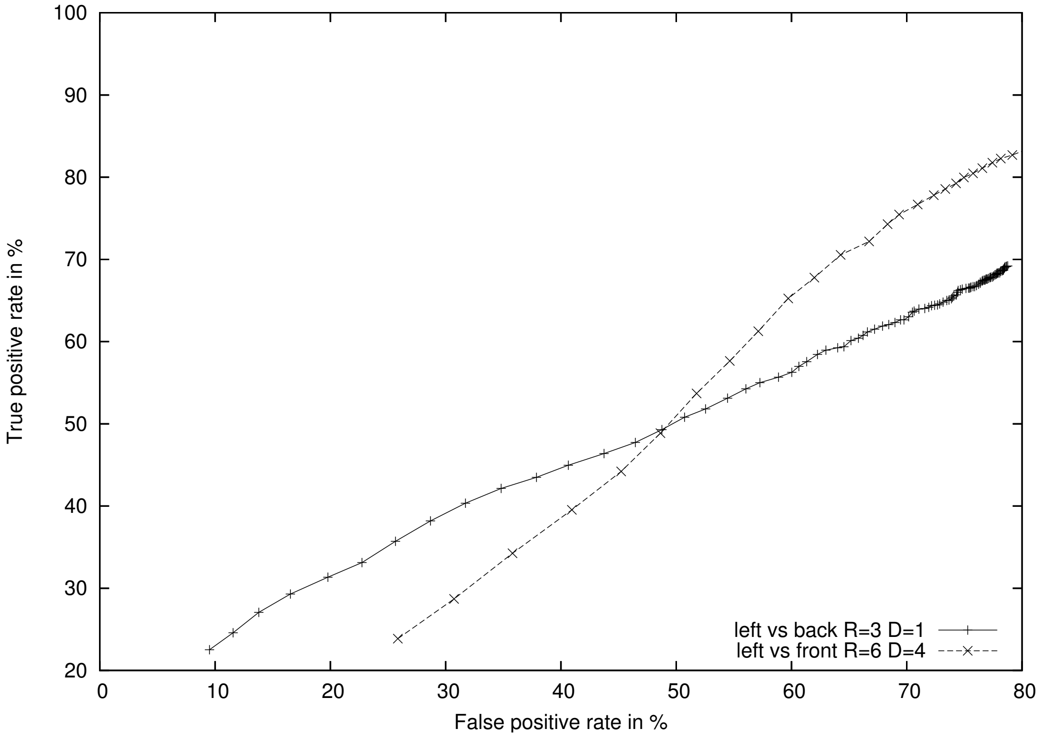


Figure 5.8: *ROC curves for the **1 versus 1** classifiers using the fixed cropping data set*

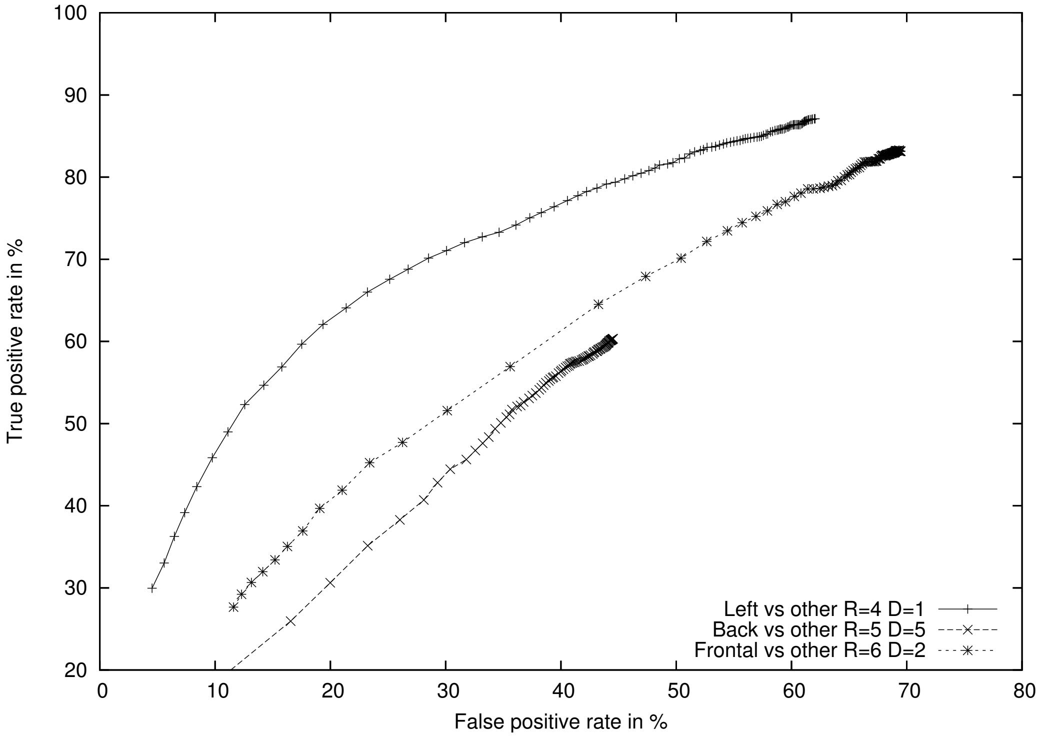


Figure 5.9: *ROC curves for the **1 versus 3** classifiers using the head/shoulder data set*

Classifier	TPR	FPR
back versus front	74%	17%
left versus right	71%	22%
left versus front	69%	40%
left versus back	51%	18%
right versus back	53%	18%
right versus front	68%	39%

Table 5.7: *Performance per classifier in percentages*

	front	back	left	right
front	657	228	192	453
back	723	1189	258	1550
left	1153	217	263	747
right	1177	344	185	674

(a) *Confusion matrix using absolute terms*

	front	back	left	right
front	43%	15%	12%	30%
back	19%	32%	7%	42%
left	49%	9%	11%	31%
right	49%	15%	8%	28%

(b) *Confusion matrix using percentages*

Table 5.8: *Performance of the BFD estimator using a two-stage hierarchy of* ***1 versus 1*** *classifiers. Tests were performed using the same classifiers and test set corresponding to those shown in table 5.7*

shown in table 5.8a and clearly show that the performance of the ***1 versus 1*** classifiers is insufficient to be able to determine a person's BFD by themselves.

5.3 Combining shape model labeling and texture classification

In the previous section we found that a hierarchy of neural networks is unable to classify an input pattern into one of the four main facing directions. We believe that this is mainly due to the bad performance of the {***left,right***} ***versus front*** and {***left,right***} ***versus back*** classifiers. We therefore only wish to use the ***back versus front*** and ***left versus right*** classifiers. This however, requires a selection method which determines if a person is standing towards/away or side ways to the camera.

In chapter 1 we have determined three features which might be useful in determining a person's BFD. Two of these, *shoulder width* and *head location relative to the body*, are also apparent in the shape models used to detect the location of a person in an image. Thus by labeling the shape models as a front/back or side view model, this label can be used during matching to select either the ***back versus front*** or ***left versus right*** classifier.

To test the method we use a 20 second video sequence taken by TNO from an elevated viewpoint (see figure 1.2). The original image resolution is 720×576 pixels which we scaled down to 360×288. This reduces the search space and makes the people correspond more with the available shape models.

Creating and labeling head/shoulder shape models

We have seven different shape model sizes to our disposal, these are for the full body case, 48, 54, 60, 66, 72, 78 and 84 pixels in length. The original database contains 5322 models for differing BFDs per size set. Thus we have a total of

$7 \times 5322 = 37254$ models. These were cropped to 22% to contain only the head and shoulders.

We have manually selected 60 models, 30 front/back and 30 left/right, from one of the size sets. These were manually labeled to the corresponding BFD. We also tried to automatically label the shape models by considering the maximum and minimum shoulder width, however due to differing sizes of people (e. g. thin and thick people, different clothing) this did not always lead to correct labeling. Using the labeled models the corresponding models in the other size sets were labeled with the same BFD as the manually labeled models.

Using the cropped shape models we have build a hierarchy using two prototypes at level 1 for each size set. This means we have $2 \times 7 = 14$ shape models at level 1 and $7 \times 60 = 420$ models at level 2.

Detection range

As mentioned earlier, the shape models do not represent people viewed from above well. Hence in our scenario people standing further away from the camera should match more accurately than people standing close to the camera. In the test sequence most people are found in the middle of the image. We therefore concentrate on people in the middle of the image and want to scale the input image (i. e. TNO test sequence) down to make the people correspond better with the available template sizes.

We expect people to be between $2.00m$ and $1.40m$. Using the calibration parameters as found best in section 5.1, we have calculated the largest and smallest needed template size for different amounts of image resolutions. We found that the available shape model sizes correspond best when scaling the image resolution to 360×288.

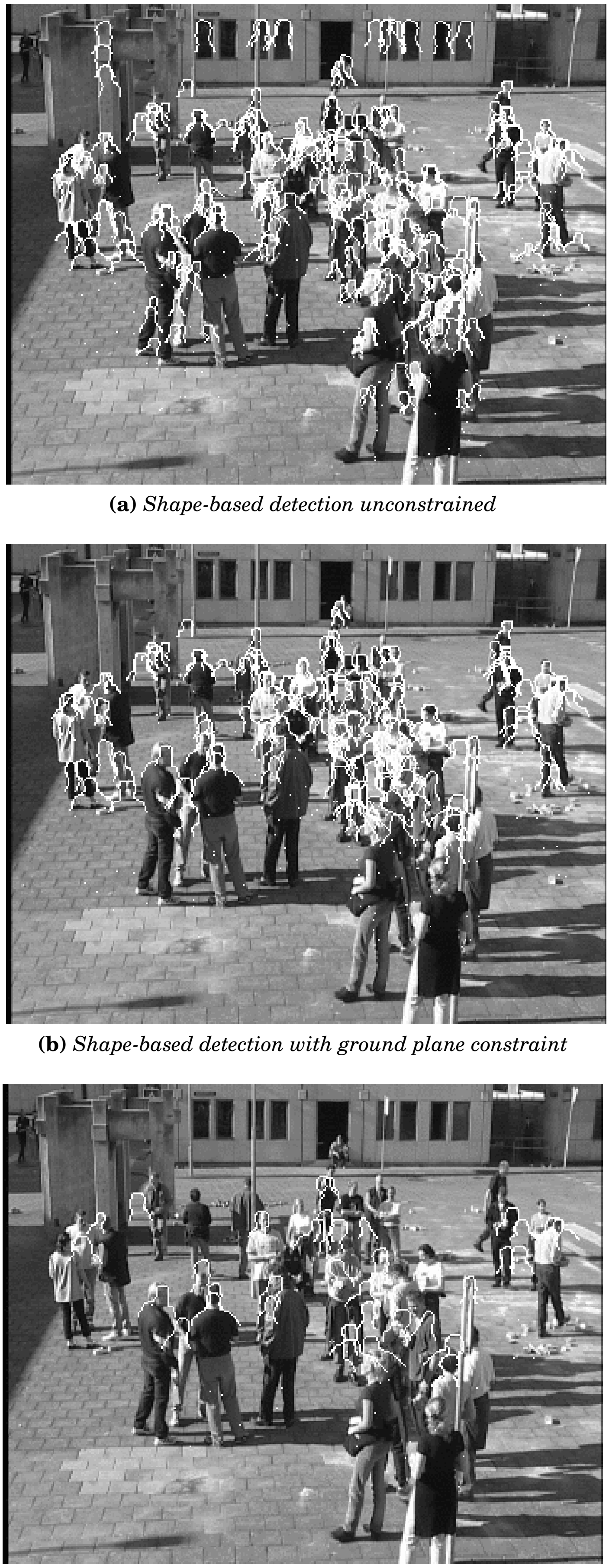
This results in a largest template needed for a person of length $2.00m$ of 129 pixels to represent a person's whole body standing at the bottom of the image (i. e. with his or her feet at image location $y = 288$). The building in the back limits the distance at which people may be visible, hence at image location $y = 117$ the largest template needed should be 66 pixels in length. For a person of length $1.40m$ the largest template should be 89 and the smallest 46 pixels. This means that people standing below approximately two-third of the image (i. e. their feet below $y \approx 192$) are too big for the currently used template sizes and will therefore not be detected.

Detection results

In figure 5.10 one frame of the TNO sequence is shown three times, once for detecting people using *no constraints*, once using the *ground plane constraint* and once using both the *ground plane constraint* and the *person/non-person classifier*. Unfortunately we do not have a version of the test sequence containing labeled people and therefore we cannot provide a quantitative measure of the performance of the detection method. The results, as given in figure 5.10, are merely used to illustrate the effect of using the ground plane constraint and texture classification.

As the advantage of using a hierarchy is little we have chosen to set the threshold for the first level relatively high and control the detection rate by adjusting the threshold of the second level. The results as shown figure 5.10 were performed using a threshold of 4 for the first and 2 for the second level of the hierarchy, using a step size of 4 and 1 respectively.

Using *no constraints* as shown in figure 5.10a, we get a lot of false positives



(a) *Shape-based detection unconstrained*

(b) *Shape-based detection with ground plane constraint*

(c) *Shape-based detection with ground plane constraint and person/non-person classifier*

Figure 5.10: *Detection results for the TNO sequence*

(e. g. people's arms and legs, the windows in the back and the art structure). However the templates that match represent a person's length which is not very likely to occur at that location. In figure 5.10b the result is shown using the same threshold values as in figure 5.10a, but now using the ground plane constraint. The windows and a lot of people's arms and legs are no longer detected as humans.

The *head/shoulder* shape model is not very characteristic as it matches on a lot of similar shapes which are not a person's head and shoulders. We therefore used the person/non-person classifier for the head/shoulder input patterns from the previous section to verify the detection results as given by the Chamfer System. In figure 5.10c one frame of the TNO sequence using the *ground plane constraint* and the *person/non-person classifier* is shown. We see that the amount of false positives is reduced however some good detected people are discarded as well. However analyzing the result on the TNO sequence we must note that performance is less than the performance given in table 5.5.

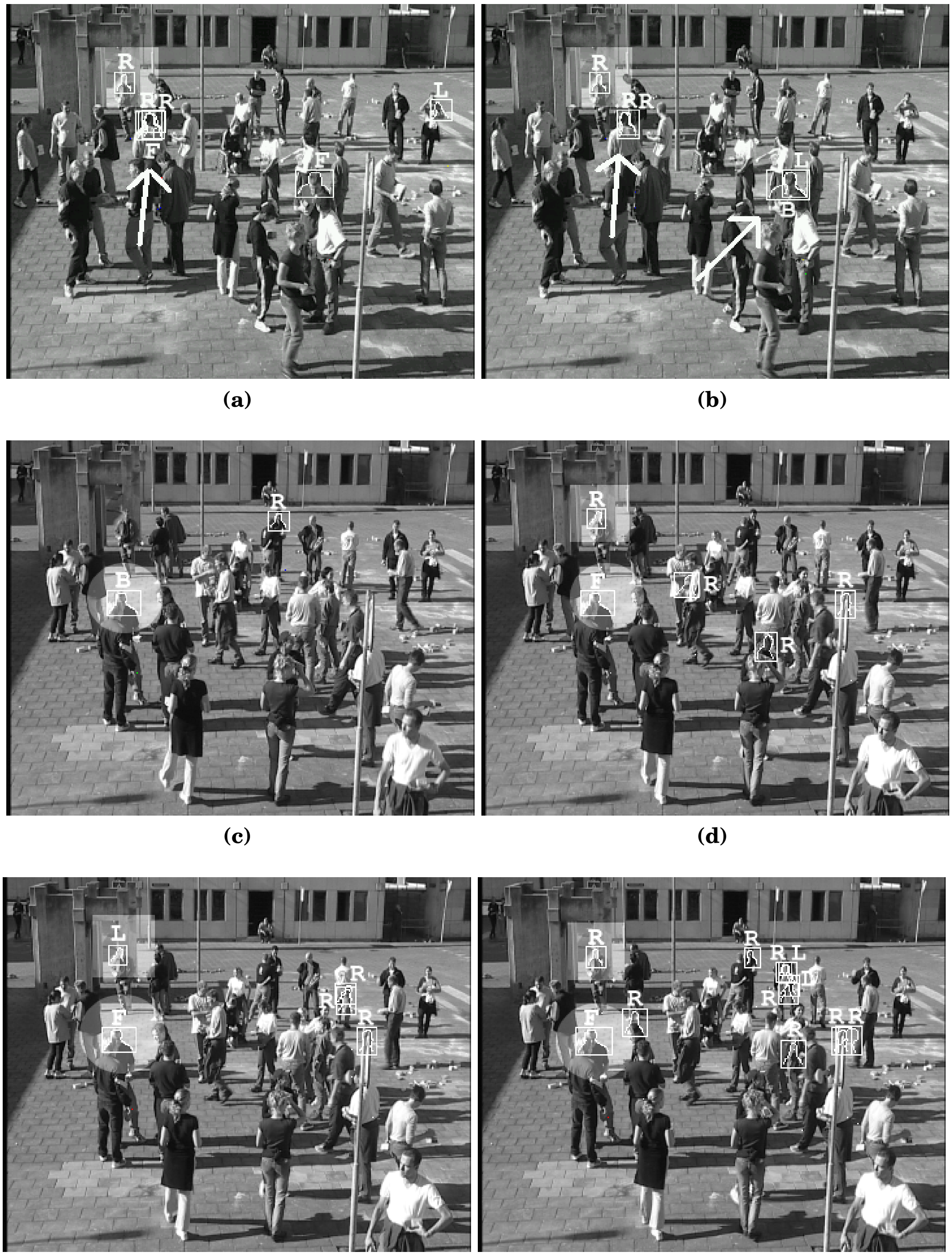
Estimating the Body Facing Direction

In figures 5.11 and 5.12 the results for estimating the BFD are shown for a couple of frames taken from the test sequence. The rectangles point out the location of detected people together with the matching shape model. Alongside each rectangle the estimated BFD is given: (R)ight, (L)eft, (F)ront and (B)ack. As noted earlier, we are unable to provide any quantitative measurements. The results are merely given to highlight some specific issues with detecting and estimating the BFD of people.

The thresholds as used to illustrate the effect of using a ground plane constraint and the person/non-person classifier were not set very strict. Here we tighten the thresholds a lot more to reduce the amount of false positives considerably. This provides us with results of which we more clearly can see the matching accuracy and BFD estimation. A disadvantage of setting the threshold more tight is that some people are no longer detected. We have taken a threshold of 4 for the first and 1.2 for the second level using a step size of 4 and 1 respectively.

In figures 5.11c–d and 5.12a–f some frames are shown, where one person is highlighted by an ellipse. This person is labeled mostly correctly as **front** and in figure 5.12e as **right**, even though he is partly occluded by the person standing in front of him. Figures 5.11c and 5.12f show two frames where he is incorrectly labeled as **back**. Figures 5.11a,b,d –f and 5.12a–d show a person highlighted by a rectangle. The person is standing towards the camera, however looking to the right. In figures 5.11a,b,d,f and 5.12d the person is labeled incorrectly as **right**. In figures 5.11e and 5.12a,b,c the person is labeled incorrectly as **left**. Figures 5.11a and 5.11b illustrate another problem where people are detected more than once (indicated by arrows).

Considering the shape models that match we see that incorrect labeling is often caused by an incorrect matching shape model. When the person highlighted by a rectangle is labeled as **right** a different side view shape model matches than when labeled as **left**. In figure 5.12f we see that the person highlighted by an ellipse is detected using a front/back view shape model, whereas a side view shape model would be more appropriate. Although accurate shape matching is crucial for correct labeling, figure 5.11b shows that incorrect labeling is not always due to incorrect shape matching. The person pointed out by an arrow and labeled as **back** is accurately matched by a front/back shape model, however labeled incorrectly as **back**.



(a)

(b)

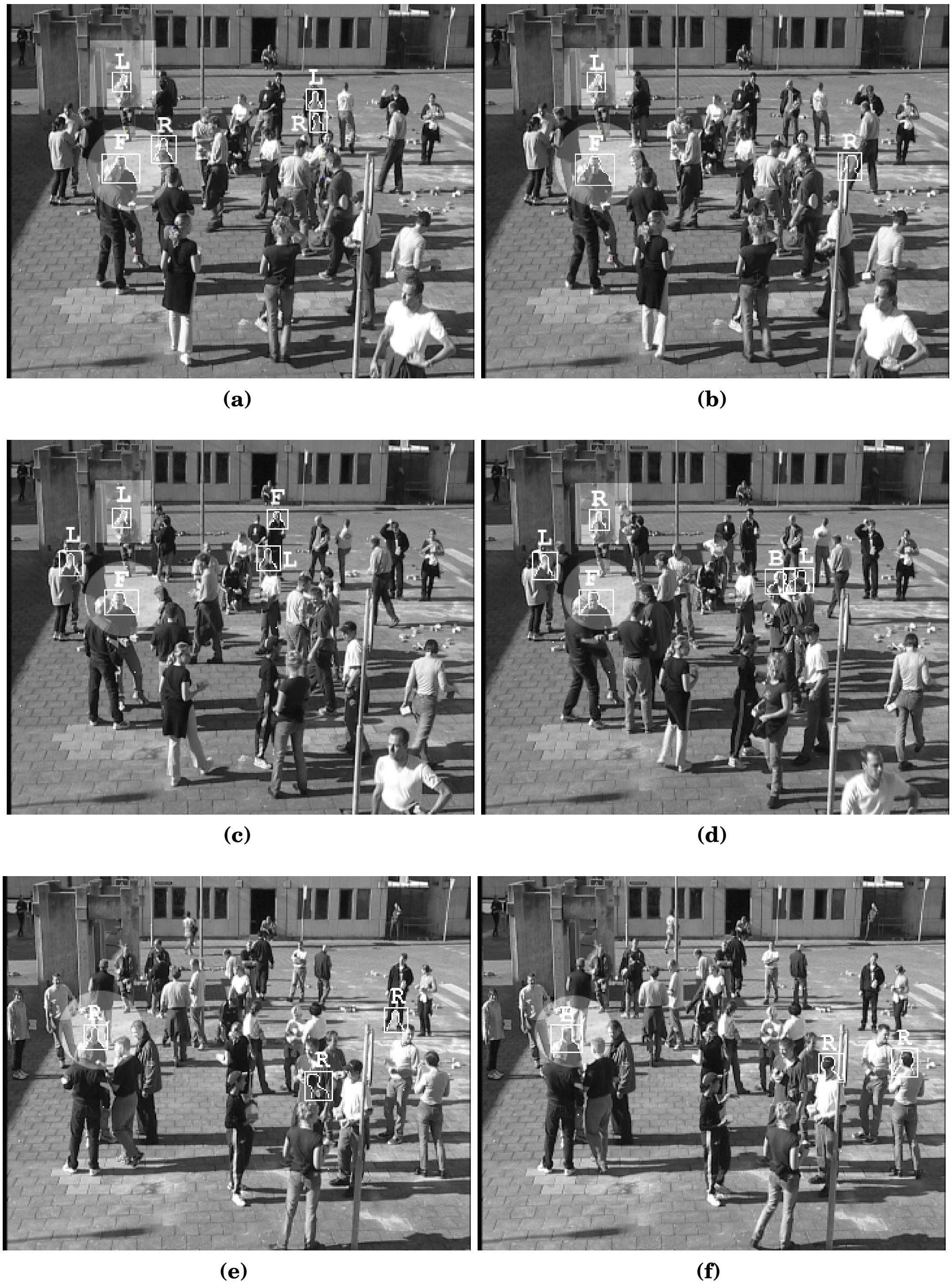
(c)

(d)

(e)

(f)

Figure 5.11: *Body facing estimation results for the TNO sequence. Detected locations are marked with a rectangle, together with the matching shape model. Alongside each rectangle the BFD is represented with R, L, B or F which are respectively,* ***R****ight,* ***L****eft,* ***B****ack or* ***F****ront*



(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.12: *Body facing estimation results for the TNO sequence continued. Detected locations are marked with a rectangle, together with the matching shape model. Alongside each rectangle the BFD is represented with R, L, B or F which are respectively, **R**ight, **L**eft, **B**ack or **F**ront*

A conclusion is the place where you got tired of thinking.

Martin H. Fischer

6 Conclusion

In this chapter we conclude the work done in relation to the goal as described in chapter 1. We were able to successfully estimate the camera parameters with only limited knowledge about the camera used. Using texture classification we were able to enhance the detection of people in complex dynamic scenes using a shape-based detection method called the *Chamfer System*. It turned out however that, estimating the actual BFD is a hard task to achieve with texture classification alone. Preliminary results have shown that a combination of shape model labeling and texture classification to estimate the BFD looks promising. Further research involves accommodating shape models to actual camera viewpoint, use of wavelet features for classification, incorporation of temporal information (tracking), auto-calibration and detecting people interacting.

6.1 Camera calibration

Using the method described by Lv et al. [19] and assuming off-the-shelf hardware was used, we were able to successfully calculate the camera parameters of the camera used in the TNO test sequence. We took an average digital camcorder with a CCD imaging plane as our camera model. This allowed us to constrain the camera parameters sufficiently to calculate the camera parameters. Instead of using a walking human, as is used by Lv et al. [19], we manually selected straight lines defining the three vanishing points. By using an initial guess for people's length and analyzing the calibration results we were able to interactively estimate the camera parameters.

6.2 Detecting people

We have extended the detection method developed by Gavrila and Philomin [12] with texture classification to reduce the number of false positives. We used the camera parameters to set the ground plane constraint which limits the image regions a particular shape model can match. This reduced the amount of false positives considerably. To verify the detected image regions given by the Chamfer System a neural network using local receptive fields was trained to discriminate between images of people and scenery. We achieved a performance of 95% true positive and 15% false positive rate for the neural network to discriminate people from background.

6.3 Estimating the Body Facing Direction

We have used a classification technique based on local processing as described by Wöhler and Anlauf [33] to discriminate between four different facing directions. The results have shown that discriminating between a frontal and rear view and a left and right view works well. We've achieved a performance for the ***front versus back*** classifier of 74% true positive and 17% false positive classification. For the ***left versus right*** classifier a performance was achieved of 71% true positive and 22% false positive classification. The classifiers for the other facing directions performed far less and were found to be incapable to determine the corresponding facing direction. We found that the *facial skin and hair contrast* feature is most apparent for the ***back versus front*** classifier. Considering the ***left versus right*** classifier we noticed that the *head position relative the body* is the feature most distinctive for the two classes.

The results have shown that our initial proposal of using a hierarchy of networks as shown in figure 4.3 to estimate the body facing direction of people is not possible. We think that the most apparent feature for discriminating a side from a front/back view is the *shoulder width*. However due to scaling of the input patterns for the classifiers this feature is discarded. Therefore we manually selected a few shape models, used by the detection method, representing a front/back or a side view and labeled these to the corresponding BFD. We then used the labeling of the model to determine if the ***front versus back*** or ***left versus right*** classifier should be used. A limited analysis of the results show that the estimation of the BFD depends mostly on the detection accuracy and that the combination of shape model labeling and texture classification looks promising.

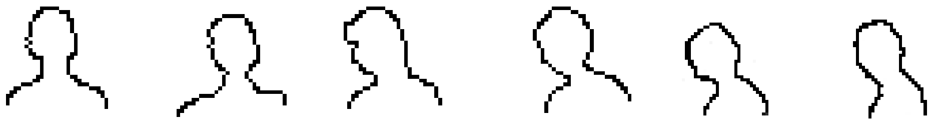


Figure 6.1: *Contours of a human's upper body rotating* 90°

6.4 Further work

Accommodating shape models to viewpoint changes

One of the problems of the current system is that the shape models used do not represent people viewed from a high viewpoint under a certain angle well. This requires the correlation thresholds to be set relatively high which results in an increasing amount of false positives. In figure 6.1 a couple of contours are shown of a woman's upper body. From these we can see that the shoulders get lowered or raised when moving towards to or away from the camera. These changes do not occur when viewing a person standing parallel to the camera (i. e. camera height at approximately the average height of people's chests). To allow for lower correlation thresholds the models should be adopted to the viewpoint changes. This would allow for more accurate matching and less false positives. It would also improve the BFD estimation based on labeling of the shape models.

Wavelets

Instead of using receptive fields, Viola and Jones [30] and Papageorgiou and Poggio [23] use Haar wavelets to extract relevant features from imagery. The method developed by Viola and Jones [30] can search an image for objects of interest very quickly. They use an efficient preprocessing method called the ”integral” image which allows the features to be computed very quickly. The method selects those features that are critical for the task at hand. Hence minimizing the amount of features needed during detection. By using a ”cascade” of classifiers, simple classifiers are used to quickly discard background regions while more complex classifiers are used to analyze more thoroughly interesting image regions. The method be applicable to detect the BFD without the need for a separate detection method. This might result in higher detection speed.

Tracking of people

Adjusting the shape models to better represent the possible shapes of people of course will make the system more robust. However, tracking may help when occasionally a shape occurs for which no shape model is present. This also may help the BFD estimation as it is most unlikely, depending on the frame rate of course, that in one frame a person is facing left and in the next facing right. By keeping a short history of previously detected locations and BFDs the system's performance might be significantly improved.

Labeling shape models representing a left or right view

People naturally tend to lean their head forward. This feature may indicate if a person is facing left or right and is apparent from a person's shape (see figure

6.1). Hence, labeling the side view shape models as a left or right view may help the ***left versus right*** classifier to reduce the number of false positives (i. e. left classified as right and vice versa).

Auto-calibration

As humans are generally the object of interest in surveillance systems, Lv et al. [19] uses a walking human to estimate the vanishing points. A walking human is supposed to be standing upright, by detecting a person and estimating its principal axis they obtain a fair measure for detecting the vertical vanishing point. The other two vanishing points are located by determining the head and feet locations and from these the line equations to calculate the horizon line.

Although Lv et al. [19] uses a background subtraction technique and analyzes the blob representing a human being, shape models might be incorporated with feat and head location information and be used to estimate the horizon line and the vertical vanishing point.

People interacting

One of the purposes of estimating a person's facing direction is that this can be used to detect a possible interaction between two or more people. This can be used to detect suspicious behavior, e. g. an individual wandering around a group of people. Further applications might be violence detection as described by Datta et al. [6], where it is crucial to pinpoint both a victim and a mugger. This is where the BFD estimation can help, as generally, the mugger is facing his or her victim.

Bibliography

[1] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles, and H.C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *International Joint Conference on Artificial Intelligence*, pages 659–663, 1977.

[2] V. Cantoni, L. Lombardi, M. Porta, and N. Sicard. Vanishing point detection: Representation analysis and new approaches. In *International Conference on Image Analysis and Processing*, 2001.

[3] B. Caprile and V. Torre. Using vanishing points for camera calibration. In *International Journal of Computer Vision*, volume 4, pages 127–140, March 1990.

[4] R. Cipolla, T. Drummond, and D. Robertson. Camera calibration from vanishing points in images of architectural scenes. In *Proc. British Machine Vision Conference*, volume 2, pages 382–391, September 1999.

[5] Antonio Criminisi, Ian D. Reid, and Andrew Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000. URL `citeseer.ist.psu.edu/article/criminisi99single.html`.

[6] Ankur Datta, Mubarak Shah, and Niels Da Vitoria Lobo. Person-on-person violence detection in video data. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 1, page 10433, August 2002.

[7] Shimon Edelman. *Representation and Recognition in Vision*. Bradford Books, 1999.

[8] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT PressCambridge, Massachusetts, London, England, 1993., 1993.

[9] Kunihiko Fukushima. A neural network for visual pattern recognition. *Computer*, 21(3):65–75, 1988. ISSN 0018-9162.

[10] D. M. Gavrila. The visual analysis of human movement: A survey. In *Computer Vision and Image Understanding*, volume 73, pages 82–98, 1999.

[11] D. M. Gavrila. Sensor-based pedestrian protection. *IEEE Intelligent Systems*, 16(6):77–81, 2001.

[12] D. M. Gavrila and V. Philomin. Real-time object detection for smart vehicles. In *Proc. of IEEE International Conference on Computer Vision*, pages 87–93, Kerkyra, Greece, 1999.

[13] I. Haritaoglu, D. Harwood, and L. Davis. W4: Who, when, where, what: A real time system for detecting and? tracking people. In *Third Face and Gesture Recognition Conference*, pages 222–227, 1998.

[14] T. Horprasert, Y. Yacoob, and L. S. Davis. Computing 3-d head orientation from a monocular image sequence. In *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, page 242. IEEE Computer Society, 1996. ISBN 0-8186-7713-9.

[15] Takahiro Ishikawa, Simon Baker, Iain Matthews, and Takeo Kanade. Passive driver gaze tracking with active appearance models. In *Proceedings of the 11th World Congress on Intelligent Transportation Systems*, October 2004.

[16] Joss Knight, Andrew Zisserman, and Ian Reid. Linear auto-calibration for ground plane motion. *Conference on Computer Vision and Pattern Recognition*, 1:503–510, June 2003.

[17] D. Liebowitz. *Camera Calibration and Reconstruction of Geometry from Images*. PhD thesis, University of Oxford, 2001.

[18] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proc. EuroGraphics*, volume 18, pages 39–50, 1999.

[19] Fengjun Lv, Tao Zhao, and Ram Nevatia. Self-calibration of a camera from video of a walking human. In *International Conference on Pattern Recognition*, pages 562–567, 2002.

[20] G.F. McLean and D. Kotturi. Vanishing point detection by line clustering. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 17, pages 1090–1095, November 1995.

[21] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *European Conference on Computer Vision LNCS 2352*, volume 3, pages 666–680, 2002.

[22] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Proc. Computer Vision and Pattern Recognitio*, pages 193–199, Puerto Rico, June 1997.

[23] Constantine Papageorgiou and Tomaso Poggio. A pattern classification approach to dynamical object detection. *International Conference on Computer Vision*, 2:1223–1229, September 1999.

[24] Seong Ho Park, Jin Mo Goo, and Chan-Hee Jo. Receiver operating characteristic (roc) curve: Practical review for radiologists. *Korean Journal of Radiology*, 5(1):11–18, March 2004. URL `http://www.kjronline.org/abstract/files/v05n01011.pdf`.

[25] John C. Russ. *The Image Processing Handbook*. CRC Press, 4th edition, 2002.

[26] R. Stiefelhagen, J. Yang, and A. Waibel. A model-based gaze tracking system. *Proceedings of IEEE International Joint Symposia on Intelligence and Systems*, pages 304 – 310, 1996.

[27] Rainer Stiefelhagen, Michael Finke, Jie Yang, and Alex Waibel. From gaze to focus of attention. In *Visual Information and Information Systems*, pages 761–768, 1999. URL `citeseer.ist.psu.edu/stiefelhagen98from.html`.

[28] William Bialek Thomas L. Adelman and Robert M. Olberg. The information content of receptive fields. *Neuron*, 40(13):823–833, 2003.

[29] Roger Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of IEEE Coonference on Computer Vision and Pattern Recognition*, pages 364–374, Miami Beach, Florida, 1986.

[30] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004. ISSN 0920-5691.

[31] Andrew Webb. *Statistical Pattern Recognition*. John Wiley & Sons, 2002.

[32] C. Wöhler and J.K. Anlauf. A time delay neural network algorithm for estimating image-pattern shape and motion. *Image and Vision Computing*, 17:281–294, 1999.

[33] C. Wöhler and J.K. Anlauf. Real-time object recognition on image sequences with adaptable time delay neural network algorithm - applications for autonomous vehicles. *Image and Vision Computing*, 19(9-10):593–618, 2001.

[34] C. Wöhler, U. Kreßel, and J. K. Anlauf. Pedestrian recognition by classification of image sequences global approaches vs. local spatio-temporal processing. *Proceedings of the 15th International Conference on Pattern Recognition*, 2:540–544, 2000.

[35] C. Wöhler, U. Kreßel, and J. Schürmann. Dimensionality reduction by local processing. *European Symposium on Artificial Neural Networks*, pages 237–244, 1999.

[36] Y. Wu and K. Toyama. Wide-range, person- and illumination-insensitive head orientation estimation. *Proceedings of International Conference on Face and Gesture Recognition*, pages 183–188, March 2000. URL `citeseer.ist.psu.edu/396012.html`.

[37] Semir Zeki. *A Vision of the Brain*. Blackwell Scientific Publications, 1993.

A
Network Architecture

The overall structure of the used neural network is shown in figure 4.2. It is an adjusted version of the network as described by Wöhler et al. [35]. They use a time delay to detect the gait pattern of a walking human. However, we are not interested in detecting patterns over time and therefore removed the temporal aspect.

A set of weight factors $\{r^s_{mn}\}$ is assigned to the layer 2 neuron in branch s located at position ij. This means that for each local receptive field connected to the same branch s the weight factors $\{r^s_{mn}\}$ are the same. Hence in figure 4.2 there are only two sets of weight factors as we have two separate branches. These branches act as filters for the underlying receptive fields and are given by the activations ξ^s_{ij} of the neurons in layer 2

$$\begin{aligned} x^s_{ij} &= \sum_{n=1}^{R_y} \sum_{m=1}^{R_x} r^s_{mn} B_{D_{x(i-1)+m}, D_y(j-1)+n} \\ \xi^s_{ij} &= g_2\left(x^s_{ij} - \theta^s\right) \end{aligned} \tag{A.1}$$

where g_2 is the standard sigmoid function $g_2(x) = \tanh(x)$, θ^s is the threshold value of the activation of the neurons in branch s, D_x and D_y are the distance between the centers of two neighboring receptive fields and R_y and R_x denote the spatial extensions of the receptive fields. In the network shown in figure 4.2 R_x and R_y are equal resulting in squared sized receptive fields. The value of $D_x = D_y = 2$ results in an overlap of one pixel between neighboring receptive fields.

The neurons in layer 3 are fully connected to the neurons in layer 2 with activations ω_k

$$\begin{aligned} h_k &= \sum_{s=1}^{N_{RF}} \sum_{j=1}^{S_y^{(2)}} \sum_{i=1}^{S_x^{(2)}} v^s_{ijk} \xi^s_{ij} \\ \omega_k &= g_3\left(h_k\right) \end{aligned} \tag{A.2}$$

where $g_3(x) = \tanh(x)$, N_{RF} the number of branches, $S_y^{(2)}$ and $S_x^{(2)}$ the number of neurons in layer 2 in the y and x direction respectively. Here we consider only $K = 2$ classes, in which the output is evaluated such that if $q\omega_1 > \omega_2$ the input image belongs to class 1, otherwise to class 2, where q is a threshold evaluated using the test set to determine the value at which the classifier performs best (see chapter 5, section 5.2, ***Receiver Operating Characteristic curves***).

The number of neurons in layer 3 depends on the number of classes we wish to discriminate. The amount of neurons in layer 2 depend on the receptive field configurations and equals

$$\begin{aligned} S_x^{(2)} &= 1 + \left\lfloor \frac{S_x - R_x}{D_x} \right\rfloor \\ S_y^{(2)} &= 1 + \left\lfloor \frac{S_y - R_y}{D_y} \right\rfloor \end{aligned} \tag{A.3}$$

where $\lfloor x \rfloor$ means the largest integer smaller or equal than the real number x.

Memory usage and computational complexity

The amount of memory needed to store the network structure is given by the amount of independent weight parameters and equals

$$n_w = R_x R_y N_{RF} + N_{RF} + S_x^{(2)} S_y^{(2)} N_{RF} K \tag{A.4}$$

One constraint of the system is that it can analyze a scene within limited time (i. e. perform in real-time), to measure the time needed to perform the classification the amount of floating-point operations is given by

$$F = F_{\text{flop}} + F_{\tanh} \tag{A.5}$$

where

$$F_{\text{flop}} = R_x R_y N_{RF} S_x^{(2)} S_y^{(2)} + S_x^{(2)} S_y^{(2)} N_{RF} K \tag{A.6}$$

and

$$F_{\tanh} = z \left(N_{RF} S_x^{(2)} S_y^{(2)} \right) \tag{A.7}$$

Here z denotes the number of operations needed to perform one evaluation of the function $\tanh(x)$.

Training the network

The weights of the network are trained using a back-propagation-like online gradient descent rule. The error measure ϵ is given by

$$\epsilon = \frac{1}{2} \sum_{k=1}^{K} (\omega_k - \tau_k)^2 \quad \text{with} \quad \tau_k = \begin{cases} A & \text{if} \quad k = c \\ 0 & \text{if} \quad k \neq c \end{cases} \tag{A.8}$$

where the constant A is slightly lower than 1, τ_k is the desired output, ω_k the actual output value of the output neurons and c is the corresponding class of the input pattern.

The variation Δv^s_{abk} of weight parameter v^s_{abk} can be obtained from equations A.8 and A.2 and is given by

$$\Delta v^s_{abk} = -\eta_v \frac{\partial \epsilon}{\partial v^s_{abk}} = -\eta_v(\omega_k - \tau_k) g_3'(h_k)\xi^s_{ab} \tag{A.9}$$

where $-\eta_v$ is the learning rate which is taken inversely proportional to the fan-in of the neurons to which the corresponding weights are connected, thus

$$-\eta_v = \frac{C_v}{S_x^{(2)} S_y^{(2)}} \tag{A.10}$$

where C_v is a constant of order $\mathcal{O}(10^{-2})$.

Similar for r^s_{ab} we obtain

$$\begin{aligned} \Delta r^s_{ab} &= -\,\eta_r \frac{\partial \epsilon}{\partial r^s_{ab}} \\ &= -\,\eta_r \sum_{k=1}^{K} (\omega_k - \tau_k) g_3'(h_k) \\ &\quad \times \sum_{j=1}^{S_y^{(2)}} \sum_{i=1}^{S_x^{(2)}} v^s_{ijk} g'{}_2 \left(x^s_{ij} - \theta^s\right) B_{\{D_x(i-1)+a\}\{D_y(j-1)+b\}} \end{aligned} \tag{A.11}$$

where $-\eta_r$ was chosen proportional to $-\eta_v$

$$-\eta_r = \frac{C_r \eta_v}{R_x R_y} \tag{A.12}$$

where C_r is a constant of order $\mathcal{O}(10^{-3})$.

Now for the threshold θ^s we obtain

$$\begin{aligned} \theta^s &= -\eta_n \frac{\partial \epsilon}{\partial \theta^s} \\ &= -\eta_n \sum_{k=1}^{K} (\omega_k - \tau_k) g_3'(h_k) \sum_{j=1}^{S_y^{(2)}} \sum_{i=1}^{S_x^{(2)}} v^s_{ij} g_2' \left(x^s_{ij} - \theta^s\right) \end{aligned} \tag{A.13}$$

where $\eta_n \approx \eta_v$ and the weight parameters and threshold values are initialized by small random numbers of order $\mathcal{O}(10^{-6})$.

B
Creating a DT image

Sobel filtering

Sobel filtering consists of two image processing methods in one

Smoothing by applying the mask $[1, 2, 1]$

Derivation by applying the mask $[-1, 0, 1]$

These are applied in both horizontal and vertical directions. The two 3×3 Sobel kernels are then as follows

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{B.1}$$

These kernels are convoluted over the input image resulting in two Sobel direction images (i. e. vertical and horizontal). The magnitude image is obtained by

$$M_{sobel}[x][y] = \sqrt{\left(p_{horizontal}[x][y]\right)^2 + \left(p_{vertical}[x][y]\right)^2} \tag{B.2}$$

where p is the pixel intensity at location (x, y) for the horizontal or the vertical Sobel direction image. To calculate the direction of the largest gradient change the inverse tangents of the pixel-values in the Sobel direction image is taken

$$\phi_{sobel}[x][y] = \tan^{-1}\left(\frac{p_{horizontal}[x][y]}{p_{vertical}[x][y]}\right) \tag{B.3}$$

Applying the Chamfer mask

If we take p to be the pixel value, r to be the image column and c the image row, than at location (r, c) the new pixel value will be for the first part of the mask (as shown in figure 3.3)

$$p_{rc} = min\big(\{0 + p_{(r,c)}, 2 + p_{(r-1,c)}, 3 + p_{(r-1,c-1)}, 2 + p_{(r,c-1)}, 3 + p_{(r+1,c-1)}\}\big)$$

and for the second part

$$p_{rc} = min\big(\{0 + p_{(r,c)}, 2 + p_{(r+1,c)}, 3 + p_{(r+1,c+1)}, 2 + p_{(r,c+1)}, 3 + p_{(r-1,c+1)}\}\big)$$

C
Pinhole Camera Model

Lets say we have a three dimensional coordinate system with its origin at the center of projection (indicated by C in figure 2.1) and whose Z axis lies along the optical axis. This coordinate system is called the World Coordinate System (WCS).

A point w with world coordinates (X, Y, Z) will be imaged at some point m with image coordinates (x, y) in the image plane. These coordinates are with respect to a coordinate system with its origin at the intersection of the optical axis and the image plane, and with the u and v axes parallel to the X and Y axes. This coordinate system is called the Camera Coordinate System (CCS).

We assume a pinhole model for our camera, therefore the relationship between the two coordinate systems (u, v) and (X, Y, Z) is a simple transformation

$$u = \frac{fX}{Z} \qquad v = \frac{fY}{Z} \tag{C.1}$$

which, using homogeneous coordinates can be written as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{C.2}$$

The actual image coordinates (x, y) are defined with respect to an origin in the top left corner of the image plane,

$$x = u_0 + \frac{u}{\text{pixel width}} \qquad y = v_0 + \frac{v}{\text{pixel height}} \tag{C.3}$$

By combining equations C.3 and C.1 and multiplying by Z, the 3×4 matrix from equation C.2 can be written as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{\text{pixel width}} & 0 & u_0 \\ 0 & \frac{f}{\text{pixel height}} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{C.4}$$

Hence, there are five camera parameters, namely the focal length f, the pixel width, the pixel height and the principal point (u_0, v_0).

The projection matrix from equation C.4 describes the perspective projection process when the u and v axes are orthogonal (i. e. zero skew). When these are not orthogonal it has to be incorporated in the projection matrix.

We do not need to know the actual pixel width and height but only their ratio

$$\lambda = \frac{\text{pixel width}}{\text{pixel height}} \tag{C.5}$$

We can therefore rewrite equation C.4 to

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & s & u_0 \\ 0 & \lambda f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{C.6}$$

where s is a factor which is zero in the absence of skew and λ is the aspect ratio.

In general, the three dimensional world coordinates of a point will not be specified in a frame which has its origin at the center of projection and its Z axis alongside the optical axis. Some other, more convenient frame, will more likely be specified. For this, we have to include a change of coordinates from this other frame to the standard coordinate system.

$$\mathbf{P} = \mathbf{K} \cdot \mathbf{T}$$

where $\mathbf{K}$ is the calibration matrix from C.4 and $\mathbf{T}$ is a 4×4 homogeneous transformation matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0_3^\top & 1 \end{bmatrix} \tag{C.7}$$

where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ a 3×1 translation vector. The matrix $\mathbf{T}$ has six unknown variables, three for the orientation, and three for the translation of the camera, defining the *extrinsic* parameters. Matrix $\mathbf{K}$ has five unknown variables, defining the *intrinsic* parameters. This makes a total of eleven parameters to be estimated.

Abbreviations

BFD	Body Facing Direction	5
CCS	Camera Coordinate System	11
DT	Distance Transform	20
FNR	False Negative Rate	29
FPR	False Positive Rate	29
ICS	Image Coordinate System	11
MLP	Multilayer Perceptron	26
NN	Neural Network	8
ROI	Region Of Interest	23
TDNN	Time Delay Neural Network	28
TNO	Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek//Dutch Organization for practical scientific research	34
TNR	True Negative Rate	29
TPR	True Positive Rate	29
WCS	World Coordinate System	11