Robust Real-time Vision-based Human Detection and Tracking

Adrian Leu

Universität Bremen 2014

Robust Real-time Vision-based Human Detection and Tracking

Vom Fachbereich für Physik und Elektrotechnik

der Universität Bremen

zur Erlangung des akademischen Grades

Doktor–Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von

Dipl.-Ing. Adrian Leu

wohnhaft in Bremen

Referent:	Prof. Dr.-Ing. A. Gräser
Korreferent:	Prof. Dr.-Ing. U. Frese
Eingereicht am:	10. Oktober 2013
Tag des Promotionskolloquiums:	17. Juni 2014

Publication Series of the Institute of Automation
University of Bremen

Series 5-Nr.4

Adrian Leu

Robust Real-time Vision-based Human Detection and Tracking

D 46 (Diss. Universität Bremen)

Shaker Verlag
Aachen 2014

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Zugl.: Bremen, Univ., Diss., 2014

Publication Series of the INSTITUTE OF AUTOMATION, UNIVERSITY OF BREMEN:

1 Colloquium of Automation
2 Automation
3 Robotics
4 Control Theory
5 Image Processing
6 Virtual and Augmented Reality
7 Brain Computer Interface

Printed in Germany.

ISBN 978-3-8440-3046-4
ISSN 1861-5457

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen
Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9
Internet: www.shaker.de • e-mail: info@shaker.de

To my wife Roxana and my son Eric

Acknowledgements

First of all I would like to thank Prof. Gräser for offering me a research position as PhD student at the Institute of Automation (IAT) and for his valuable guidance as my thesis supervisor. Secondly I would like to thank Ms. Danijela Ristić-Durrant for her constant support and constructive feedback regarding my research and my publications. I would also like to thank prof. Frese for accepting to be second reviewer of my thesis, as well as prof. Anheier and prof. Garcia for accepting to be examiners.

Further I would like to thank Mr. Saravana Kumar Natarajan, Mr. Uwe Lange and Mr. Bashar Enjarini whom I shared an office with for many years for the valuable discussions and their valuable feedback regarding my research in general and my thesis in particular. Additionally I would like to thank Mr. Henning Kampe for reading my thesis and providing constructive feedback. I would also like to thank Dr. Matthias Spranger for his great support during the development of the gait analysis system, as well as everyone who participated in the performance evaluation of the system.

During my stay at the IAT I had the opportunity to supervise almost 20 students for their Master Projects or Master Theses. Out of them I would like to particularly thank Mr. Xiangpeng Liu and Mr. Angel Mota who both contributed to my research. I would also like to thank Ms. Ge Gao who is carrying on my work as part of her own research.

There are a few persons that were particularly important to me during my studies at the "Politehnica" University of Timişoara and who considerably contributed to my future career. I would like to thank Mr. Florin Drăgan and his team, as well as Prof. Radu Precup and Prof. Ştefan Preitl for supporting me and always giving me good advices. My thanks also go to Mr. Antonius Stanciu, who believed in me and who helped me to discover my talents and to improve my skills in the field of automation and computer science. I would like to thank one more person that marked me more than he is probably aware of: Mr. Adrian Mihăilescu. I strongly believe that he considerably contributed to my career as engineer with his lecture about digital logic.

Last but not least I would like to thank my wife Roxana and my son Eric for their constant support and their understanding, as well as my mother Silvia for always being there for me.

Bremen, August 2014 *Adrian Leu*

Kurzfassung

In den letzten zwei Jahrzehnten hat sich die Technik in unserer Gesellschaft immer weiter durchgesetzt. Mittlerweile gibt es überall Geräte die auf mehr oder weniger intuitiver Weise mit Menschen interagieren: zu Hause, am Arbeitsplatz, im Fahrzeug und überall dort wo sich Menschen aufhalten. Einige dieser Geräte verwenden Kameras um die Menschen zu beobachten oder mit ihnen zu interagieren. Nichts desto trotz ist es immer noch eine große Herausforderung, einer Maschine beizubringen, Menschen in Bildern und Videos zu erkennen und zu verfolgen. Die Gründe hierfür liegen in der unterschiedlichen Wahrnehmung der Menschen wegen ihrer Statur, ihrer Kleidung und ihren dynamischen Bewegungen.

Der Schwerpunkt der vorliegenden Arbeit ist die Entwicklung von zuverlässigen Algorithmen zur Echtzeiterkennung und –verfolgung von Menschen, sowohl in Innenräumen als auch im Freien. Um dieses Ziel zu erreichen wurden die Algorithmen für traditionelle, passive Kameras entwickelt, da sie in beiden Umgebungen zuverlässig funktionieren. Die im Rahmen der vorliegenden Arbeit neu entwickelten Ansätze zur visions-basierten Echtzeiterkennung und –verfolgung von Menschen werden anhand von drei Anwendungen verdeutlicht: Ganganalyse, Fußgänger-erkennung und Interaktion zwischen Mensch und Maschine. Diese drei Anwendungen haben gemeinsam, dass sie in Echtzeit Menschen erkennen und verfolgen sollen und dabei anwendungs-spezifische Daten im Bezug zum Menschen extrahieren müssen.

Um die rechenaufwendige Erkennung und Verfolgung von Menschen in Echtzeit durchführen zu können, werden neuartige hardware-spezifische Optimierungen der vorgeschlagenen Bildverarbeitungsalgorithmen vorgestellt. Hierfür werden GPU Implementationen für die Fußgängererkennung eingeführt, deren Bearbeitungszeiten mit denen von CPU und GPU Implementationen verglichen werden. Für die Interaktion zwischen Mensch und Maschine wird ein verteiltes Rechensystem eingeführt, welches die Echtzeitfunktionalität der Algorithmen ermöglicht.

Abstract

In the last couple of decades technology has made its way into our everyday lives including our homes, our offices and the vehicles we use for travelling. Many modern devices interact with humans in a more or less intuitive way and some of them use cameras for observing or interacting with humans. Nevertheless, teaching a machine to detect humans in an image or a video is a very difficult task. There are many aspects that contribute to the complexity of this task, such as the many variations in the humans' perceived appearances: their constitution, the clothes they wear and the dynamic nature of the activities performed by humans.

The focus of this thesis is on the development of reliable algorithms for real-time vision-based human detection and tracking in indoor as well as in outdoor applications. In order to achieve this, the algorithms presented in this thesis were developed for traditional passive cameras, as they perform well in both environments. The novel approaches for vision-based human detection and tracking are presented for three different applications: gait analysis, pedestrian detection and human-robot interaction. All these approaches have in common the need for real-time human detection and tracking in video sequences in order to extract application-specific data regarding the tracked human.

In order to cope with human detection and tracking as a computational expensive task, novel hardware-specific optimizations of the proposed image processing algorithms are presented, that allow the algorithms to run in real-time. For this purpose GPU implementations are presented for pedestrian detection and the processing times are compared to CPU and FPGA implementations. In the case of human-robot interaction the real-time human tracking is achieved by using distributed computing.

Table of contents

1

Introduction

One of the fundamental prerequisites of human interaction is at the same time one of the most significant challenges in computer vision. Recognizing other humans comes natural to all of us. Teaching a machine to detect humans in an image or a video is on the other hand a very difficult task. There are many aspects that contribute to the complexity of this task, such as the many variations in the humans' perceived appearances: their constitution, the clothes they wear and the dynamic nature of the activities performed by humans.

In the last decades a vast research field has emerged, where researchers strive to find reliable solutions for the complex problem of vision-based human detection and tracking. The developed applications include, among others, surveillance [70], pedestrian detection for automotive collision warning systems [63], clinical gait analysis [42] and human-machine interfaces including human-robot interaction based, for example, on hand gesture recognition [68][69]. Due to the complex nature of vision-based human detection and tracking, algorithms attempting to work reliably for multiple applications tend to be complex and time consuming. An example of such an algorithm is presented in [40], where 30-40 seconds are required for estimating the 3D human pose using images obtained from three different camera views. The reliable pose extraction of the above mentioned method enables it to be used for a wide range

of applications, but the required processing time makes it unsuitable for on-line applications.

A recent attempt to enable the on-line use of vision-based human detection and tracking algorithms was the development of so-called RGBD cameras such as the Microsoft Kinect [76] and the Asus Xtion PRO LIVE [77]. These cameras provide a depth image together with the color image of the scene. Open-source libraries such as OpenNI [78] use this additional information provided by the RGBD cameras to enable on-line human detection and tracking. As shown in section 2.7 of this thesis, RGBD cameras project an infrared light pattern, which cannot be detected by the cameras in direct sunlight, therefore limiting the range of applications to indoor applications. An example of such an indoor application is the clinical gait analysis. However, the OpenNI library cannot be used on its own to extract the gait features required for clinical gait analysis. Therefore, in section 4.4 it is explained how the methods introduced in section 4.3 for traditional cameras can be applied to extract gait features from images provided by RGBD cameras.

For outdoor applications a typical example is pedestrian detection. Most state-of-the-art algorithms developed for pedestrian detection are suitable for on-line use. For example in [41] the processing time for one frame is about 100ms and is reduced to 20ms by reducing the image resolution. Generally, state-of-the-art systems for pedestrian detection have a processing time for one frame of about 30-60ms [63]. However, as will be shown in Chapter 5 of this thesis, shorter processing times are required for reliable pedestrian detection with the purpose of collision warning for vehicle velocities higher than 50 km/h, corresponding to driving outside urban environments. Due to the fact that in a fixed time unit, longer distances are travelled as the vehicle speed increases, the environment should be imaged more often in order to maintain the same spatial resolution. For this purpose high-speed cameras are used in the application presented in Chapter 5.

Generally, in order to achieve real-time image processing, vision-based human detection and tracking algorithms are often adapted and optimized for implementation on specialized hardware platforms such as GPU (Graphical Processing Unit) and FPGA (Field Programmable Gate Array) [18]. In Chapter 5 such specialized hardware implementations are proposed for real-time processing of images obtained from high-speed cameras for the purpose of vision-based human detection and tracking. Another approach for achieving real-time image processing is presented in Chapter 6, where distributed computing is used for vision-based Human-Robot Interaction (HRI).

Main Contributions of this thesis

The focus of this thesis is on the development of reliable algorithms for real-time vision-based human detection and tracking in indoor as well as in outdoor applications. Therefore, the algorithms presented in this thesis were developed for traditional passive cameras, as they perform well in both environments. The novel approaches for vision-based human detection and tracking are presented for three different applications: gait analysis, pedestrian detection and human-robot interaction. All these approaches have in common the need for real-time human detection and tracking in video sequences in order to extract application-specific data regarding the tracked human. In order to cope with human detection and tracking as a computational expensive task, novel hardware-specific optimizations of the proposed image processing algorithms are presented, that allow the algorithms to run in real-time.

In summary, the main contributions of this thesis can be listed as follows:

- Novel methods for robust markerless vision-based clinical gait analysis [5][8] and gait feature extraction [7]. In particular novel methods are presented for robust closed-loop segmentation of subtracted images in section 3.1 and automatic extraction of gait parameters based on 3D reconstruction of virtual joints in section 4.3.3. The gait patterns of 40 subjects have been extracted using the proposed algorithm. A comparison between the healthy and the pathological gait patterns is presented in section 4.3.4.
- A novel disparity map segmentation method for detection and tracking of objects, including humans as particular type of objects, which is presented in section 3.2. This method can be used in various applications. Two possible applications are presented in this thesis: pedestrian detection for automotive collision warning systems [2][3][4] presented in Chapter 5 and human tracking for human-robot collaboration in hazardous environments, where a robotic platform has to follow a human co-worker [1], presented in Chapter 6. Even though the novel disparity map segmentation method was developed for human detection and tracking, due to its generic nature it has been successfully used for 3D object reconstruction in service robotics as well [6].
- Novel optimizations of image processing algorithms for human detection and tracking and adaptations to specific hardware platforms in order to enable real-time operation. In particular two applications are presented, for which the real-time capability was achieved either with the help of a GPU [9], as shown in Chapter 5 or by using distributed computing [1], as presented in Chapter 6.

Organization of the thesis

This thesis is organized in seven chapters. Chapter 3 presents the two novel algorithms for robust image segmentation. Starting from these segmentation algorithms, Chapters 4, 5 and 6 present novel algorithms for human detection and tracking in three different indoor as well as outdoor applications. The theoretical background of all these algorithms is given in Chapter 2.

Each of the three application chapters contains a general description of the developed vision-based algorithms followed by proposed optimizations as required for real-time image processing. The performance evaluation of each algorithm is presented within each chapter.

Chapter 7 summarizes the most important conclusions from all chapters and presents the outlook.

2

Vision-based object recognition

The main goal of this chapter is to provide the necessary theoretical background of vision-based object recognition to support the methods that are described in detail in the following chapters. Therefore, vision-based object recognition is introduced, followed by particularities that arise in vision-based human detection and tracking where humans are treated as particular types of objects.

Vision-based object recognition is widely used in many robotics and computer vision applications such as 3D reconstruction of objects for manipulation in service robotics. Even though various computer vision applications for object recognition have different goals, they include either all or some of the steps illustrated in Fig. 2.1.

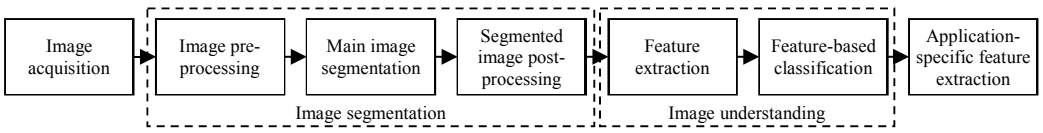


Fig. 2.1 Block diagram of a typical computer vision application for object recognition

Image acquisition is the first step in a vision-based object recognition application where a digital camera is used to capture images of the environment including the object of interest. Next the image is segmented, which generally results in simplifying the image content and preparing it for image understanding. *Image segmentation* can typically include up to three steps: image pre-processing, the main image segmentation

and segmented image post-processing. The goal of image pre-processing is to enhance the image quality or to reduce the region that needs to be segmented by selecting an object ROI (Region of Interest). After the appropriate image segmentation algorithm is applied, in the segmented image post-processing step the content of the segmented image can be filtered in order to exclude regions that do not contain objects of interest.

Image understanding contains a feature extraction and a feature-based classification step. Depending on the type of input image and on the previously applied segmentation algorithm, various features are extracted for potential objects, which are used for classification in order to detect the object or objects of interest. Once the objects of interest have been localized in the image, *application-specific features* are extracted for them, which represent the output of the image processing system.

If the above mentioned steps are repeated for a series of consecutive images of a video, also called frames, the application-specific features can be tracked in space and time. This means that besides the values of the features, also their positions in image coordinates are monitored from frame to frame.

Digital images

The main data types used in vision-based object recognition applications are digital images. A digital image, like the one shown in Fig. 2.2, is a discrete representation of a two-dimensional image as a finite set of integer values, called picture elements or pixels, which are organized in a fixed number of rows and columns.

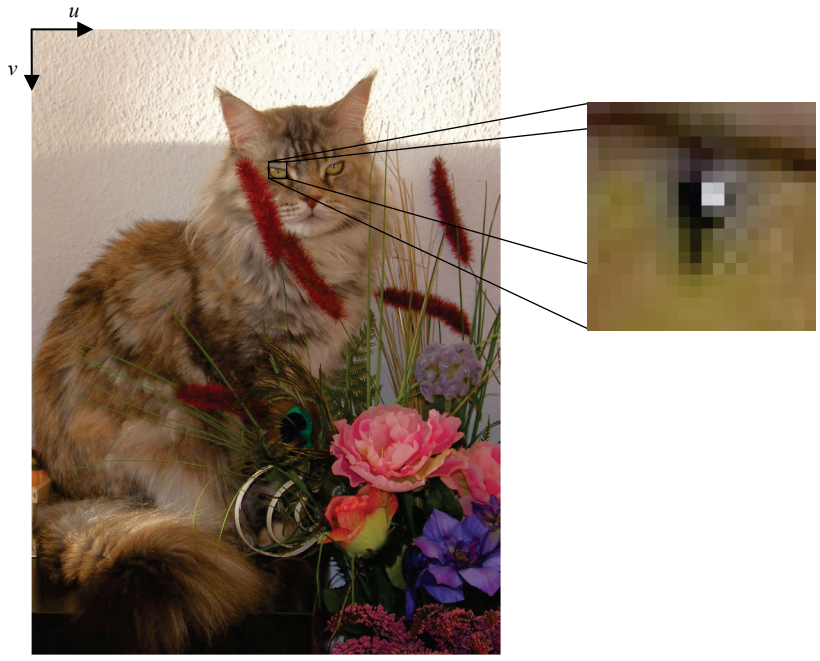


Fig. 2.2 Digital image example illustrating discrete pixel values and the image coordinate system

Pixels are the smallest individual elements in an image, which hold quantized values that represent the brightness of a given color at any specific point. Each pixel has two coordinates describing its location with respect to the image coordinate system. The origin of this coordinate system is usually in the upper left corner having the *u* axis parallel to the image rows and pointing to the right and the *v* axis parallel to the image columns and pointing down, as can be seen in Fig. 2.2. This convention for defining the image coordinate system is used throughout this thesis.

The magnified region in Fig. 2.2 illustrates the discrete nature of digital images. The individual pixels arranged in rows and columns can be clearly seen.

Color spaces

Several different color spaces are used for representing digital images. One of the most commonly used formats is the RGB color space. The abbreviation RGB comes from Red, Green, Blue, which name the three color channels used to store the information for each pixel. For each of these channels usually an 8-bit value is stored and the combination of all representable values composes the color space.

The RGB color space illustrated in Fig. 2.3 a) is a cube-shaped additive color space, which means that if for a pixel all color channels hold the smallest representable value (no color is added), the resulting color is black, while in the case of holding the highest possible value (all colors are added), the resulting color is white.

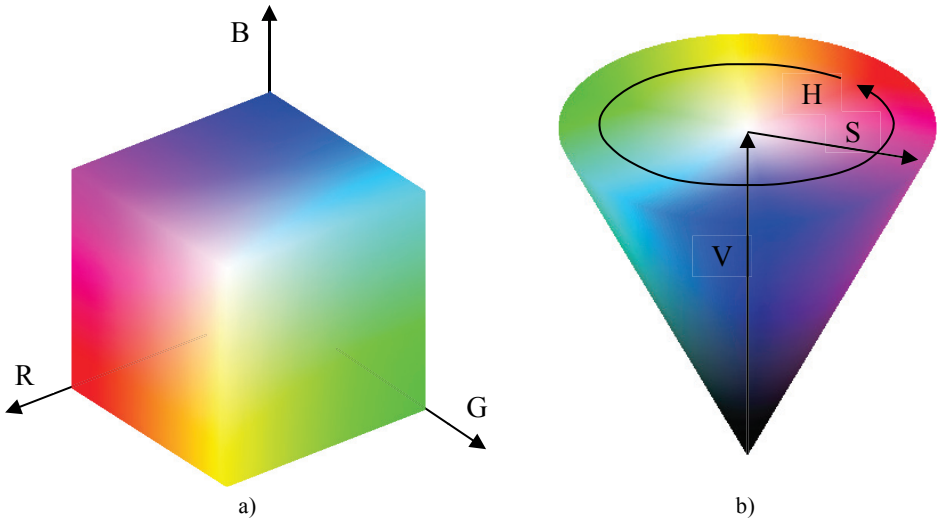


Fig. 2.3 Color representations: a) The RGB color space and b) The HSV color space

Another popular color space is the HSV (Hue, Saturation, Value) color space presented in Fig. 2.3 b), which has the shape of a cone and can be used for color segmentation. A third commonly used color space is YCbCr, which is the digital counterpart of the famous analog YUV color space used in television transmissions.

Besides these, in image processing gray-scale images are often used if only luminance information is required. The conversion from RGB to gray-scale is often performed using the following equation, which accounts for the perception of the brightness of different colors by the human eye:

$$Y = 0.299R + 0.587G + 0.114B \tag{2.1}$$

where Y represents the equivalent gray-scale pixel value of the perceived luminance and R, G and B represent the values of the three color channels of the color pixel.

Region of Interest (ROI)

In some situations only a sub-region of a digital image needs to be processed, in which case this region is often called Region of Interest (ROI) and is typically considered to be rectangular, as can be seen in Fig. 2.4. The selection of the ROI can be performed "manually" if throughout the application the same image region is used or it can be automatically adapted to the imaged scene by a pre-processing algorithm.

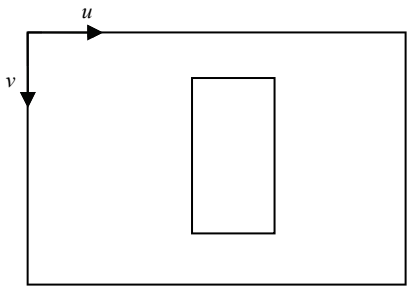


Fig. 2.4 Illustration of the concept of Region of Interest (ROI)

Reducing the region of the image to be processed has a beneficial influence on the required processing time, which is of crucial importance in real-time image processing applications. Therefore, fast pre-processing algorithms are able to reduce the overall time consumption by reducing the region that has to be processed by subsequent time-consuming algorithms.

2.1 Image acquisition

Vision-based algorithms involve the processing of images recorded by a digital camera. The term "camera" comes from the Latin "camera obscura", which was an early device developed for helping painters in creating realistic paintings by projecting the light reflected by objects from the real world onto a plane through an opening. This resulting projection was then used as a template for the painting.

Modern cameras use the same principle during image acquisition for projecting the light reflected by objects onto a surface, which is either a photographic film or a digital sensor. However the light does not travel anymore through a simple opening, but through various lenses, which allow more light to enter and to be properly focused on the image plane.

In the past couple of decades digital photography has gained huge interest due to reasonable prices of the devices, high quality of the images and practically costless image recording and storage. Hence the big focus on digital image processing. If a series of images is acquired, this series is referred to as a video consisting of multiple consecutive frames or images.

During image acquisition light enters through the camera lens and is focused on the image sensor. After a certain amount of time, called *exposure time*, either a mechanical shutter obscures the sensor so that no light can fall on it anymore or the image data is electronically read out very fast, making a mechanical shutter unnecessary. The exposure time influences the image brightness since a longer exposure time allows more light to be captured, leading to a brighter image. Other important parameters during image acquisition are the *aperture size* and *light sensitivity* of the sensor.

The term aperture size refers to the diameter of the opening through which the light travels to the sensor while the shutter is open. A bigger diameter is desirable in most situations as it enables the camera to work well in low lightning conditions. However, a smaller diameter allows less light to pass and can be useful in very bright conditions if the minimum shutter speed is reached. Also, through a smaller opening the incident angle of the light rays is almost the same for objects which are close to the camera and for the ones further away, allowing therefore a larger part of the scene to appear sharp, an effect also known as a wider depth of field.

The sensor's light sensitivity describes the minimum amount of light the sensor needs in order to register a difference from complete darkness. A sensitive sensor can perform well in low light condition but usually at higher cost and not without a good lens. Regular sensors produce darker images in low light conditions, which are electronically amplified, including the noise. This amplified noise is one of the reasons why in a series of images taken in low light conditions, the outcome of applying the same image processing algorithms can differ from frame to frame even if the images look identical to the human eye.

Color digital images are mostly captured by placing a so-called Bayer filter [13] over the image sensor, which usually only captures light intensity. The Bayer filter is a

mosaic filter which allows for each individual pixel only the red, the green or the blue parts of the light to reach the sensor surface. Afterwards the color image can be calculated using so-called debayering algorithms. However, in applications with low light conditions or when a high frame rate is required, sometimes gray-scale images are directly captured by the camera and the Bayer filter is omitted in order to increase the amount of light that reaches the sensor. Such a high-speed application is presented in Chapter 5.

Monocular camera

Even though modern cameras use complex optical lenses for focusing the incident light, the simple pinhole camera model [12] still applies to these cameras, with the difference that additional parameters are introduced, which model the distortions introduced by the lenses. A model of the pinhole camera can be seen in Fig. 2.5.

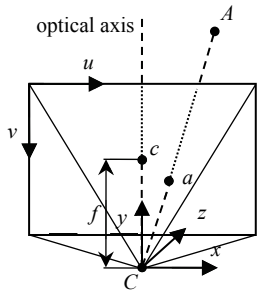


Fig. 2.5 Coordinate systems of a monocular camera

As can be seen in Fig. 2.5, a camera has two coordinate systems: an image coordinate system *(u, v)* having the origin in the upper left corner of the image and a camera coordinate system *(x, y, z)* having the origin in the optical center of the camera *C*. The distance between the center of the image plane and the optical center is called focal length *f*. The center of the image plane is the projection of the optical camera center on the image plane and is called principle point *c*. For any world point *A*, a projection point *a* on the image plane exists at the intersection of the ray *AC* with the image plane. However, as the image sensor surface has a limited size, projected points that lie outside the sensor surface will not appear on the resulting image.

The relationship between the two coordinate systems of the camera is characterized by the so-called intrinsic camera parameters:

$$K = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where (c_u, c_v) are the image coordinates of the principal point c and f_u and f_v represent the number of pixels that would fit in the focal length f and are measured in pixels:

$$f_u = \frac{f}{px_{width}}, f_v = \frac{f}{px_{height}} \tag{2.3}$$

where px_{width} and px_{height} represent the width and the height of a pixel, expressed in meters. Therefore, if the image pixels are rectangular, f_u and f_v are equal to each other.

Usually, the origin of the world coordinate system is considered to be in the optical center of the camera. However, if the origin of the world coordinate system is at a different location the so-called extrinsic camera parameters characterize the rotation and translation of the camera frame with respect to the world frame:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.4}$$

where R is the Direction Cosine Matrix (DCM) also known as rotation matrix and t is the translation vector giving the coordinates of the origin of the camera frame in the world frame. If the frame of the world coordinate system is identical to the frame of the camera, the rotation matrix is an identity matrix and the translation vector is the zero vector.

Generally, the two-dimensional (2D) projection $a(u, v)$ on the image plane of a three-dimensional (3D) point $A(x, y, z)$ from the world is obtained through the following relationship:

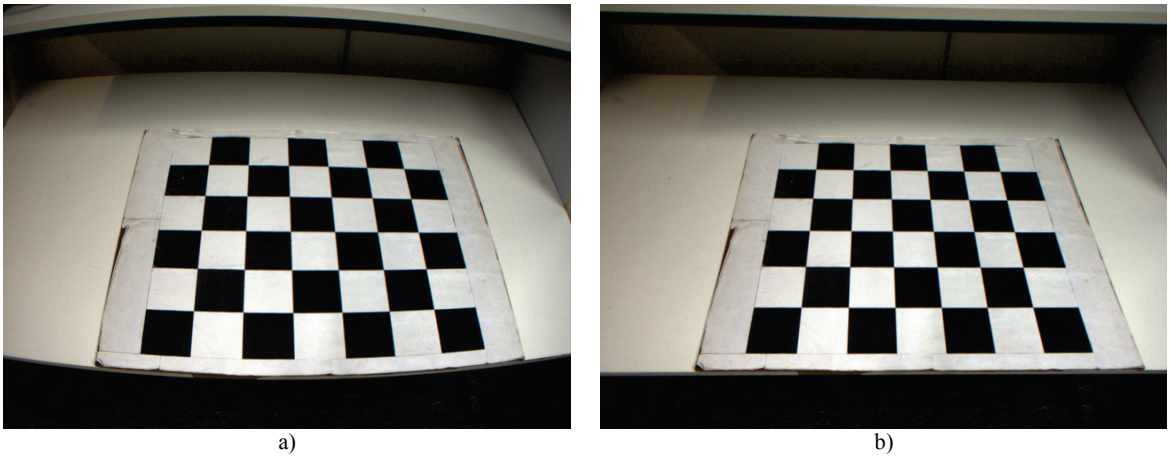
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \cdot \begin{bmatrix} x \\ v \\ z \\ 1 \end{bmatrix} \tag{2.5}$$

where P is the so-called *projection matrix* obtained by multiplying the intrinsic parameters matrix K with the extrinsic parameters matrix $[R|t]$:

$$P = K \cdot [R \mid t] = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{2.6}$$

Additionally to the intrinsic and extrinsic camera parameters, four more parameters are often used. Two of them, namely k_1 and k_2, model the radial distortion, while the other two, p_1 and p_2, model the tangential distortion coefficients caused by the lens.

Based on these four distortion coefficients the images can be transformed so that straight lines, which appear distorted in the original image, appear straight in the resulting image. This process is also called undistortion. The effect of undistortion can be seen in Fig. 2.6.



a) b)

Fig. 2.6 Effect of undistortion: a) Original distorted image and b) Resulting image after undistortion

Camera calibration

Many computer vision algorithms require a precise estimation of the intrinsic and extrinsic camera parameters. Even though the intrinsic camera parameters are often given in the camera datasheet, due to imperfections in the construction process, they might not be accurate enough, as required for some vision-based applications. The extrinsic parameters are always application dependent, as the origin of the world coordinate system might be different for each scenario.

The estimation of the intrinsic and extrinsic camera parameters is done by the so-called *camera calibration*, a process in which a known pattern is imaged multiple times from different viewpoints and based on the pattern's geometrical properties a system of equations is solved in order to obtain the camera parameters. A widely used algorithm for camera calibration is Tsai's calibration method [43].

One of the most common patterns used for camera calibration is a chessboard with alternating black and white squares, as illustrated in Fig. 2.7. Although the intrinsic camera parameters are determined only based on the ratio of the square's sides, for estimating the extrinsic parameters the exact size of the chessboard is required. Further, the pattern should not be symmetrical along both the horizontal and the

vertical axes, as in this situation the origin of the world coordinate system located on the board could not be uniquely identified. Asymmetrical chessboard patterns have an even number of rows and an odd number of columns or vice versa.

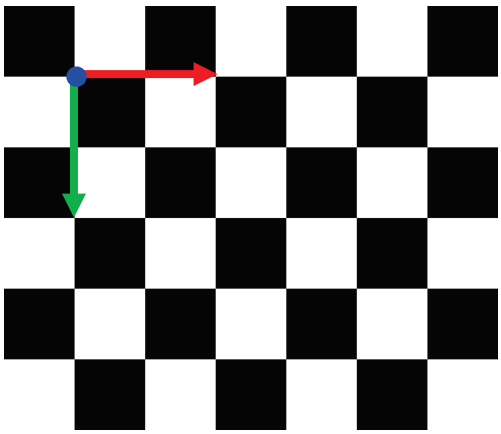


Fig. 2.7 Example of chessboard pattern used for camera calibration

For a chessboard calibration pattern the world origin is usually chosen to be on the chessboard plane and two of the axes are along the chessboard rows and the columns while the third axis is normal to the plane defined by the chessboard, as illustrated in Fig. 2.7. Knowing the size of the chessboard squares, the 3D location of the corners is computed with respect to the world origin. Afterwards the image points of the chessboard corners are detected and the 2D-3D corresponding points are used for solving a system of equations which give the camera parameters [43].

After estimating the intrinsic and extrinsic camera parameters it is important to test whether the accuracy of camera calibration is satisfactory, as the camera parameters play an important role in subsequent image processing algorithms. This is done by projecting the 3D chessboard corners onto the image as shown in (2.5) by using the projection matrix (2.6). Ideally the projected chessboard corners perfectly overlay on the detected chessboard corners. The distance for any pair of projected-detected corners is denoted as corner error. By computing statistical measures like average and standard deviation on the series of corner errors, the calibration accuracy is described.

2.2 Image segmentation

The goal of image segmentation is to simplify and/or change the representation of an image, so that the content appears more meaningful and is easier to analyze [12]. In other words, an image of a complex scene is transformed during the segmentation process in order to enhance the application-specific, relevant features and to suppress unimportant image details. Depending on the type of application and on the input

image there are many different segmentation methods, such as threshold-based binary segmentation, color segmentation, etc.

Many of the existing segmentation methods require some form of image pre-processing in order to prepare the image for segmentation. Fig. 2.8 illustrates different types of segmentation, having different types of images as input.

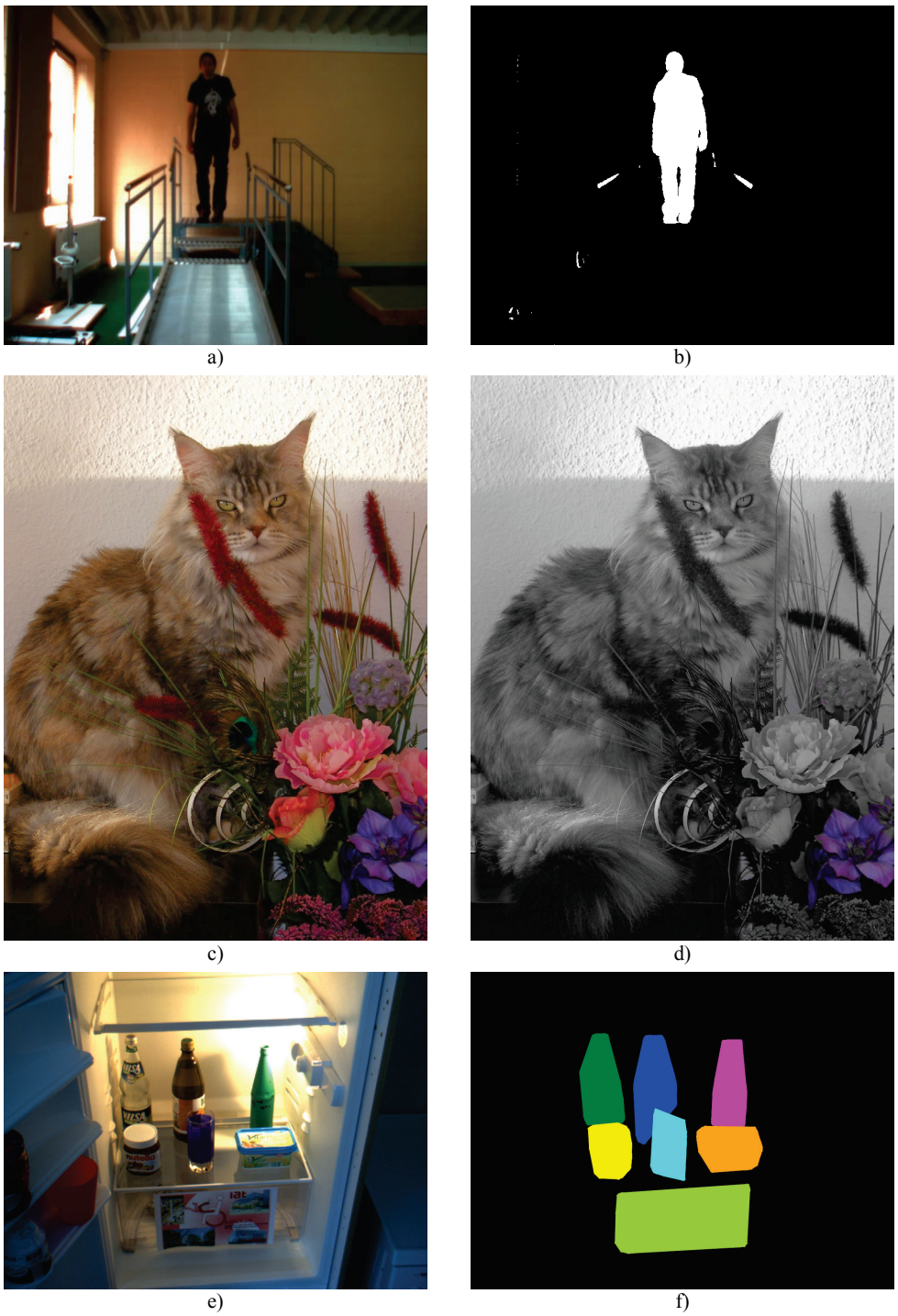


Fig. 2.8 Original images and segmented images: a), b) Binary segmentation of background subtracted image, c), d) Color segmentation using the Hue channel of the HSV color space and e), f) Disparity map segmentation using connected components labeling and after convex hull fitting

Fig. 2.8 a) and b) show an example of separation of foreground objects from the background by using background subtraction and binary segmentation, as will be shown in section 3.1. Fig. 2.8 c) and d) show a typical application of color segmentation, in which objects of a specific color are separated from the rest of the scene, using the Hue channel after converting the RGB image to the HSV color model. Fig. 2.8 e) and f) show an application for which the objects in the scene are separated from each other and from the background by computing a disparity map and segmenting it, as will be shown in section 3.2.

2.3 Feature extraction

Based on the application, certain features might be needed to describe the shape of the segmented objects for the purpose of classifying the object as belonging to a particular object class. Some of the most popular features are: size, volume, length of boundary and size of the bounding rectangle. Additionally, there are shape descriptors which are invariant to translation, rotation and scaling. A set of such invariant shape descriptors, that are widely used, are the so-called Hu moments [16]. Out of the seven Hu moments only the first three are used in this thesis as they provided the best results for object classification:

$$\begin{cases} hu_1 = \eta_{20} + \eta_{02} \\ hu_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\ hu_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \end{cases} \tag{2.7}$$

where η_{pq} denote the normalized central moments:

$$\eta_{pq} = \sum_u \sum_v (u - \overline{u_c})^p (v - \overline{v_c})^q I(u,v) \tag{2.8}$$

where p and q represent the order of the moments, u and v are image coordinates and $I(u, v)$ represents the pixel value at the given coordinates. These invariant moments are very useful for object classification, as they reduce the amount of data to be processed by describing the shape of the objects.

2.4 Feature-based object classification

Most applications require some form of classification for inferring to which class an object belongs to. Therefore, after segmenting the image, depending on the application, some or all of the features described above can be extracted for each

segmented object. These features are then used for object classification. An overview of feature-based classification is given in Fig. 2.9.

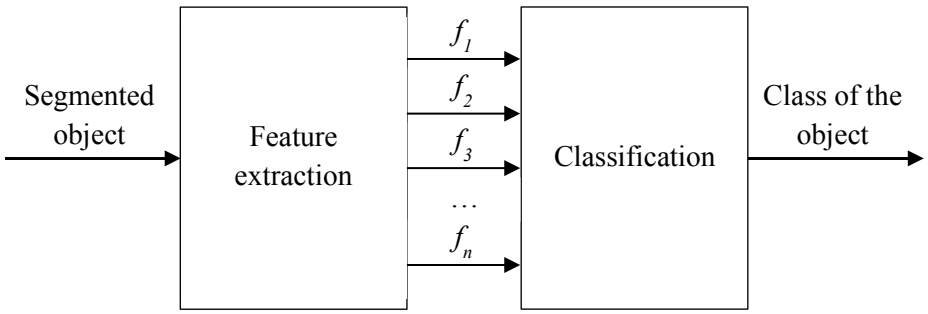


Fig. 2.9 Illustration of feature-based classification

In order to perform classification, a classifier is first trained using a set of objects, for which the class they belong to is known. During this training phase, also known as learning phase, some parameters of the classifier are adapted based on specific rules so that the probability to correctly classify an unseen object is maximized.

Minimum Distance Classifier

The minimum distance classifier, often called Euclidian distance classifier is one of the simplest classifiers. Assuming k different classes to which an object can belong to, the classifier only requires the mean feature vectors m_i of each class and a distance function like the Euclidian distance [39]. The learning process consists in computing the mean of the representative feature vectors for each class. For classification the distance between the new object x and each of these means is computed and the smallest distance gives the class c to which the object most likely belongs to:

$$c = \begin{cases} c_1, \textit{ if } \| \overline{f} - \overline{m_1} \| = \min(\| \overline{f} - \overline{m_i} \|), \textit{ where } i = \overline{1..k} \\ c_2, \textit{ if } \| \overline{f} - \overline{m_2} \| = \min(\| \overline{f} - \overline{m_i} \|), \textit{ where } i = \overline{1..k} \\ \quad \ldots \\ c_k, \textit{ if } \| \overline{f} - \overline{m_k} \| = \min(\| \overline{f} - \overline{m_i} \|), \textit{ where } i = \overline{1..k} \end{cases} \tag{2.9}$$

where c is the class to which the object belongs to, $\bar{f}$ is the feature vector of the object x, m_i are the means of the feature vectors representative for each class and k is the number of considered classes.

Besides its simplicity, the minimum distance classifier has the advantage that its parameters can be easily interpreted and boundaries can be added to the mean feature vectors. These boundaries can be given for example by the standard deviation of the samples within a class and can be used to build a binary classifier even without having negative samples. In other words, if the properties of a class of objects are represented

by a series of samples of feature vectors, the minimum distance classifier can be trained to detect the membership to that class even without knowing the properties of objects that do not belong to that class.

One of the disadvantages of this classifier is that it can only work well in simple feature spaces. As the feature space becomes more complex by adding more dimensions, the distance might not give enough information for properly determining the class an object belongs to and selecting proper boundaries can become a complex problem.

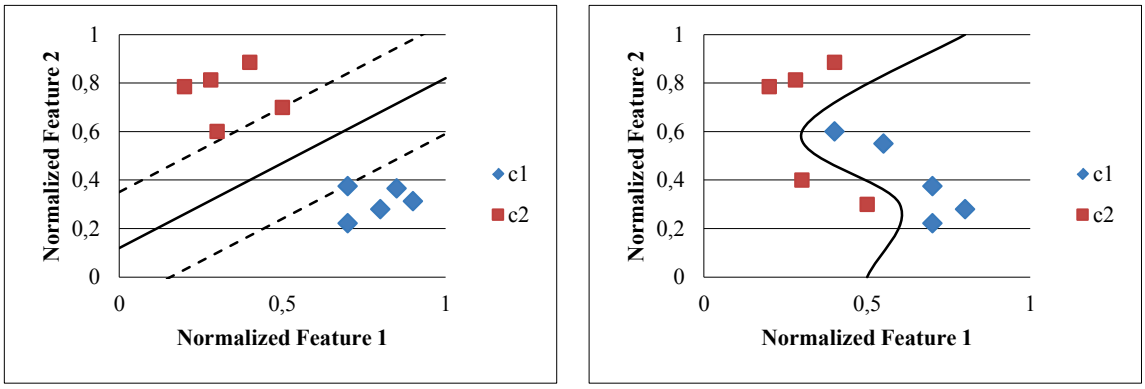
Support Vector Machines

Support vector machines (SVM) were originally designed for binary linear classification, which means that a hyper plane separates the two possible classes in the feature space. Although the linearity and the binary nature present clear limitations of this classifier, methods were developed for dealing with non-linear feature distributions by remapping the feature space to a higher-dimensional feature space in which the feature distribution is linear [37] [38]. Also, it is possible to build a multiclass support vector machine by combining multiple binary support vector machines. Two widely used approaches are the "one-versus-all" [37] and the "one-versus-one" [38] approach.

The one "one-versus-all" approach constructs k binary classifiers, where k is the number of classes. Each of these classifiers discriminates between the considered class and a group formed by all other classes. The final output is selected based on the highest outcome given by the k binary classifiers, similarly to how the minimum distance classifier analyzes the distances to each mean feature vector.

The "one-versus-one" approach considers $k(k-1)/2$ binary classifiers, where each two classes are directly compared. Each of these classifiers casts a vote for a specific class and the class with the largest number of votes wins.

The most important advantage of SVM is the automatic supervised learning process, in which a series of objects are presented to the classifier together with their class membership. The learning process consists in finding the hyper plane that best separates the two classes, meaning that the largest possible boundary is selected. An example of two linearly separable classes, c_1 and c_2, in a two-dimensional feature space together with the hyper plane and the boundaries can be seen in Fig. 2.10 a).



a) Linear separation b) Non-linear separation

Fig. 2.10 Examples of linear and non-linear separation of features

However, there are situations in which it is not possible to find a hyper plane to separate the two classes, like in the case presented in Fig. 2.10 b). There has been research on transforming the feature space to a higher-dimensional feature space in which the classes are linearly separable [37] [38], but this solution implies additional computation which differs depending on the application and might not always yield the best results.

Neural Network Classifier

An Artificial Neural Network (ANN), often called Neural Network (NN) is a mathematical model, which was inspired by biological neural networks. It consists on artificial neurons located on different layers, which are connected by artificial synapses, similar to biological neurons [17].

There are different types of neural networks, the simplest of which being the feed-forward neural network, in which data only travels in one direction, as illustrated in Fig. 2.11.

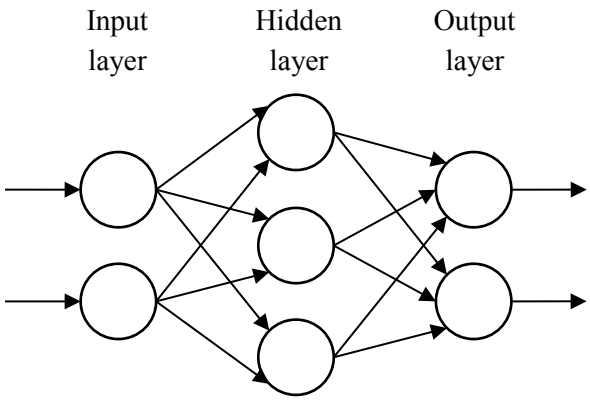


Fig. 2.11 Example of artificial neural network

In the presented example, there is one input layer, one output layer and one hidden layer. Clearly the number of neurons depends on the application as the input layer contains as many neurons as there are data inputs which should be used for classification (size of the feature vector $\bar{f}$) and the output layer contains the same number of neurons as many object classes there are. Choosing the right number of hidden layers and neurons is a very complex problem, however it is commonly agreed that one hidden layer having a number of neurons equal to the mean between the number of neurons in the input and the output layer is fairly suitable in most situations.

Numerical data is transmitted among neurons via synapses, which store a weight that controls how much the value of that particular input influences the output value of the neuron. Therefore the output of a neuron is a weighted sum of all its inputs.

$$y_i = \sum_j w_j g_j \tag{2.10}$$

where y_i is the output of neuron i, w_j is the weight of the synapse on which the value g_j is being received from the previous neuron. In many situations however y_i is not directly the output of the neuron, but it is the input to an activation function like a sigmoid or a step function, whose output is used as the output of the neuron.

The classification process for a neural network is therefore determined by all these weights and by the way the neurons are connected. During the learning process these weights are automatically adjusted so that the classification results for the training data are as good as possible. The most popular training algorithm is back propagation, in which the weights are randomly assigned and after presenting an object, the output error with respect to the desired output is propagated back to the input, adjusting the weights. If the input data is consistent, after the training phase the neural network should be able to correctly classify the training data. Further, if the training data was representative for the population, it should also perform well on new data, which has not been used for training.

The biggest advantages of using neural networks for classification are the fact that it can deal with data that is not linearly separable, as illustrated in Fig. 2.10 b) and that the data distribution does not need to be known, as is the case for many other classifiers. Some of the disadvantages include a long training time until the weights converge and a higher complexity than the SVM and minimum distance classifier.

2.5 Object tracking

Object tracking is the process of locating one or more moving objects in a video sequence, which is a series of individual images also called frames. Depending on the application, these objects can be for example vehicles, various containers such as boxes or bottles or humans. However, common to all object tracking applications is that the goal of object tracking is to detect the object of interest in all frames of the video. In case of on-line applications, as presented in Chapter 5 and Chapter 6, object tracking has to be performed in real-time, which means that each captured frame has to be processed until the next frame is received.

In object tracking applications the central role is played by the so-called tracker or tracking filter, which is the module responsible for establishing to which object in the previous frame the currently processed object in the current frame corresponds to. There are various approaches for object tracking, i.e. establishing the correspondences between objects in subsequent frames [11]. One of the most widely used methods is to use time consuming object recognition algorithms only for initializing the tracker, which then uses other types of features that can be computed very fast. The disadvantage of this method is that sometimes the tracker loses the object and it needs to be reinitialized by performing object recognition again. Another approach, on which the algorithms presented in Chapters 5 and 6 are based, uses an object recognition algorithm optimized for on-line use, which is able to process each incoming frame before the next frame is captured. Such optimization typically consists in using application dependent a priori knowledge and specific image segmentation algorithms, which can significantly reduce the overall computation time.

The presented tracking methods can be improved by adding a motion filter to them, such as a Kalman filter [35]. The purpose of this filter is to interpolate among data samples as well as to predict future values in order to help the tracking process. The main limitation of the Kalman filter is that it can be successfully applied only for objects moving with constant speed or constant acceleration. This filter was used in this thesis for the application presented in Chapter 6 as the above mentioned conditions are met.

2.6 Stereo camera vision-based 3D reconstruction

The main disadvantage of monocular cameras is that due to the projection of the 3D objects from the imaged scene on the 2D image surface the distance information to the objects is lost. However, this information can be recovered if two or more cameras

are imaging the object of interest and if the relative positions of these cameras are known. A camera system that includes two monocular cameras is called stereo camera.

Stereo camera

A stereo camera is generally composed of two monocular cameras which are both imaging the scene that should be observed, similarly to how the human eyes are used for perceiving depth. In contrast to a monocular camera, the second perspective allows the 3D location of real-world points to be estimated based on the position of their projections on the two image planes. This process is also known as triangulation and is covered by the epipolar geometry [10].

Based on the orientation of the two cameras with respect to each other the stereo camera can be convergent if the optical axes intersect in front of the stereo camera (the cameras look to each other), divergent if the optical axes do not intersect in front of the stereo camera (the cameras look away from each other) or parallel if their optical axes are parallel. The parallel configuration additionally allows the construction of so-called disparity maps, which are specific gray-scale images in which the pixel intensity is inversely proportional to the distance between the camera and the imaged point in the scene. However, the construction of disparity maps additionally requires the focal lengths of the two cameras to be identical, so that both image planes lie in the same geometric plane. More information on disparity maps is given later in this section.

An example of convergent stereo camera can be seen in Fig. 2.12. For a stereo camera, each of the two cameras has two coordinate systems, as already shown for monocular cameras: an image coordinate system *(u, v)* and a camera coordinate system *(x, y, z)*, which are usually denoted differently to indicate to which camera the coordinate applies. An additional parameter of the stereo camera is the distance between the two camera centers, which is known as base line *b*. The intersection of the base line with the two image planes generates the two epipoles, one in each image.

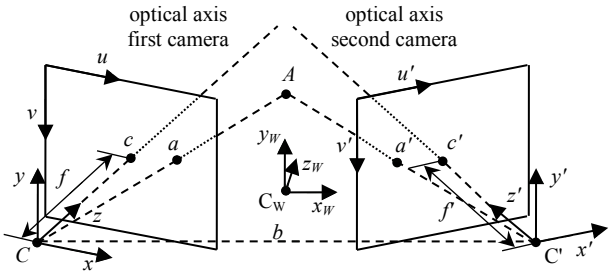


Fig. 2.12 Illustration of relationships between the first camera, the second camera and the world coordinate system

The camera parameters already given for monocular cameras apply to each of the two stereo cameras and they are denoted differently as illustrated in Fig. 2.12. It can be observed that the two camera centers form a plane together with point A, which is the world point that is imaged. This plane is called epipolar plane and its intersection with the two image planes generates two epipolar lines, one on each image plane. It can be seen that the two corresponding points a_L and a_R of an imaged point A are constrained to lie on the epipolar lines. This property is used in Chapter 4 in order to improve the vision-based gait analysis system.

In this thesis, the stereo camera with parallel optical axes is of particular interest for the applications presented in Chapters 5 and 6. Fig. 2.13 illustrates the coordinate systems of a stereo camera with parallel optical axes. The two cameras are denoted in this case as left camera L and right camera R.

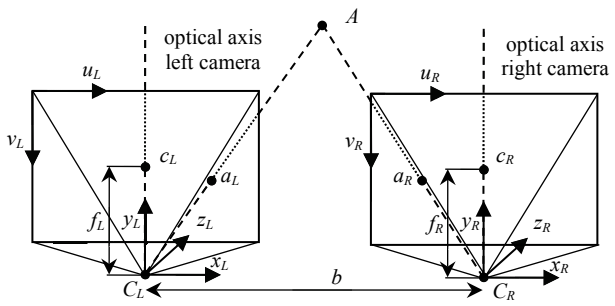


Fig. 2.13 Stereo camera system with parallel optical axis with the image and camera coordinate systems of the left and right camera

For stereo cameras it is common to consider the origin of the stereo camera system in the optical center of the left camera. Stereo cameras with parallel optical axes have particular values for the rotation and translation matrices. Namely, the "ideal" extrinsic parameters are:

$$R_L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad t_L = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.11}$$

$$R_R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad t_R = \begin{bmatrix} -b \\ 0 \\ 0 \end{bmatrix} \tag{2.12}$$

where R_L and t_L are the rotation matrix and translation vector for the left camera and R_R and t_R are the rotation matrix and translation vector for the right camera. The base line

b is exactly the distance for which the right camera is translated along the *x* axis, as can be seen in (2.12).

As this ideal case is very unlikely to occur in real-world applications, the extrinsic parameters must be estimated along with the intrinsic parameters during the stereo camera calibration procedure, which is described next.

Stereo camera calibration

As already mentioned, the stereo camera contains two monocular cameras and therefore the camera parameters can be estimated as already described, including the estimation of the rotation matrix and the translation vector. These two extrinsic parameters can be inferred from the extrinsic parameters of each camera with respect to the world origin [10].

In order to measure the accuracy of stereo camera calibration, the already described tests using the chessboard calibration pattern can be used. Additionally, the accuracy of estimating the rotation matrix and translation vector of one camera with respect to the other can be tested. This is done using epipolar geometry and the property that corresponding points must lie on epipolar lines. Therefore the chessboard corners from the left image are used to compute the corresponding epipolar lines in the right image. Next, the distance between each of the detected corners in the right image and the corresponding epipolar line is computed. The same operation is then also applied from the right to the left camera and by computing statistical measures like average and standard deviation on the series of distances, the stereo calibration accuracy is calculated.

Stereo triangulation

Generally, regardless of how the two cameras are oriented, the 3D location of the point *A(x, y, z)* can be obtained by stereo triangulation. For this operation the two projections of the point *A* in the two images are required, *a(u, v)* and *a'(u', v')*, together with the two projection matrices *P* and *P'*. Triangulation implies solving the following set of equations:

$$\begin{bmatrix} uP_3^T - P_1^T \\ vP_3^T - P_2^T \\ u'P_3'^T - P_1'^T \\ v'P_3'^T - P_2'^T \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.13}$$

where the generic terms P_n^T and $P'_n{}^T$ refers the *n*-th row of the cameras' projection matrices P and P' respectively.

Equations of the type $Ax=0$ can be solved either by Singular Value Decomposition (SVD) or by computing Eigenvectors. In order to assess how precise the result of each of the two methods is, the error *e* can be defined as the sum of absolute values obtained from multiplying the matrix *A* with the computed *x*:

$$e = |R_1| + |R_2| + |R_3| + |R_4| \tag{2.14}$$

where R_1, R_2, R_3 and R_4 are the four components of the error vector, which ideally should be equal to zero.

Stereo rectification

As previously mentioned, for stereo cameras with parallel optical axes, so-called disparity maps can be computed if their focal lengths are equal so that the two image planes lie in the same geometric plane. If this condition is not met, as is commonly the case due to mechanical misalignments of the cameras, the images must be stereo rectified.

Stereo rectification is the process of transforming a pair of images obtained from a converging or diverging stereo camera. This results in images with the content as if they were captured by a pair of cameras with perfectly parallel optical axes and equal focal lengths. The result of stereo rectification is considerably improved if the actual camera setup is close to these ideal conditions.

One of the most important properties of stereo rectified images is that the epipolar lines are parallel to the *u* axis of the image. Further, a pair of corresponding points, that is the two projections of the same 3D point on the two images, is located on the image line with the same *v* coordinate. Fig. 2.14 illustrates this property.

In Fig. 2.14 a) two example chessboard corners were selected for the left image and for each of these corners a green line illustrates the *v* coordinate at which the corresponding corner points should lie in the right image. It can be seen that for both corners in the left image, the corresponding corners in the right image are located at a smaller *v* coordinate.

In Fig. 2.14 b) the image pair after stereo rectification is shown. In contrast to the case of the original images, for the rectified image pair the corresponding corner points in the right image lie on the same *v* coordinate as the selected corners in the left image, as illustrated by the two green lines.

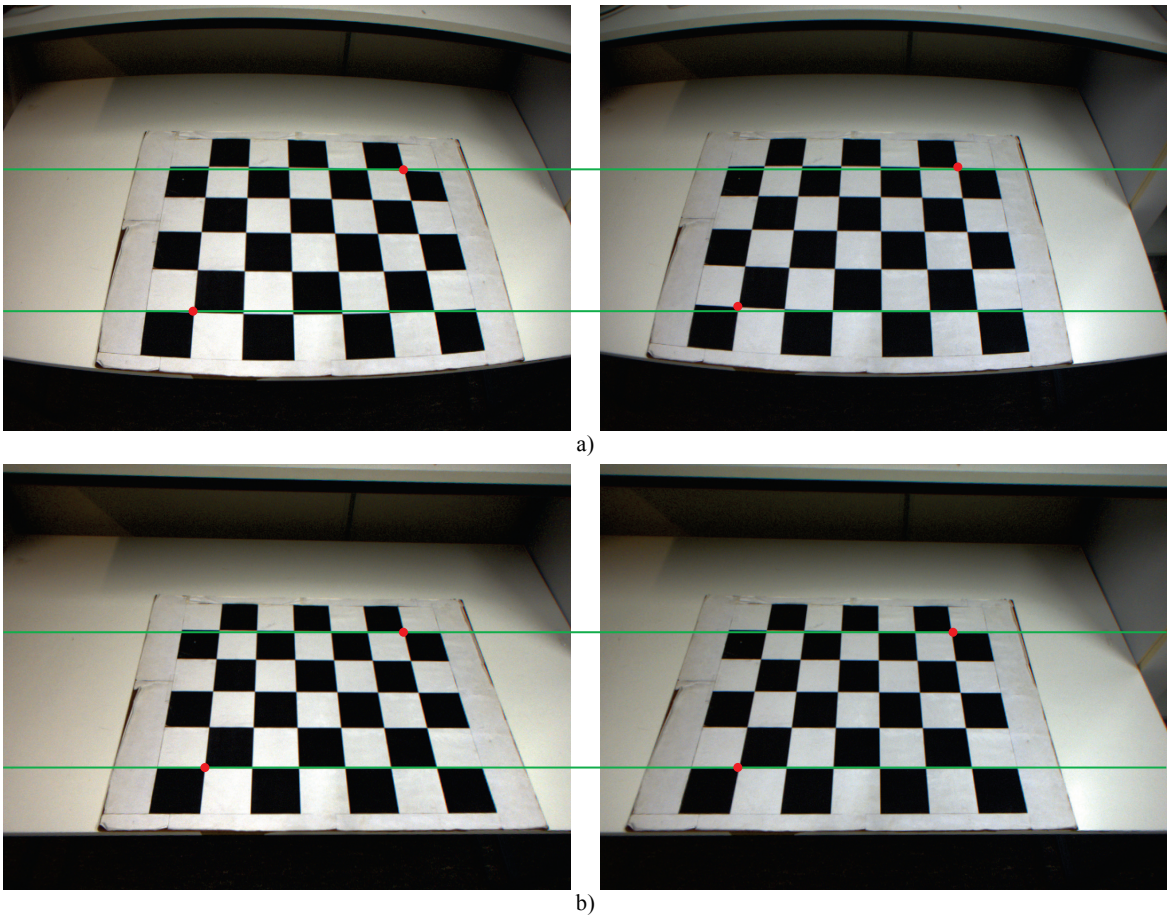


Fig. 2.14 Example of stereo rectification: a) Original image pair and b) Stereo rectified image pair

In order to perform stereo rectification, the intrinsic camera parameters are required, together with the rotation matrix and translation vector, which describe the rotation and translation of the right camera with respect to the left camera. These parameters can all be determined using camera calibration.

Even though the location of the projected 3D point can be computed using triangulation for all stereo camera setups, the fact that for stereo rectified images the corresponding points lie on the same image line speeds up the searching process considerably. Due to this property disparity maps can be computed very fast by using adequate methods.

Disparity map computation

Stereo rectification of images enables the computation of so-called disparity maps. Disparity maps are images in which each pixel value encodes the distance between the camera and the 3D point projected on the left image at the same image coordinates. In other words, for each image point *a(u, v)* in the left image, the corresponding disparity value in the disparity map is *d(u, v)*. For visualizing the disparity maps, they are most commonly represented either as grayscale image, for which the pixel intensity is

inverse proportional to the distance, or as color image in which the distance is encoded with colors between red and blue, where red points are close to the camera (hot) and blue points are far (cold). Fig. 2.15 shows an example of a stereo rectified image pair, the disparity map using grayscale representation and the disparity map using red-blue representation.

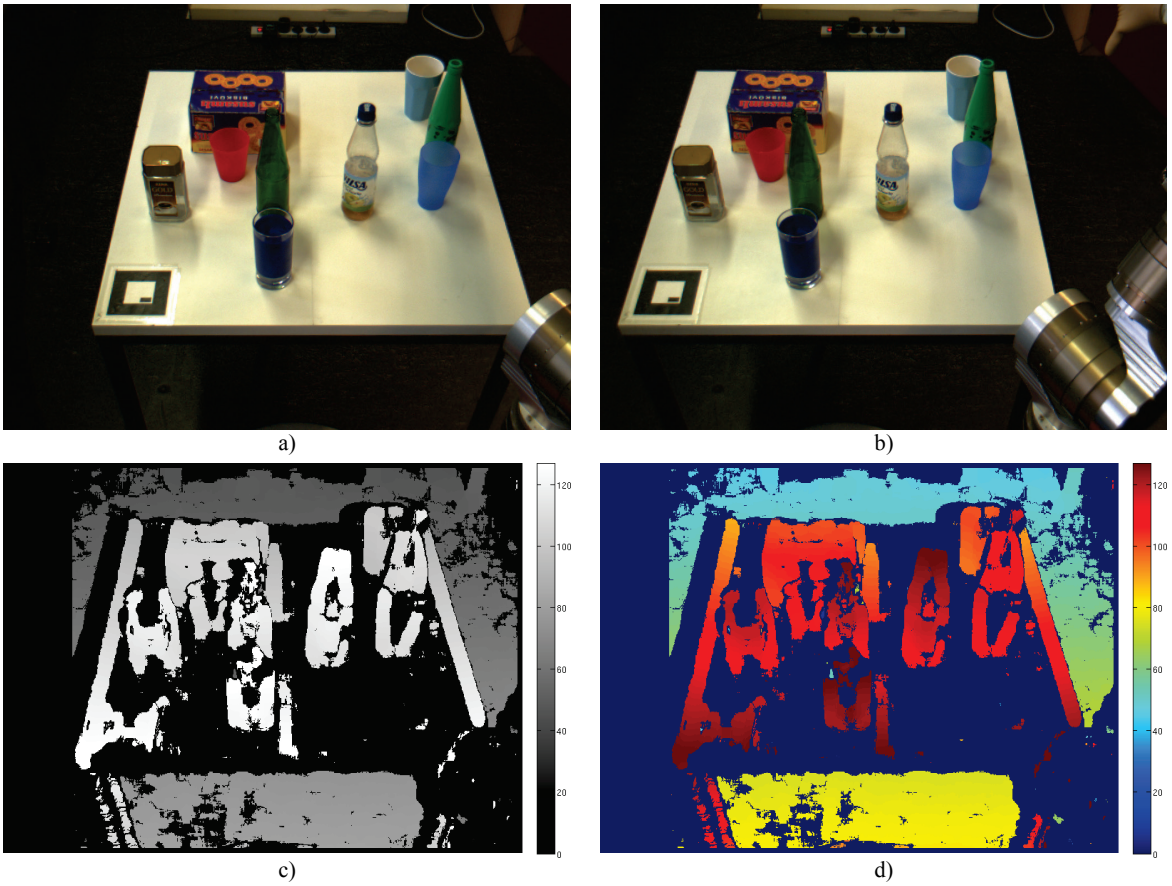


Fig. 2.15 Example of disparity map representations: a) Rectified left image, b) Rectified right image, c) Gray-scale representation and d) Colored representation

One of the biggest problems faced in disparity map computation is finding corresponding points in the two images, a process also known as stereo matching. There are various ways for computing disparity maps, which can be mainly divided in three categories: local methods, which only use the close neighborhood of a pixel in the left image for finding its corresponding pixel in the right image; global methods which extract features from both full images and match them; and hybrid methods, which use a combination of the two.

Usually, local methods are very fast compared to global methods. However they are not able to compute disparity values for objects that are occluded or if the difference in perspective between the two cameras is too big. The reason for this is that local methods only use a limited number of pixels in the selected pixel's neighborhood

for matching. Global methods, on the other hand, contain information almost over the whole disparity map, but are relatively slow due to the series of required optimizations and due to using all pixels for computing the disparity value for each selected pixel.

Throughout this thesis the local disparity map computation method called *block matching* [44] is used due to the fair results obtained in most scenarios and especially due to the short computation time, which is critical for real-time applications.

In block matching disparity map computation for each pixel in the left image an n x n block of pixels having the selected pixel in the center is compared to all blocks of pixels of the same size from the right image, where n is always an odd number. Due to the usage of stereo rectified images, two constraints are added, that speed up the matching process. The center pixel of a block in right image lies on the image line with the same v coordinate as the considered (reference) pixel in the left image. Additionally the u coordinate of the center pixel of a block in the right image is always smaller than that of the reference pixel in the left image. Having these constraints in mind, the best matching block is determined based on a cost function. The most commonly used cost function is the Sum of Absolute Differences (SAD) of pixel values from the left and right image within the block:

$$SAD(u,v,d) = \sum_{i=u-\frac{n}{2},j=v-\frac{n}{2}}^{i=u+\frac{n}{2},j=v+\frac{n}{2}} \left| a_L(u+i,v+j) - a_R(u+i-d,v+j) \right|, \quad d = \overline{0..d_{\max}} \tag{2.15}$$

where u and v are the coordinates of the pixel from the left image for which the disparity value is calculated, d is the current disparity value, n is the block size used for stereo matching and a_L and a_R represent the pixel values of the left and right image respectively.

The above SAD value is computed for all possible disparity values ranging from 0 to the maximum expected disparity value $d_{\max}$. The best match for a pixel is found for the disparity value that generates the smallest sum:

$$d(u,v) = MIN(SAD(u,v,d)), \quad d = \overline{0..d_{\max}} \tag{2.16}$$

If the minimal sum is too big with respect to a chosen threshold, no correspondence could be found (due to occlusions or difference in perspective). If two different sums, corresponding to two different disparity values, are similar, it means that more than one correspondence was found (due to lack of texture), so the correct match could not

be uniquely identified. Certain post-processing steps are often applied in order to eliminate such regions with low texture or occlusions.

After two corresponding image points $a_L(u_L, v_L)$ and $a_R(u_R, v_R)$ have been found, the disparity value is defined as:

$$d = u_L - u_R \tag{2.17}$$

where d is the disparity and u_L and u_R are coordinates of the corresponding points in the left and right image respectively.

The size of the block used for matching plays an important role. A too small block does not overcome the problem of matching uncertainty which appears especially for image regions with low texture. On the other hand, a too big block can cause the selected regions of the two images to look too different due to the two different perspectives. This applies especially to objects which are close to the camera. The influence of the block size on the resulting disparity map can be seen in Fig. 2.16.

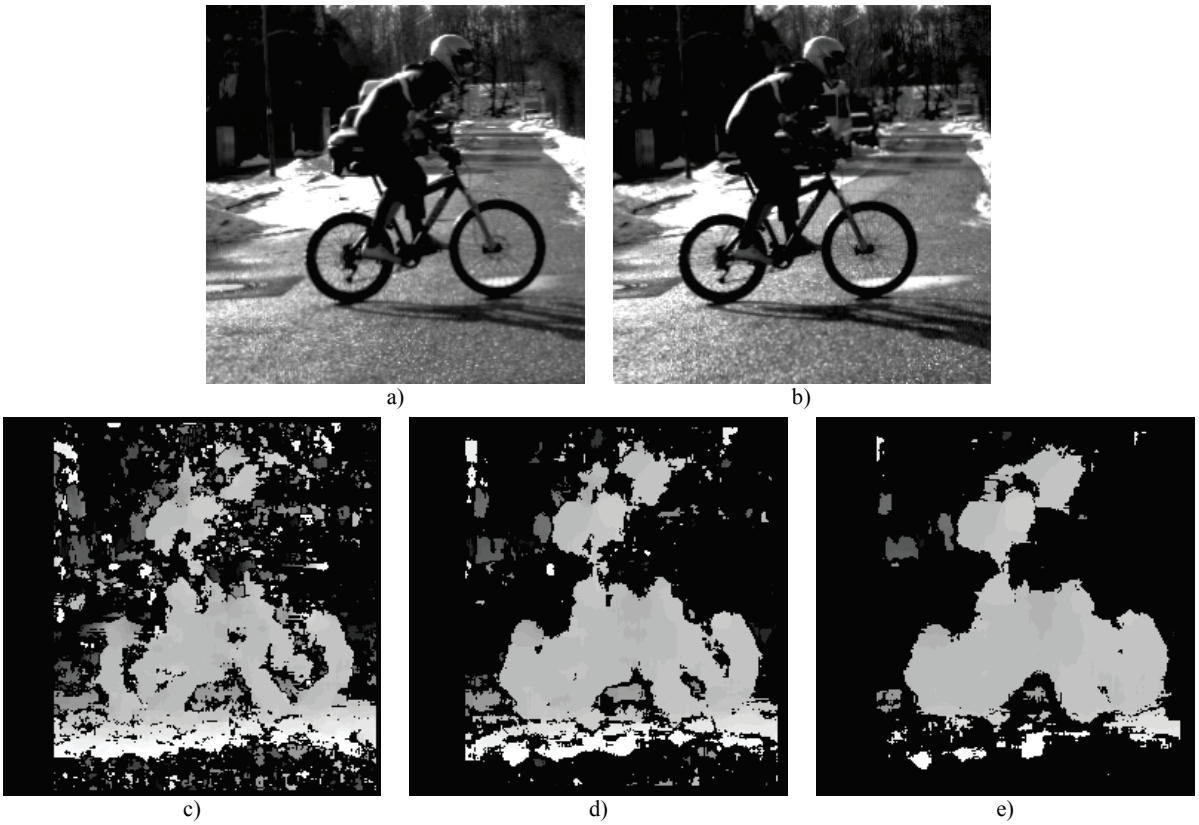


Fig. 2.16 Example of disparity maps using block matching: a) Original left image, b) Original right image, c) Block size 9x9, d) Block size 15x15 and e) Block size 21x21

The regions for which no correspondence was found are represented with black in the disparity maps c)-e) in Fig. 2.16. For smaller blocks the shape of objects is more accurately represented, but at the same time there are more regions without

correspondence and more noise. In case of the bigger blocks there are fewer unmatched areas and less noise, but the shape of the object is altered due to the so-called "thickening effect", also known as "fattening effect" or "bleeding", which is characteristic for large blocks.

3D point reconstruction

Knowing the disparity value d for a point $a_L(u_L, v_L)$, the 3D location of the original point $A(x, y, z)$, with respect to the stereo camera coordinate system is given by:

$$x = \frac{b \cdot (u_L - u_{cL})}{d} \quad y = \frac{b \cdot (v_{cL} - v_L)}{d} \quad z = \frac{b \cdot f}{d} \tag{2.18}$$

where f is the focal length of the left camera, b is the base line of the stereo camera and $c(u_{cL}, v_{cL})$ is the principal point of the left camera.

2.7 Vision-based human detection and tracking

Vision-based human detection and tracking is a particular case of vision-based object recognition in the sense that humans can be seen as objects of particular size and shape. Therefore, the typical block diagram of the image processing chain used for human detection and tracking, shown in Fig. 2.17, has certain similarities to the diagram of object recognition shown in Fig. 2.1.

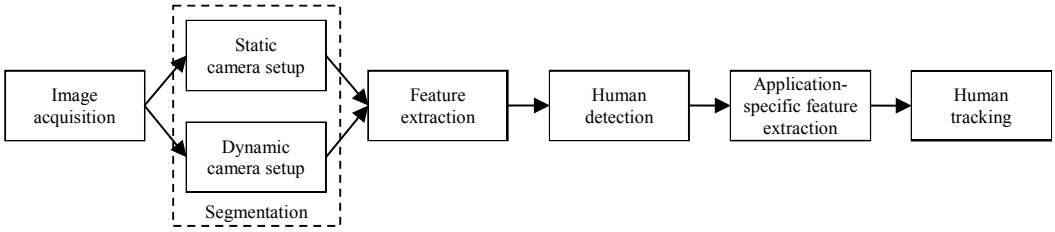


Fig. 2.17 Block diagram of vision-based human detection and tracking

The image segmentation algorithms used for human detection and tracking are optimized either for static or dynamic camera setup, as can be seen in Fig. 2.17. In static setups the camera is fixed, like in the case of vision-based clinical gait analysis or video surveillance, while dynamic setups imply that the camera is moving, like for in-vehicle cameras used for pedestrian detection or on-board cameras used for navigation of robotic systems. Static setups are often used in indoor applications, while dynamic setups are often used outdoors.

As image segmentation has a crucial impact on the reliability of subsequent human detection, feature extraction and tracking, a large proportion of the image processing

literature is dedicated to this topic [28][50]. However, in spite of the number of published papers, there is still no general method for determining the segmentation parameters which will lead to reliable segmentation results in the presence of the numerous external influences that can arise in real-world human detection applications.

As image segmentation is still an unsolved problem, researchers work on the development of new algorithms but also on new hardware to help human detection and tracking. Therefore, recently so-called RGBD cameras such as the Microsoft Kinect [76] and the Asus Xtion PRO LIVE [77] have been developed for indoor human detection and tracking. These cameras provide color information (RGB) like a traditional camera but also depth information (D), which gives information about the distance between the camera and particular regions of the scene, similarly to the information obtained from disparity maps. Structured infrared light is used for estimating the depth, which means that an infrared light pattern is projected on the scene and observed with a camera that is only sensitive to infrared illumination. Fig. 2.18 shows a set of sample images recorded with the Kinect camera.

a) b) c)

Fig. 2.18 Sample images recorded with the Microsoft Kinect camera: a) RGB image, b) Infrared image and c) Depth image

The RGBD cameras are usually calibrated by the manufacturer, so they can be used immediately after they are set up. One advantage of this technology is that no environment light is required for depth sensing, as this type of camera is an active sensor that projects all light that it needs on the scene. However, at the same time this light projection is also the biggest disadvantage, as natural light contains a significant amount of infrared light, so this technology cannot be used in direct sunlight, as the projected light cannot be distinguished from the natural light. As the goal of this thesis is to develop algorithms for both indoor and outdoor human tracking, traditional passive cameras are used, which perform well in both environments.

Human Detection

Vision-based human detection is the process of detecting the image region which contains a person. The output of this image processing step can be a rectangular region which bounds the person, also known as Region of Interest (ROI), but sometimes a more complex shape can better describe the region containing the human by including fewer background pixels. Such shape can be the contour of the person or a convex hull bounding the person's contour.

Depending on the nature of the application, images to be processed for human detection can be independent of each other (e.g. if they are taken in very large time intervals one after another and/or in different locations) or they can be part of a series of images, like in a video. Human detection can be performed on individual images, while for image series, the person has to be tracked, which means that the person is followed in all frames. This makes it possible to benefit from advantages like prediction of ROI, as will be shown in Chapter 6.

Human Tracking

The process of human tracking can be defined as locating the same person in a series of consecutive frames and the result of this process could be a series of ROIs. Clearly human tracking heavily depends on human detection since it is essential to detect the person before deducing whether the person detected in the current frame is the same person as in the previous frame, but at different image coordinates. In tracking, if human detection fails in one frame, values from previous frames can be used to estimate where the person is most likely to be located in the current frame. Further, the estimated location of the human can be used to improve the performance of the tracking algorithm. This additional information can help making the detection process faster (e.g. by using the previous location and/or size of the ROI as a starting

point) and it can also increase the detection accuracy if a filtering and prediction algorithm is implemented.

Human tracking algorithms can be divided mainly in two approaches. The first is to detect the person in each frame and to use the image coordinates of some of the person's features for tracking, for example the location of the center of mass and the size of the human ROI. In the second approach the person is only detected in the first frame to initialize a low-level algorithm, which then tracks the region of the person based on low-level features such as distribution of pixel intensities within that region. In this second situation, the detection of the person can be performed again at regular intervals to update the model used for low-level tracking [11].

However, tracking also differs with respect to the types of required features. In the case of clinical gait analysis the extracted and tracked features are gait parameters such as the angles between body parts. For pedestrian detection the extracted and tracked feature is the center of mass of the ROI containing the human. In vision-based Human-Robot Interaction (HRI), the extracted and tracked feature is the human position, but could also include body poses or gestures, which would enable the system to better understand what the human wishes to communicate to the robot.

Indoor human detection and tracking

There are many indoor applications for human detection and tracking such as surveillance, teleconferencing, clinical gait analysis and interactive video games. Indoor human tracking applications often include particularities such as constant artificial illumination and little to no influence of natural light in rooms with small windows or with no windows at all. Also indoor application are often characterized by fixed positions of the cameras, as they are often mounted on walls or supports and by relatively low scene dynamics due to the limited number of different objects that can be in the camera view. Nevertheless indoor human tracking is still a challenging topic as the appearance of humans can differ due to their size, skin color and clothing. In this thesis an indoor application of vision-based gait analysis is presented in Chapter 4. New algorithms for robust closed-loop segmentation of subtracted images and automatic extraction of gait parameters based on 3D reconstruction of virtual joints are presented along with a comparison between healthy and pathological gait patterns using the extracted gait features.

Outdoor human detection and tracking

Human detection and tracking is used in outdoor scenarios for different purposes such as surveillance, human-machine interaction and pedestrian detection for collision warning systems in the automotive industry. One of the most distinctive characteristics of outdoor applications is the natural light during day and its absence during night. Additionally, even though for outdoor applications the positions of the cameras can be fixed, such as in surveillance applications, for the majority of applications the cameras are moving. These characteristics are challenges that have to be addressed by vision-based algorithms that are designed for outdoor human detection and tracking. In this thesis a typical outdoor application for pedestrian detection is presented in Chapter 5. A new algorithm for disparity map segmentation for human detection and tracking is presented, which enables real-time outdoor human tracking.

2.8 Hardware acceleration of image processing algorithms

Many image processing algorithms for the purpose of objects recognition and tracking (including humans) are very time consuming due to the large number of pixels of an image that have to be processed, even if only simple operations are applied to each individual pixel. Regular PCs usually have a single CPU (Central Processing Unit) which nowadays has 2-4 cores. However PCs often run multiple applications in the background, so not all resources are available for image processing. This often leads to applications in which the image processing algorithms are run sequentially on the input images, as illustrated in Fig. 2.19.

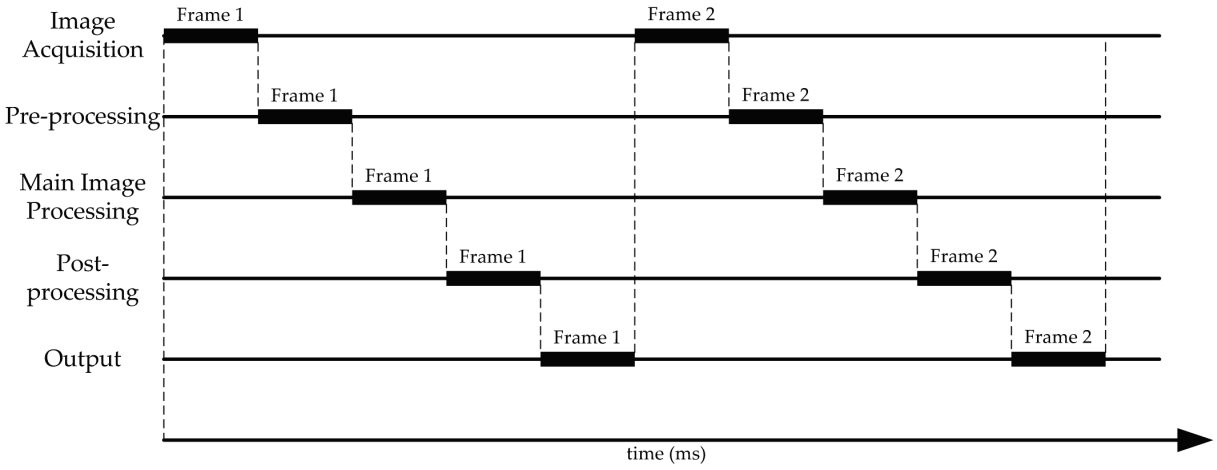


Fig. 2.19 Illustration of sequential image processing

It can be seen that each image processing algorithm is applied to the output of the previous algorithm as they are all part of a chain. However, the first step of acquiring a

new image can only start again after the last step of the chain has finished if the PC does not offer enough computational power to run all steps in parallel.

In order to overcome these limitations, different approaches have been developed, depending on the particular application and on the available physical system [18]. Dedicated hardware is often used for specialized image processing applications, two of the most popular solutions in state-of-the-art systems being the use of GPUs (Graphical Processing Units) and of FPGAs (Field Programmable Gate Arrays) as in the application presented in Chapter 5. Another approach is the use of distributed computing for which multiple PCs are working together, each of them running one or multiple algorithms, as in the application presented in Chapter 6.

Depending on the particular image processing algorithm, sometimes multiple regions of the image can be processed in parallel to speed up the execution. This is particularly well supported by GPUs, as they have a large number of cores that can run the same instruction set in parallel on different image regions according to the SIMD (Single Instruction Multiple Data) principle [21].

Another way to speed up image processing is pipelining, which can be used almost every time when different algorithms are applied successively to a series of images, as illustrated in Fig. 2.20. This approach is particularly well supported by distributed computing, especially if a relatively small amount of data is exchanged between computers.

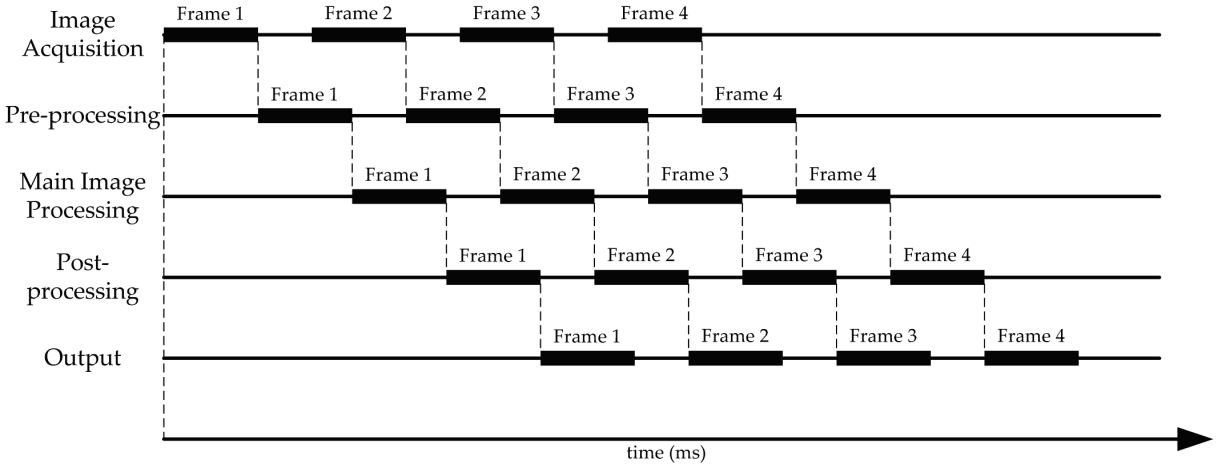


Fig. 2.20 Illustration of the pipeline principle

It is noticeable that even though individual algorithms do not necessarily run faster than in the sequential approach, pipelining allows them to run in parallel, which considerably reduces the processing time, therefore allowing more images to be processed in the same amount of time.

The FPGA combines both of these advantages as it allows parallel processing within the image and also supports pipelining, but at the cost that algorithms are considerably more difficult to implement than on CPUs and GPUs.

In order to evaluate the processing speed of an image processing chain that runs on a particular hardware system, two parameters can be defined: the processing time T_P and the latency T_L. The processing time is defined as the time interval between two consecutive outputs, while the latency is the time interval between image acquisition and output for the same frame.

In the case of sequential processing, the processing time and the latency are identical and for the example given in Fig. 2.19 they are:

$$T_L = T_P = T_{Acq} + T_{Pre} + T_{Main} + T_{Post} + T_{Out} \tag{2.19}$$

where T_{Acq} is the time required for acquiring an image, T_{Pre} is the time required by pre-processing, T_{Main} is the time needed by the main image processing algorithm to finish, T_{Post} is the time required by post-processing and T_{Out} is the time needed to compute and deliver the output to the subsystem requiring it.

In the case of pipelining, the latency remains the same, as evidently each image has to pass through all steps, but the processing time is reduced to the time needed by the slowest step to finish processing. Hence in the example illustrated in Fig. 2.20, the processing time is:

$$T_P = MAX(T_{Acq}, T_{Pre}, T_{Main}, T_{Post}, T_{Out}) \tag{2.20}$$

It can be easily observed that pipelining does not bring any benefit if a single image is processed, as the operations still run sequentially on the same image. However if image sequences are processed, after the first frame passes through the pipeline all other frames are delivered at the interval T_P. In (2.19) and (2.20), the transmission time between the modules is already included in each processing time. The transmission time is neglectable for CPU and GPU, as for CPU memory access is very fast and for GPU the input image is only once copied in the GPU memory, where all image processing algorithms are applied to it. However, in the case of distributed computing transmission times can play an important role, so it is important to properly choose which steps to distribute on which hardware platforms.

Hardware acceleration using the Nvidia Tesla C1060 GPU

Dedicated graphic cards became very popular in the last decade, as high end PC games started to demand advanced GPUs to support the CPU in efficiently rendering game scenes. GPUs are optimized for parallel processing by having a large number of cores that significantly contribute to increasing the system's processing power. The fact that nowadays every high-end PC comes with a dedicated graphics card brought up the idea of enabling this dedicated hardware to also support other time-consuming operations besides the traditional game scene rendering, like the ones required by image processing systems.

The Nvidia Tesla C1060 GPU is a high-end device with 30 multiprocessors (MP) of 8 cores each, summing up to 240 scalar processor cores (SP) [83]. Besides the large number of cores, the Tesla C1060 offers a so-called shared memory, which is basically a 16 KB cached memory for each MP, which is shared among the threads running on a particular MP. This local storage can highly improve the overall performance of an algorithm, as the access times to the shared memory are significantly shorter than to the global memory, making it suitable especially for repeatedly accessing the same memory locations, as is often the case for image processing algorithms. The Nvidia GPUs can be programmed using CUDA, which is an API (Application Programmable Interface) offered by Nvidia to make GPU programming simpler [84].

The software architecture offered by CUDA version 1.3 consists of threads, which are somewhat similar to the threads of a CPU, but they are more light-weight, meaning that switching between them can be done very quickly without involving many operations, being therefore very efficient. Each thread is programmed to run a set of operations on a subset of image pixels. This set of operations is called kernel and each thread executes the same kernel, according to the SIMD (Single Instruction Multiple Data) principle [21], which only allows one algorithm to run at a time, so pipelining is not supported. Within each kernel the thread ID can be determined in order to know which image region to process. In this way image processing algorithms are speeded up by allowing different regions of the image to be processed at the same time.

When executing the code, threads are grouped into blocks of up to 512 threads each, where each block will be executed on one MP. Therefore threads being part of the same block can use shared memory to improve the processing speed. A so-called warp, consisting of 32 threads, is physically executed on the Tesla C1060 in 4 clock cycles in the MP and a scheduler switches between warps. The warp is the smallest possible execution unit in CUDA. The maximum total number of threads is 65536.

Having all these details in mind, it becomes clear that writing an efficient program for a GPU is fundamentally different from writing an efficient program for a CPU, as all these hardware and software details have to be known in order to write an efficient program. However, the major difference compared to CPU programming consists in rethinking each algorithm, so that as many regions of the image as possible are processed in parallel.

Hardware acceleration using FPGAs

In contrast to CPUs and GPUs, FPGAs have the ability to run the algorithms in a pipeline as various parts of the FPGA can be programmed to run different algorithms at the same time [18]. In this way, the first frame will be processed by the first algorithm and by the time the second algorithm starts processing the first frame, the next frame is already being processed by the first algorithm. Therefore, even though the total latency of the system is unavoidable, the processing time of one frame is considerably reduced, allowing more frames to be computed in the same amount of time, as can be seen in Fig. 2.20.

FPGAs are often used in applications where low power consumption is critical, as they offer parallel processing and pipelining without the typical high energy consumption of PCs and particularly of GPUs. However, there is often no standard implementation of image processing algorithms offered for FPGAs and almost each solution has to be customized and implemented manually. This implies a big effort on the developing part but also bears potentially huge benefits for a mass product as FPGAs are low-cost chips.

As both GPUs and FPGAs come with the possibility to highly parallelize image processing algorithms, the design of image processing algorithms is considerably different compared to CPU implementations, as CPUs typically have 2-4 cores, while GPUs often have hundreds or even thousands of them. FPGAs are fully customizable, so the programmer can decide how many processing units to use.

Even though the principle of parallelizing image processing algorithms is the same for both GPUs and FPGAs, programming a GPU is pretty straight-forward compared to programming an FPGA. This is due to the fact that modern GPUs can be programmed by the same programming languages used for writing CPU programs, but by employing additional libraries and different paradigms and by having the hardware particularities in mind. This makes the GPU implementation a valuable step towards FPGA implementation by acting as a rapid prototyping environment for parallelizing time consuming image processing algorithms without the need of having deep

knowledge about the architecture of particular FPGA chips, which is usually the case when porting algorithms to an FPGA.

Hardware acceleration using distributed computing

In applications where it is possible to use multiple PCs and high power consumption is not an issue, distributed computing can be used to accelerate image processing algorithms. In this case multiple computers are used and on each of them one or more steps from the image processing chain are running. The fact that different systems are used leads to the necessity to exchange data between them and efficient data exchange becomes imperative. Therefore considerable benefits can be obtained if the sum between the time needed to exchange data and the time needed for processing the data on the sub-system is much smaller than if the whole image processing chain would run on one system.

The most important benefit of distributed computing is that it combines the advantages brought by GPU and FPGA over the CPU: the possibility to process different regions of the same image in parallel using a fast multi-core CPU or even a GPU, while at the same time the other systems run different steps of the image processing chain, creating therefore a pipeline. Similar benefits could be obtained by using multiple GPUs in one high-end PC.

Distributed computing can be used beyond the scope of pure image processing systems, for example for robotic systems. In this case the very demanding image processing chain can run on a high-end PC that can be off-board, while other functions such as motor control can run on-board on low-power or embedded PCs, as is the case for the application presented in Chapter 6.

3

Robust image segmentation for object recognition

It has been shown in Chapter 2 that image segmentation is one of the most important steps for object recognition including human detection. The reason for this is the fact that the subsequent feature extraction and feature-based classification heavily rely on segmented images of good quality. Depending on the particular application, the desired output of image segmentation can differ, but it generally concerns separating the objects of interest from each other and from the background.

In this chapter two novel segmentation methods for object detection are presented, where human detection is considered as a particular case of general object detection. Hence, the first proposed method, robust closed-loop segmentation of subtracted images, is explained for the case of human body segmentation. This method is applicable for static camera positions as often used indoors. The second method, disparity map segmentation, is explained for the case of general object recognition including human detection. This method can be used both indoors and outdoors. As it will be shown, both methods are robust against variable scene illumination and against the various appearances of different objects including humans.

3.1 Robust closed-loop segmentation of subtracted images

In vision-based applications such as surveillance and human gait analysis, for which the camera position is fixed, human detection can be supported by background subtraction and by segmenting the subtracted image, as it will be shown in the following sections.

Background subtraction

The main idea behind background subtraction is to use an image of the background and to subtract images containing the person from it. An image containing only the background is known as background image *b* while the image with the person is known as foreground image *f*. The background subtraction operation is defined as:

$$r_{u,v,c} = \left| b_{u,v,c} - f_{u,v,c} \right|, \; u = \overline{0..imWidth-1}, \; v = \overline{0..imHeight-1} \tag{3.1}$$

where u and v are the pixel image coordinates, $imWidth$ is the image width and $imHeight$ is the image height, c represents the color channel and r is the resulting subtracted image. The subtracted color image is then converted to gray-scale according to (2.1). Fig. 3.1 illustrates the process of background subtraction: Fig. 3.1 a) shows the background image, Fig. 3.1 b) shows the foreground image and Fig. 3.1 c) shows the result of background subtraction (3.1) after color to gray-scale conversion.

a) b) c)

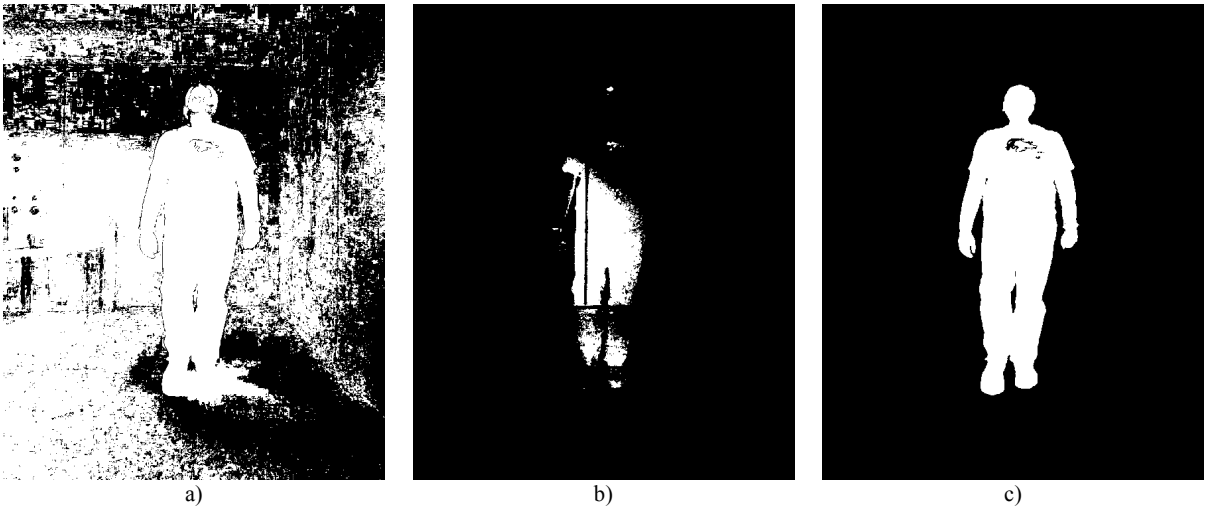
Fig. 3.1 Example of background subtraction: a) Background image, b) Image with person and c) Subtracted gray-scale image

Ideally, the image resulting from background subtraction should contain bright pixels only in regions belonging to the human body and all regions belonging to the background should be black. Such an image would be an “ideal” input to segmentation by simple thresholding [28]. The equation of simple threshold segmentation is:

$$s_{u,v} = \begin{cases} 0, if\ p_{u,v} \le Th \\ 1, if\ p_{u,v} > Th \end{cases} \tag{3.2}$$

where u and v are the image coordinates, s is the pixel value in the resulting segmented image, p is the pixel value in the input gray-scale image and Th is the threshold level. In (3.2) 0 denotes black pixels and 1 denotes white pixels

The goal of simple thresholding is to obtain a binary image for which all pixels belonging to the human are white and all background pixels are black. However, the choice of the threshold level Th is a challenging task, as a too low value causes over segmentation as shown in Fig. 3.2 a) and a too high value causes under segmentation, as shown in Fig. 3.2 b). Only the optimal threshold value leads to a well segmented image, as shown in Fig. 3.2 c), for which the human can be properly detected.



a) b) c)

Fig. 3.2 Example of image segmented using various threshold values: a) Th=15, b) Th=100 and c) Th=33

In real-world applications where natural light is present in the scene and no specially colored backgrounds are used, subtracted images often contain bright pixels in the background area besides the bright pixels in the region of the human body. This happens due to the fact that, in contrast to laboratory environments of rooms with only artificial light, the ability to control the experimental conditions, such as lighting conditions and background environment is very limited for real-world scenes. Because of this, the illumination of the background image can differ considerably from the illumination of the foreground image. The results of the corresponding background subtraction and segmentation are illustrated in Fig. 3.3 which represents a real-world application, in contrast to the laboratory environment shown in Fig. 3.1.

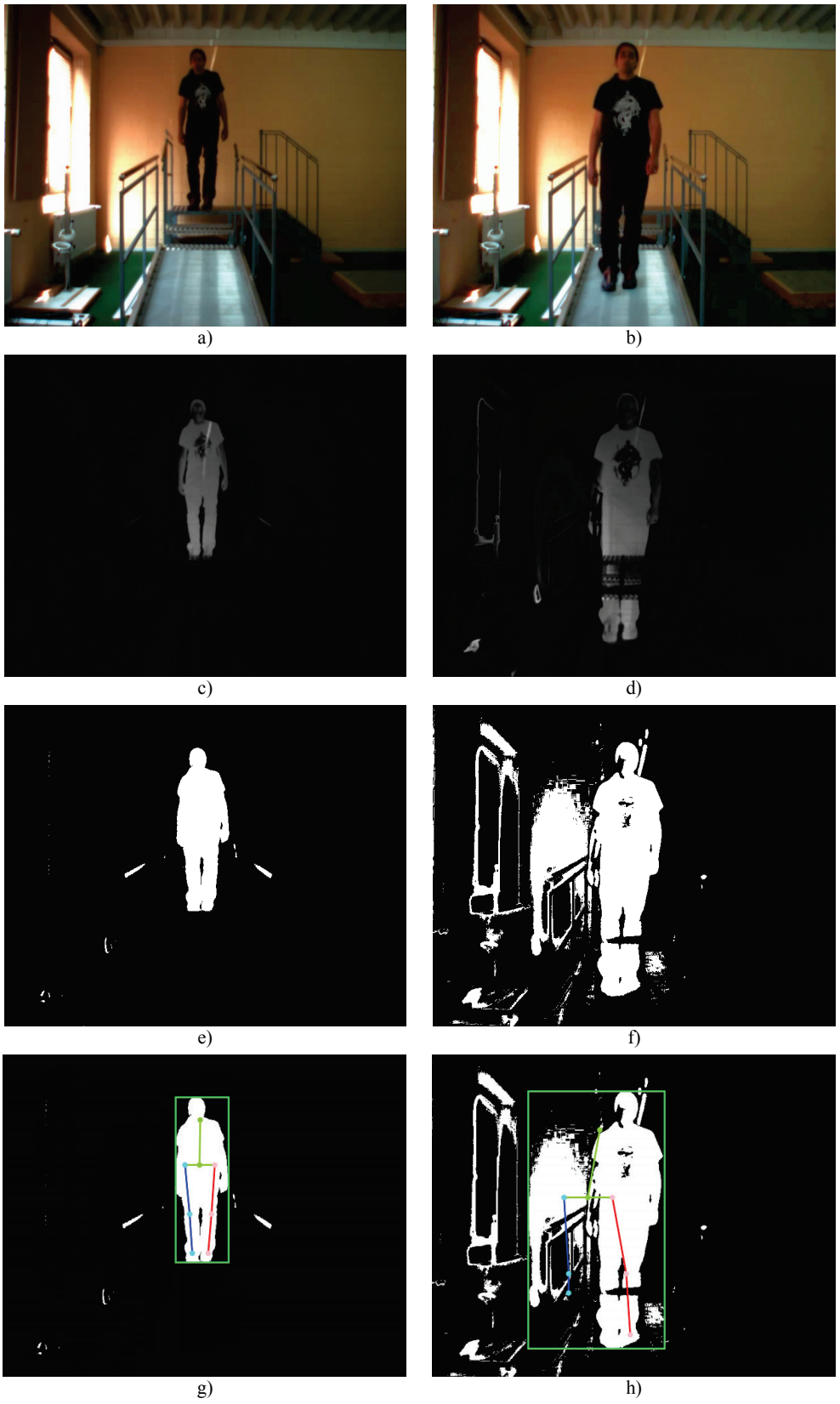


Fig. 3.3 Two video frames of different quality due to different lighting conditions: a) and b) are original images, c) and d) are the corresponding subtracted images obtained by background subtraction and conversion to grayscale, e) and f) are the segmented images obtained by segmentation of the subtracted images using the same threshold and g) and h) are binary segmented images overlaid with the extracted stick figures

Fig. 3.3 a) and b) show two example video frames of a person walking along the walking platform. The video was taken in the rehabilitation room of the "OT-Team" in Bremen [72]. As can be seen in Fig. 3.3 a) and b), in this video the walking platform is located directly in front of a large window. Such large windows create dynamic lighting conditions which may differ between the frames of the same video. Because of this, the images in Fig. 3.3 e) and f), obtained by thresholding of the subtracted images using the same segmentation parameter, are of significantly different quality.

The segmented image in Fig. 3.3 e), which corresponds to the first considered frame, is of good quality since it contains a full and "well shaped" segmented human body represented by white pixels on a black background. This image is obtained by applying threshold segmentation to the image in Fig. 3.3 c) with the manually determined optimal threshold Th=36. However, the threshold value which is optimal for one image is not necessarily optimal for another image frame captured in potentially different illumination conditions. This is evident from the segmented image in Fig. 3.3 f), which is obtained by segmenting the image in Fig. 3.3 d) using the same threshold value Th=36. This segmented image is of poor quality since it contains a lot of noise (white segmented pixels in regions belonging to the background) and a "broken" segmented human body.

If the segmented images are used for gait features extraction for human gait analysis, as will be shown in Chapter 4, the consequence of bad segmentation is a poor extraction of gait features as shown in Fig. 3.3 h) by the incorrectly extracted stick figure representing a simplified human body skeleton. In contrast to this, the segmented image of good quality shown in Fig. 3.3 e) enables the reliable feature extraction result shown in Fig. 3.3 g) by the stick figure which is correctly overlaid on the segmented human body.

The quality of image segmentation can be improved in two ways. The first approach is to adapt the background image to the illumination condition of the current frame that has to be subtracted. The second approach is to automatically select the optimal segmentation threshold for each frame. Both approaches are explained in the following sections. However, these approaches complement each other, so using both approaches at the same time gives better results than using each approach individually.

Adaptive background subtraction

In order to compensate the illumination change, the following running average background adaptation algorithm is employed [46], where the background image is updated using background information obtained from the current frame:

$$b_{u,v,c} = \begin{cases} \alpha \cdot f_{u,v,c} + (1-\alpha) \cdot b_{u,v,c}^{prev}, if\ s_{u,v} \neq 0 \\ b_{u,v,c}^{prev}, if\ s_{u,v} = 0 \end{cases} \quad (3.3)$$

where u and v are the image coordinates, c represents the color channel, b stands for background image, f is the foreground image (current frame), b^{prev} is the background image before updating, s is the pixel value in the segmented image and α represents the adaptation rate.

In order to update the background, all pixels of the foreground image are used which are assumed to not belong to the person image region. These pixels are black in the segmented image obtained after subtracting the current frame from the background image and segmenting it using the closed-loop segmentation method described later in this section. By using the background information from the current frame, the background image is updated to reflect the current scene illumination. The updated background image is then used for the next frame for improving the subtraction result.

If the parameter α is large, sudden illumination changes are immediately compensated for in the background image. A smaller value prevents pixels of the person, which have not been successfully segmented, to significantly contribute to the new background image. Depending on particularities of the scene where the system is set up, this parameter might be adjusted in order to obtain the best possible results.

Robust closed-loop segmentation

After successfully subtracting the background from the current frame, an intelligent segmentation must be applied to the subtracted image in order to obtain an optimally segmented image. An optimally segmented image does not contain noise in contrast to an over segmented image, where not only human body pixels but also background pixels are segmented. Also, an optimally segmented image does not have “holes” in the segmented human body region in contrast to an under segmented image, where important information is lost due to missing human body parts.

As already shown in Fig. 3.3, due to external influences, such as variable lighting conditions, the optimal threshold value is not the same for all the frames of a video sequence. In order to automatically select the appropriate threshold value for each frame and to make the operation robust against external influences, closed-loop segmentation is suggested in this thesis.

As in traditional control theory, a closed-loop in image processing always includes an actuator variable which influences the controlled variable. The actuator variable, which is an input variable, in image processing influences the image quality. The

measure of image quality is a controlled variable, which is an output variable. The chosen actuator variable for closed-loop segmentation is the threshold value, as it directly influences the segmented image characteristics. For the controlled variable the so-called two-dimensional (2D) entropy of segmented pixels introduced in [29] is used. The 2D entropy S is defined as:

$$S = -\sum_{i=0}^{8} p_{(1,i)} \log_2 p_{(1,i)} \tag{3.4}$$

where $p(1,i)$ is the relative frequency, that is, the estimate of the probability of occurrence of a pair $(1,i)$ representing the segmented pixel 1 (white) surrounded with i segmented pixels in its 8-pixel neighborhood:

$$p_{(1,i)} = \frac{\text{number of segmented pixels surrounded with } i \text{ segmented pixels}}{\text{number of segmented pixels in the image}} \tag{3.5}$$

Based on (3.4) it can be seen that the value of the 2D entropy is high when the probabilities of occurrence of multiple pairs $p(1,i)$, $i=0..8$ are high. This corresponds to image disorder, as many segmented pixels have a different number of directly connected segmented neighbors. In contrast to that, if the pixels are well connected, the majority of pixels will have exactly eight pixels in their neighborhood. This causes $p(1,8)$ to be close to 1 and all other probabilities to be close to zero, so the 2D entropy value to be small. In other words, a high value of the 2D entropy corresponds to images having scattered segmented pixels (for Fig. 3.2 a) S = 0.37 and for Fig. 3.2 b) S = 0.27). A lower value of the 2D entropy corresponds to segmented images of good quality having well connected segmented pixels. For example for Fig. 3.2 c) S = 0.10. Because of this, the problem of finding the optimal threshold can be defined as the problem of finding the minimum 2D entropy. The input-output characteristic corresponding to the segmentation of the image in Fig. 3.1 c) is given in Fig. 3.4. This characteristic shows the values of the 2D entropy of the segmented images obtained using different threshold levels Th from the possible range of threshold levels, which is 0-255 in the case of segmentation of gray-scale subtracted images. Fig. 3.4 also shows the number of segmented pixels in the image (scaled by dividing by 1.000.000) and the gradient of segmented pixels in the image (scaled by dividing by 100.000).

It can be seen in Fig. 3.4 that the 2D entropy is zero or has small values for very small thresholds, as all the image pixels would be segmented in this case and they would appear connected. Such small 2D entropy corresponds also to high thresholds

where only few grouped segmented pixels appear. The low values of 2D entropy for these two threshold regions indicate images that are not well segmented, as either all images pixels are segmented or no pixels are segmented at all. In order to overcome this shortcoming of the presented "searching for the minimum 2D entropy" approach, the number of segmented pixels and the gradient of the number of segmented pixels are used.

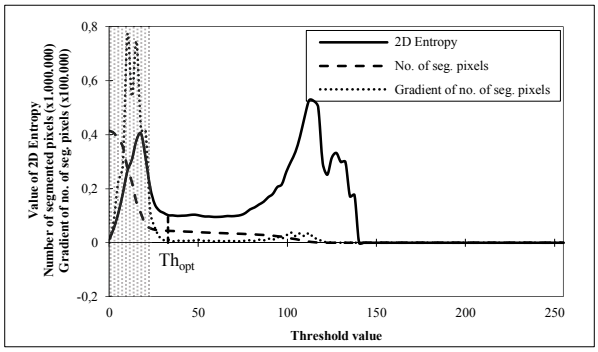


Fig. 3.4 Input-output characteristic "threshold-2D entropy" corresponding to segmentation of the image in in Fig. 3.1 c)

It can be observed that for the low threshold values, the total number of segmented pixels is reduced significantly between two consecutive threshold values. Therefore, in order to select the valid input region, the absolute value of the gradient of the number of segmented pixels is used to exclude threshold values that lead to a poorly segmented image. Multiple trials have shown that if the gradient of the number of segmented pixels is higher than 0.2% of the total number of pixels in the image, the segmentation result is poor. In Fig. 3.4 this excluded region appears darkened out and the optimal threshold value Th_{opt} is found at the first local minimum of the 2D entropy within the valid region and is marked with a dotted vertical line.

In order to find the optimal threshold, i.e. to find the first local minimum of the 2D entropy within the valid region, a control based on the gradient of the 2D entropy is employed. Extremum points can be detected by using the gradient of the 2D entropy, as the gradient equals to zero both for local minima and for local maxima. The sign of the gradient before and after the extremum point provides information of whether the extremum point is a local maximum or a local minimum. If the gradient turns from positive to negative values, the extremum point is a local maximum, otherwise it is a local minimum. Therefore the optimal threshold Th_{opt} is found at the first transition from negative to positive values of the gradient of the 2D entropy within the valid region. Fig. 3.5 shows the block diagram of the considered closed-loop segmentation algorithm.

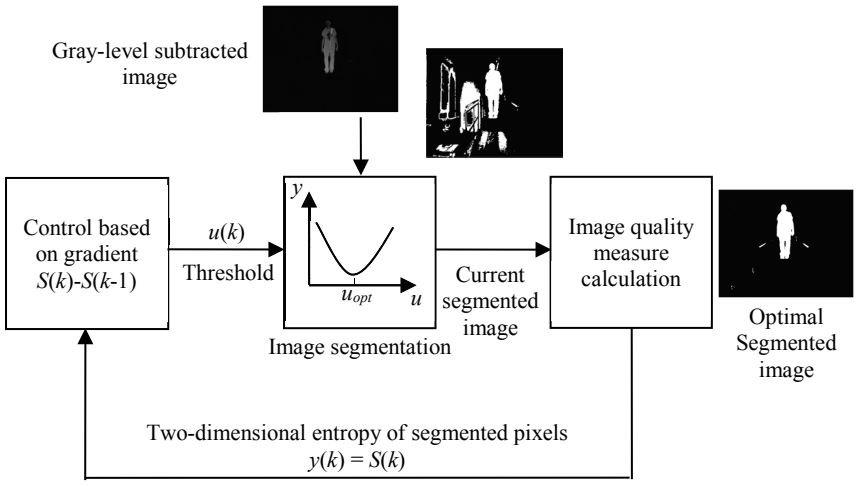
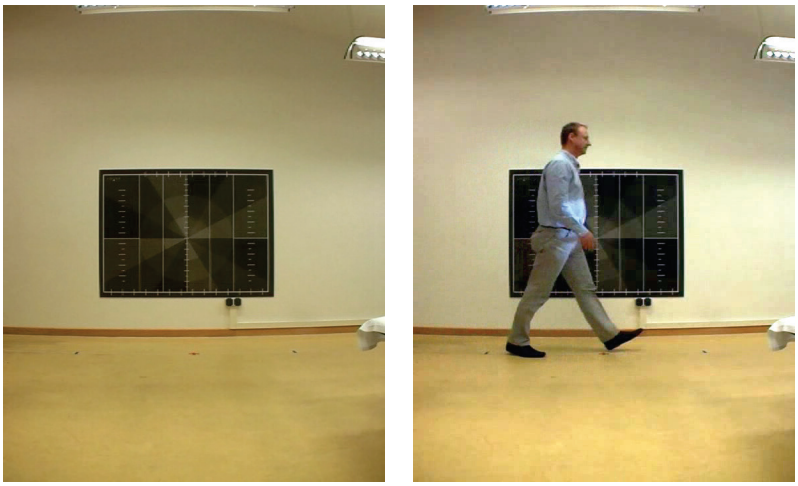


Fig. 3.5 Closed-loop segmentation of the gray-scale subtracted image

By using the closed-loop control, the presented system is robust to illumination changes that often appear during a video sequence, as it finds the optimal threshold for each frame separately.

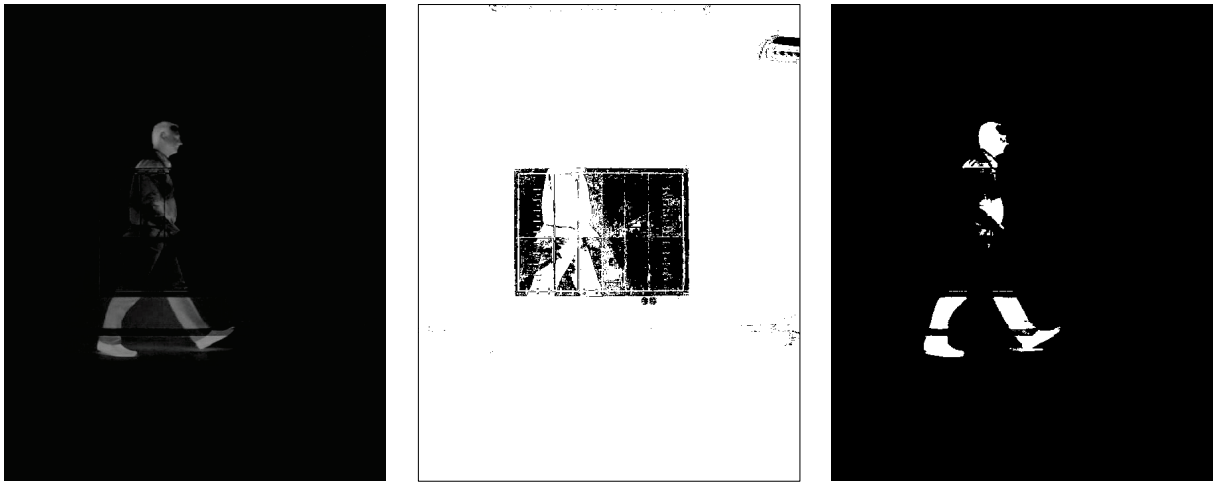
The presented closed-loop segmentation works well if either the complete person region is brighter than the background or the other way around. However, it does not work well if only parts of the person are brighter than the background while other parts are darker. This is the case in the clinical setup at the Neurological Rehabilitation Centre Friedehorst in Bremen, Germany [73] where the experiments needed for the development of the vision-based gait analysis system presented in Chapter 4 were carried out. In this setup the wall in front of which the person walks has both bright and dark regions, as shown in Fig. 3.6.



a) b)

Fig. 3.6 Example of images taken in clinical setup: a) Background image and b) Frame containing a person walking

Due to the bright and dark regions of the background, almost independent of the colors of the persons' clothing, there will be parts of the person which are darker and parts that are brighter than the background. This leads to the fact that the absolute background subtraction in (3.1) introduces considerable noise, as illustrated in Fig. 3.7 a) by the bright regions of the image, which do not belong to the person. Such noisy background subtracted image cannot be optimally segmented using a single threshold, as can be seen in Fig. 3.7 b) and c) as parts of the image are either over segmented or under segmented.



a) b) c)

Fig. 3.7 Example of: a) Absolute background subtraction and subtracted image segmentation using: b) Th = 18 and c) Th = 80

In order to overcome this problem, instead of using the absolute background subtraction introduced in (3.1), two operations are proposed, resulting in two subtracted images:

$$r1_{u,v,c} = b_{u,v,c} - f_{u,v,c} \tag{3.6}$$

$$r2_{u,v,c} = f_{u,v,c} - b_{u,v,c} \tag{3.7}$$

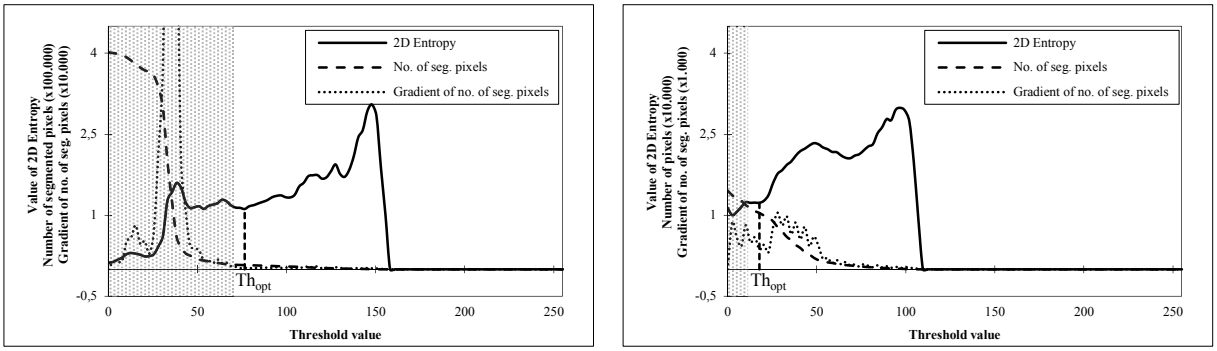
where u and v are the pixel image coordinates, c represents the color channel, b stands for the background image, f is the foreground image and $r1$ and $r2$ are the two resulting subtracted images shown in Fig. 3.8 a) and b) respectively.

These two images are then segmented separately by finding the optimal thresholds Th_{opt1} and Th_{opt2} using the proposed closed-loop segmentation method. Finally the two segmented images are merged by the logical OR operation. The result after combining the two segmented images can be seen in Fig. 3.8 c).

a) b) c)

Fig. 3.8 Example of background subtraction: a) Background – Foreground (r1), b) Foreground – Background (r2) and subtracted image segmentation using: c) $Th_{opt1} = 80$, $Th_{opt2} = 18$

In order to automatically determine the two optimal threshold values Th_{opt1} and Th_{opt2}, the presented closed-loop algorithm is applied individually on the two images. As in Fig. 3.4, the regions excluded with the help of the gradient of the number of segmented pixels appear darkened out in Fig. 3.9, so the optimal threshold value is found at the first local minimum of the 2D Entropy outside the excluded region.



a) b)

Fig. 3.9 Input-output characteristic of the 2D entropy corresponding to segmentation of the images in: a) Fig. 3.7 b) (Thopt1=80) and b) Fig. 3.7 c) (Thopt2=18)

The separate segmentation of the two images introduces additional robustness against variations in background colors and the persons' clothes compared to segmenting the image obtained by absolute background subtraction. Therefore this method can also be applied for the previously shown scenario in which the background does not contain such different colors. However, computing two threshold values requires more processing power, so it should only be used if necessary.

3.2 Disparity map segmentation

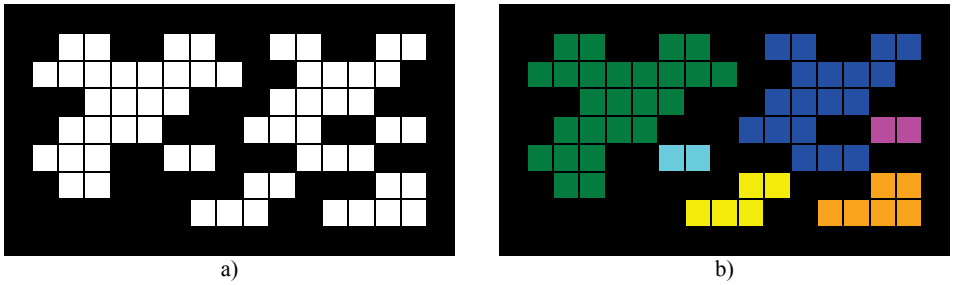
In vision-based applications where stereo cameras with parallel optical axes and equal focal lengths are used, disparity maps can be computed after performing stereo rectification on the stereo image pair, as described in section 2.6. In order to enable feature extraction for object recognition, the disparity map has to be segmented, which means that objects have to be separated from each other and from the background. This applies to human detection as well, as humans are considered to be a particular type of objects.

In some applications such as automotive applications, which are the object of consideration in Chapter 5 of this thesis, there are mainly two directions of disparity map segmentation. One direction is to use the so called bird perspective [31], which presents the top view of the scene, as known from radar applications, where after removing the ground plane the pixels are grouped into objects. The second main direction is the so called U/V Disparity, which consists in two projection images of the disparity map. In these projections, the u respectively the v image axis is the axis on which the projection is applied while the other axis is the disparity value. The pixel intensity represents the number of pixels having a specific coordinate and a specific disparity value. The V Disparity is used to detect the road surface and obstacles lying on it as a surface in the disparity map is mapped to a line in the V Disparity [60][61]. Object detection under these circumstances however is not always reliable enough, so another approach helps segmentation by using pre-processed radar signals [59] in order to generate "masterpoints" that validate the result. These methods output a rectangular region of interest (ROI) for each object of the scene.

In contrast to state-of-the-art methods, the novel segmentation algorithm presented in this thesis uses directly the disparity map, and does not depend on the image content. The method is based on the two-pass algorithm for blob detection in binary images introduced in [12] and improved in [23]. However, the method has been adapted in this thesis to segment gray-scale disparity maps. In the first pass the segmented pixels are labeled and an equivalence list is kept, while in the second pass this list is used to update the labels as final.

The output of the algorithm is a labeled image containing separated objects, as it gives a different label for each object. This labeling is also known as blob detection and can be performed for 4-neighbors connectivity or for 8-neighbors connectivity. The number of considered neighbors influences the way pixels are grouped together. If 4 neighbors are considered, the upper, lower, left and right neighbors of a pixel will

bear the same label as the considered pixel. In the case of considering 8 neighbors, all neighbors of a pixel will get the same label. Fig. 3.10 illustrates the result of 4-neighbors blob detection applied to a binary image, where, for better visualization, each label is represented by a color.



a) b)

Fig. 3.10 Illustration of blob detection: a) Binary image and b) Extracted blobs

In the first pass of the 4-neighbors blob detection the input binary image is scanned row by row starting from the left upper image corner and only two already labeled neighboring pixels, left and upper, are checked when labeling the current pixel. The labeling of the pixel (u,v) is performed according to:

$$l_{u,v} = \begin{cases} l_{u-1,v}, & l_{u-1,v} \neq 0 \; AND \; l_{u,v-1} = 0 \\ l_{u,v-1}, & l_{u-1,v} = 0 \; AND \; l_{u,v-1} \neq 0 \\ \min(l_{u-1,v}, l_{u,v-1}), & l_{u-1,v} \neq 0 \; AND \; l_{u,v-1} \neq 0 \\ next \; label \; value, & otherwise \end{cases} \tag{3.8}$$

where $l_{u,v}$ is the current label and $l_{u-1,v}$ and $l_{u,v-1}$ are the labels of the left and upper neighbor respectively. Besides the labeling, an equivalence table *eq* is kept in the first pass in order to memorize equivalent labels. This table is updated if the current pixel to be labeled has segmented left and upper neighbors with different labels. If so, the equivalence table *eq* will memorize the fact that the two different neighboring labels actually belong to the same region (are equivalent).

In the second pass, each label gets the new value of its minimum equivalent according to the equivalence table:

$$l_{u,v} = \min(eq[l_{u,v}]), \; u = \overline{0..imWidth-1}, \; v = \overline{0..imHeight-1} \tag{3.9}$$

where $l_{u,v}$ is the label for the pixel located at the coordinates *(u, v)*, *imWidth* is the image width and *imHeight* is the image height.

The result of the two pass pixel labeling can be seen in Fig. 3.11. It can be easily seen that each label in Fig. 3.11 b) corresponds to a color in Fig. 3.10 b).

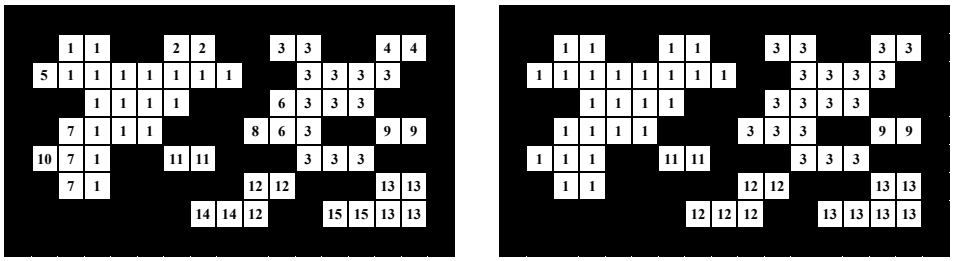


Fig. 3.11 Illustration of blob detection algorithm: a) Result of first pass and b) Result of second pass

As previously mentioned, this algorithm was designed for binary images. However, disparity maps are gray-scale images, in which darker pixels correspond to objects which are further away, while brighter pixels correspond to objects which are closer to the camera. In order to segment disparity maps, the above described pixel labeling algorithm was adapted to group pixels together that belong to the same object, therefore assigning them the same label.

The main assumption for disparity map segmentation is that neighboring pixels in the disparity map belonging to the surface of an object, have similar disparity values, while on the edges of the object the difference in disparity value between the pixels of the object and of the background is large. These transitions in disparity values are used for the segmentation. Accordingly, the pixel labeling algorithm (3.8) was modified to consider neighboring pixels that have disparity values close to the current pixel as belonging to the same object. If the difference in disparity values is bigger than a set threshold *Th*, the current pixel will be considered to be part of a different object. The modified equation (3.8) is:

$$l_{u,v} = \begin{cases} l_{u-1,v},\ p_{u,v} - p_{u-1,v} < Th\ AND\ l_{u-1,v} \neq 0\ AND\ l_{u,v-1} = 0 \\ l_{u,v-1},\ p_{u,v} - p_{u,v-1} < Th\ AND\ l_{u-1,v} = 0\ AND\ l_{u,v-1} \neq 0 \\ \min(l_{u-1,v}, l_{u,v-1}),\ (p_{u-1,v}\ AND\ p_{u,v-1} are\ ok)\ AND\ l_{u-1,v} \neq 0\ AND\ l_{u,v-1} \neq 0 \\ next\ label\ value,\ otherwise \end{cases} \tag{3.10}$$

where $p_{u,v}$ are the pixel values and $(p_{u-1,v}\ AND\ p_{u,v-1}\ are\ ok)\ AND\ l_{u-1,v} \neq 0\ AND\ l_{u,v-1} \neq 0$ is the shorter form of: $p_{u,v} - p_{u-1,v} < Th\ AND\ p_{u,v} - p_{u,v-1} < Th\ AND\ l_{u-1,v} \neq 0\ AND\ l_{u,v-1} \neq 0$. The threshold *Th* varies from pixel to pixel and depends on the disparity value of the current pixel, being equal to 10% of the disparity value. For example, if the disparity value of the pixel is 10 then *Th*=1 and if the disparity value is 30 then *Th*=3. Therefore the nonlinear nature of the disparity values is accounted for. An illustrative example of the segmentation of a disparity map according to (3.10) is given in Fig. 3.12.

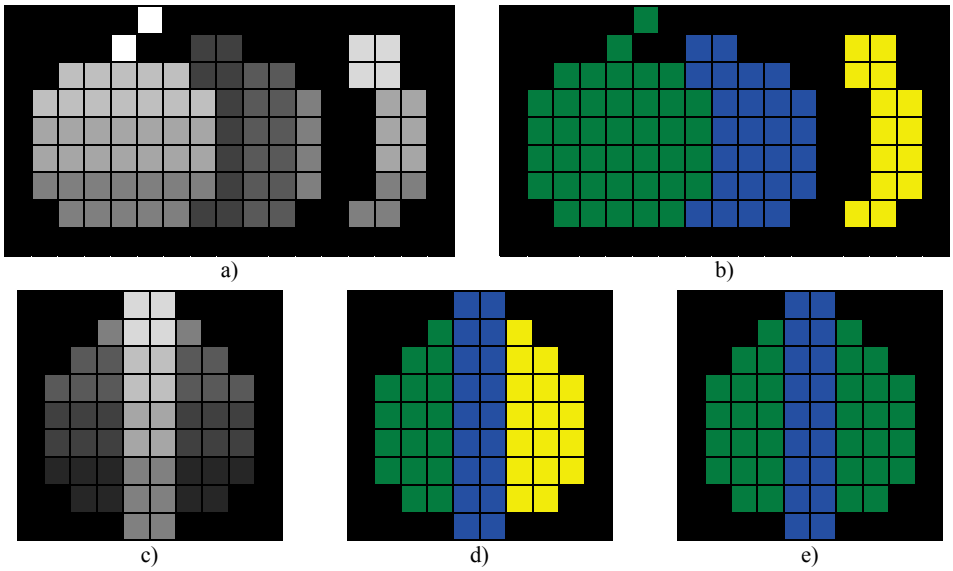


Fig. 3.12 Illustration of disparity map segmentation and merging: a) Disparity map, b) Segmented disparity map, c) Disparity map, d) Segmented disparity map and e) Merged segmented disparity map

Fig. 3.12 a) and c) show synthetic disparity maps that are to be segmented so that pixels belonging to the same object are grouped together. It can be observed in Fig. 3.12 d) that after applying the proposed segmentation method, the vertical object is dividing the object behind it in two parts. However, after applying the segmentation in (3.10), the disparity values on the edges of the differently segmented objects are analyzed. This post-processing step reveals for instance that the right edge of the green object in Fig. 3.12 d) has the same disparity values as the left edge of the yellow object in Fig. 3.12 d), as can be seen in Fig. 3.12 c). This indicates the fact that these two parts belong to the same object and that they should bear the same label. The result of merging the two object parts is shown in Fig. 3.12 e).

After segmentation and merging object parts, another post-processing step is applied to remove segmented objects having a smaller area than a given threshold. This post-processing step removes noise from the segmented image, as the objects of interest always have a known minimum size, which is chosen as threshold value.

In order to evaluate the performance of the presented algorithm and to prove its generic nature the famous Tsukuba image from the Middlebury dataset [33] was used as benchmark. The results are shown in Fig. 3.13.

Fig. 3.13 Disparity map segmentation for the Tsukuba image: a) Original left image, b) Original right image, c) Ideal disparity map, d) Disparity map computed using block matching, e) Segmented ideal disparity map, f) Segmentation of computed disparity map

Fig. 3.13 a) and b) show the original left and right Tsukuba images. Fig. 3.13 c) shows the ground truth disparity map provided with the original images and Fig. 3.13 d) shows a real disparity map computed using the block matching algorithm. Fig. 3.13 e) and f) show the result of segmentation of the ideal and of the computed disparity maps using the proposed algorithm. It can be noticed that the ideal disparity map does not contain noise and therefore the resulted segmented image contains all objects pixels. The real disparity map contains small regions that are considered as noise and removed during segmentation due to their small size. In the Tsukuba image the difference in disparity values between the video camera and the background is the

same as the difference between the regions of the statue. However, the threshold *Th* used for segmentation according to (3.10) is proportional to the disparity value, which is higher for the head statue. Therefore the video camera is not merged with the background, but the regions of the statue are merged.

In real-world disparity maps pixels belonging to an object often have slightly different disparity values if the objects are not parallel to the image plane. This is often the case in service robotics applications, when the camera is tilted with respect to the imaged scene, as illustrated in Fig. 3.14 b).

Fig. 3.14 Examples of object detection for street scene: a) Original left image, c) Disparity map and e) Segmented disparity map and home (service robotics) scene: b) Original left image, d) Disparity map and f) Segmented disparity map

Fig. 3.14 b) shows an image of a home scene captured with the stereo camera of a service robot. In this service robotics application, as described in [6] the goal is the 3D reconstruction of the objects on the table to be grasped by the robot. The corresponding computed disparity map, obtained using the block matching method is shown in Fig. 3.14 d). In contrast to service robotics applications, in automotive applications the objects are usually perpendicular to the ground and almost parallel to the camera as can be seen in Fig. 3.14 a). However, there are still different disparity values on the object's surface if the object is not perfectly parallel to the camera plane, as is the case of the bicycles shown in Fig. 3.14 c). Even though the two scenarios are different, the algorithm presented in this thesis performs well in both scenarios, as illustrated by in Fig. 3.14 e) and f).

It can be observed that for the Tsukuba image as well as for the two real-world scenes, after segmentation the objects are separated from each other and from the background. However, as explained in section 2.6, real-world disparity maps often contain regions without disparity information. This strongly influences the quality of the segmentation as can be seen in Fig. 3.15 a)-c) and Fig. 3.16 a)-e), where the segmented regions are overlaid on the original left image.

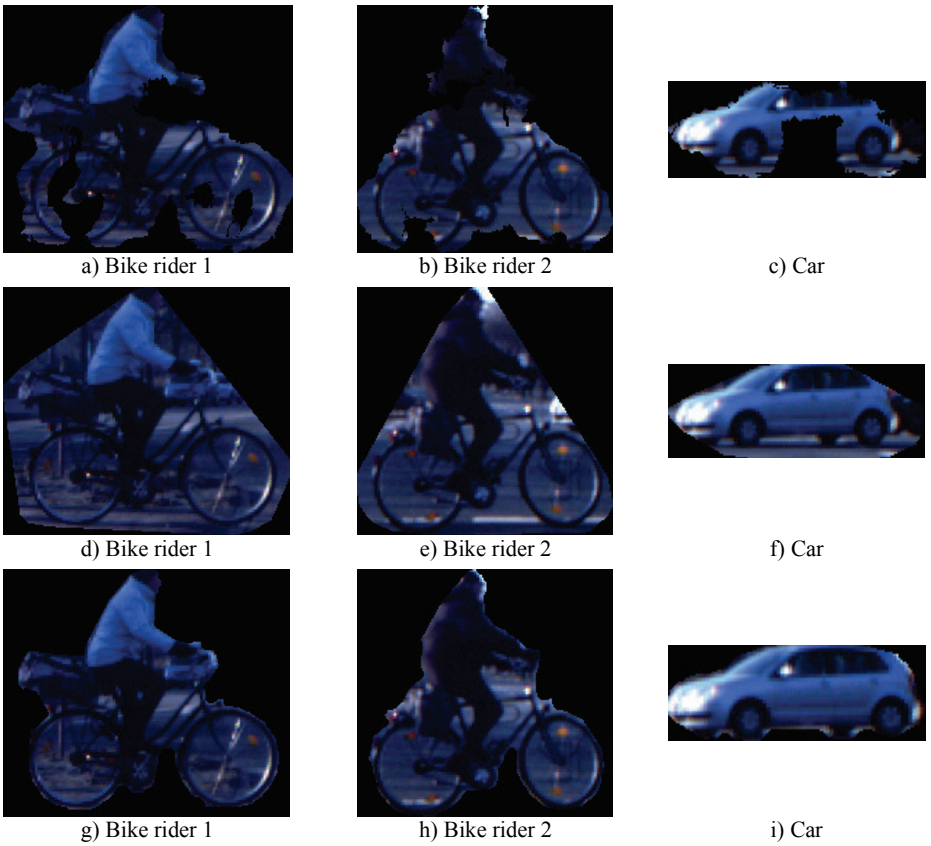


Fig. 3.15 Result of segmentation for the street scene: a)-c) Segmentation of individual objects from the street scene, d)-f) Segmentation result after convex hull fitting, g)-i) Manually obtained ground truth

Fig. 3.16 Result of segmentation for the home scene: a)-e) Segmentation of individual objects from the home scene, f)-j) Segmentation result after convex hull fitting, k)-o) Manually obtained ground truth

In order to improve the segmentation result by compensating for the missing regions, a convex hull is fitted on each of the segmented objects. The convex hull fitting is performed by finding points on the edge of the segmented object which, if consecutively connected with straight lines, generate a convex polygon [32]. The result of convex hull fitting on the objects shown in Fig. 3.15 a)-c) and Fig. 3.16 a)-e) is presented in Fig. 3.15 d) – f) and Fig. 3.16 f)-j). As evident, much larger portions of the object are segmented and the result is close to the manually extracted ground truth shown in Fig. 3.15 g) – i) and Fig. 3.16 k)-o).

The fitted convex hulls define objects ROIs of irregular shape which fit the objects better than traditional rectangular bounding boxes. Such ROIs are appropriate inputs for reliable object features extraction as they contain mainly object pixels and less background pixels. A problem however appears with object occlusion as is the case of the box shown in Fig. 3.16 b): this object is partially occluded by two other objects. As shown in Fig. 3.16 g), after convex hull fitting, the object ROI includes two other objects. This problem can be solved by using the distance information encoded in the disparity map. Evidently objects further away from the camera are occluded by the ones closer to the camera. Having this in mind, the convex hulls of occluding objects are subtracted from the convex hull of the occluded object, as shown in Fig. 3.17 b).

a) b)

Fig. 3.17 Effects of occlusion handling: a) Convex hull result and b) Occlusion handling result

It can be observed that after occlusion handling the result of convex hull fitting contains almost only object pixels and is therefore close to the manually extracted ground truth shown in Fig. 3.16 l).

3D distance and size reconstruction for segmented objects

After segmentation and convex hulls fitting, the distance from the left camera to each segmented object can be computed and the 3D size (width and height) of the objects can be reconstructed. For computing the distance, the image coordinates of each object's closest point to the camera are computed and applied in equation (2.18). The result is a 3D point *A(x, y, z)* for each object, which describes the 3D position of the object's closest point with respect to the left camera.

Based on the computed 3D point, the distance between the camera and each object can be computed using this equation:

$$D = \sqrt{x^2 + y^2 + z^2} \tag{3.11}$$

where D is the distance from the camera center to the object's closest point and x, y, z are the 3D coordinates of the object's closest point.

Additionally to the distance, for each of the objects in Fig. 3.15 a)-c) and Fig. 3.16 a)-e) the 3D width and height of the object were computed and compared to ground truth data in order to evaluate whether these features can be reliably used for classification. The 3D with W and height H of objects express the object size in meters and are independent of the distance between the object and the camera. In contrast to this, the 2D with w and height h of objects are expressed in image pixels and change depending on the distance between the object and the camera (distant objects appear smaller).

As already explained, each segmented object is bounded by a convex hull that defines a ROI of irregular shape. Additionally, objects are bounded with a rectangular bounding box. For computing the object's 2D width w and height h, firstly the left upper corner $c_{LU}(u_{LU}, v_{LU})$ and the right bottom corner $c_{RB}(u_{RB}, v_{RB})$ of the 2D bounding

box are calculated. From these, the 3D coordinates of the left upper corner $C_{LU}(x_{LU}, y_{LU}, z_{LU})$ and right bottom corner $C_{RB}(x_{RB}, y_{RB}, z_{RB})$ are computed by applying (2.18).

Subsequently the object's 3D width W and height H in meters are computed using:

$$W = x_{RB} - x_{LU}, \; H = y_{LU} - y_{RB} \tag{3.12}$$

Table 3.1 and Table 3.2 show the reconstructed sizes of the objects (width and height) and their distances to the camera. In this table the object sizes and distances extracted using the presented algorithm are denoted by "Vision" and the real-world values of object sizes and distances are denoted by "Ground Truth" (ground truth data). The ground truth distance was measured with a Bosch PLR 50 laser range finder [81], while the ground truth sizes were manually measured. In Table 3.1 and Table 3.2 both the absolute and relative errors are given.

Table 3.1 Distance and size for objects in the street scene

		Objects		
Dimensions		Bike 1	Bike 2	Car
Width (m)	Vision	2.12	1.94	4.41
	Ground Truth	2.05	2.05	4.21
	Error	0.07 (3%)	0.11 (5%)	0.20 (5%)
Height (m)	Vision	1.82	1.88	1.62
	Ground Truth	1.80	1.85	1.62
	Error	0.02 (1%)	0.03 (2%)	0.00 (0%)
Distance (m)	Vision	10.29	10.79	31.74
	Ground Truth	10.15	10.56	30.50
	Error	0.14 (1%)	0.23 (2%)	1.24 (4%)

Table 3.2 Distance and size for objects in the home scene

		Objects				
Dimensions		Jar	Box	Bottle	Glass	Mug
Width (m)	Vision	0.11	0.23	0.09	0.10	0.09
	Ground Truth	0.10	0.20	0.08	0.08	0.09
	Error	0.01 (10%)	0.03 (15%)	0.01 (13%)	0.02 (25%)	0.00 (0%)
Height (m)	Vision	0.17	0.19	0.24	0.15	0.16
	Ground Truth	0.18	0.12	0.24	0.15	0.14
	Error	0.01 (6%)	0.07 (58%)	0.00 (0%)	0.00 (0%)	0.02 (14%)
Distance (m)	Vision	1.34	1.51	1.32	1.43	1.58
	Ground Truth	1.33	1.50	1.32	1.42	1.57
	Error	0.01 (1%)	0.01 (1%)	0.00 (0%)	0.01 (1%)	0.01 (1%)

It can be seen that the accuracy of object feature extraction compared to the ground truth data is mostly below 5%, which indicates the accuracy of the proposed disparity map segmentation to separate objects from each other and from the background.

However, for small objects large relative errors can occur (over 10%), which indicates that the presented method is not accurate enough for precise object reconstruction.

In the presented scenes, the pixels of the dominant plane on which the objects to be segmented stand (road in the street scene and table in the home scene) did not appear in the disparity map because of the lack of texture and due to the property of the used block matching method of not being able to find correspondences for textureless surfaces.

If the dominant plane is well textured or other algorithms than block matching are used, the disparity map may contain pixels of the dominant plane that could negatively influence the segmentation result. In this situation, the pixels belonging to the dominant plane have to be removed prior to applying the proposed disparity map segmentation. A simple but effective way for ground plane removal, which is the dominant plane for human tracking applications, is proposed in section 5.5.3. However, a more general approach can be found in [62], where values of the disparity map are processed using the so-called RANSAC method in order to estimate the equation of dominant plane. Subsequently, all pixels belonging to the estimated dominant plane are removed from the disparity map.

4

Indoor human tracking – application in clinical gait analysis

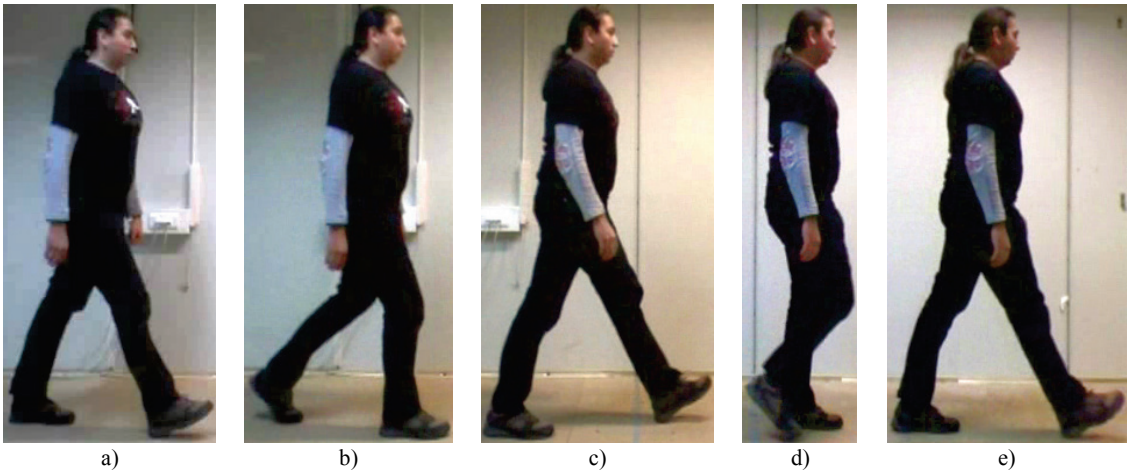
Individuals who suffer from gait abnormalities, due to stroke, an orthopedic surgery, or other motor function or neurological disabilities, can benefit from supervised gait training in order to relearn independent walking. The gait rehabilitation is a complex and long lasting process. During this process medical experts and physiotherapists spend significant time and effort together with their patients to continuously monitor and correct the patients' gait patterns. For the gait assessment observational gait analysis is traditionally used by clinicians who observe their patients during walking in order to deduce whether their gait has improved. However, there are several concerns regarding the limitations of observational gait analysis, as it is subjective and highly dependent on the experience and judgment of the clinician [24]. In order to get objective results, as necessary for successful gait rehabilitation, a variety of motion capture systems are used in the diagnosis and monitoring of gait irregularities [25]. In the last decades, gait rehabilitation has arisen as a growing application field for the reconstruction of human motion from image sequences [26][19].

Currently, the majority of commercially available visual systems used for motion analysis are marker-based, such as the Vicon system [71]. Marker-based solutions rely

primarily on markers or sensors attached at key locations on the human body. Although they can accurately measure the gait parameters, existing marker-based motion capture systems have a number of disadvantages including high expense, complexity of setup, marker placement error and their invasive nature, which is a major disadvantage for patients already suffering of various gait disorders. In order to overcome these problems, significant effort has been made in recent years to develop markerless motion capture methods [45][47]. However state-of-the-art vision-based markerless motion analysis systems often require a carefully controlled environment for reliable functioning, including a specially colored background and specially colored patients' clothes [48]. The reason for these constraints is the difficulty of image segmentation, which is the key component in the majority of current markerless vision-based motion analysis systems [47][48][49]. An approach that does not require a controlled environment is described in [50]. However, in this approach the human body has to be scanned using a 3D scanner before running the vision-based algorithm.

4.1 Features required for gait analysis

The human gait is a cyclical process that can be divided in two main phases: stance phase and swing phase [15]. During the stance phase the leg supports the person's weight, while during the swing phase the leg is swung from back to front. These two phases are complementary for the two legs, as while one leg is in the stance phase the other leg is in the swing phase. Between these two phases there are double support phases, when both feet are on the ground. The gait cycle can also be divided in five phases: initial contact, loading response, terminal stance, initial swing and terminal swing. These five phases, exemplified for the right leg, are illustrated in Fig. 4.1.

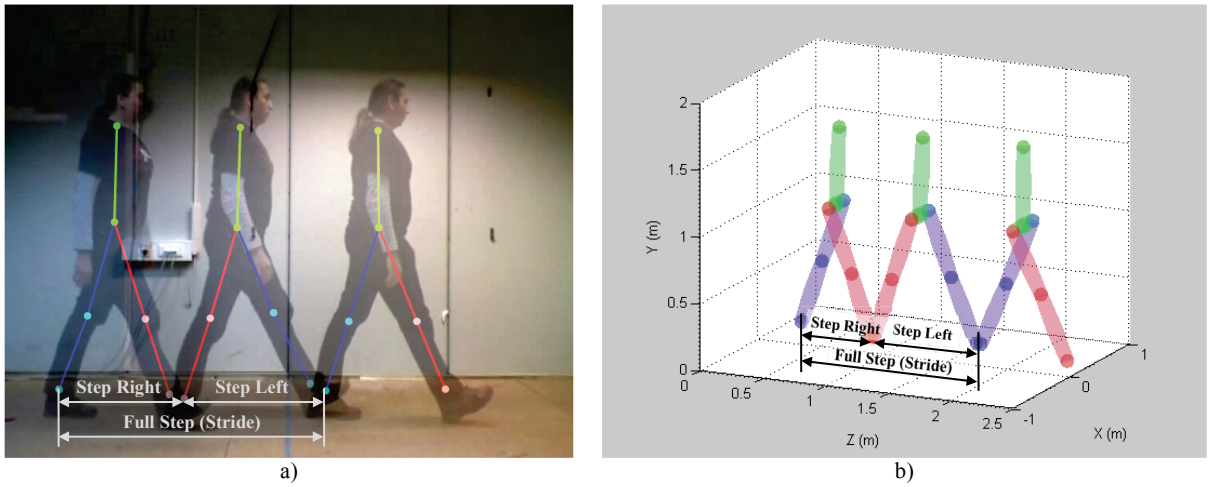


a) b) c) d) e)

Fig. 4.1 Illustration of the five phases of the human gait, exemplified for the right leg:
a) Initial contact, b) Loading response, c) Terminal stance, d) Initial swing, e) Terminal swing

In clinical gait analysis images of the frontal and the sagittal plane of the patient are important as gait parameters from both planes are needed to effectively describe the gait. Also the combination of these views offers important visual information to the clinicians.

According to physicians there are a series of gait parameters which are relevant for analyzing the gait during both the diagnosis and the rehabilitation process. These parameters are defined based on the gait phases and include step length, stride length, cadence, speed and angles between body parts like hip and knee angles. Gait parameters can be defined from the 2D or 3D human body model extracted from images as illustrated in Fig. 4.2 and Fig. 4.5.



a) b)

Fig. 4.2 Illustration of step length measurement during one gait cycle: a) 2D image with overlaid extracted human body model and b) 3D human body model in the form of a stickman and step length and stride length definition

In order to measure step length, stride length, cadence and speed, the location where the foot sets on the ground must be detected in the image. Step and stride length can be measured by computing the distance between the ankle joint locations in three different gait phases, as illustrated in Fig. 4.2. These gait phases are: right initial contact, left initial contact and right initial contact again.

After the step length has been determined, speed is computed as the distance walked during a known amount of time, while the cadence represents the number of steps walked per minute. Additional to these standard gait parameters, joint angle variations can be computed if the 3D model of the person shown in Fig. 4.3 is projected on the frontal (XY) and sagittal (YZ) plane, as illustrated in Fig. 4.4 a) and b).

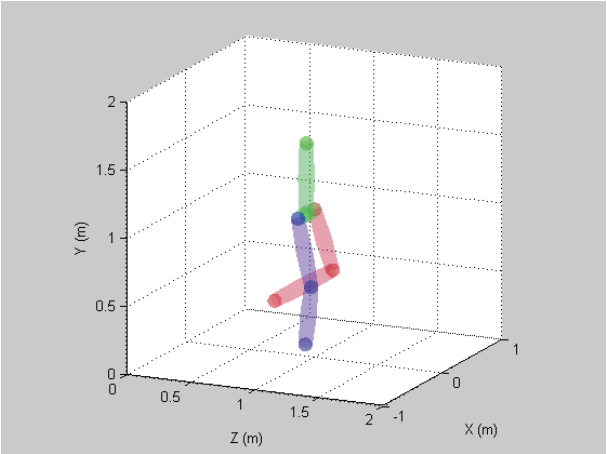
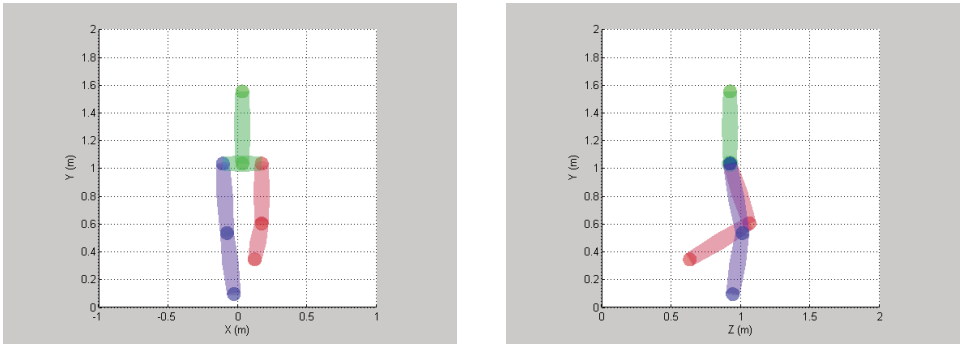


Fig. 4.3 Example of 3D model of the human body

For better visualization the two legs of the 3D model of the human body in the form of stick figure are represented with different colors. However, there is no convention which leg should be represented with which color. In the 3D model the joints are represented by spheres while the limbs are drawn as cylinders.



a) b)

Fig. 4.4 Example of back projected 3D model a) on the XY plane (frontal) and b) on the YZ plane (sagittal)

In a number of published works it has been shown that joint angles in the frontal and sagittal plane are an effective means for characterizing human gait [14][15]. These angles, θ_{Fi} and θ_{Si} where i=1..5 are illustrated in Fig. 4.5 a) and b) as the angles between the parts of the human body model in the form of stick figures. The indices i=1..5 denote the following angles:

1 – torso angle;

2, 4 – left and right thigh angles;

3, 5 – left and right shank angles.

Beside these angles it is of particular interest for gait analysis [14][15] to extract the knee angles θ_{K1} and θ_{K2} which are the angles between the thighs and shanks in the sagittal plane as shown in Fig. 4.5 b), as well as the hip ankles θ_{H1} and θ_{H2}.

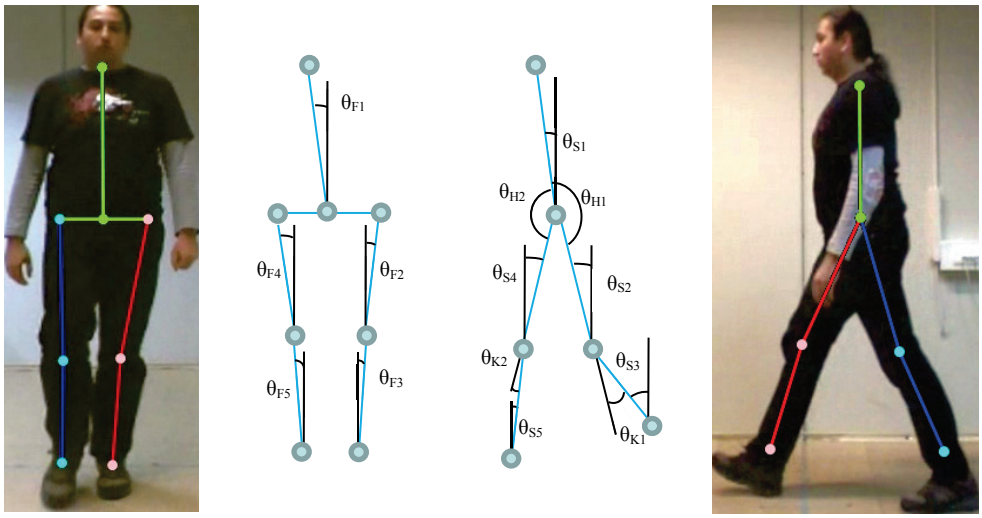


Fig. 4.5 Illustration of extracted gait features: Original image of a person walking in a) frontal and d) sagittal plane overlaid with stick figures; human body models in b) frontal and c) sagittal plane in the form of stick figures with the joint angles

Even though angles between body parts can be extracted from single 2D images, they need to be compensated due to the perspective transformation applied by the camera when the scene is projected on the 2D image plane. The compensation is needed as perspective projections do not preserve angles. Additionally certain gait parameters such as step length and walking speed cannot be extracted from single 2D images, as there is no relationship between the step length in image coordinates and the step length in real-world coordinates, expressed in meters. This is due to the fact that the depth is lost when the scene is projected on the image sensor.

For these reasons two cameras are used for imaging the walking person, which are placed orthogonal to each other, as illustrated in Fig. 4.6. This enables the reconstruction of the 3D location of individual points, particularly of joints, with the help of epipolar geometry, as will be shown in the following sections. These joints are then used to extract gait features such as step length in real-world coordinates. Also, angles extracted using 3D joints are already compensated for projective distortions.

4.2 Vision system for markerless gait analysis

The considered vision-based gait analysis system uses two Panasonic NV-GS 330 camcorders, which are placed orthogonal to each other and a notebook on which the image processing algorithms run. In the following sections, the camera imaging the person from the front will be referred to as “frontal camera”, while the camera imaging the walking person from the side will be referred to as “sagittal camera”, as depicted in Fig. 4.6.

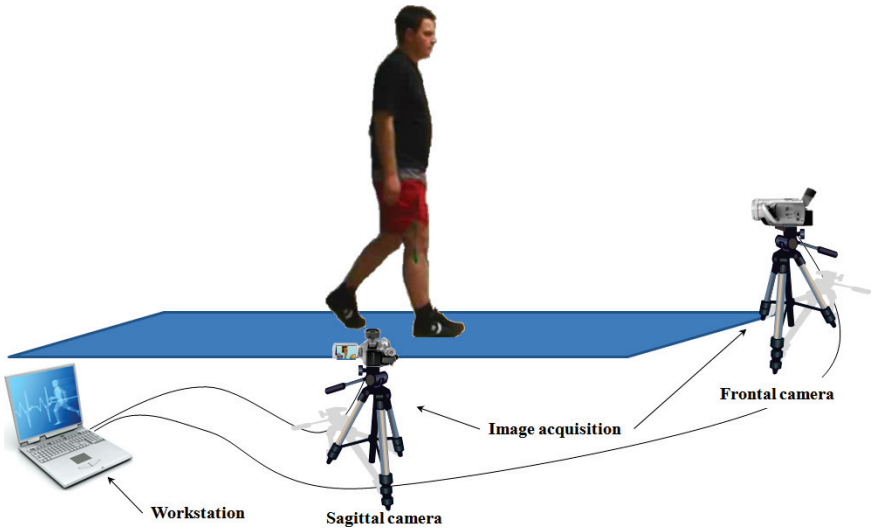


Fig. 4.6 System layout for vision-based markerless clinical gait analysis

In order to ensure that the cameras are orthogonal to each other a special cubic calibration pattern is used, introduced in [8]. This pattern can be seen in Fig. 4.7 a). Depending on the position of the camera relative to the calibration pattern, the image of the calibration pattern can appear as the images shown in Fig. 4.7 b)-d). The appearance shown in Fig. 4.7 d) assists the user in properly placing the cameras as it ensures that each camera is parallel to one side of the calibration cube. Since the sides of the cube are orthogonal to each other this implicitly means that the cameras are orthogonal to each other.

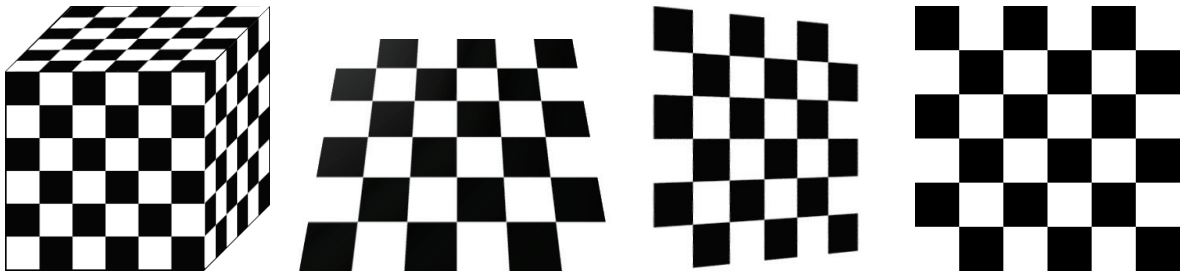


Fig. 4.7 Camera calibration cube a) and camera views of the calibration pattern for different positions of the camera with respect to the calibration cube: b) camera places too low, c) camera is rotated to the left and d) camera is parallel to the calibration cube

In order to actively help the clinician in placing the cameras, the angle between the image plane and the facing side of the calibration cube is continuously extracted with the help of chessboard corner detection and Tsai's calibration method [14]. Information about the positioning of the cameras and how they should be adjusted in order to be parallel to the calibration pattern is given on the graphical user interface (GUI) as visual feedback.

However, due to small mechanical misalignments, the cameras are unlikely to be placed perfectly orthogonal to each other. Nevertheless, the exact positioning of the

cameras is not critical for the presented system, as their relative position is determined during calibration and corresponding compensations are done when extracting the features, as will be explained in the following sections.

4.3 Markerless vision-based clinical gait analysis

In order to reliably extract the required gait features, videos of the human are recorded for both the frontal and the sagittal plane. For each pair of frontal and sagittal images, in both images 2D joint features are extracted, which are then combined into a simplified 3D model of the imaged person. This model enables the computation of all required gait features, as presented in the previous section, which are then used for gait analysis. An overview of the proposed image processing chain for robust gait features extraction for the purpose of gait analysis is shown in Fig. 4.8.

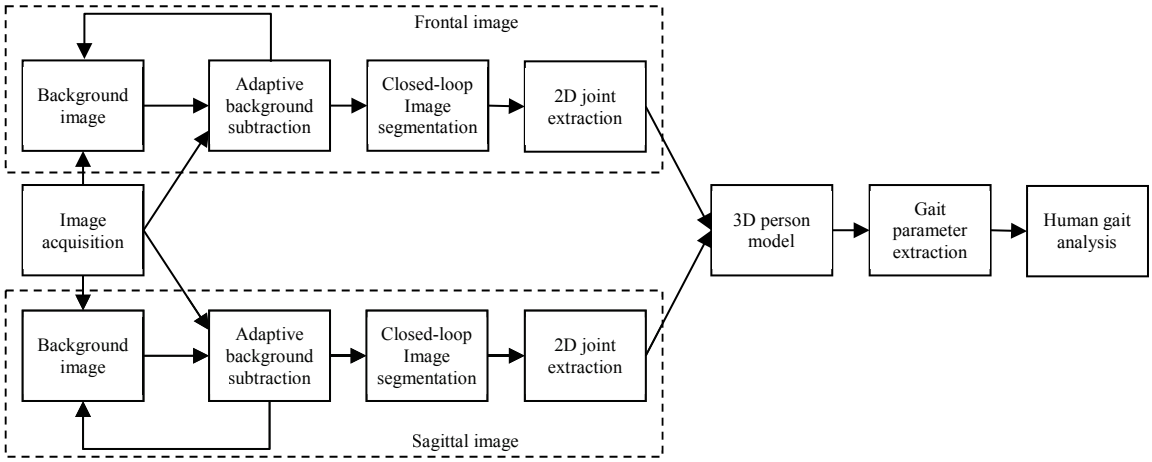


Fig. 4.8 Block-diagram of the proposed vision system for gait analysis

The image processing system was designed having in mind the indoor nature of the application, so adaptive background subtraction is employed as described in section 3.1. For the frontal video a background image without person is captured separately, while for the sagittal video the background is automatically extracted as will be shown in the following sections. For both the frontal and the sagittal videos, the adaptive background subtraction updates the background image to match the current illumination condition.

The subtracted images are then segmented using the closed-loop segmentation introduced in section 3.1. The result of the segmentation enables 2D joint feature extraction, as will be shown in the next section.

Automatic background extraction for human gait analysis

The adaptive background subtraction algorithm presented in section 3.1 gives the best results if the initially used background image has the same illumination as the first frame of the video sequence, as in this case the pixels belonging to the human region are bright, while the pixels belonging to the background region are dark. Such a background image can be obtained by capturing a background image immediately before the human enters the scene. However, in practice it often proves to be difficult to capture both the frontal and the sagittal background images without the person, especially in small sized rooms. Also, gait analysis protocols exist for which only the sagittal video is recorded, as it contains more valuable information for clinicians than the frontal video [15]. It will be shown in following that for the sagittal image the background can be automatically extracted for already captured videos in which the human is visible in all frames and for which no background image is available.

The solution to automatic background extraction from existing video sequences is the use of a temporal median filter [16]. The operation of the temporal median filter is similar to the widely used spatial median filter [16], which uses all pixels in a chosen neighborhood of a considered pixel and sorts them in ascending order. The output of the spatial median filter, the processed value of the considered pixel, is the value that lies in the middle of the sorted array. In contrast to this, the temporal median filter uses temporal neighboring pixels instead of spatially neighboring pixels, as can be seen in Fig. 4.9, where *P(u, v)* is the pixel intensity at the coordinates *u* and *v*.

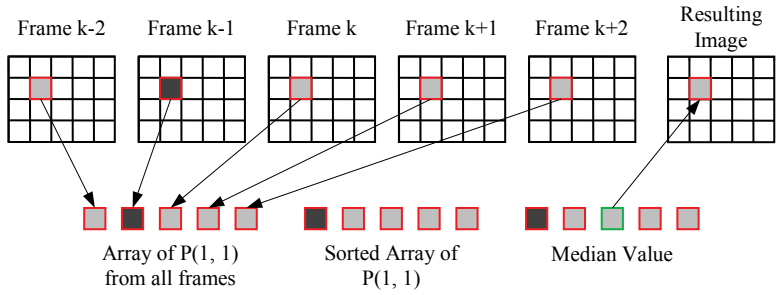


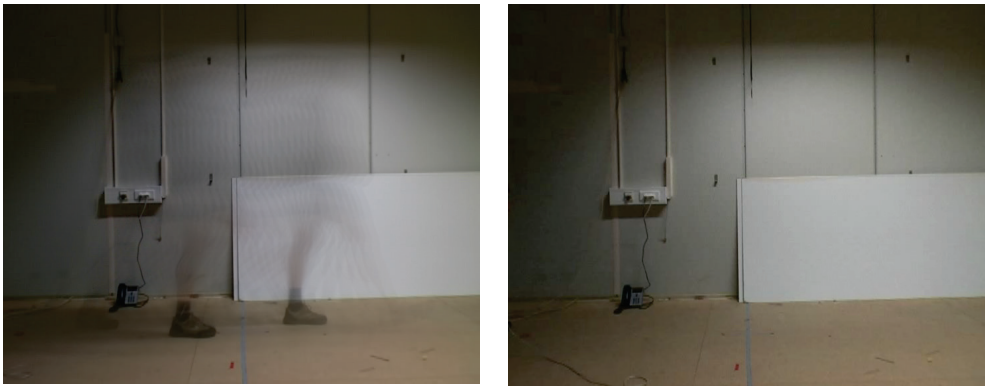
Fig. 4.9 Illustration of temporal median filtering

In other words, in a video sequence the pixels from each frame that have the same coordinates are sorted in ascending order and the median value of the sorted array is copied to the background image.

The main assumption for applying the temporal median filter is that as the human walks through the scene, for each image coordinate pair *(u, v)* the pixel *P(u, v)* will belong to the background in at least half of the frames. Therefore, at least half of the

sorted array will contain the background value for the considered pixel so that the value in the middle of the array will best describe the background.

The result of temporal median filtering is considerably better than using a mean filter [16], which replaces the considered pixel with the mean of the pixels from all the video frames with the same coordinates as the considered pixel. This means that pixel values in frames with the person for pixels that have the same coordinates as background pixels in frames without the person will influence the output mean value. Therefore the resulted image will have a so-called ghost-like effect on the background image. Fig. 4.10 shows a comparison of the results obtained by using the two filters.

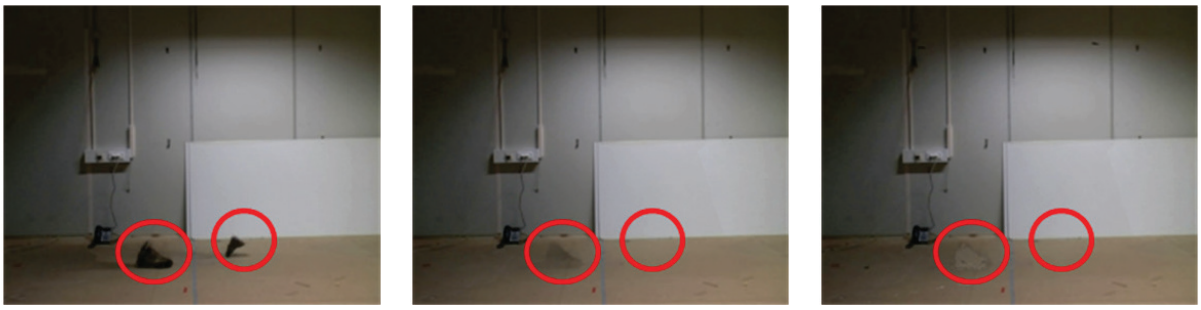


a) b)

Fig. 4.10 Comparison of outputs of automatic background extraction for: a) temporal mean filter and b) temporal median filter

As in the case of clinical gait analysis the input videos are always cut to contain exactly one full gait cycle, the temporal median filter works well for most regions of the sagittal image. However, depending on the walking speed of the person, the feet of the person are sometimes still visible in the background image as they appear to be static for more than half of the video, as can be seen in Fig. 4.10 a). In order to overcome this problem a particular property of the human gait cycle is used, namely the fact that the feet appear to be static mostly between the first 30% and the last 30% of the frames of the gait cycle. The reason for this is that at the beginning and at the end of the gait cycle, the human is standing on both legs, while in the middle of the gait cycle one of the legs supports the weight, while the other leg is swung from back to front. For this reason only video frames outside of this interval are used. Fig. 4.11 a) shows the background extracted using all the frames of a gait cycle video and Fig. 4.11 b) shows the background extracted using only the first 30% and the last 30% of the frames. It can be seen that the support legs are well visible in Fig. 4.11 a), which means that they appear to be static for more than half of the frames. In the background

image shown in Fig. 4.11 b) this effect is diminished, but in particular cases the feet can remain visible in the extracted background image.

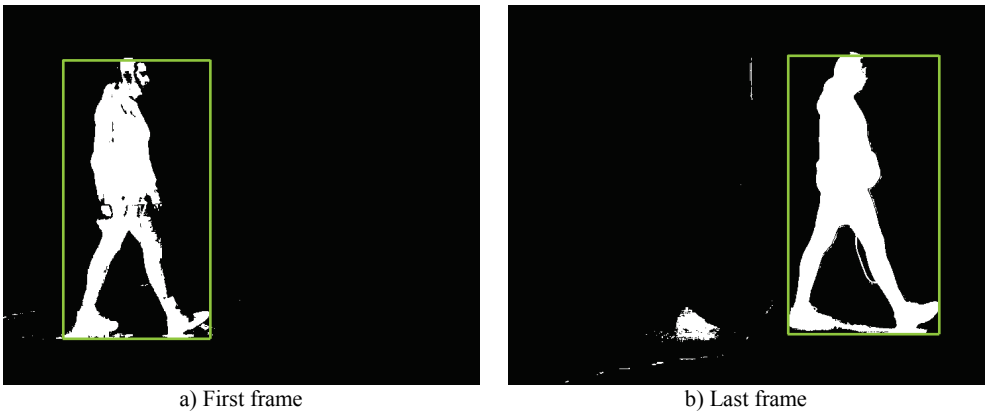


a) All frames used b) First 30% and last 30% used c) Additional filtering performed

Fig. 4.11 Results of the automatic background subtraction

Additional post-processing solves the above mentioned problem and makes background extraction more accurate. This filtering can be done using two frames of the video, for which the background that is occluded in the first frame is completely visible in the second frame and vice versa. Given the fact that for gait analysis videos, the human is walking through the scene, the first frame and the last frame of the video are most likely to meet the above condition.

The proposed additional filtering consists in using the currently obtained background, shown in Fig. 4.11 b), and performs background subtraction for the first and last frames, followed by the closed loop segmentation of the resulted image introduced section 3.1. Fig. 4.12 illustrates the result of this operation. The last step of the automatic background extraction is to update the current background using pixels from the original image that have the same coordinates as the segmented pixels that are outside the defined ROI. This operation is similar to the background adaptation shown in (3.3) with $\alpha=1$. The final result can be seen in Fig. 4.11 c).



a) First frame b) Last frame

Fig. 4.12 Additional background filtering

For the frontal video this algorithm cannot be used, because there is always a large region of the background that will not be seen in any frame of the cut video due to the person that occludes it. Therefore a background frame, which does not contain the person, must be acquired separately.

4.3.1 Human joints extraction in 2D segmented images

After segmenting the subtracted image using the closed-loop segmentation method introduced in section 3.1, the person has to be detected and bounded by a rectangle which demarcates the ROI. As in the presented gait analysis scenario there is always only one person in the image, the ROI is defined as the rectangle bounding the largest region of connected segmented pixels in the segmented image, as can be seen in Fig. 4.13 d). The ROI, which is the region of the segmented image which contains the person, is used further as input to 2D joint extraction.

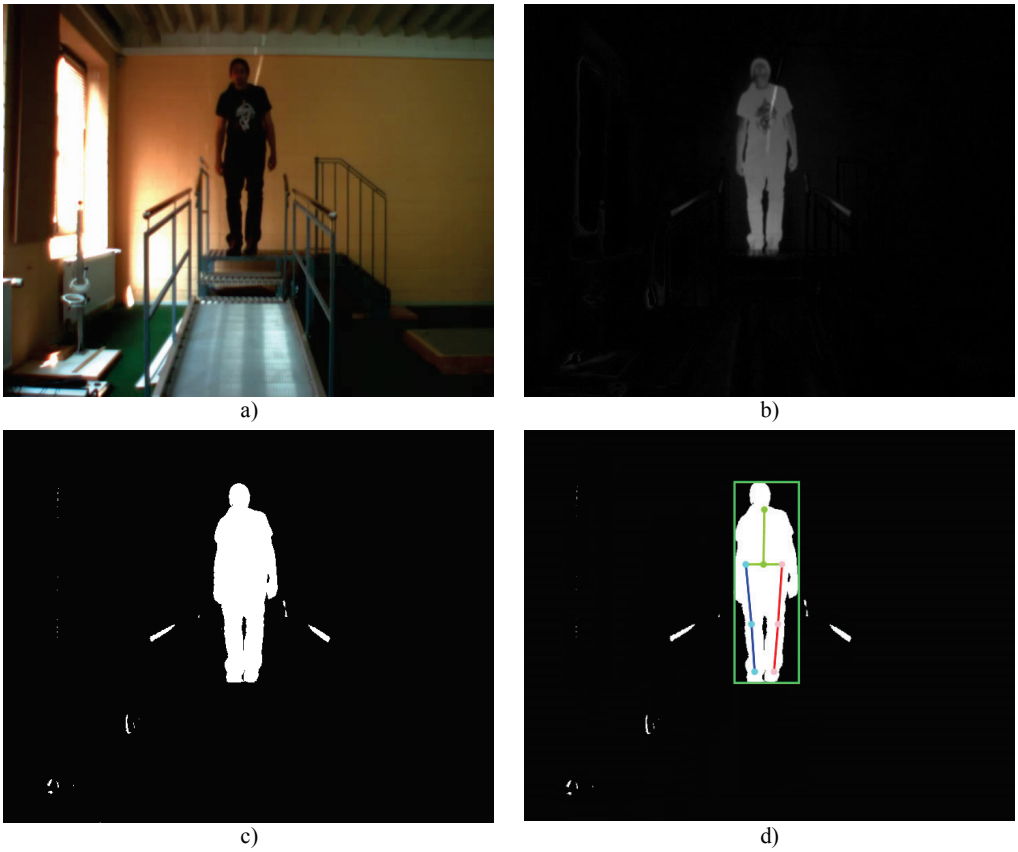


Fig. 4.13 Overview of image results for the important steps of the proposed markerless gait analysis system: a) Original image, b) Subtracted gray-scale image, c) Segmented image obtained by closed-loop segmentation and d) Binary segmented images overlaid with the extracted ROI and stick figure

An illustration of the process of joint detection is presented in Fig. 4.14. The graphs displaying the number of segmented (white) pixels in each row of the ROIs, known as horizontal projections, are shown in Fig. 4.14 at the left of the ROIs of segmented images.

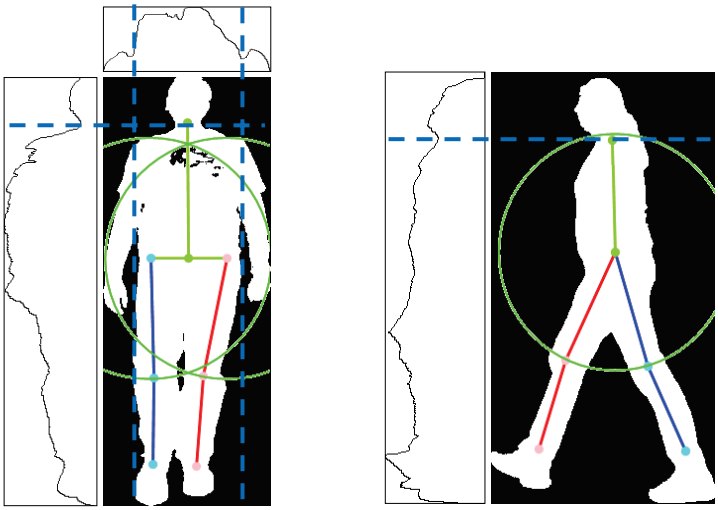


Fig. 4.14 Process of joints localization based on segmented images of the human body: a) Frontal plane and b) Sagittal plane

The horizontal projections are used for determining the location of the first joint, the neck joint. The *v* image coordinate of the neck joint corresponds to the top located global minimum of the horizontal projection while the *u* coordinate corresponds to the middle point of the segmented region in the defined *v* row of the ROI of the segmented image. Additional to the horizontal projection, the frontal binary image of the ROI is accompanied with the graph displaying the number of segmented (white) pixels in each ROI column, known as the vertical projection. As illustrated in Fig. 4.14 a), the vertical projection is used for identifying arms regions in the segmented human body as arms correspond to the end left and right "hill" of the vertical projection. The locations of the other joints are determined by combining the segmented image data, the extracted location of the neck joint and, depending on the availability, either measured lengths of the particular person's body parts or statistical anatomical measures of the human body segments.

An overview of the statistical anatomical measures of the human body that can be found in [14] is illustrated in Fig. 4.15, where all body segment lengths are represented as a percentage of body height *H*. The person's height in the image is considered to be equal to the height of the image ROI. Therefore, based on these human body ratios, the location of the human joints in image coordinates are extracted from the segmented image. For example, the hip joint is located at 0.53 of the total body height with respect to the ground. Therefore, the *v* coordinate of the hip joints is located at the ratio of 0.53 of the ROI height with respect to the lower boundary of the ROI.

For the frontal plane, the hip width is equal to 0.191 of the human height, so the *u* coordinates of the left and right hip are located at half of this distance to the left and

right of the vertical human symmetry axis (trunk). This axis is in the centre of the segmented body region after excluding the arms. For the sagittal plane, the *v* coordinate of the hip is detected in the same way as for the frontal plane, while the *u* coordinate is in the middle of the segmented human body at the already computed *v* coordinate.

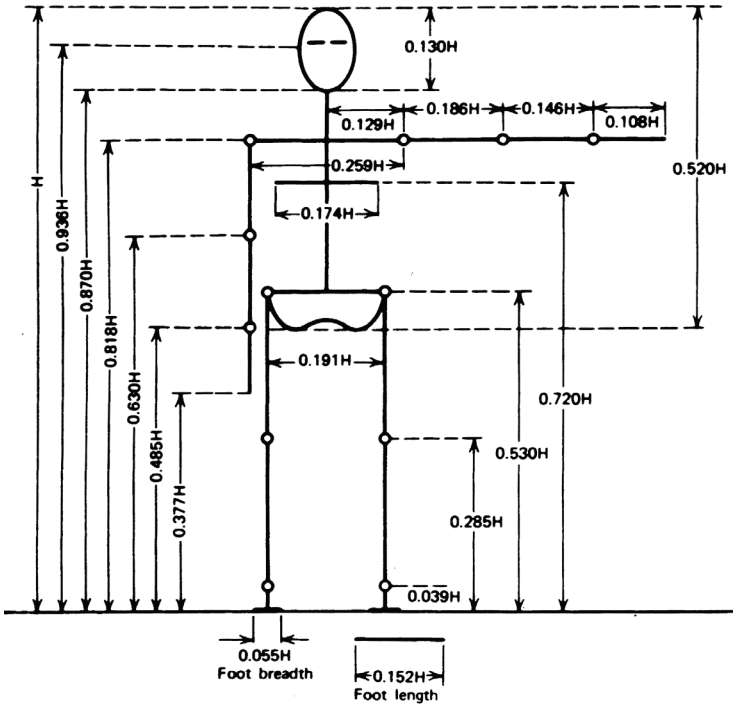


Fig. 4.15 Body segment lengths expressed as a fraction of body height H [14]

The other joints are extracted by using biomechanical knowledge on the human joints, such as the fact that the hip and knee joint are spherical. Therefore the hip joints are considered centres of circles having radiuses equal to the thigh length, so the knee joints are defined as middle points of the arcs that represent the intersections of these circles and the two segmented body parts located below the hips, which are the legs.

In the sagittal plane ankle joints are easily detectable as they lie on a circle with the radius equal to the shank length and with the centre in the corresponding knee. In this context correspondence means left knee-left ankle and right knee-right ankle. However, for the frontal plane an analogue detection is not possible, as the shank length as projected on the image plane, varies depending on how much the knee is bent. In this situation, as well as in the case when joints are occluded, the other view helps reconstructing the joint location, as described in the following section.

4.3.2 Improvement of 2D joint extraction based on epipolar geometry

Similarly to the case of converging cameras presented in Chapter 2, for each of the two cameras in the orthogonal setup of the presented vision system for gait analysis, two coordinate systems can be defined: an image coordinate system *(u, v)* and a camera coordinate system *(x, y, z)*. The relationships between these coordinate systems are expressed by the intrinsic (2.2) and extrinsic (2.4) parameters. Fig. 4.16 illustrates how a 3D point is projected on the two orthogonal cameras and presents part of the camera parameters, which are the focal length *f*, the optical center *c* and the camera center C. These parameters are illustrated for both the frontal and sagittal camera, so they are denoted with the indices *F* and *S*.

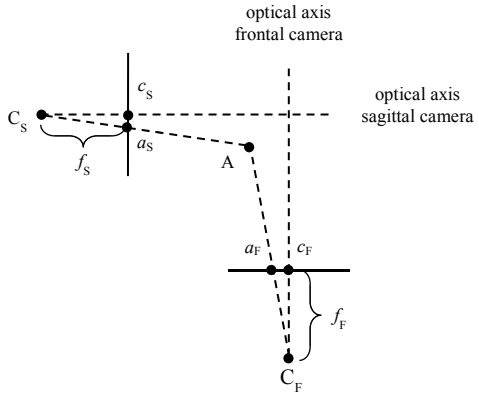


Fig. 4.16 Illustration of 3D point projection on the orthogonal cameras

In epipolar geometry [10] for each pair of cameras a matrix can be defined which describes the relationship between pairs of corresponding points in the two images. This matrix, also known as fundamental matrix *F*, is defined as:

$$a_F{}^T \cdot F \cdot a_S = 0 \tag{4.1}$$

where a_F and a_S are corresponding points in homogenous coordinates, which are projections of the same 3D point onto the frontal and sagittal image respectively.

According to the epipolar constraint, the corresponding point in the second image of an image pair lies on a special line known as epipolar line for a point in the first image and the other way around. The fundamental matrix allows epipolar lines to be computed using the following relationship:

$$l_F = F \cdot a_S \tag{4.2}$$

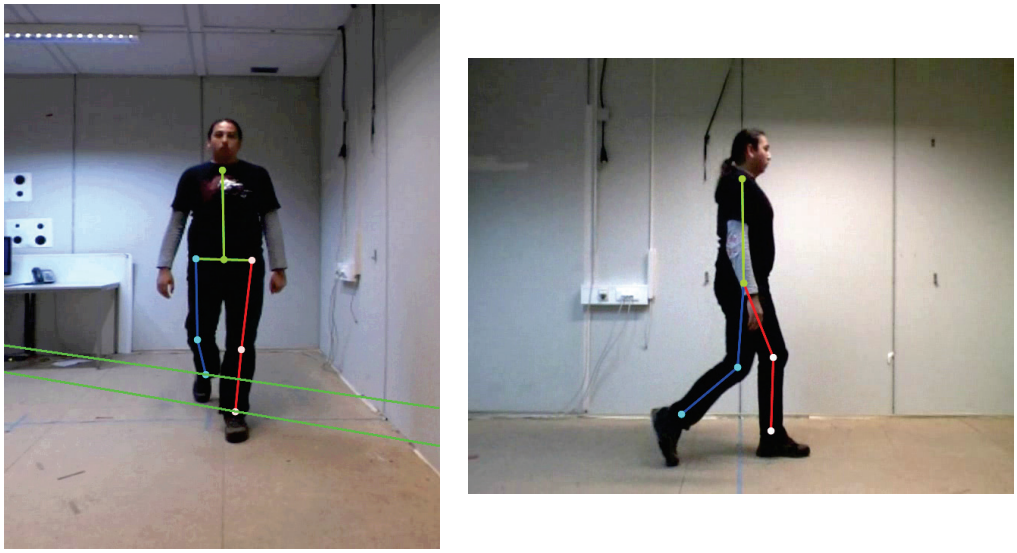
where l_F is the epipolar line in the frontal image, which corresponds to the image point a_S in the sagittal image.

This information can be used to help detecting the corresponding point by reducing the search space to a single line, which means that the corresponding point a_F lies on the line l_F, therefore the following equation is satisfied:

$$a_F^{\ T} \cdot l_F = 0 \tag{4.3}$$

Analogue epipolar lines in the sagittal image can be computed based on image points in the frontal image.

The epipolar lines can be used to help detecting joints that are visible in one view, but are occluded in other view. A typical example where the epipolar lines are useful is detecting the ankles in the frontal image. The fact that the ankle can be well detected in the sagittal image is used to help its detection in the frontal plane by computing its epipolar line for the frontal image. Fig. 4.17 illustrates this process.



a) b)

Fig. 4.17 Illustration of using the epipolar constraint for detecting the ankle joint in the frontal image: a) Frontal plane and b) Sagittal plane

It can be observed that the two epipolar lines intersect the person's legs and the ankles are detected as being located in the middle of the line segments that result from intersecting the epipolar lines with the segmented legs.

The presented joint extraction algorithm for the frontal and the sagittal plane is applied separately to all frames of a video and an example can be seen in Fig. 4.18.

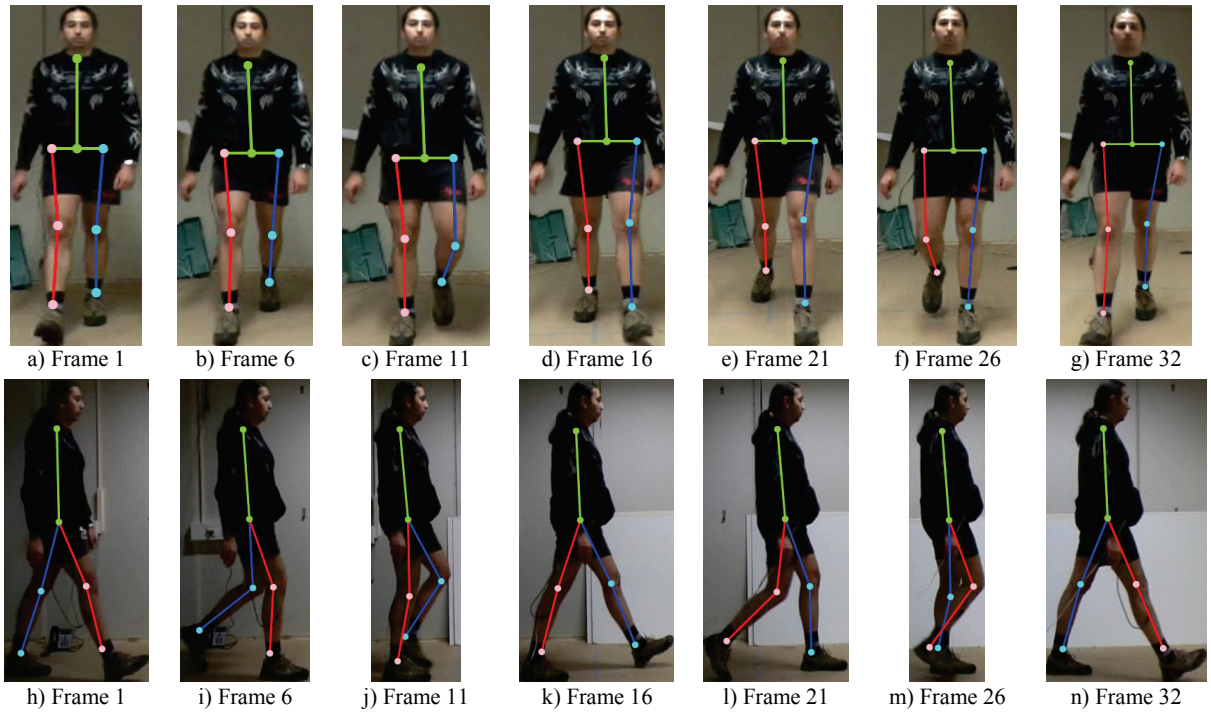


Fig. 4.18 Examples of frames form a video of a walking person with overlaid stick figures: a) – g) Frontal plane and h) – n) Sagittal plane

It can be seen that the presented algorithm performs well for both the frontal and the sagittal plane, as the stick figures are properly overlaid on the human body. The person shown in Fig. 4.18 is wearing shorts in order to allow precise placement of the electrogoniometer used for performance evaluation, as explained in the next section.

4.3.3 3D reconstruction of the human gait

After the 2D joints have been successfully detected in a pair of images, 3D joint reconstruction is performed. This enables features such as step length to be extracted in real-world coordinates. Additionally, using 3D points for computing the angles between body parts eliminates the effects of perspective projection that are introduced by the camera when capturing the scene, so the angles do not need to be compensated.

Looking at (2.5), the relationship illustrating the projection of 3D world points on the image plane, it becomes clear that the original 3D point A cannot be obtained from a single image just by knowing the image coordinates of its projection a. This is due to the fact that the system is underdetermined, having three unknowns and only two equations. However, if two cameras are looking at the point A, the equation system composed of a total of four equations given by the two projections a_F and a_S is over determined and its solution is the 3D point A. This process is known as stereo triangulation and requires accurate calibration, including stereo calibration and the computation of the fundamental matrix.

In epipolar geometry, two projections of the same 3D point on two different images are known as corresponding points. In the presented application however, a virtual point is reconstructed which is not seen by any of the two cameras, as it lies inside the human joint, as illustrated in Fig. 4.19 on the example of the knee joint.

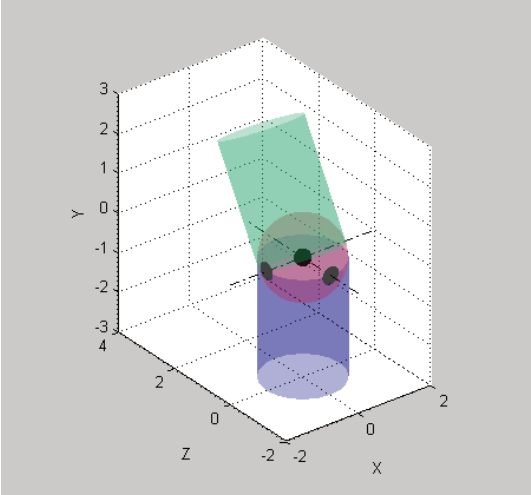


Fig. 4.19 3D reconstruction of a virtual point located in the knee joint

It can be seen that each of the two cameras sees a projection of the 3D point inside the knee joint on the surface of the human body, but the reconstructed point lies inside the joint as it is in reality. The same applies for all other extracted joints. Other similar approaches for 3D reconstruction of joints in setups with two orthogonal cameras have been presented in [22] and [27]. However, the first method in [22] does not consider and thus does not compensate the effects of perspective projections and the second method in [27] attempts to reconstruct a point on the surface of the human body in contrast to the virtual point proposed in this thesis.

After obtaining the two virtually corresponding points of one joint in both the frontal and the sagittal image as explained in section 4.3.1, the location of the virtual 3D point can be obtained using triangulation [10][52]. This requires the following equation to be solved:

$$\begin{bmatrix} u_S P_{3S}^T - P_{1S}^T \\ v_S P_{3S}^T - P_{2S}^T \\ u_F P_{3F}^T - P_{1F}^T \\ v_F P_{3F}^T - P_{2F}^T \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.4}$$

where $p_F(u_F, v_F)$ and $p_S(u_S, v_S)$ are the two corresponding image points in the frontal and sagittal plane respectively, which are used for triangulation. P^T_{nF} is the n-th row of

the frontal camera's projection matrix P_F and P^T_{nS} is the n-th row of the sagittal camera's projection matrix P_S. The resulting equation system is solved using Singular Value Decomposition (SVD) [10]. After obtaining the 3D coordinates of all joints, a 3D model can be constructed for each frame, as exemplified in Fig. 4.20.

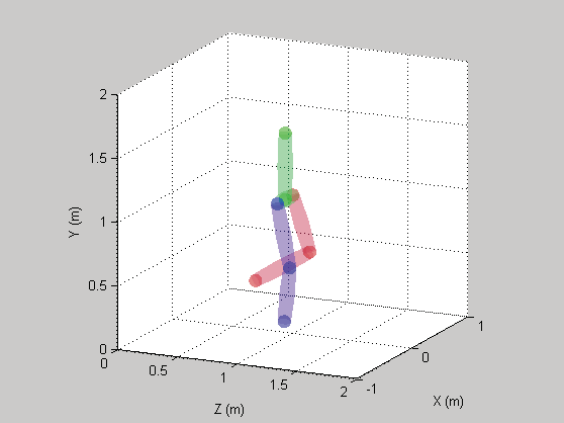


Fig. 4.20 Example of 3D model of the human body

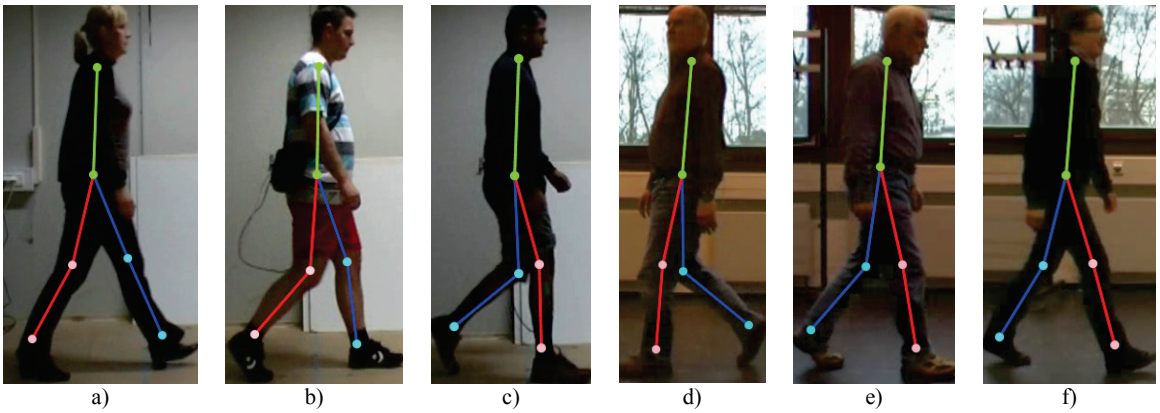
The process of computing the 3D coordinates of the joints by using triangulation is repeated for each video frame and an animated 3D model is obtained by tracking the 3D model in time. Some of the advantages of 3D reconstruction are the possibility to easily follow the joints of the same leg based on their 3D position and to extract gait parameters such as step length and walking speed. The obtained joints are the basis for extracting gait parameters as they are used to extract angles between body parts.

In order to compute the required angles for the frontal and the sagittal plane, the 3D model is projected on the *XY* and *YZ* plane using orthographic projections. By using these projected angles instead of angles from the 2D images, the effects of perspective projection introduced by the camera are eliminated as previously mentioned. Other gait parameters such as step length and speed are extracted from the 3D model as well.

One particular difficulty of this application is tracking the two legs during the whole gait cycle, having in mind that the legs get switched when viewed from the sagittal plane. However, using the 3D joint locations the two knees and the two ankles can be distinguished during the whole gait cycle, as their coordinate on the *X* axis does not get switched.

Performance evaluation

In order to test the proposed algorithm, 20 healthy subjects of different age, height and gender were imaged while walking. The subjects were wearing normal clothes and were walking in different natural indoor environments at different times of the day, as well as in laboratory environments with controlled illumination as shown in Fig. 4.21.



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

Fig. 4.21 Examples of filmed persons walking in different environments, wearing different clothes and walking either left to right or right to left; the filmed persons are overlaid with back projected stick figures

The processing time required by the complete presented vision-based gait analysis system to process one frame is about 30ms on a standard Dual-Core Desktop PC running at 2.4 GHz. This enables the system to be useful in clinical environments for supporting medical doctors in gait diagnosis and rehabilitation, as it delivers results shortly after the person's gait was recorded.

In order to evaluate the performance of the presented system, ground truth was necessary, which was provided by a Biometrics flexible electrogoniometer SG150 [74]. The sensor accuracy is ±3° according to the manufacturer's data and results reported in [30]. During performance evaluation experiments the goniometer was attached to the subject's right leg, as shown in Fig. 4.22, as well as on the right hip.

Fig. 4.22 Goniometer attached to the human leg at the right knee joint

In order to increase the measurement accuracy, 10 out of the 20 healthy subjects were recorded with the goniometer placed directly on the skin. The angles measured with the goniometer were compared to the values of the knee and hip angles that were automatically extracted from each video frame.

An example diagram showing the automatically extracted and the goniometer measured knee and hip joint trajectories is given in Fig. 4.23.

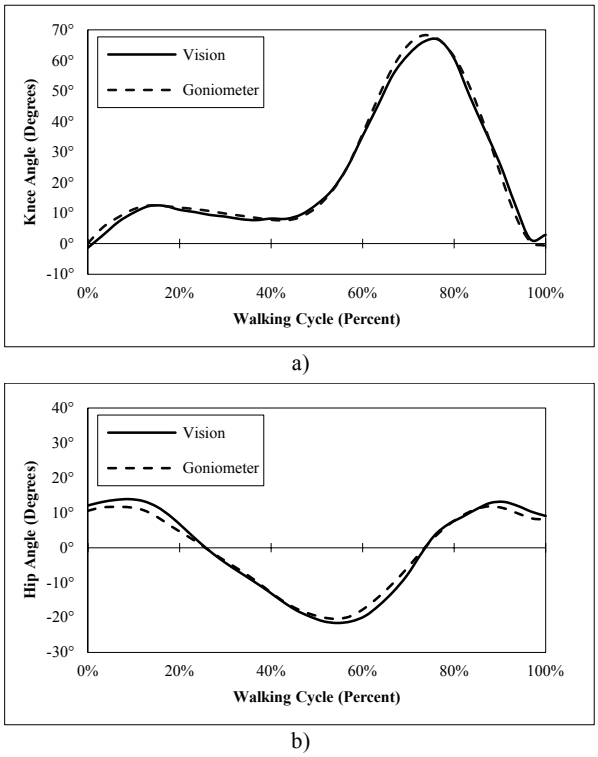


Fig. 4.23 Comparison of automatically extracted knee and hip angles from image sequences with a) knee angles and b) hip angles measured using the electrogoniometer for one subject

It can be seen that the automatically extracted angle trajectories almost perfectly follow the curves obtained using the electrogoniometer. This indicates that the data obtained using the presented system is reliable. The shapes presented in Fig. 4.23 also correspond very closely to the shape of the reference knee and hip angle trajectory of a healthy walking subject that can be found in the reference literature [14][15][20].

Fig. 4.24 shows the knee joint angle change during a gait cycle, computed from the automatically extracted joints from the videos of 20 examined healthy subjects. The bold curve represents the mean knee joint angle across the twenty subjects. These knee angle trajectories were used to compute the mean and standard deviation, as presented in [5].

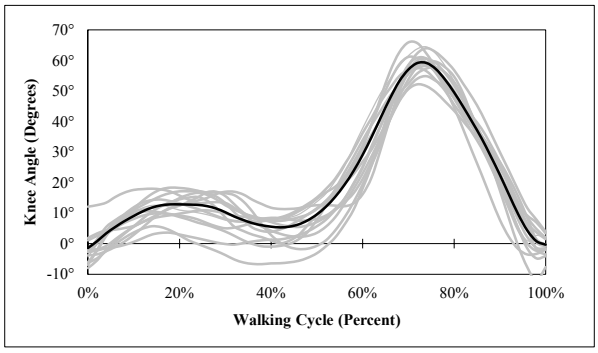


Fig. 4.24 Knee angles trajectories of 20 subjects during one gait cycle

In order to illustrate the improvement achieved by extracting joint angles from the 3D model with respect to extracting them from the 2D image features directly, Fig. 4.25 shows the mean and standard deviation (bright gray) for features extracted from the 2D image, the same measurements obtained using the presented 3D model (dark gray) as well as a reference knee joint angle pattern presented in [14] (black).

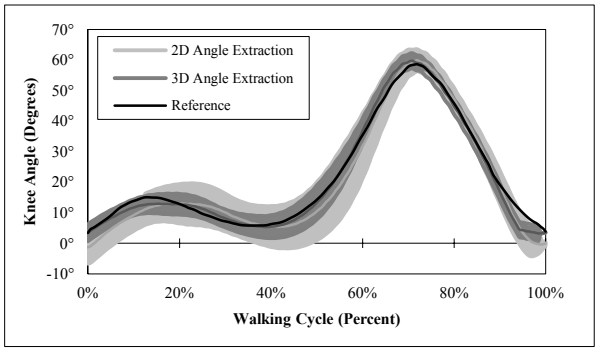


Fig. 4.25 Illustration of vision-based gait analysis performance improvement after extracting angles from 3D model rather than from 2D image features

It can be observed that the results obtained using the 3D model follow the reference curve more closely and the smaller standard deviation indicates reliable results. The reason is that the 3D reconstruction eliminates the effects of perspective projection, which are introduced by the camera when capturing the scene on the image sensor.

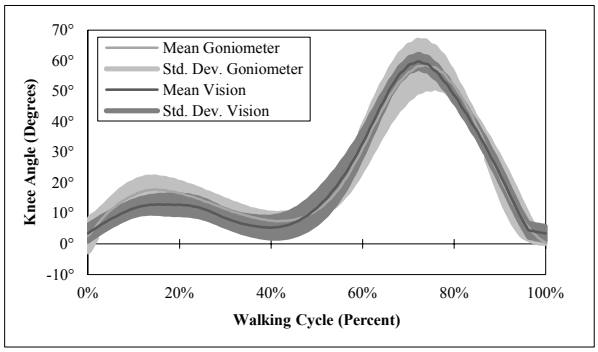


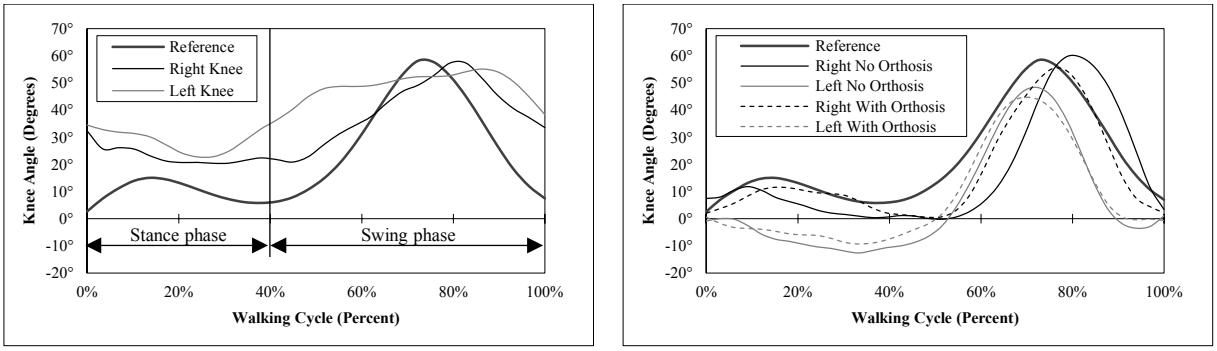
Fig. 4.26 Mean knee angle and standard deviation obtained using the proposed vision-based algorithm overlaid on the mean and standard deviation obtained using the goniometer

In order to additionally validate the results, the mean and standard deviation for the knee angle trajectories obtained using the electrogoniometer were computed. Fig. 4.26 presents the mean knee angle and standard deviation obtained using the proposed vision-based algorithm overlaid on the mean and standard deviation obtained using the goniometer. It can be seen that the result is close to the values obtained using the presented vision system. The performed tests indicate that the system extracts gait data reliably and can be used for gait analysis.

4.3.4 Human gait analysis for gait rehabilitation

The goal of the presented vision system for gait analysis is to assist clinicians in diagnosing gait disorders as well as for recording the progress of gait rehabilitation. In order to demonstrate the usefulness of the approach, videos of 20 patients have been recorded in the gait laboratory of the Neurological Rehabilitation Centre Friedehorst in Bremen, Germany [73].

A common cause for pathological gait is spastic cerebral palsy, which is a neurological impairment characterized by velocity dependent resistance to stretch of the muscles [20]. This reflects in the patients' inability to fully stretch the legs, as can be seen in the diagram presented in Fig. 4.27 a).



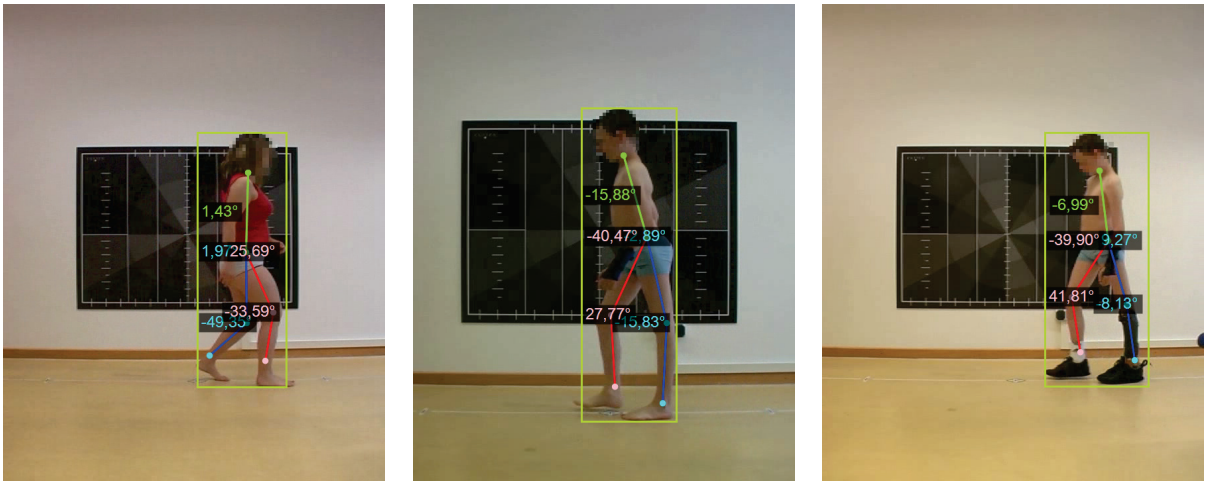
a) b)

Fig. 4.27 Examples of clinical gait analysis: a) Knee angle illustration for a patient suffering of spastic cerebral palsy and b) Knee angle comparison for a patient with hemiparesis on the left side before and after putting an orthosis on the left leg

It can be observed that neither the left nor the right leg is completely stretched at any moment during the entire gait cycle. The reference gait cycle, which illustrates the knee angle variation for a healthy person, contains a stance phase (40%) and a swing phase (60%). Between these phases the leg should be almost fully stretched, as shown by the reference knee angle trajectory, but spastic patients cannot fulfill this, as their flexor muscles are shortened. Fig. 4.28 a) shows a video frame of this patient. The fact that such impairments can be observed by looking at these diagrams illustrates the capability of the presented system to assist clinicians in diagnosing gait disorders.

Depending on the diagnosis it can be particularly useful to compare the gait parameters from the left and the right side of a patient, as many types of gait disorders are characterized by asymmetric gait, e.g. in stroke patients. The proposed system can also assist clinicians in observing the process of gait rehabilitation by recording the patient prior and after applying some form of gait correction, like an orthosis.

Fig. 4.27 b) shows the knee angle for a patient suffering of hemiparesis on the left side, before and after putting an orthosis on the left leg. Hemiparesis is the medical term for describing the paralysis of the arm and the leg on one side of the body. It can be observed that the knee cycle for the right leg is close to the reference knee cycle, but the swing phase starts and ends late. The left leg on the other hand is overstretched during the stance phase at over 15 degrees, as shown in Fig. 4.28 b) and the leg is not bent enough during the swing phase. After fitting the orthosis it can be observed that the left leg is less overstretched during the stance phase, as shown in Fig. 4.28 c), but also the swing phase of the right leg begins and ends earlier than before.



a) b) c)

Fig. 4.28 Illustration of pathological gait patterns: a) Patient suffering of cerebral palsy and b) Patient with hemiparesis on the left side

Fig. 4.29 shows the knee angle for a patient suffering of hemiparesis on the right side, before and after putting an orthosis on the right leg. Initially the swing phase for the healthy leg (left leg) starts very late, similarly to the other patient suffering of hemiparesis. The right leg is neither stretched enough during the stance phase nor bent enough during the swing phase. After the orthosis was fitted, the gait significantly improved. This can be observed by looking at the swing phase for the left leg, which now starts and ends similarly to the healthy reference. Also the complete cycle for the right leg has improved by allowing the patient to stretch the leg completely during the stance phase and bend it more during the swing phase.

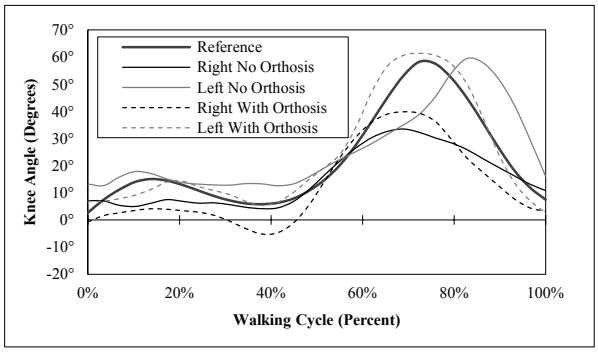


Fig. 4.29 Knee angle improvement for a patient with hemiparesis on the right side

Similarly to the extracted knee angle trajectories, all other extracted features can be used to draw such kind of curves. In the medical literature however, the knee angle is the most frequently used feature for describing the human gait. The progress of gait rehabilitation can also be recorded on a longer period, for example if the patient goes to physiotherapy. In this case it might be useful to record the patient's gait for instance every week and to observe whether the applied therapy brings the expected results.

Besides various angles between different body parts, additional features are extracted, which characterize the human gait. These features are: step length, stride length, cadence and speed.

As illustrated in Fig. 4.2 b), the step length represents the distance between the two 3D points where the two feet have consecutively touched the ground. The stride length represents the distance between the two 3D points where the same foot has touched the ground. In order to automatically determine the locations of these points in 3D space, the most natural way would be to detect the ground and to determine when and where the feet touch it. However the exact detection of the ground plane is not a simple task and even small errors in detection can introduce big errors in calculation, as the foot only lifts off the ground for a few centimeters.

A far better solution is to analyze the speed of the ankles, as the speed is zero while the foot is on the ground and the person stands on it. Therefore it is proposed to analyze the speed along the *Z* axis and to store the 3D location of the ankles for the moments in which the speed of both ankles is zero. Fig. 4.30 shows an example of left and right ankle velocities along the Z axis for a right leg gait cycle.

It can be seen in Fig. 4.30 that the speed of both ankles is zero (both feet are on the ground) at the beginning and at the end of the gait cycle, but also at about 50% of the gait cycle.

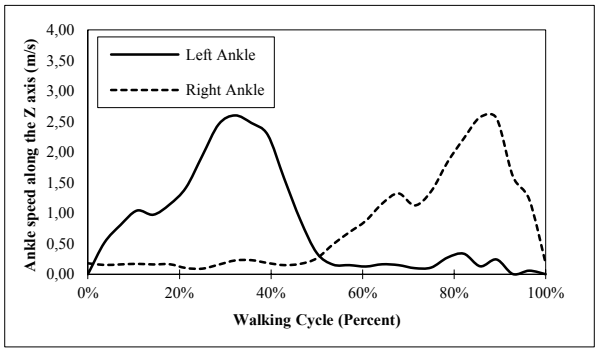


Fig. 4.30 Example of left and right ankle speed along the Z axis

As the gait cycle in the example begins and ends with the right initial contact, the step length of the right leg can be computed as the distance between the left ankle and the right ankle either at the beginning or at the end of the gait cycle. The step length of the left leg is computed as the distance between the left ankle and the right ankle in the middle of the gait cycle, as the velocities for both ankles equal to zero and therefore both legs are on the ground.

The stride length can only be computed for the right leg in the given example as it contains a gait cycle of the right leg. The stride length equals to the distance between the right ankle position in the first and the last frame of the gait cycle.

The speed of the person is defined by the distance walked in a certain amount of time. As the sampling rate is given by the video frame rate, it can be easily determined how long a person needs for walking one step. After determining the step length, the speed can be calculated. The cadence represents the number of steps walked by a person within a minute. Knowing how much time a person needed to walk a step, the cadence can be directly deduced.

4.3.5 Other applications of vision-based 3D gait reconstruction

Besides clinical gait analysis there are other applications for which the presented 3D joint reconstruction can be used. One of these applications is the 3D modeling and simulation phase in the development of robot assisted gait rehabilitation systems.

Motion data obtained with the presented system, namely joint angles, was used to drive the model of the human body which was attached to a platform in the simulation software MSC Adams [7]. Fig. 4.31 shows five frames of a video and the corresponding poses of the human model as obtained from MSC Adams.

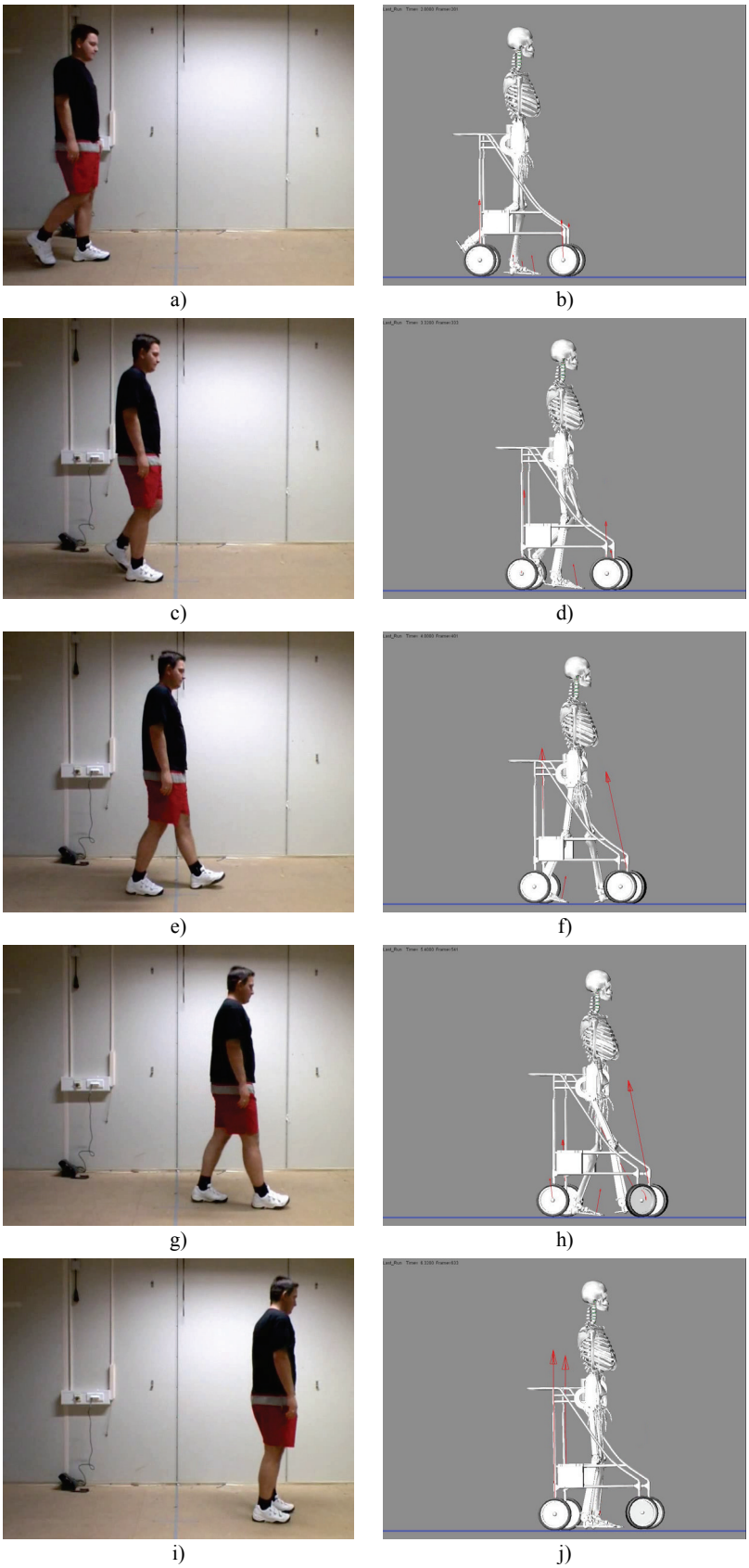


Fig. 4.31 Animation of a human body model in MSC Adams using motion data obtained using the presented 3D gait reconstruction: a), c), e), g), i) Original frames with the human and b), d), f), h), j) Corresponding frames from simulation

It can be observed that the pose of the modeled human body corresponds to the pose of the human in the video, indicating that the data extracted from the video can be used for simulating the motion of the human body.

4.4 Clinical gait analysis using RGBD cameras

As already mentioned in section 2.7, recently so-called RGBD cameras such as the Microsoft Kinect have been developed for indoor human detection and tracking. The depth information recorded with these cameras simplifies the process of detecting and tracking the human even in cluttered scenes. Additionally, open-source libraries such as OpenNI [78] exist, which allow certain joints of the human body to be tracked. However, these cameras have been developed mainly for interactive computer games and the corresponding libraries are designed for building Human-Machine Interfaces (HMI) by offering functionality such as finger tracking and gesture recognition. Therefore these cameras cannot be used "out-of-the-box" for human gait analysis.

Even though most of the joints required by clinical gait analysis can be extracted from RGBD images with the OpenNI library, the joint detection accuracy is poor compared to marker-based systems and compared to the algorithms presented previously in this chapter. This holds especially if the person is imaged from the sagittal plane, as can be seen in Fig. 4.32 a)-e).

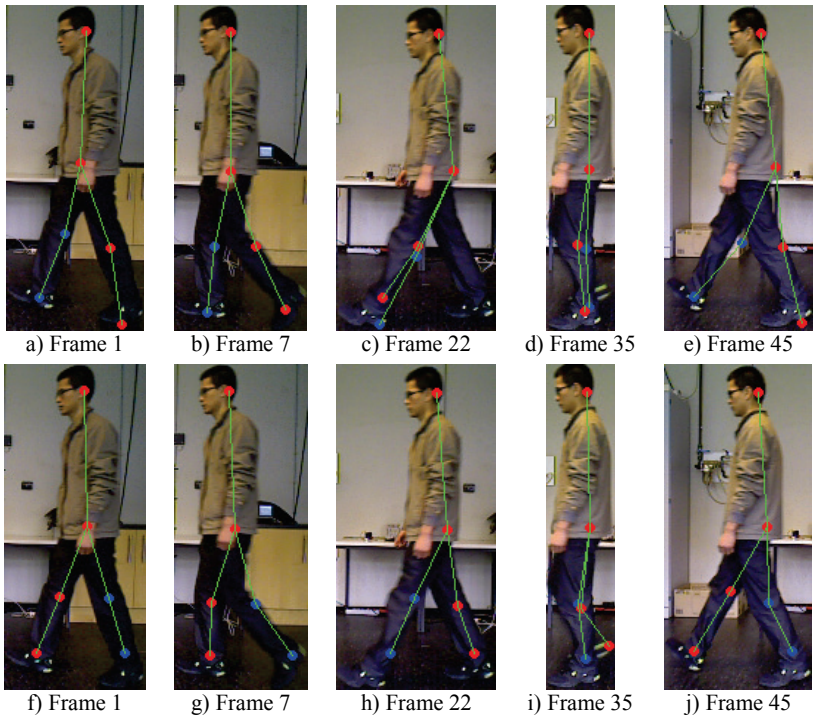


Fig. 4.32 Examples of frames captured with a Kinect camera and processed with: a) – e) the OpenNI library and f) – j) the proposed joint extraction algorithm

The reason for this is that the OpenNI library was designed to work well for the frontal plane, as the user usually faces the screen while playing or interacting with the computer. However, as imaging the sagittal plane is necessary for clinical gait analysis, joint extraction can be improved by refining the results obtained from OpenNI using the 2D joint extraction algorithm presented in section 4.3.1, as shown in Fig. 4.32 f)-j).

As the RGBD camera provides depth images, the process of separating the human from the background is simplified compared to the case presented in section 3.1. Using the depth image, OpenNI provides a binary image containing the segmented human. Additionally, OpenNI provides 3D joint locations for the required joints. Considering the data provided by the OpenNI library, the gait analysis algorithm is changed compared to the one presented in Fig. 4.8, as can be seen in Fig. 4.33.

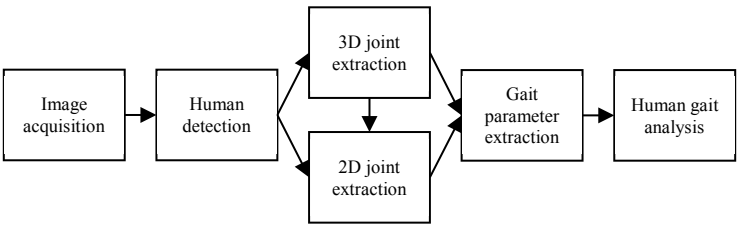


Fig. 4.33 Block-diagram of the improved gait analysis system

After acquiring an RGBD image, the OpenNI library is used for detecting the human in the image. The result of this operation is a binary image containing the segmented human body. OpenNI uses this binary image further as mask for the depth image and extracts the 3D locations of the joints. However, only the location of the neck joint extracted by OpenNI is reliable. Therefore the binary image is additionally used for 2D joint extraction as shown in section 4.3.1, with the difference that the neck joint provided by OpenNI is used as starting point. The final result of 2D joint extraction, overlaid on the original images, is shown in Fig. 4.32 f)-j). After all 2D joints are extracted, their image coordinates *(u, v)* and the pixel value in the depth image at the coordinates *(u, v)* are used as presented in section 2.6 for reconstructing the 3D location of the joints. These 3D joints are used for gait parameter extraction and human gait analysis in the same manner as presented earlier in this chapter.

As already mentioned, the RGBD cameras provide on one hand RGB images that are useful for clinicians to observe the recorded gait and on the other hand depth images, which allow extraction of 3D positions of joints independently of the camera position. Even though both frontal and sagittal angles can be extracted from the 3D model by applying an orthographic projection on the frontal and the sagittal plane

respectively, if the camera images the frontal plane, the sagittal angles are inaccurate and vice-versa. Additionally, RGB images are only available either for the frontal or for the sagittal plane. Therefore, an RGBD camera cannot fully replace the images acquired by two cameras. Hence, in future two RGBD cameras could be used to image the patient from both the frontal and sagittal plane at the same time, as these cameras clearly help in making the human detection and tracking more robust against various external influences. Several performed tests have shown that the frontal and the sagittal RGBD cameras do influence each other due to the fact that their projected structured light patterns overlap as shown in Fig. 4.34, but the influences are minimal.

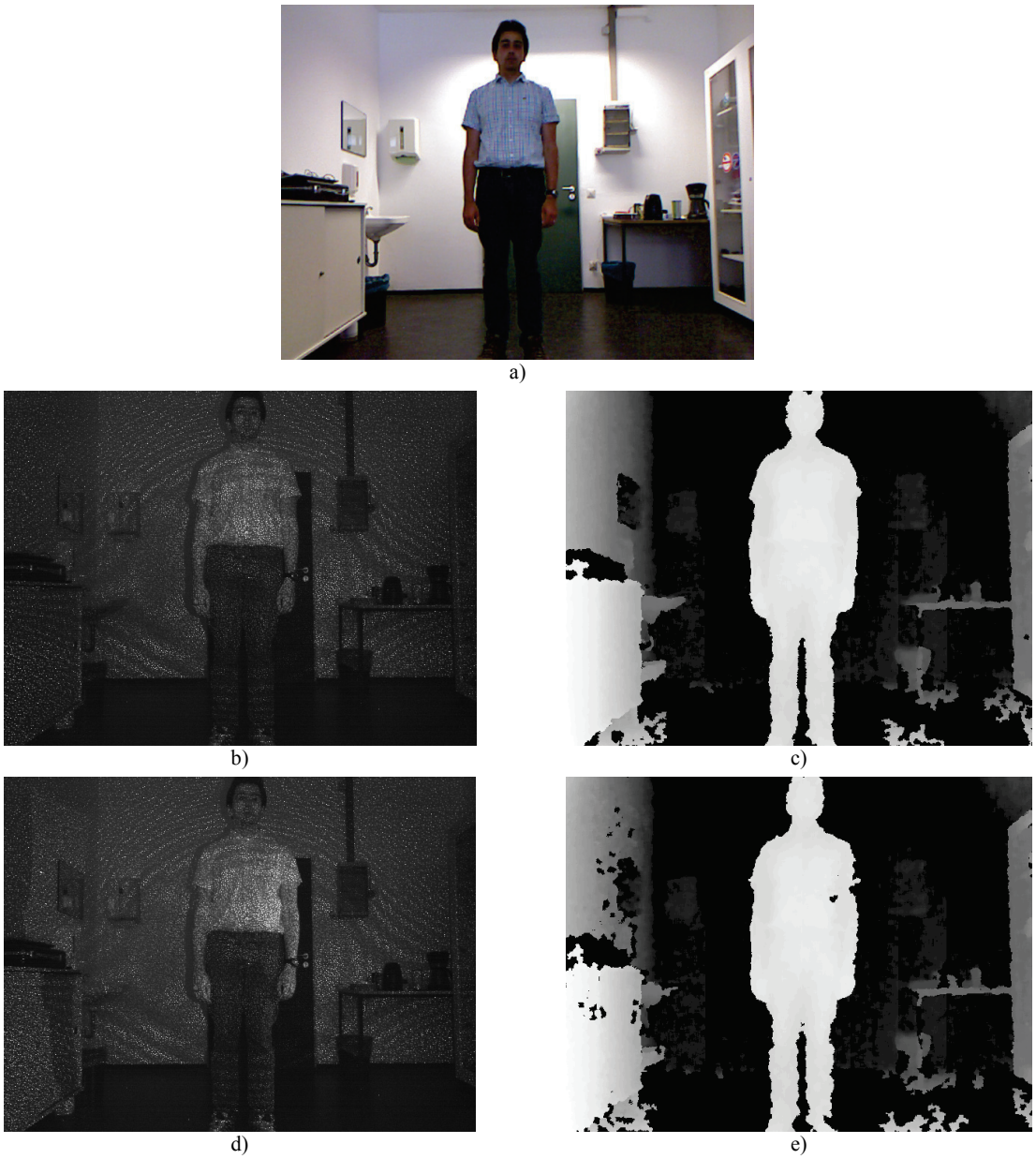


Fig. 4.34 Interference between two Kinect cameras: a) RGB image, b) Frontal infrared image and c) Frontal depth image without sagittal Kinect camera and d) Frontal infrared image and e) Frontal depth image when the sagittal Kinect camera is used

Fig. 4.34 shows the RGB image and infrared and depth images recorded by a Kinect camera which is imaging the person from the front. At the same time, a second Kinect camera was imaging the sagittal plane of the person in order to evaluate the interference between the two cameras. In Fig. 4.34 d) the additional infrared light pattern projected by the sagittal Kinect camera can be seen, as the scene is brighter and the person casts a shadow in the left part of the image, compared to Fig. 4.34 b) where this shadow is not cast. In the depth image shown in Fig. 4.34 e) additional noise regions (black regions) are visible on the left shoulder of the person and on the wall on the left part of the image, compared to Fig. 4.34 c). These regions are caused by the overlapping patterns of the two cameras. However, these regions are relatively small compared to the whole scene and are unlikely to considerably influence the human tracking and the subsequent gait feature extraction. Therefore, multiple RGBD cameras could be used in future for gait analysis, as their interference in the presented camera setup of using a frontal and a sagittal camera is small.

5

Outdoor human tracking – application in pedestrian detection

Advanced driving assistance systems (ADAS) are systems which are nowadays present in most personal vehicles and help the driver during the driving process by augmenting the driving capabilities. Their purpose is to increase driving safety and implicitly road safety. Some examples of commercially available ADAS systems are adaptive cruise control (ACC), lane departure assistant, lane change assistant, intelligent speed adaptation (ISA), collision warning and the well-known in-vehicle navigation system.

Among advanced driving assistance systems, collision warning is one of the most discussed topics in the automotive industry. The purpose of collision warning systems is to autonomously observe the environment in front of the car and to warn the driver if the probability for a collision to occur is high. This could be the case if another vehicle, a pedestrian or a person on a bicycle is in front of the vehicle.

Various systems for collision warning have been developed and tested, which make use of different types of on-board sensors. The approaches can be divided according to the types of used sensors. Some approaches use passive sensors, such as monocular cameras [53] and stereo cameras [54][55] while others use active sensors such as laser scanners [56][57] and radars [58]. The main difference between these types of sensors

is that passive sensors sense naturally available energy, such as the reflected sunlight, while active sensors provide their own source of energy, which is reflected by objects in the vehicle environment. Since each sensor has advantages and disadvantages, authors have developed sensor fusion approaches to make use of the advantages of different sensors by merging the data obtained from the used sensors. Such an example is the fusion of stereo camera with radar [59].

Common to these approaches is that state-of-the-art collision warning systems mostly operate at a frequency of 15-30 Hz, which is only suitable for relatively low vehicle velocities (30-50 km/h) [41][63]. This detection rate is not sufficient for higher vehicle velocities as the braking distance exponentially increases with the vehicle speed and at high velocities the detection time would be too long for warning the driver on time for safely stopping the vehicle. More complex human tracking algorithms which are designed to deal with partial occlusions, such as [55], often only achieve a frequency of 10 Hz or less.

In the system presented in this thesis a stereo camera is used for perceiving the environment as it does not require a special illumination source. This is of particular importance to the automotive industry, as many vehicles drive on the road facing each other so that active sensors such as Time-of-Flight cameras (ToF) [75] located on approaching vehicles might interfere with each other.

Another particularity of vision-based automotive applications such as an ADAS vision system is that during the day the natural light existing in the environment contains a high amount of infrared light, especially on sunny days. This can interfere with active sensors like RGBD cameras which project a structured light pattern for sensing the depth of the scene [76].

An advantage of using cameras to sense the environment, compared to non-visual tracking systems such as radars, is that the complete scene is imaged. By using stereo cameras, as in the system presented in this thesis, three-dimensional (3D) data can be obtained, whereas radars provide only two-dimenstional (2D) information. This 3D data can be used to classify the sensed objects, as will be shown in this chapter.

5.1 System requirements

The system requirements have been defined in order to develop a vision-based collision warning system which will overcome the shortcomings of state-of-the-art systems. It is assumed that the proposed system will be installed in personal vehicles and that it will be used in various driving scenarios such as driving inside cities

(vehicle speed up to 60 km/h) and driving on land roads (vehicle speed up to 90 km/h). It is also desired that the system works on highways during the day (vehicle speed up to 130 km/h) as well as during the night. For this purpose an operating frequency of 100Hz is required, as will be explained in the following sections. However in low light conditions the operating frequency of the system is expected to be lower, as the stereo camera system would need longer exposure times than during the day for capturing an image pair. Therefore, during the night only a slower vehicle speed is supported.

The system should be able to detect different obstacles on the road such as cars and motorcycles. However, the main objective is the detection of pedestrians and persons on bicycles, as they are easily overseen by the driver and can suddenly appear in front of the car, so that they represent the highest danger for collision. A frontal collision is defined as a contact between the front bumper of the car and another object.

The safest action for avoiding a collision is braking, as it does not interfere with other vehicles' paths. The total braking distance depends on the vehicle speed, the road surface condition and of course on the driver's reaction time until hitting the brake. As the vehicle speed and the road surface condition cannot be influenced, the aim of collision warning systems is to reduce the driver's reaction time by providing an alert as soon as a high probability for a collision to occur is detected, so that the braking process starts as early as possible.

The German automobile club (ADAC) published a study on average braking distances with respect to the vehicle speed [82]. Table 5.3 presents the average braking distance on dry roads with respect to the driving speed together with the distance the vehicle is driving in 1 second (human reaction time), which corresponds to an obstacle detection system operating at 1 Hz. Travelled distances between two samples of obstacle detection systems operating at 15 Hz, 25 Hz and 100 Hz are provided as well.

Table 5.3 Illustration of required braking distance at different driving speeds

Driving speed (km/h)	Average braking distance (m)	Human reaction time 1 Hz (m)	Inter-sample distance at 15 Hz (m)	Inter-sample distance at 25 Hz (m)	Inter-sample distance at 100 Hz (m)
30 km/h	4.34	8.33	0.56	0.33	0.08
50 km/h	12.06	13.89	0.93	0.56	0.14
60 km/h	17.36	16.67	1.11	0.67	0.17
90 km/h	39.06	25.00	1.67	1.00	0.25
100 km/h	48.23	27.78	1.85	1.11	0.28
120 km/h	69.44	33.33	2.22	1.33	0.33
130 km/h	81.50	36.11	2.41	1.44	0.36

It can be seen that the vehicle travels four times longer distances between two samples in the case of an object detection system operating a 25 Hz and even more

than six times longer for 15 Hz compared to the proposed operating frequency of 100 Hz. For instance, at 60 km/h the traveled distance for the proposed frequency is 0.17 m between two samples, compared to 0.67 m for 25 Hz systems and 1.11 m for 15 Hz systems. Additionally, one should keep in mind that on wet or snow covered roads the braking distance is considerably longer. In case of car accidents, especially when pedestrians are involved, each gained meter is important as it could make the difference between being able to stop the vehicle before the impact or not.

The main goal of the presented system is to warn the driver as fast as possible if there is a high probability for a collision to occur. For this reason the proposed system images the environment more frequent than the previously mentioned state-of-the-art systems, namely it captures images and processes them at 100Hz. Therefore the proposed system has the potential to detect obstacles and warn the driver significantly earlier than state-of-the-art systems.

5.2 Vision system for outdoor human detection and tracking

A stereo camera system with parallel optical axes was chosen for sensing the environment, as stereo cameras enable 3D reconstruction of object points and therefore the distance between the camera and the detected objects can be computed. This choice is justified by the fact that for collision warning it is at first necessary to detect objects in front of the vehicle, then to classify them as belonging to a particular object class and, at the end, to estimate the distance to the detected objects. As presented in section 2.6, a stereo camera system with parallel optical axes allows computing disparity maps after rectifying the stereo images, which can be used for object detection.

The complete image processing system including the cameras and their setup has been introduced in [3] and [4]. In [3] a Point Grey Bumblebee XB3 camera was used [79], which could be bought off-the-shelf and whose images were already stereo rectified. However the XB3 camera was not sensitive enough for low-light conditions and did not support high-speed operation at 100 Hz, so a different stereo camera was used in [4]. The stereo camera system in [4] had two pco.1200s high speed cameras [80], as shown in Fig. 5.1 a) (courtesy of PCO AG, Kelheim, Germany), which capture frames at 100 Hz with a resolution of 1280x480 pixels and a pixel depth of 10 bit, monochrome. This high number of frames per second opens the possibility of reliably tracking the objects in consecutive frames, which enables the computation of the objects' path and allows a more accurate estimation to be made of whether a collision

is likely to occur. The cameras capture gray-scale images, so that more light reaches the sensor than if a Bayer filter would be used for capturing color images, as explained in section 2.1. This approach allows the camera to operate at high frequency even in low-light conditions.

The cameras were placed behind the windscreen, as this location ensures good visibility for the cameras. The assumption is that the driver regularly cleans the windscreen in order to have a clean view while driving. Fig. 5.1 b) shows the positioning of the cameras on a special support behind the windscreen.

a) b)

Fig. 5.1 Stereo camera systems used for collision warning: a) pco.1200s camera and b) Camera placement inside the vehicle

The 10 bit dynamics of the cameras enable capturing images of good contrast in various illumination conditions such as sunny or cloudy days. The cameras' exposure time is automatically adapted using the image histogram in each frame for ensuring a good exposure. The 10 bit images are then scaled to 8 bit for further processing.

5.3 Vision-based human detection and tracking in street scenes

Starting from a stereo image pair, there are two main approaches for detecting a person imaged with stereo cameras. The first approach is to find the person in both images, to extract the person feature points from both images and afterwards to use triangulation as shown in section 2.6 in order to calculate the distance to the person. The second approach is to compute a disparity map as shown in section 2.6, to segment the disparity map with the method presented in section 3.2 and to classify the segmented objects as belonging to a particular object class as shown in section 2.4.

The biggest advantage of the second approach is that the objects can be easily separated from each other due to the depth information provided by the disparity map. This allows the shape of the object to be reliably extracted and the size of the object in meters to be computed. By using these features, the classifier becomes faster and more

reliable. Additionally, this method can be used in clustered environments as it is often the case in outdoor scenes. The biggest shortcoming of this method is that the image pair must be stereo rectified, which requires a reliable estimation of the camera parameters. On the other hand, this approach implies mostly simple operations that are suitable for hardware implementation and therefore for real-time operation.

Taking all the presented facts into consideration, an image processing system has been developed, which is able to detect and track various objects including pedestrians and persons on bicycles based on the novel disparity map segmentation method introduced in section 3.2. Fig. 5.2 presents an overview of this image processing system.

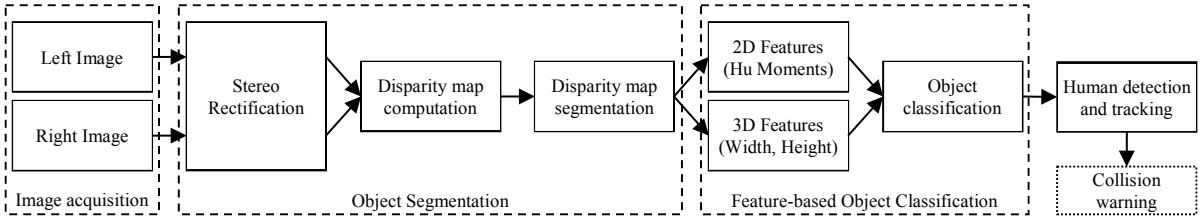


Fig. 5.2 Flow chart of the image processing system

In short, the captured stereo image pair is rectified and a disparity map is computed. The disparity map is segmented and the segmented objects are classified and tracked before the collision warning module estimates whether a collision is likely to occur.

5.3.1 Disparity map computation and segmentation

As shown in Fig. 5.2, the image pair is stereo rectified and a disparity map is computed using the block matching algorithm [44] introduced in section 2.6. For stereo rectified images each object has the same size in both images and each pair of corresponding points lie on the same image line. For better visualization, Fig. 5.3 a) and b) show the original stereo image pair and Fig. 5.3 c) and d) present the stereo rectified images.

It can be observed that in the original image pair, the bike is located considerably higher in the left image than in the right image, while in the rectified image pair the bike is on the same image line. The two rectified images are subsequently used to compute the disparity map using the block matching method with the following parameters: block size of 21x21 and a maximum disparity of 64. Afterwards the resulting disparity map shown in Fig. 5.3 e) is segmented using the novel method presented in section 3.2 and the result can be seen in Fig. 5.3 f).

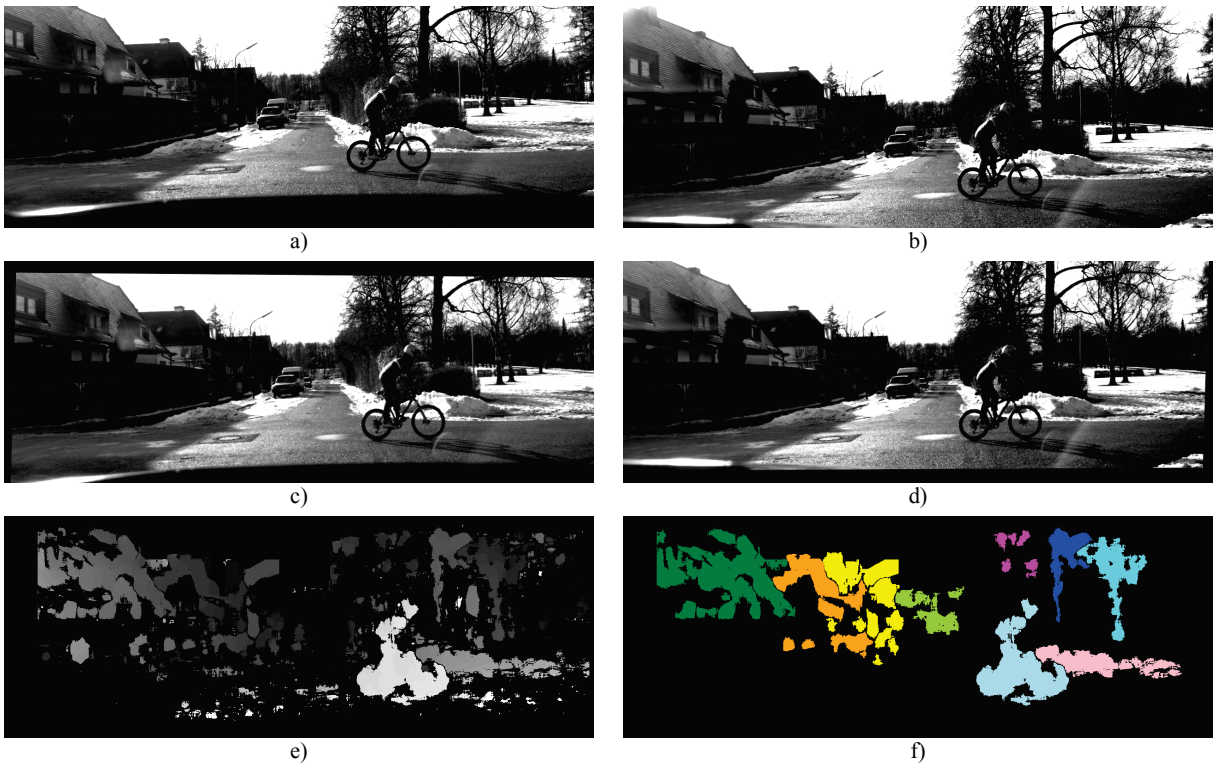


Fig. 5.3 Illustration of object detection in a street scene: a) Original left image, b) Original right image, c) Stereo rectified left image d) Stereo rectified right image, e) Computed disparity map using block matching and f) Segmented disparity map

The purpose of segmenting the disparity map is, on one hand, the separation of objects from each other and from the background and, on the other hand, noise filtering.

5.3.2 3D distance calculation

After object segmentation and convex hulls fitting, the distance from the camera to each object is computed, so that the probability of a collision to occur can be estimated. For this purpose the image coordinates of each object's closest point to the camera is computed and applied in equation (2.18). The result is a 3D point *A(x, y, z)* for each object, which describes the 3D position of the object's closest point with respect to the left camera. Based on the computed 3D point, the distance from the camera to each object is computed using (3.11).

As described earlier, the camera is installed behind the windshield, so the vehicle coordinate system {V} and the camera coordinate system {C} are different. Therefore the relationship between the two coordinate systems must be calculated in order to allow reliable computation of the objects' distances with respect to the front bumper of the car. Fig. 5.4 illustrates the two coordinate systems.

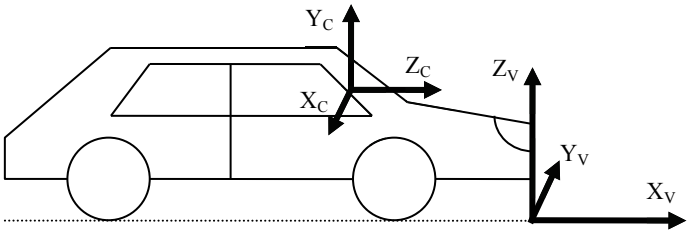
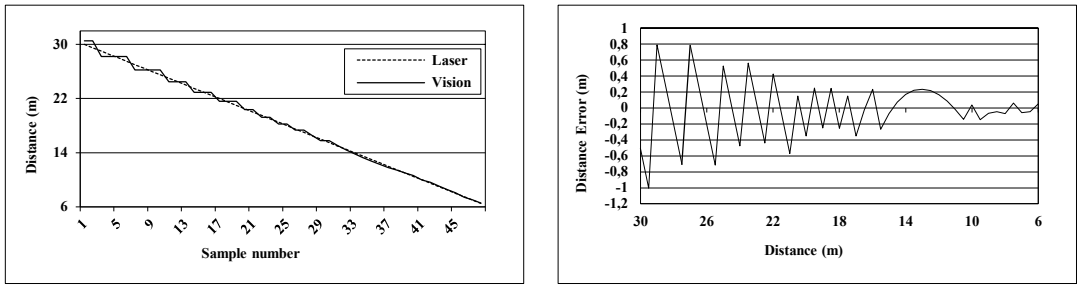


Fig. 5.4 Relationship between the camera coordinate system and the vehicle coordinate system

The translations between the camera coordinate system and the vehicle coordinate system are: T_x = -0.30m, T_y = -1.20m and T_z =1.65m, where T_x, T_y and T_z are translations along the x, y and z axes of the camera coordinate system. The axes orientations are: X_C is parallel to Y_V, Y_C is parallel to Z_V and Z_C is parallel to Y_V.

Performance evaluation

The main outcome of the presented algorithm is the distance between the vehicle and each of the objects in front of it, including pedestrians and bicycles. Therefore, in order to evaluate the performance of the algorithm, the computed distances were compared to ground truth data measured using a Bosch PLR 50 laser range finder [81]. In this setup, a person was standing in front of the car at a certain distance, measured using the laser range finder. At the same time, the vision system calculated the distance to the person using the proposed algorithm. This process was repeated for 49 distances ranging from 30m to 6m, with a step of 0.5m. Fig. 5.11 shows the distance results and the error between the measured distance and the data obtained using the proposed vision-based system.



a) b)

Fig. 5.5 Comparison between data obtained from the laser range finder and data obtained using the proposed stereo vision system a) and distance error b)

It can be seen that the absolute error decreases as the distance gets smaller and the relative error is not higher than 5% for the entire tested range of 6m to 30m.

5.3.3 Human detection and tracking

Even though the distance to each object in the scene in front of a vehicle can be very useful for collision warning, it is not sufficient for a reliable collision warning system. The simple distance of an object to the vehicle can give a hint on how likely a collision is to occur, but the object's own speed can offer additional information for a more intelligent collision warning system. For this purpose each target object in the scene should be tracked among a series of consecutive frames in order to compute its speed and to analyze its behavior. In the presented application, pedestrians and persons on bikes are the particular objects which should be distinguished from other objects in the scene and whose speed should be determined.

As already presented in section 2.4, various classifiers exist, which often use shape descriptors such as the Hu Moments for classifying objects in images. The purpose of using various features such as shape descriptors and the object's size is to reduce the feature space in a meaningful way compared to using e.g. all pixels on the boundary of the segmented object [65]. In order to illustrate the usefulness of Hu Moments as shape descriptors, Fig. 5.6 shows different examples of pedestrians and bikes together with their corresponding first three Hu Moments.

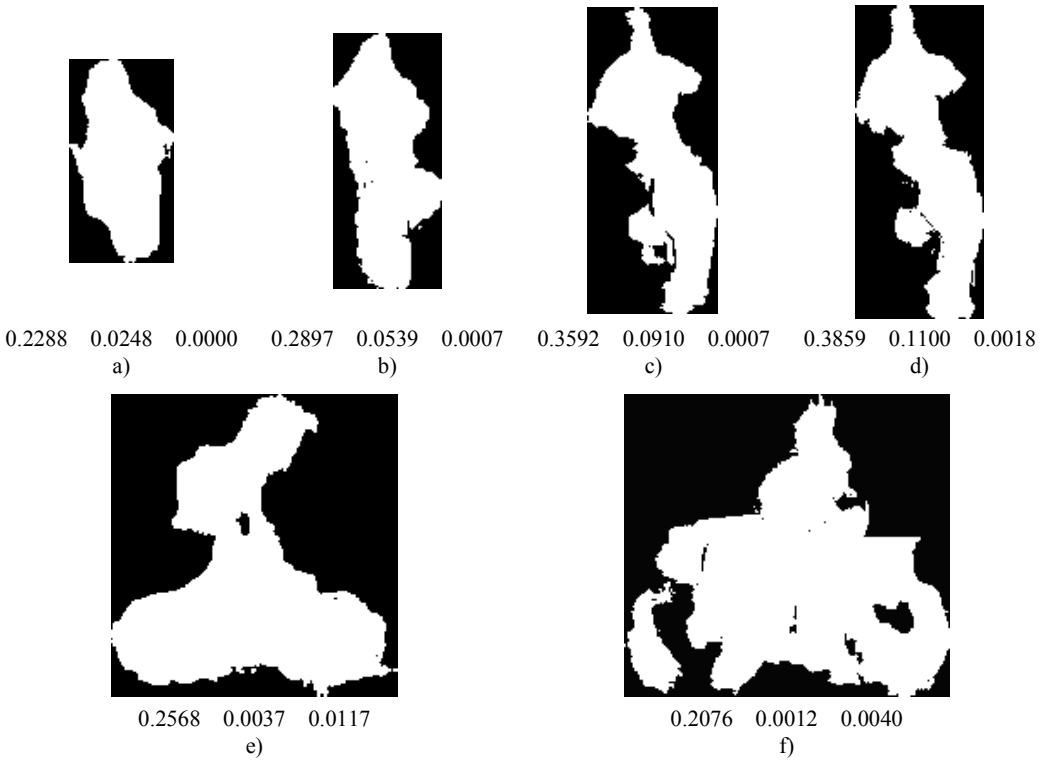


Fig. 5.6 First three Hu Moments as shape descriptors for the human in: a) Frame 350, b) Frame 375, c) Frame 399 and d) Frame 400 and two persons on bikes in e) and f)

Fig. 5.6 a)-d) show four image regions containing a running person in different frames of the same video together with their corresponding first three Hu Moments. Fig. 5.6 e) and f) show two segmented persons on bikes from two different scenarios together with their corresponding first three Hu Moments. The corresponding original image ROIs are shown in Fig. 5.7.

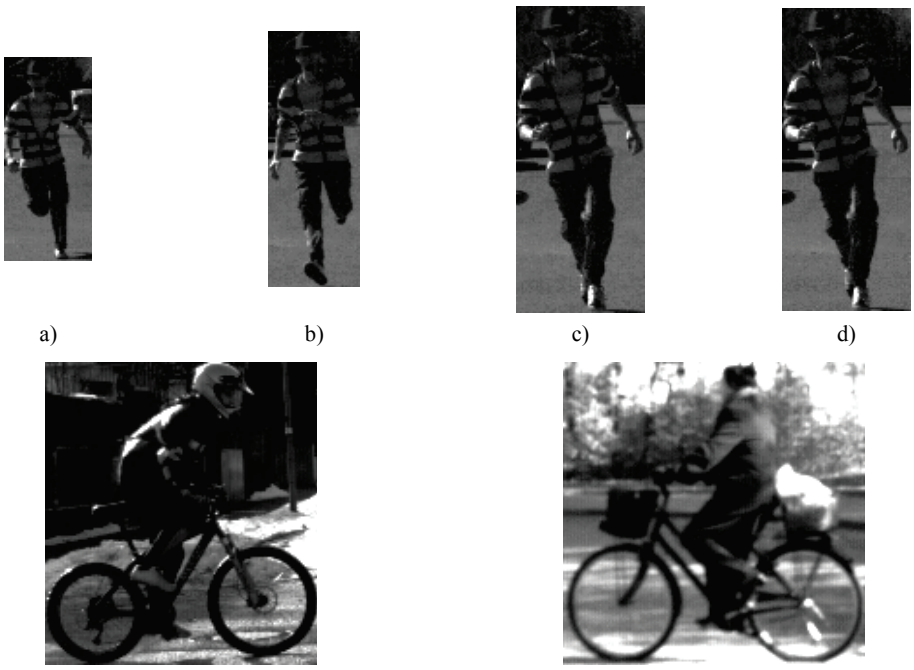


Fig. 5.7 ROIs of the original left image corresponding to the images in Fig. 5.6

It can be observed that the values of the first three Hu Moments are similar within each of the object classes, but are different between pedestrians and bikes. This indicates that the combination of these three features can be used for object classification. Additionally, the 3D width and height of the objects, as estimated by the proposed vision system with the method explained in section 3.2, are used for a more robust classification. The 3D width and height of objects express the object size in meters and are independent of the distance between the object and the camera. In contrast to this, the 2D with and height of objects are expressed in image pixels and change depending on the distance between the object and the camera (distant objects appear smaller).

Even though, on first sight, the object type might not seem important for the presented application, knowing to which class an object belongs to can help to better understand and predict its behavior and so to estimate its speed. For instance if an object is classified as person on a bike, the knowledge about its maximum speed can be used to estimate its most probable trajectory. On the other hand, if a human is

detected, it would be evident that the speed is very slow, but on the other hand the walking direction could suddenly change, so the path prediction has to consider this.

Minimum Distance Classifier

The minimum distance classifier is considered as a basic linear classifier which, as described in Chapter 2, only requires positive samples of the classes and no negative samples. This means that it does not need to know how an object does *not* look like. In its original form the minimum distance classifier requires only the mean value of the feature vector. However using only the distance to the sample means would lead to classifying every sample into one of the defined classes, as no distance boundary is considered. Therefore, appropriate boundaries should be set in order to best represent the distribution of the samples. This is achieved by setting the size of the region based on the standard deviation of the sample. This enhancement is also known as parallelepiped classifier, as it defines parallelepipeds within which objects are classified as belonging to a certain class.

A sample of 146 frames of pedestrians and 233 frames of bikes were used for establishing the mean and interval and 178 frames of pedestrians and 286 frames of bikes were used for testing. Even though the minimum distance classifier does not require negative samples for training, 100 frames of various objects like trees and bushes have been used to test how likely false positives would occur.

Table 5.4 shows the mean and standard deviation of the first three Hu Moments and the 3D size for the classes Pedestrian and person on a bike, denoted as Bike.

Table 5.4 Mean of the first three Hu Moments and size for the classes Pedestrian and Bike

Feature		Pedestrian (146 Frames)	Bike (233 Frames)
Hu Moment 1	Mean	0.326	0.276
	Std. Dev.	0.045	0.066
Hu Moment 2	Mean	0.065	0.003
	Std. Dev.	0.030	0.006
Hu Moment 3	Mean	0.002	0.016
	Std. Dev.	0.002	0.017
Height (m)	Mean	1.80	1.82
	Std. Dev.	0.10	0.06
Width (m)	Mean	0.73	2.03
	Std. Dev.	0.07	0.15

It can be seen that part of the feature values have overlapping intervals that can cause ambiguities during classification. However the combination of these five features has the potential to clearly differentiate between these two classes and also to separate them from “other” objects that could appear in the scene. In order to better

illustrate the class separation, Fig. 5.8 shows the sample distribution considering the simple two-dimensional feature space consisting of the first two Hu Moments.

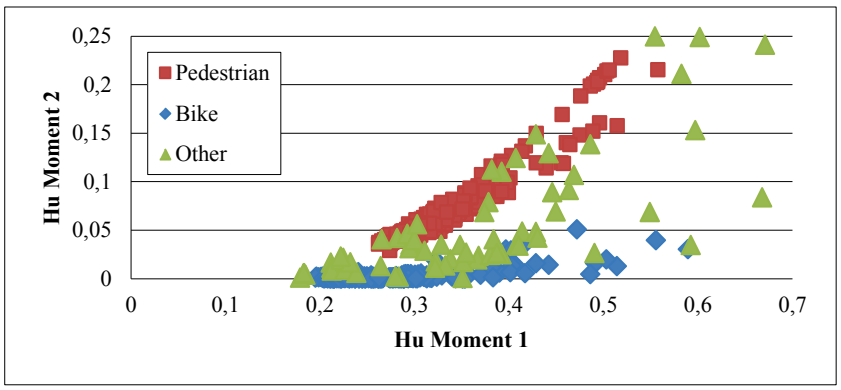


Fig. 5.8 Sample distribution in a two-dimensional feature space

It can be clearly seen that the Bike and the Pedestrian samples are well grouped. However, other random objects in the scene such as trees and bushes have similar values to the first two object classes, which underline the importance of using all five proposed features for classification.

Looking at the sample distribution in Fig. 5.8 it becomes obvious that for the minimum distance classifier the interval size plays an important role, as a wider interval has a higher probability to include all correct matches. However, the probability to include false positives increases with the interval size. In order to find the most suitable interval I_i for of the five features, several tests have been performed. In these tests the interval was varied according to:

$$I_i = m_i \pm s \cdot \sigma_i, \quad i = \overline{1..5} \tag{5.1}$$

where m and σ are the mean and the standard deviation for the five features according to Table 5.4 and s is the parameter that was changed to adjust the interval size. The values of s were chosen having in mind that for a normal distribution 99.7% of the samples would be included for s=3. The test results are presented in Table 5.5.

Table 5.5 Various intervals used for classification and the corresponding results

		Pedestrian (178 Frames)		Bike (286 Frames)		Other (100 Frames)
Test setup	Interval size	Correct matches (%)	False matches (%)	Correct matches (%)	False matches (%)	False matches (%)
Test 1	s = 1	78.41	0.00	76.27	0.00	4.00
Test 2	s = 1.5	88.89	0.00	83.05	0.00	11.50
Test 3	s = 2	91.61	0.00	87.29	0.00	15.50
Test 4	s = 3	94.34	0.00	91.53	1.70	27.50

It can be observed that the higher the interval is the more false positives appear in the object class "other". On the other hand, making the interval narrower also decreases the number of correct matches. This indicates that choosing the interval is not a trivial task. However, the best compromise in this situation appears to be the third test, as it has a relatively low number of false matches while still achieving a high number of correct matches.

The slightly smaller number of correct matches for bikes can be explained by the various appearances depending on the viewing angle. This is due to the fact that the difference of a pedestrian's appearance when viewed from the front and from the side is significantly smaller than for a person on a bike.

The biggest advantage of using the minimum distance classifier with respect to other classifiers is that it is very simple to implement and also to set its parameters. However it only works well in simple scenarios and selecting the right interval is not always trivial, as could be seen in Table 5.5, as a too high value increases both the correct and the false matches and a low value decreases them both.

Support Vector Machines

Support vector machines (SVM) are more elaborate classifiers which can be trained with a large sample, even in multivariate feature spaces, but the basic implementation can only differentiate between two classes, as was explained in section 2.4. The main benefit of using support vector machines is that the largest boundary between the two classes is automatically computed rather than the interval containing the samples, as is the case for the minimum distance classifier. However support vector machines need linearly separable feature distributions as well in the basic implementation, which limits the number of suitable applications.

As could be seen in Fig. 5.8, the data is not linearly separable which indicates the possibility that standard SVM might not be able to perform well on this data set. In order to enable classification of the three classes, a "one-versus-one" approach was used, as explained in section 2.4. SVM attempts to find the hyper planes which best separate each of the two classes and leave the biggest possible buffer between them. In order to test the performance of the classifier, the same data used for the minimum distance classifier was used to train the SVM classifier as well. Fig. 5.9 illustrates the result of SVM hyper plane detection for a simplified two-dimensional feature space for the classes Pedestrian and Bike.

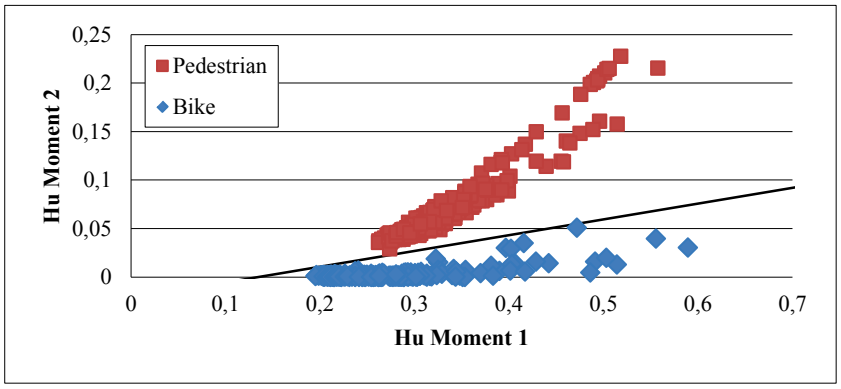


Fig. 5.9 Illustration of the classification hyper plane generated by SVM for a simplified two-dimensional feature space for the classes Pedestrian and Bike

The classification result was slightly improved compared to the minimum distance classifier. However, the linear nature of the classifier causes an important number of other objects to be classified as either being pedestrian or bike. A comparison of the performance of all tested classifiers is shown in Table 5.6.

Neural Network Classifier

The third classifier tested on the sample data is a back propagation neural network [36], which can directly classify between three classes and can deal with data that is not linearly separable. As introduced in Chapter 2, the network has been adapted in order to match the presented feature data. Accordingly, the network has five nodes in the input layer (one for each feature), three nodes in the output layer (one for each output class) and four nodes in the hidden layer. Fig. 5.10 illustrates the structure of the used neural network.

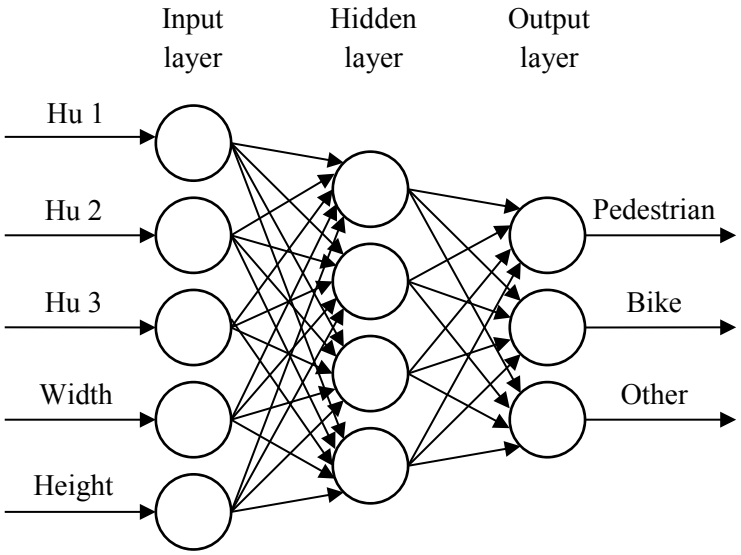


Fig. 5.10 Artificial neural network used for classification

It can be seen that each of the five inputs is connected to a node in the input layer and the output layer contains nodes for the three possible outputs: pedestrian, bike and other. The purpose of the hidden layer is to offer a good performance on data which is not linearly separable, as is the case for this application.

The nodes in the output layer work in a normed complementary way, meaning that the sum of all three nodes is always equal to one. Ideally, if an object is classified as belonging to one of the classes, the corresponding node's output should be 1 and both other outputs should be 0. In practice the maximum of the three outputs determines to which class an object belongs.

The training and testing of the network was performed using the same dataset that was used for the other two classifiers and the performance evaluation of all three methods can be seen in Table 5.6.

Table 5.6 Comparison of classification results for the three presented classifiers

	Pedestrian (178 Frames)		Bike (286 Frames)		Other (100 Frames)
Classifier	Correct matches (%)	False matches (%)	Correct matches (%)	False matches (%)	False matches (%)
Minimum Distance Classifier	91.61	0.00	87.29	0.00	15.50
Support Vector Machines	92.33	0.00	88.54	0.00	9.85
Neural Network Classifier	95.77	0.00	93.21	0.00	2.35

The neural network classifier exhibits the best results, as it is able by its nature to deal with data which is not linearly separable. The biggest improvement with respect to the other two classifiers can be seen in the amount of falsely classified "other" objects. Only 2.35% of other objects were falsely classified either as pedestrian or as bike. All three classifiers were able to correctly distinguish between pedestrians and bikes, as neither of them falsely classified bikes as pedestrians or vice versa.

Human tracking

After the objects belonging to one of the two important classes (pedestrian or bike) were filtered out from the rest of the segmented objects, the final step is to track them in a series of consecutive frames and to compute their own speed. For this purpose a tracker filter is used.

Basically a tracker is a low-pass filter, which is initialized with a region within a particular frame of a series of images, like a video or live stream and its goal is to detect the same region in the next frame. As described in section 2.7, there are different methods for object tracking in video sequences, which mainly differ in the way the data that is collected from the images and how they are processed.

Since all objects of the scene are already separated from each other and from the background, the tracking process is reduced to matching objects detected in one frame to the objects already detected in previous frames. The five features that were used for classification, together with the 3D position of the objects are used for object tracking. Hereby for newly detected objects, which were classified either as pedestrian or as bike, a new tracker is initialized, while for existing objects the tracker should update the position of the object and predict their position in the next frame in order to ensure robustness against image processing failures, in the case where the object failed to be detected in a frame, but is afterwards detected again.

The implemented tracker has two operation modes: initialization and tracking. During initialization the tracker is empty and all pedestrians and bikes detected in the first frame in which at least one object is classified as belonging to one of these classes are added to it. Afterwards the tracker switches to tracking mode until no object is tracked anymore, when it returns back to the initialization mode. During tracking there are three phases: tracking existing objects, adding new objects, removing out-of-range objects.

For each new frame the tracker receives the list of objects belonging to either the class pedestrian or bike together with their shape descriptors (Hu 1, Hu 2, Hu 3) and 3D size (Width, Height), as well as the 3D coordinates of the center of their bounding cuboid. The bounding cuboid is considered as the smallest cuboid that fits the 3D object. An example of bounding cuboid is shown in Fig. 5.11 b).

The tracker compares the values of the shape descriptors and of the 3D coordinates to the values of each currently tracked object in order to check whether any of the objects from the new frame corresponds to a tracked one. Detected objects are considered to correspond to tracked objects if the differences for their shape descriptors values, their size and their position are smaller than the set threshold values. This operation is based on the nearest neighbor method [11]. If more than one object in the current frame corresponds to a tracked object, the most likely correspondence is used to update the position and speed of the tracked object along each of the three axes of the car's coordinate system and the new object is removed from the list of new objects. After either all tracked objects have been updated or no correspondences are found anymore, all remaining objects in the list of objects from the new frame are added to the tracker as new objects to be tracked.

When adding new objects, their shape descriptors and 3D coordinates of the bounding cuboid are stored. After at least three consecutive correspondences have

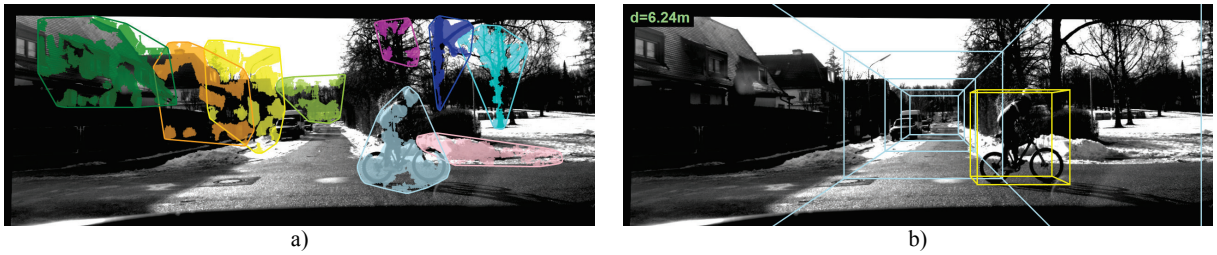
been found, the mean speed among these three frames is computed and fed to the collision warning module.

If for an object no correspondence could be found for a particular frame, the shape and size are presumed to stay constant, as it logically should be, since neither of them changes significantly between five consecutive frames. The 3D coordinates of the bounding cuboid, on the other hand, are updated based on the previously computed speed and considering that the speed remained constant. This prediction is performed for no more than five consecutive frames. If in the sixth frame still no correspondence is found, the object is removed from the list of objects to be tracked.

5.4 Collision warning for automotive applications

The final step of the presented application is to warn the driver whether a collision is likely to occur. For this purpose it is common in the automotive industry to define a so called driving tunnel, which represents a virtual corridor through which the vehicle has to pass in order to follow its course. A similar concept exists in aeronautics, where such a virtual corridor is used for ensuring that a plane has enough space for safe flying. In the presented application this driving tunnel is used to estimate possible collisions by detecting if any object either intersects this driving tunnel or is about to enter the driving tunnel closely to the vehicle.

The collision warning algorithm is based on the previously computed objects positions and on the defined driving tunnel. Assuming the car is heading forward, the driving tunnel is defined as the space required by the car in order to safely pass and it is 50% wider and 50% higher than the car width and height respectively. An example of street scene with overlaid driving tunnel is shown in Fig. 5.11 b), which is the output of the algorithm for the scene shown in Fig. 5.3. Fig. 5.11 a) shows the result of the disparity map segmentation after convex hull fitting, overlaid on the rectified left image, which is used for classification and for detecting the closest object.



a) b)

Fig. 5.11 Result of disparity map segmentation overlaid on rectified left image: a) Segmented disparity map and b) Driving tunnel and bounding cuboid for the closest object

The driving tunnel in Fig. 5.11 b) contains markings every 5 meters from the bumper, until a distance of 20 meters. Also a cuboid is displayed, which bounds the closest object: the bike.

If an object is inside the driving tunnel or even close to it, the driver should be warned. For this, the distance between the front bumper of the car and the closest point of the object is used. The computed distance is shown in Fig. 5.11 b) in the upper left corner. If an object is inside the driving tunnel, based on its distance to the vehicle, the collision warning returns a severity level W_L according to:

$$W_L = \begin{cases} low, if\ D \geq 20m \\ medium, if\ D < 20m\ AND\ D > 10m \\ high, if\ D \leq 10m \end{cases} \tag{5.2}$$

where D is the distance defined in (3.11).

If more than one object is in the driving tunnel at a time, the severity level is given by the closest object. However, if the object is outside the tunnel and no object is inside the tunnel, the severity level is either medium or low, depending on the distance between the object and the tunnel. In Fig. 5.11 b) the bike is located at 7.89m with respect to the camera. The shift between the camera coordinate system C and the vehicle coordinate system V is 1.65m on the Z axis, so the distance from the front bumper to the bike is actually 6.24m. But because the bike is outside the tunnel the returned severity level W_L is *medium* as depicted by the yellow colored bounding cuboid. In order to better illustrate the functionality of the collision warning system, Fig. 5.12 shows four additional frames from the same street scene.

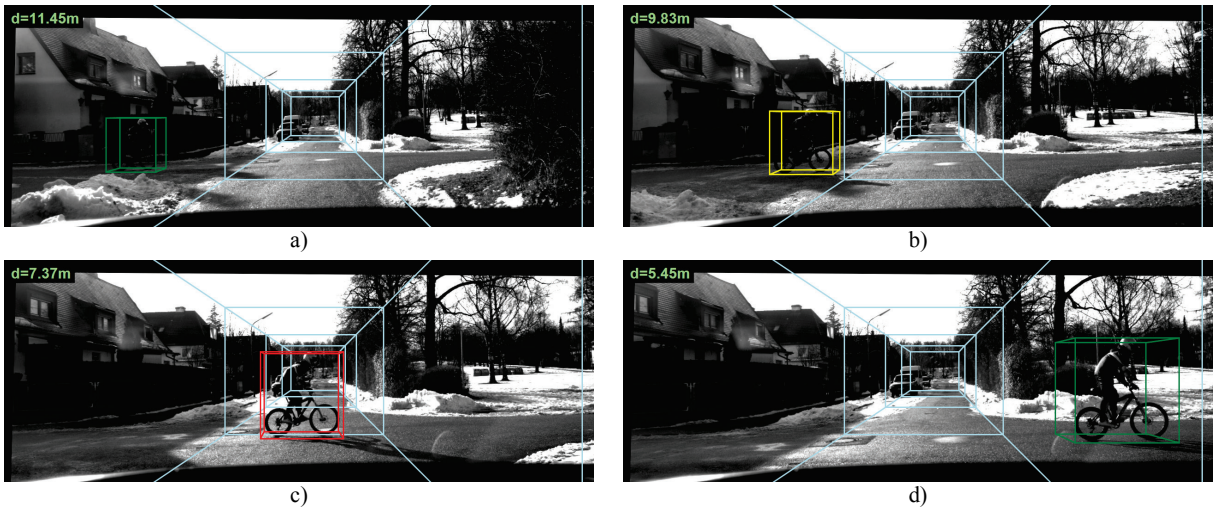


Fig. 5.12 Illustration of the collision warning system for a bike driving across the driving tunnel: a) Bike outside driving tunnel, b) Bike on the edge of driving tunnel, c) Bike inside driving tunnel and d) Bike outside driving tunnel

It can be seen that while the bike is outside the tunnel and far away from it, the severity is *low*, when it gets closer to the tunnel the severity level is *medium* and inside the tunnel due to the small distance to the car it is *high*. Later, as the bike exits the tunnel, the severity level is *medium* again as shown in Fig. 5.11 b) and finally it is *low* as the bike exits the scene.

As could be seen, in the presented street scene the bike was entering the scene on the left side, crossing the street in front of the car and exiting on the right side. Another street scene is presented in Fig. 5.13.

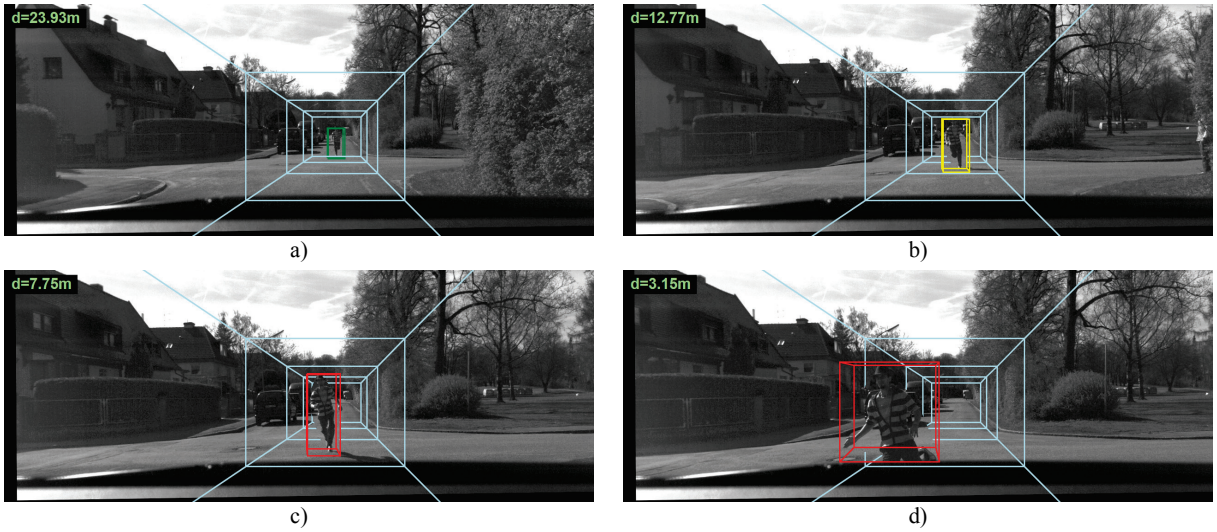


Fig. 5.13 Illustration of the collision warning system for a pedestrian running towards the car inside the driving tunnel: a) Pedestrian at far distance, b) Pedestrian at medium distance, c) Pedestrian at close distance and d) Pedestrian at very close distance

Fig. 5.13 shows a pedestrian who is running in front of the car and runs towards it, while it only avoids the collision just in front of the car by running out of the driving tunnel to the left of the car. It can be observed that the image series from Fig. 5.13 contains a more accurate augmented reality, as the person appears to really be inside the tunnel. This is achieved by using the result of the convex hull fitting to mask which image region should not be covered by the tunnel lines.

5.5 Hardware acceleration of image processing algorithms

In order to ensure that all presented algorithms run fast enough to maintain the desired 100Hz frequency, hardware acceleration is necessary, as CPUs can't achieve the desired performance. Even though GPUs are not suitable for automotive applications due to the high power requirement, they are very helpful as fast prototyping platform for developing the parallelized algorithms for low-power FPGAs.

As the GPU can only perform one operation at a time with all its kernels, according to the SIMD principle, the operations run sequentially, as already presented in Fig.

2.19, exactly as is the case for CPUs if a single core is used. This means that the last algorithm in the chain has to finish processing a frame before the first algorithm can start processing the next frame.

5.5.1 Stereo rectification

As already shown, stereo-based image processing algorithms usually require each image pair to be stereo rectified. After calibration, knowing all intrinsic and extrinsic camera parameters, so-called rectification maps can be computed for both the left and right image and which encode the relationship between the pixels in the rectified image and the pixels in the original image. Fig. 5.14 shows a numerical example of image to be rectified and corresponding rectification maps obtained after calibration.

v \ u	104	105
153	99	146
154	121	133

a)

v \ u	130	131
180	104.4876	105.4591
181	104.5085	105.4799

b)

v \ u	130	131
180	153.1628	153.1587
181	154.1522	154.1481

c)

v \ u	130	131
180	123	172
181	147	177

d)

Fig. 5.14 Illustration of the stereo rectification process with bilinear interpolation: a) Original image b) Rectification map *u* c) Rectification map *v* d) Resulting image

Numerical values of a small region of an image are presented in Fig. 5.14 a). The rectification maps required for stereo rectification are presented in Fig. 5.14 b) and c) for the *u* and *v* coordinates respectively. The resulting pixel *r(u, v)*of the rectified image depends on the two values from the rectification map at the specified coordinates *(u, v)*. For example, $r(130,180)=f_b(104.4876, 153.1628)$, where f_b is the bilinear interpolation function.

It can be seen that *r* depends on floating point values of pixel addresses, which means that for each pixel its four neighboring pixels will be used. For the presented example these four pixels are presented in Fig. 5.14 a). If these pixel values are denoted left-right and top-bottom f_{00}, f_{01}, f_{10} and $f_{11}$the following formula illustrates the bilinear interpolation function f_b:

$$r(u,v)=f_{00}\cdot(1-x)\cdot(1-y)+f_{10}\cdot x\cdot(1-y)+f_{01}\cdot(1-x)\cdot y+f_{11}\cdot x\cdot y \tag{5.3}$$

where *x* and *y* represent the fractional part given in the rectification map. In the given example, $x=0.4876$ and $y=0.1628$ and the final rounded result is $r(130, 180)=123$.

The bilinear interpolation is performed sequentially on a CPU. If multiple cores are used, the transformation is speeded up since the computation of one pixel only depends on the four pixels specified by the rectification map. However, as GPUs were initially designed to be used in computer games, they allow 2D and 3D data structures

to be loaded as so-called textures. A texture in computer graphics has the property that it can be addressed with floating point indices and the required bilinear interpolation is done in hardware and transparently to the user in order to allow quick modifications of the displayed scene, such as slightly changing the viewpoint. Stereo rectification transforms the original images by bilinear interpolation specified by the rectification maps. Therefore, using textures considerably speeds up the process, as the rectification is reduced to traversing the image once and directly reading the output value specified in the rectification map.

5.5.2 Disparity map computation

One of the more time consuming image processing algorithms is the disparity map computation based on block matching, which is specific for stereo vision and has been introduced in section 2.6. The basic idea is to find for each block of pixels in the left image the corresponding block of pixels in the right image that correlates best. These corresponding blocks are then used to compute the disparity value for the central pixel of the block, which can be later used for 3D reconstruction. In order to compute a disparity map, the input images must be stereo rectified as explained in section 2.6.

The most commonly used cost function for block matching is the SAD (Sum of Absolute Differences) due to its computational simplicity and yet reliable results. Another widely used cost function is the Euclidean distance, but it implies squaring and calculating square roots, both of which are very time consuming compared to computing absolute differences and summing them. A third commonly used cost function is the SSD (Sum of Squared Differences), which is similar to the Euclidean distance, but without extracting the square root. However, squaring often leads to huge numbers which can lead to overflows especially when operating on integers, as is the case for many embedded systems. Even though SAD has a relatively low reliability when compared to the other two cost functions, it is still widely used due to its simplicity and execution speed.

The basic principle of block matching using SAD is to iterate through each pixel in the left image, to extract a block having the considered pixel in the center, surrounded by a number of neighbors and to match this block to blocks of the same size in the right image with the help of SAD. Initially, the block in the right image contains pixels located at the same coordinates as in the left image and is later on shifted to the left pixel by pixel and the SAD value is computed again. This operation is repeated until all disparity levels have been analyzed, meaning the maximum distance on the v axis between the blocks in the left and in the right image has been reached. The best

matched block generates the smallest SAD value and represents the most probable disparity value. Several post-processing steps refine this search by testing for the reliability of the result, by eliminating textureless regions and by adding sub-pixel accuracy.

The pseudo code for CPU implementation considering the presented block matching algorithm with SAD and without post-processing can be seen below.

```
FOR V = MIN_V to MAX_V
    FOR U = MIN_U to MAX_U
        IDX = V * IMG_WIDTH + U
        MIN_SAD = MAXINT
        DISP = 0
        FOR D = MIN_D to MAX_D
            UL = U
            UR = U - D
            CURR_SAD = computeSAD (UL, UR, V)
            IF CURR_SAD < MIN_SAD
               MIN_SAD = CURR_SAD
               DISP = D
            END IF
        END FOR
        DISP_IMG [IDX] = D
    END FOR
END FOR
```

In order to obtain the *U* and *V* limits only pixels that have all required neighbors are considered, together with the maximum disparity and the size of the SAD window:

```
MIN_U = MAX_DISP + SAD_SIZE/2 – 1
MAX_U = IMG_WIDTH - SAD_SIZE/2 – 1
MIN_V = SAD_SIZE/2
MAX_V = IMG_HEIGHT - SAD_SIZE/2 – 1
```
(5.4)

The minimum and maximum disparities are obtained from the application requirements as these limits depend on the distance for which objects are expected as well as on the camera parameters and image resolution. Typical values for the presented application are `MIN_D = 20` and `MAX_D = 64` as too far (small disparity values) as well as too close (large disparity values) objects can be ignored. The *computeSAD* function computes the sum of absolute differences for a pair of given blocks in the left and in the right image. It takes as an input the location of the pixels to be matched, which are in the center of the block to be matched.

The simplest GPU implementation leads to slight changes to this pseudo code by letting each thread process only a few pixels, as can be seen below.

```
V = THREAD_IDX
U = BLOCK_IDX
IDX = V * IMG_WIDTH + U
MIN_SAD = MAXINT
DISP = 0
FOR D = MIN_D to MAX_D
    UL = U
    UR = U - D
    CURR_SAD = computeSAD (UL, UR, V)
    IF CURR_SAD < MIN_SAD
       MIN_SAD = CURR_SAD
       DISP = D
    END IF
END FOR
DISP_IMG [IDX] = D
```

In the presented pseudo code, the changes with respect to the CPU version are presented in bold and `BLOCK_IDX` and `THREAD_IDX` are values delivered by the CUDA API. The block ID is common to all threads running on an MP (and which can collaborate via the shared memory) and the thread ID is unique for each thread. It can be seen that the *for* loops have been replaced by block and thread indices, which indicate which thread of which block is currently accessing the function. Since the number of rows and columns of the image might be smaller than the number of used threads (multiple of 32) and blocks, the validity of the indices has to be checked within the *computeSAD* function. A more complex numerical example is given in [9].

It can be seen that the pseudo code for the GPU implementation is not very different from the pseudo code for the CPU implementation, but the operating principle, as well as the underlying hardware are considerably different.

As could be seen, the disparity map computation can be parallelized very well, but this parallelization reduces the efficacy at the same time, as many mathematical operations have to be repeated. A novel method for parallelizing the disparity map computation introduced in [2] makes full use of the high number of cores of a GPU, keeping at the same time the number of redundant operations at a minimum.

On a CPU the block matching algorithm runs sequentially if only one core is available. If multiple cores are used, the process can be fastened up as the Sum of Absolute differences (SAD) has to be computed only within a given window. If the

SAD window has the size of n x n pixels and the maximum disparity is d_M, for each pixel a window of n x (n+d_M –1) has to be processed. While the window is moving, many of the summed values can be reused, as presented in [44], which leads to a big improvement compared to the straight-forward implementation. For each additional core the processing time is improved but not halved due to the fact that some of the calculations which could have been reused on a single core must be repeated by the second core in order to initialize the algorithm.

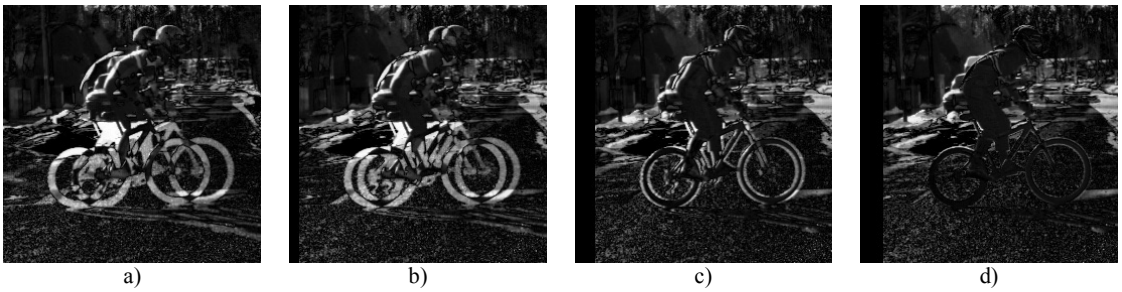
This could lead to the idea that block matching is not suitable for GPU implementation, as the strongest attribute of the GPU is the multitude of cores. Another downside of SAD is that it contains two operations: sums and differences, while on a GPU the operations are performed according to the SIMD principle [21], where all cores perform the same operation on different datasets.

In order to obtain significant improvement, for GPU implementation the processing steps of the algorithm have been redesigned and divided in three steps:

- Computing the absolute differences
- Computing the sums
- Determining the disparity

Computing the absolute differences

Given the maximum expected disparity d_M, the absolute difference is computed between the whole left and right images and afterwards the right image is shifted one pixel to the right and the process is repeated until the maximum expected disparity is reached. Fig. 5.15 shows the result of computing the differences between the left and right image at different disparities.



a) b) c) d)

Fig. 5.15 Absolute difference between the shifted right image and the left image for different disparities: a) d = 0, b) d = 10, c) d = 20 and d) d = 23

It can be observed that in Fig. 5.15 a) the bike does not overlay at all, while it perfectly overlays in Fig. 5.15 d). A good overlay is characterized by regions where the resulting image is black. Thus the disparity value of the bike in this scene is d=23.

Computing the sums

In order to judge whether a region of the image has been well overlaid or not, a sum is computed for each pixel within the defined neighborhood window. For the CPU version optimizations can be done to reuse partial sums or to use integral images [64]. On a GPU, the task is split among the multitude of cores so that each core computes the sums for certain pixels. Hereby the tasks are performed in parallel and the summing is performed very fast.

Determining the disparity

The first two steps are repeated for all expected disparities resulting in an array of d_M values for each pixel in the image. This array holds the Sum of Absolute Differences (SAD) for each potential window. At the minimum value of the array the best match is found and its index represents the most likely disparity value.

The biggest advantage of the restructured algorithm is that a large number of cores bring a huge improvement by parallelizing the operations, including the subtracting and summing.

Recent GPU boards and programming libraries allow individual cores to be assigned different functions. This feature could improve the benefits of using a GPU even more.

5.5.3 Disparity map segmentation

The novel disparity map segmentation algorithm presented in section 3.2 is by its nature a sequential algorithm, which cannot be parallelized, as each pixel depends on two previously computed neighboring pixels. However, by processing sub-regions of the image in parallel and merging them later, the real-time capabilities of the algorithm are improved without introducing a big overhead.

Therefore, the challenging parallelization task was solved by splitting the image into many sub-images, as one core would always just process one sub-image. After all sub-images have been processed, they are merged by investigating the border pixels. If border pixels belong to the same region, both blobs from the two sub-images are merged in the second pass. For minimizing the ratio of border pixels to total number of pixels, ideally the sub-images should be square. For the example presented in section 3.2, more precisely in Fig. 3.10 and Fig. 3.11 the image is split in (almost) square sub-images in which blobs are detected in the first pass and joined in the second pass. Fig. 5.16 a) and b) present numerical values after the two-pass segmentation algorithm has been applied to the image in Fig. 3.10 a) using a single core, and Fig. 5.16 c), d) and e)

using two cores. It can be seen that even though the label numbers in Fig. 5.16 e) are not the same as in Fig. 5.16 b) due to the multi-core approach, the segmented regions are identical.

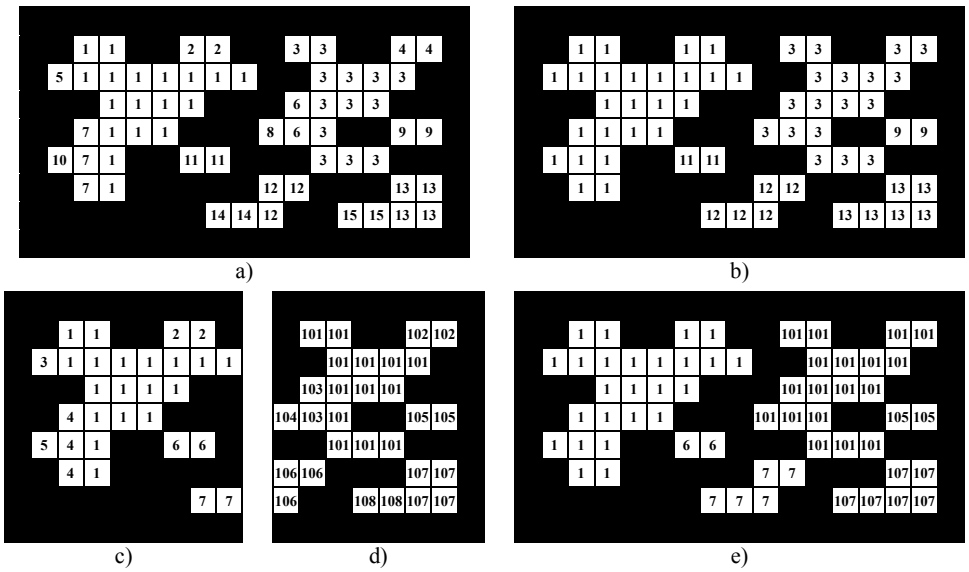


Fig. 5.16 Illustration of disparity map segmentation on one and two cores: a) First pass on single core, b) Second pass on single core, c) First pass on the first of two cores, d) First pass on the second of two cores and e) Second pass on both cores after merging

The main advantage of this algorithm is that in the second pass it is already known to which blob every pixel belongs, so each GPU core can run in parallel for determining the final blobs.

Similarly to this two-core example, the algorithm can be distributed to any number of cores having in mind that the distribution is efficient only if the number of border pixels is considerably smaller than the total number of pixels in each of the sub-windows processed by individual cores.

5.5.4 Processing time comparison for CPU, GPU and FPGA implementation

On both the CPU and the GPU, due to the lack of pipeline abilities, the processing time for all algorithms is added up and the result for each frame is available after this time interval. However, the GPU allows all algorithms to run faster than on the CPU, as different image regions are processed in parallel. The FPGA enables algorithms to run faster as multiple processing units can be defined, but it also supports pipelining, achieving therefore even faster image processing.

The big advantage of pipelining could be observed in Fig. 2.20. Even though the latency of the collision warning system implemented in FPGA is 9.5ms, the frame processing time for the FPGA is given by the most time consuming operation and is equal to 6.8ms, which means that 9.5ms after the first frame has been acquired, the

corresponding collision warning output is available and every 6.8ms afterwards a new collision warning output is ready. As the image acquisition time is smaller than 0.5ms, it is neglected. Table 5.7 shows a comparison of average processing times required by CPU, GPU and FPGA to process individual frames of the same video sequences.

Table 5.7 Comparison of processing time required by CPU, GPU and FPGA

Operation	CPU	GPU	FPGA
Stereo rectification	3.5 ms	0.2 ms	1.0 ms
Disparity map computation	16.7 ms	5.4 ms	6.8 ms
Disparity map segmentation	14.1 ms	3.3 ms	1.4 ms
Collision warning	3.1 ms	0.7 ms	0.3 ms
Frame processing time	37.4 ms	9.6 ms	6.8 ms
System latency	37.4 ms	9.6 ms	9.5 ms

For both the CPU and GPU, the total processing time and the system latency are equal, since each frame has to be completely processed before the next frame can enter the system. It can be observed that computation on GPU is about 3 times faster than on CPU, but it is topped by the FPGA, which also requires less than 10W compared to the 200W of the GPU. Distributed computing might have enabled even faster processing than the FPGA, but running multiple PCs in an automotive application is not an option.

It can be seen that disparity map computation runs slower on the FPGA than on the GPU. The reason for this is the difference in the clock frequency of the two systems. While the GPU runs at 500 MHz, the used FPGA runs only at 125 MHz and no additional benefit is obtained by parallelization in FPGA, as the GPU already splits and processes the image very efficiently. Details on GPU and FPGA implementations can be found in [9].

6

Real-time human tracking for Human-Robot Interaction (HRI)

Nowadays a lot of effort is put to enable humans and robots to work together. Based on the algorithm presented in section 3.2, the vision system of an intelligent robot has been developed, with the goal to enable human-robot cooperation in investigation of hazardous environments. Hereby the robot should act as a mobile carrier platform for collected samples. For that the robot should detect and follow the human co-worker who is collecting samples and putting them onto the robot platform. In order to achieve this, the robot calculates the distance between the target person and itself using stereo vision and follows the person with the appropriate speed to keep the distance constant. After sensing the reduction in distance indicating the human's intention to approach the robot, the robot has to stop and to allow the human to place the containers with the collected samples onto the robot's mobile platform.

The developed distributed system architecture was first introduced in [1] and enables real-time vision-based robot motion control by employing multiple computers to work together. It will be shown in this chapter that the performance boost obtained by this architecture is comparable to the improvements achieved by the previously shown hardware implementations with the help of GPU and FPGA.

6.1 Real-time capabilities through distributed computing

The time effectiveness is mainly achieved through the distributed processing that was implemented using ROS (Robot Operating System) [34], but also by making use of a Kalman filter [35] for predicting the most likely image ROI (Region of Interest) that contains the tracked human, hereby reducing the effective image data that has to be processed.

In the presented scenario the vision-based system for human tracking has to be able to detect the human, to calculate the distance to the human and to track the human, similarly to how the pedestrian detection application in Chapter 5 is operating. However, in this scenario the 3D position of the human is used by the robot in order to keep a constant distance to him instead of warning the driver, as was the case in Chapter 5. An illustration of the visually controlled robotic system for human tracking is shown in Fig. 6.1.

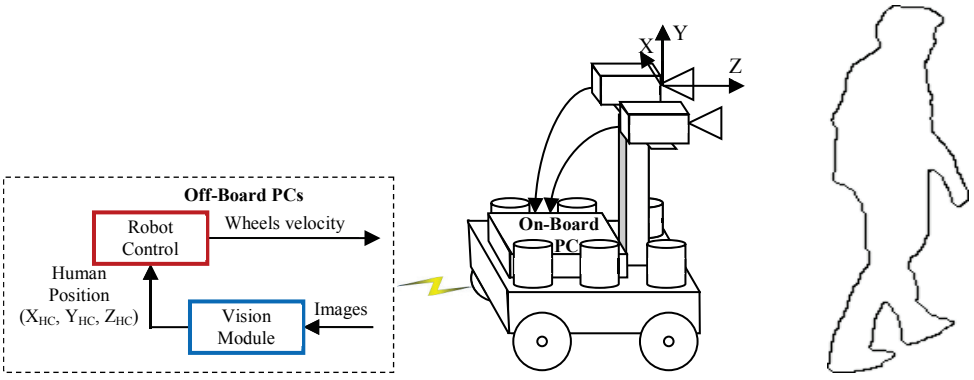


Fig. 6.1 Principal layout of the robotic system for following the human co-worker

The presented robotic system has a Point Grey Bumblebee XB3 stereo camera system on board [79], as well as a Nexcom NISE 3500 P2 low-power industrial PC (Intel i7-620M @ 2.66 GHz, 4GB of RAM, 64 GB SSD, ~65 Watt) [86]. As the platform is intended to run on batteries in order to move independently of external power supplies, low power consumption of the on-board components is critical. Therefore, the on-board PC functionality is limited to capturing images from the stereo camera and sending them compressed over the wireless link to the off-board vision module as well as sending direct commands obtained from the robot control to the wheels controllers.

In order to ensure that the control commands are sent at regular time intervals, the vision module and the robot control module run on different PCs as illustrated in Fig. 6.1 with differently colored blocks. The computational expensive vision algorithms run on a dedicated high-end PC (Intel Xeon E5520 @ 2.26 GHz, 6GB of RAM, Nvidia

Tesla C1060 GPU) with the goal of tracking the human co-worker in front of the robot and determining its 3D position with respect to the coordinate system of the left stereo camera, as depicted in Fig. 6.1.

After obtaining the 3D position, it is sent to the robot control module located on a separate desktop PC (Intel E4700 @ 2.6 GHz, 2GB of RAM), which computes the required velocities for each wheel such that the robot keeps following the human. The velocities are sent at regular time intervals over the wireless link to the on-board PC, which sends direct commands to the wheels controllers. If the robot control module would run on the same PC as the vision module, it might happen that it would not be able to send the new velocities out in time due to the vision module blocking the CPU.

The communication between the three computers is done via ROS [34], which is a widely used communication framework that, among other features, allows easily configuring multiple computers for cooperation in order to improve the time effectiveness of the system. This time effectiveness is achieved by splitting computationally expensive tasks into modules, which run on different computers.

6.2 Vision-based human detection and tracking

The image processing system is similar to the one presented in Fig. 5.2 and is shown in Fig. 6.2. The main difference is the ground plane removal applied on the disparity map before segmentation. This is necessary in order to enable the disparity map segmentation module to reliably work both indoors and outdoors, independently of whether the ground has a structured surface or not. Also, the Bumblebee XB3 camera already provides stereo rectified images in contrast to the high-speed stereo camera system presented in Chapter 5, so stereo rectification is not required. In contrast to the pedestrian detection application, no high-speed cameras are required for the robotic follower application, as both the human speed and the robot speed are low.

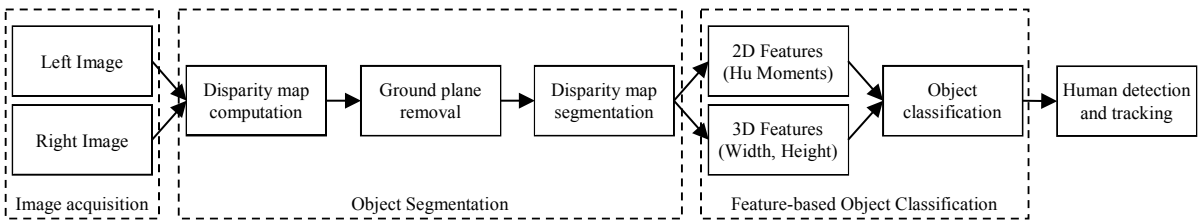
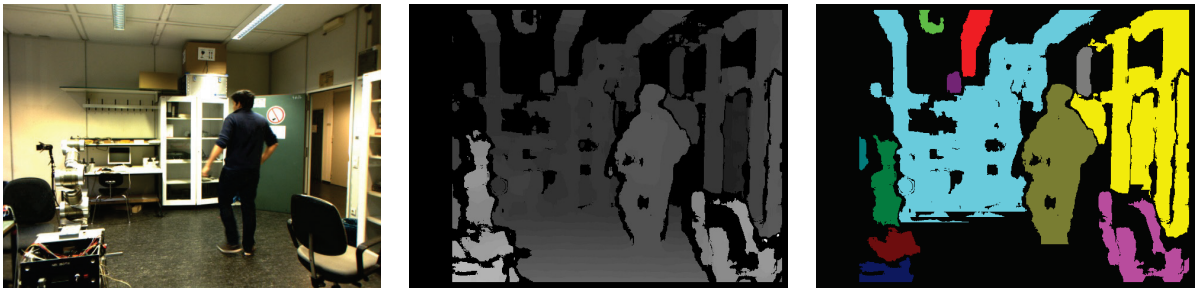


Fig. 6.2 Block diagram of the stereo-vision based human detection for robotic follower

Similarly to the pedestrian detection application, a disparity map is obtained from the stereo image pair, which is afterwards segmented using the connected pixel labeling based method presented in section 3.2. The segmentation result for the

considered human tracking robot scenario is shown in Fig. 6.3 c). Differently colored regions in the image in Fig. 6.3 c) represent different objects which are at different distances to the robot's camera. As it can be seen the ground has been removed from the segmented image (represented by black color) in order to avoid merging with other objects, including the human, which are placed on the ground.



a) b) c)

Fig. 6.3 Example of human walking in front of the robot: a) Original image, b) Disparity map and c) Segmented disparity map

The ground plane removal was achieved by detecting the regions in the lower part of the disparity map whose disparity values gradually change from dark to bright as the v coordinate of the pixel gets larger. This gradual change occurs for pixels belonging to the ground, as pixels having a smaller v coordinate in the disparity map are further away from the camera (are darker) compared to pixels having a larger v coordinate, which are closer to the camera (are brighter). For the ground removal a window of 11 rows and 1 column (11 x 1) was used to assess this vertical gradual change of disparity values for the central pixel of the window. In contrast to the ground plane pixels in the disparity map, pixels belonging to the regions of object surfaces have almost constant disparity values, so they are not removed.

After segmenting the objects in the disparity map, 2D and 3D features are extracted, which enable the classifier to distinguish humans from other objects in the robot's perceived environment. Similarly as in the pedestrian detection application the Hu Moments are extracted together with the objects' 3D width W and height H according to (2.7) and (3.12).

For classification the same classifier as in Chapter 5 was used, namely a Back-propagation Neural Network with one hidden layer [36]. A set of 577 feature vectors (H_1, H_2, H_3, H, W) were used for training the classifier. These feature vectors were extracted from segmented human regions in the disparity maps of stereo image pairs acquired indoor as well as outdoor. Another 423 feature vectors were used for testing the developed classifier. These test features were obtained by extraction from segmented regions of different objects including humans in disparity maps of stereo

image pairs acquired indoor as well as outdoor. The obtained classification result on the training set was very good, as indicated by the fact that the classification performance rate was 96%. Misclassification happened in cases of significant human occlusion or segmentation of the human as connected to objects from its environment.

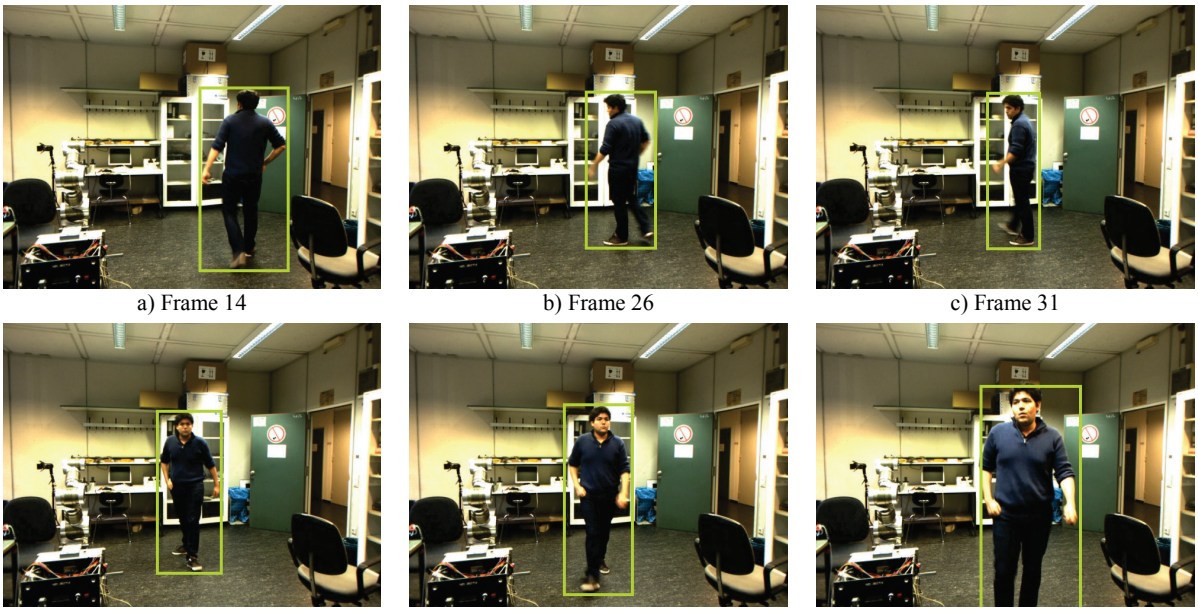
After successfully detecting the human, the 2D coordinates of the human's center of mass with respect to the camera are computed as average of all image coordinates of segmented pixels belonging to the human. After applying (2.18) to the 2D center of mass, the 3D coordinates of the center of mass are obtained. Additionally, the 3D coordinates of the corner points of the human bounding box were computed as explained in section 2.6, which are further used by the tracking module.

The tracking module is based on the modified Kalman filter introduced in [1], which recursively predicts the estimates of the 3D coordinates of the corner points. For this purpose, the modified Kalman filter receives the 3D coordinates of the corner points obtained from the current frame. The filter then estimates the actual 3D coordinates, considering that the measurements were affected by noise. Additionally, the filter outputs the predicted 3D coordinates of the corner points for the next frame, considering that the human velocity is constant between two frames. Afterwards the predicted 3D corner points are projected back on the 2D image plane using (2.5), which enables the prediction of the position of the region of interest (ROI) in the robot's camera images. The predicted position of the ROI enables human detection to be performed on the image region of interest rather than on the whole image, contributing to the cost effectiveness of the proposed human tracking algorithm.

6.3 Performance evaluation

The performance of the presented stereo-vision system for human tracking was tested within the working scenario of a mobile robot intended to follow a human co-worker in indoor applications as well as in outdoor applications.

Experiments were conducted where a human walking in front of the mobile robot was imaged by a Point Grey Bumblebee XB3 [79] stereo camera mounted on the robot. The image pairs were grabbed at full resolution (1280x960 pixels) at 12 fps (frames per second). Each pair of stereo frames was processed in order to extract the 2D coordinates of the left upper corner and the bottom right corner of the bounding box with respect to the camera coordinate system, as explained in section 2.6. Examples of processed images with superimposed extracted bounding boxes of humans are shown in Fig. 6.4 and Fig. 6.5.

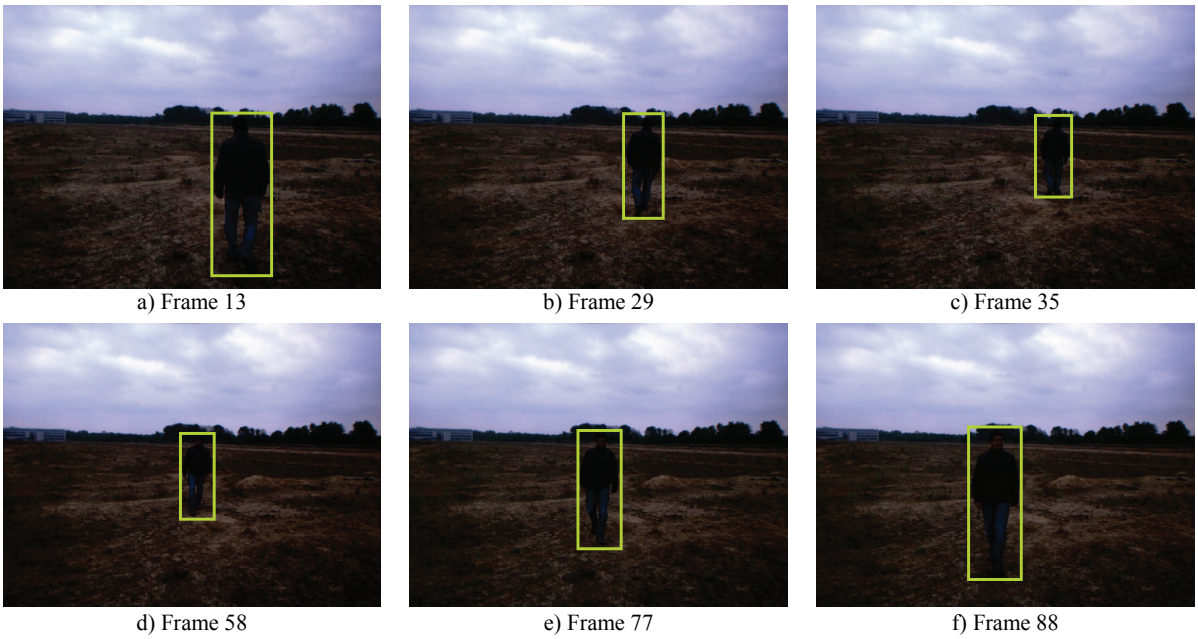


a) Frame 14 b) Frame 26 c) Frame 31

d) Frame 43 e) Frame 53 f) Frame 65

Fig. 6.4 Human detection in sample frames of the videos captured indoor

The result of human detection in 6 frames from a video captured indoor is shown in Fig. 6.4 a) – f). Fig. 6.5 a) – f) shows the result of human detection in 6 frames from a video captured outdoor.

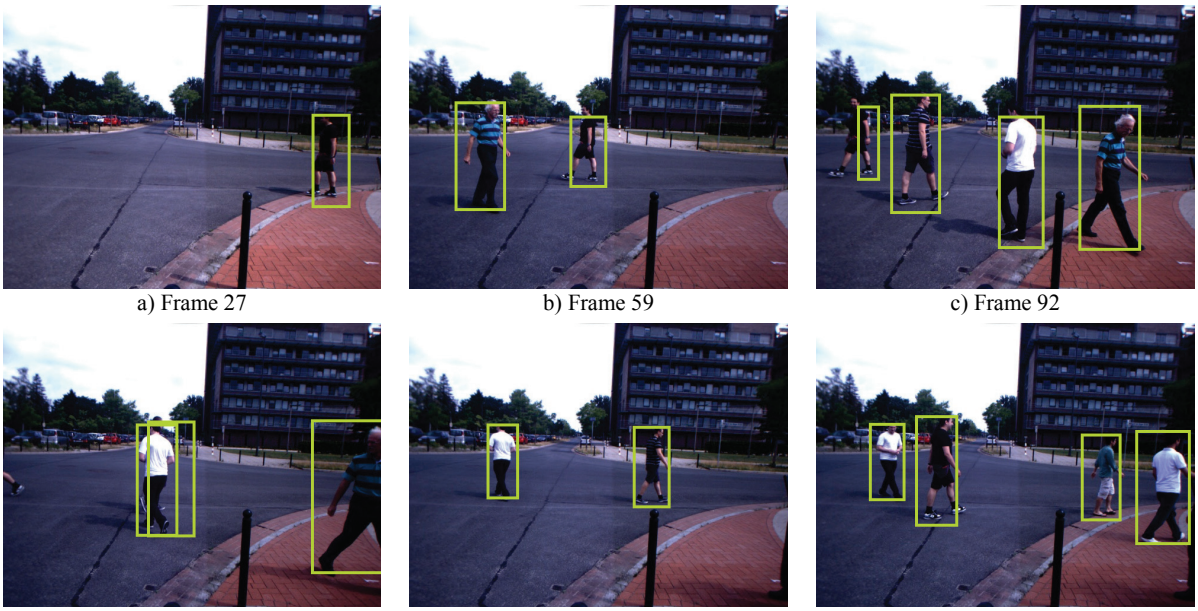


a) Frame 13 b) Frame 29 c) Frame 35

d) Frame 58 e) Frame 77 f) Frame 88

Fig. 6.5 Human detection in sample frames of the videos captured outdoor

In the presented robotic follower application only a single person is intended to be detected and tracked by the vision system. However, a scene with more persons was recorded in order to evaluate the performance of the vision module to track multiple persons simultaneously. Fig. 6.6 shows frames of a video captured outdoor, in which multiple persons enter and exit the scene.



a) Frame 27 b) Frame 59 c) Frame 92

d) Frame 104 e) Frame 132 f) Frame 203

Fig. 6.6 Human detection in sample frames of the videos captured outdoor, containing multiple persons

It can be seen in Fig. 6.6 d) that even though one person occludes another, the tracker is still able to keep following the occluded person, which afterwards becomes visible again, as can be seen in Fig. 6.6 e).

The presented application assumes only one human co-worker in the robot's environment. However, the presented method is applicable also if there is more than one human in the robot's environment. In that case, the human closest to the robot is selected during initialization as the person to be followed and is continuously tracked. In this way, even if other humans enter the scene, the robot is able to continue tracking the person selected for following in the initialization phase. In order to fulfill this task, the tracking system works under the assumption that the 3D position of the tracked person cannot suddenly change between two consecutive frames and therefore it is able to correctly locate the selected person in successive frames. This holds as long as the humans in the scene can be clearly distinguished from each other.

However even if one of the humans temporarily occludes the human selected to be tracked and the person is not in the camera's view anymore, the modified Kalman tracking filter still continues to estimate the person's position as explained in [1].

The performance of the presented human detection and tracking algorithm has been evaluated by comparing it to the Hog-Processing library [85], which is a Java implementation of the pedestrian detection based on the so-called Histogram of Oriented Gradients (HOG) presented in [66]. Human detection based on HOG is widely used for applications where monocular cameras are used. Therefore, the left image of the presented stereo camera system was used as input for the Hog-Processing

library. The Hog-Processing library offers two methods for human detection. The first method divides the image into blocks of fixed size that partially overlap (with overlapping blocks), while the second method considers that these blocks do not overlap (no overlapping blocks).

Table 6.1 illustrates the results obtained by both of these methods and manually obtained ground truth data for the videos shown in Fig. 6.4 (indoor), Fig. 6.5 (outdoor single person) and Fig. 6.6 (outdoor multiple persons). The third video contains many frames in which up to four persons are visible at the same time. In some of the frames the persons partially or fully occlude each other, as for example in frame 104 shown in Fig. 6.6 d). For comparison purposes, the raw output of the classifier was evaluated as well, besides the tracker output, as the Hog-Processing library only classifies and does not track humans.

Table 6.1 Comparison of the novel human detection and tracking algorithm and the HOG-based pedestrian detection

Operation	Indoor single person (200 frames)		Outdoor single person (400 frames)		Outdoor multiple persons (300 frames)	
	Detected humans	False positives	Detected humans	False positives	Detected humans	False positives
Ground truth	76 (100%)	–	238 (100%)	–	805 (100%)	–
Proposed classifier	69 (91%)	3 (2%)	175 (73%)	0 (0%)	549 (68%)	1 (0%)
Proposed tracker	71 (93%)	0 (0%)	181 (76%)	0 (0%)	614 (76%)	0 (0%)
HOG w. overl. bl.	45 (59%)	34 (17%)	102 (43%)	1 (0%)	283 (35%)	33 (11%)
HOG no overl. bl.	63 (83%)	26 (13%)	165 (69%)	0 (0%)	267 (33%)	16 (5%)

It can be seen that the proposed classifier performs better than both methods offered by the Hog-Processing library. This can be explained by the additional information obtained from the disparity map, which makes it considerably easier to separate humans from the rest of the image. The result is improved even more by using the presented Kalman-based tracker, compared to the raw output of the classifier. The tracker is particularly helpful in the video that contains multiple persons, as it helps to follow persons that partially or completely occlude each other and which the classifier could not detect. Another benefit of using the tracker compared to using only the classifier is the false positive rejection, as the location of false positives is mostly random, so they are eliminated as their position is not consistent among consecutive frames.

In order to evaluate the performance of the system with respect to the accuracy of reconstruction of 3D coordinates of the person and so of the reconstruction of the robot's distance to the person, the experimental results were compared with ground truth obtained in two ways. In the first experiment the ground truth distance to the person was obtained by a Bosch PLR 50 digital laser rangefinder [81]. In the second

experiment a reference path to be followed by the person was drawn on the floor. During these experiments the robot was just observing the person without following, so that the errors possibly occurring in vision could be decoupled from possible errors introduced by robot control. In the first experiment, the distance from the output of the proposed stereo-vision based tracker is computed according to:

$$D = \sqrt{x_{HC}^2 + y_{HC}^2 + z_{HC}^2} \tag{6.1}$$

where x_{HC}, y_{HC} and z_{HC} are the three coordinates of the human's center of mass.

The computed distance was compared to the distance obtained by the digital laser rangefinder. According to the datasheet, the accuracy of this device is ± 2 mm regardless of distance. The system was tested on 1055 frames and the average error in distance was 2.38% at a standard deviation of 2.12 %.

Fig. 6.7 shows the result of the second experiment where the reference path in form of a square of 2.3x2.3m is represented by the red line, while the output of the proposed stereo-vision based human detection without and with tracking filter (Kalman estimation and prediction) are represented by the blue and green line respectively.

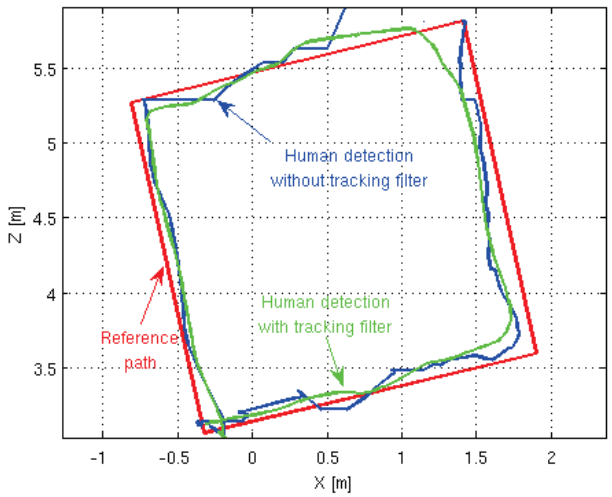


Fig. 6.7 Comparison of the reference human's path and the human's path reconstructed with the stereo-vision based tracker with and without tracking filter

It can be observed that there is a region in the reconstructed path in the upper right corner where the proposed human detection gives no results. However, the Kalman filter estimates and predicts the 3D position of the human in these frames, so that at any moment the proposed tracker outputs valid data. This is very important for proper robot control to avoid random movement of the robot. Also, the proposed tracker makes the reconstructed path smoother, which helps robot control to avoid erratic movements of the robot platform.

Timing considerations

Besides providing reliable information for robot control, Kalman prediction enables human detection to be performed on the image region of interest contributing to the cost effectiveness of human tracking. The processing time saved is between 30% and 70% of the time required to process the whole frame, depending on the size of the predicted region.

Besides the cost effective vision-based method, real-time robot control is supported with distributed computing. As mentioned before, the presented system acquires image pairs from the on-board stereo camera at a rate of 12 Hz. If image acquisition, the vision module and the robot control would all run on the on-board PC, due to the limited computational capabilities of the on-board low-power PC the vision module could run only at a rate of about 4 Hz and could interfere with robot control by constantly blocking the CPU with image processing tasks. In order to avoid CPU blocking, the operations would need to run sequentially as shown in Fig. 6.8 a).

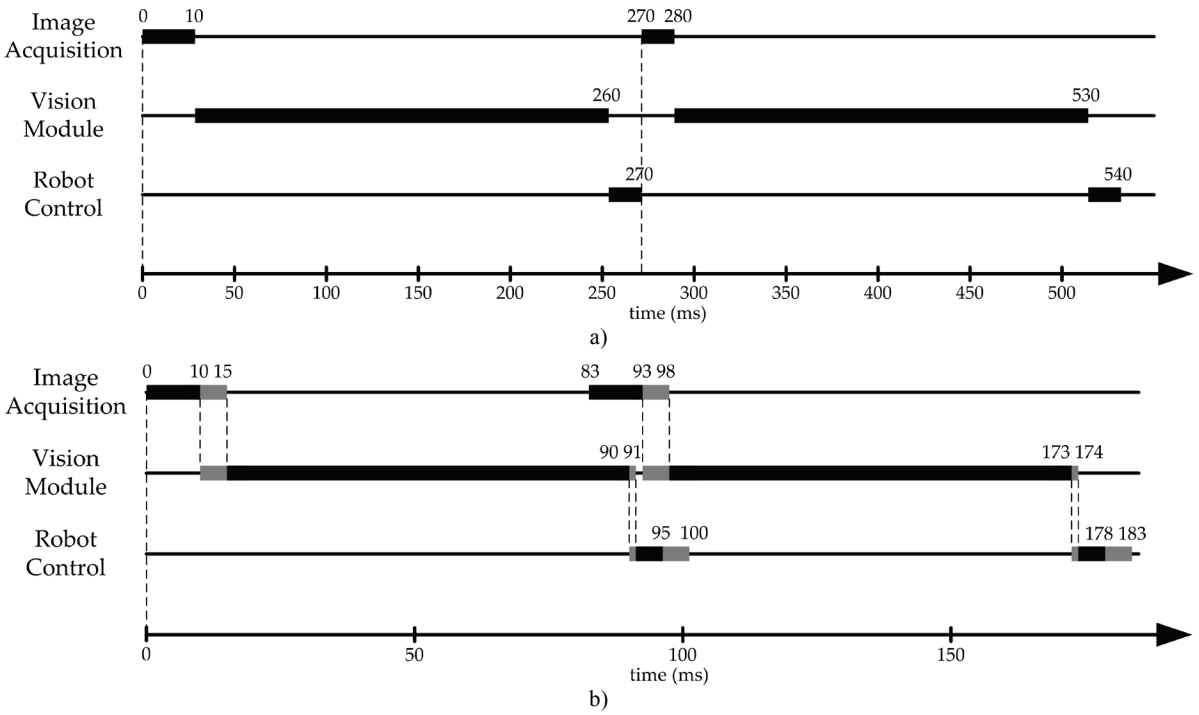


Fig. 6.8 Timing diagram in case: a) all operations run sequentially on one, low power computer and b) on different computers

It can be observed that the total processing time of the system from image acquisition to sending speed values to the wheels is the sum of the three operations:

$$T_P = T_1 + T_2 + T_3 \tag{6.2}$$

where T_1, T_2 and T_3 represent the time periods required for each of the three operations to finish and they are 10ms, 250ms and 10ms respectively. Hence, if the operations

would run sequentially the output rate of the robot control would be limited to less than 4 Hz, which could cause abrupt movements of the robot.

In order to overcome these problems distributed computing was introduced in the presented system. This allowed using of a separate high-end PC for running the vision module and a desktop PC for running the control module. In this way each individual module fully uses all available cores of the CPU without the necessity of leaving free resources to concurrent modules, as they run on separate machines. Introducing such distributed computing resulted in reducing the processing time for both the vision module from 250ms to 75ms and the robot control module from 10ms to 4ms, as can be seen in Fig. 6.8 b). The processing time for image acquisition remains the same, 10ms, as according to the given system specifications the same low power on-board PC has to be used for acquiring the images.

In the presented distributed computing system the three modules use the ROS library for communication, which implements socket communication over TCP (Transmission Control Protocol) in a way that is transparent to the user and therefore easy to use, debug and deploy.

Besides introducing distributed computing, the real-time system characteristics are supported by pipelining as the use of distributed computing allows the operations to be performed in a pipeline fashion, which means that a module which is upstream can already process new data while the downstream module processes the output of the upstream module.

However, by using multiple computers additional transmission delays are introduced in the system, which are in average about 1 ms for cabled connections (T_{23}) and 5 ms for wireless connections (T_{12} and T_{31}). Fig. 6.8 b) shows an overview of the timing when distributed computing is used. Black bars illustrate times needed for processing of image acquisition, vision and robot control, while the gray colored bars illustrate transmission delays. As the operations now run in parallel, the total processing time of a frame is given by the slowest operation, to which the communication times from and to that module are added.

$$T_P = MAX(T_1, T_2, T_3) + T_{To} + T_{From} \tag{6.3}$$

where T_{To} and T_{From} represent the transmission time required to send data from the previous module to the slowest module and from the slowest module to the next module in line. In Fig. 6.8 the slowest module is the vision module, to which the transport times from image acquisition (T_{12}) and to robot control (T_{23}) are added. In

pipelined systems there is another measurement, the system latency, which describes the time required for the final output to be delivered for a specific input. This is the sum of all operations, including all communication times.

$$T_L = T_1 + T_2 + T_3 + T_{12} + T_{23} + T_{31} \tag{6.4}$$

In other words, after T_L has passed and the pipeline is full, every other sample comes after a period T. Therefore, the pipelines are especially useful when all operations take approximately the same amount of time to be completed, since no module needs to waste PC resources by just waiting for the next input.

It can be seen that while in the case of sequential processing, according to (6.1), the total processing time and implicitly the latency would be 270ms, reaching 3.7 Hz, after using distributed computing the processing time is 81 ms leading to a potential rate of 12.34 Hz and a latency of 100 ms. Therefore the 12 Hz rate of the stereo camera can be maintained by all modules in the proposed system. Even though 12 Hz is a good rate for obtaining reference values for robot control, in order to ensure smooth movement of the robot, in the proposed architecture, the robot control additionally interpolates between two samples obtained from the vision module.

The presented application illustrates a typical scenario of human-robot interaction. Due to the distributed computing architecture, the proposed vision-based human tracking system is able process the images captured by the stereo camera system in real-time and to reliably track the human co-worker. This in turn enables the presented robotic platform to follow the human co-worker at a given distance. Additionally, the robot can sense when the human is approaching to deposit collected samples, so it knows that it should stop and wait until the human should be again followed.

7

Conclusion and Outlook

In this thesis novel approaches for vision-based real-time human detection and tracking were proposed and their usefulness has been demonstrated based on three different applications. Additionally, novel hardware-specific optimizations of the image processing algorithms were presented, which enable the proposed computer vision algorithms to run in real-time.

As image segmentation plays a crucial role in many vision-based applications, novel robust segmentation methods for detection and tracking of objects, including humans as particular type of objects, were presented in Chapter 3. The usefulness of these algorithms was demonstrated for three different applications of vision-based human detection and tracking: clinical gait analysis, pedestrian detection and human-robot interaction.

In Chapter 4 it has been shown that robust markerless vision-based clinical gait analysis is achieved by using the novel closed-loop segmentation of background subtracted images introduced in the first part of Chapter 3, as it offers optimal input for gait feature extraction. The extracted gait features can be used to support medical experts in diagnosing gait related pathologies as well as in following the process of rehabilitation by offering an objective tool for analyzing and comparing relevant gait parameters of the patients. The usefulness of the system is underlined by the fact that it

only requires about 30ms to process one frame, making it suitable for clinical environments, as it provides gait analysis results shortly after the person's gait was recorded. It also has been shown in Chapter 4 that new technologies such as RGBD cameras are expected to improve in future the markerless gait analysis by simplifying the human detection and tracking process. However, one current shortcoming is that even though RGBD cameras enable reliable detection and tracking of humans, existing software libraries for RGBD image processing cannot reliably provide the specific gait features that are necessary for clinical gait analysis. Therefore, in the last section of Chapter 4 it has been shown that the required features can be extracted from images provided by the RGBD cameras using the gait feature extraction algorithms that were proposed in this thesis.

Even though active sensors that are based on structured light like the RGBD cameras bear a huge potential for human tracking applications, they have limitations in applications where direct sunlight is encountered, such as in outdoor scenes. For this reason, in this thesis human detection and tracking algorithms were proposed that were designed to work both indoors and outdoors. This was achieved by using stereo cameras, which are passive sensors that do not need special illumination. In this context a novel disparity map segmentation algorithm was introduced in the second part of Chapter 3, which enables separating objects from each other and from the background. This algorithm was then further used for human tracking in images of stereo cameras as humans are considered to be a particular type of objects. The proposed disparity map segmentation algorithm has been shown to be useful for human tracking for pedestrian detection, for human-robot interaction as well as for 3D object reconstruction for service robotics.

In Chapter 5 a stereo-vision based pedestrian detection application was presented. The pedestrian detection operates at 100Hz on GPU and FPGA, being therefore suitable for vehicle velocities above the 50km/h limit as currently supported by state-of-the-art systems. In order to enable the developed algorithms to achieve real-time capabilities, novel hardware-specific implementations were introduced. These implementations allow either different image regions to be processed in parallel using the same algorithm or complete images to be processed in parallel by different algorithms in a pipeline fashion. For this purpose CPU and GPU implementations have been presented and compared to FPGA implementations.

Human tracking for Human-Machine Interaction was presented in Chapter 6 on the example of a robotic co-worker that should detect and follow a human. The real-time capability of this system was achieved using distributed computing based on ROS

(Robot Operating System), which is a communication framework enabling the distribution of software modules on multiple computers. It has been shown that the proposed human tracking algorithm performs well in tracking single persons or even multiple persons, so that the robot can reliably follow the human co-worker at a given distance.

It is expected that vision-based real-time human detection and tracking will continue to be an important topic in image processing and computer vision. Therefore the algorithms presented in this thesis could build the basis for new methods that could be used in future applications, especially as new devices are being developed, in order to simplify the interaction between humans and machines.

Bibliography – own publications

[1] Petrović E, **Leu A**, Ristić-Durrant D and Nikolić V (2013) Stereo-Vision Based Human Tracking for Robotic Follower. In: International Journal of Advanced Robotic Systems

[2] **Leu A**, Bacără D, Aiteanu D and Gräser A (2012) Hardware acceleration of image processing algorithms for collision warning in automotive applications. In: Methods and Applications in Automation: 32nd – 33rd Colloquium of Automation, Salzhausen/Leer, Germany, pp.1-12, Shaker Verlag GmbH

[3] **Leu A**, Aiteanu D, Gräser A (2012) High Speed Stereo Vision Based Automotive Collision Warning System. In: Applied Computational Intelligence in Engineering and Information Technology, Volume 1, pp. 187-199, Springer Verlag Heidelberg

[4] **Leu A**, Aiteanu D, Gräser A (2011) A novel stereo camera based collision warning system for automotive applications. In: Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI), May 19-21, 2011, Timişoara, România, pp. 409-414

[5] **Leu A**, Ristić-Durrant D, Gräser A (2011) A robust markerless vision-based human gait analysis system. In: Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI), May 19-21, 2011, Timişoara, România, pp. 415-420

[6] Natarajan SK, Ristić-Durrant D, **Leu A**, Gräser A (2011) Robust stereo-vision based 3D modeling of real-world objects for assistive robotic applications. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sept. 25-30, 2011, San Francisco, California, pp. 786-792

[7] Slavnic S, **Leu A**, Ristić-Durrant D and Gräser A (2010) Concept of a mobile robot-assisted gait rehabilitation system – Simulation study. In: Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 18-22, 2010, Taipei, China, pp. 6022-6027

[8] **Leu A**, Ristić-Durrant D and Gräser A (2010) A robust system for markerless vision-based human gait analysis for gait rehabilitation. In: Methods and Applications in Automation: 30th – 31st Colloquium of Automation, Salzhausen/Leer, Germany, pp. 9-20, Shaker Verlag GmbH

[9] **Leu A**, Bacără D and Jiveț I (2010) Disparity Map Computation Speed Comparison for CPU, GPU and FPGA Implementations. In: Buletinul Științific al Universității "Politehnica" din Timișoara

Bibliography – references

Books

[10] Hartley R, Zisserman A (2003) Multiple View Geometry in computer vision. Cambridge University Press

[11] Maggio E and Cavallaro E (2011) Video Tracking – Theory and Practice. John Wiley & Sons, Ltd.

[12] Shapiro LG and Stockman GC (2001) Computer Vision. Prentice-Hall

[13] Szeliski R (2011) Computer Vision: Algorithms and Applications. Springer Verlag London

[14] Winter DA (2005) Biomechanics and Motor Control of Human Movement. John Wiley & Sons

[15] Perry J (1992) Gait Analysis: Normal and Pathological Function. SLACK Incorporated

[16] Nixon M and Aguado A (2008) Feature Extraction and Image Processing, Elsevier Ltd.

[17] Bishop CM (2006) Pattern Recognition and Machine Learning. Springer Science

[18] Kanupriya G and Sunil PK (2010) Hardware Acceleration of EDA Algorithms – Custom ICs, FPGAs and GPUs. Springer Science

[19] DeLisa JA (1998) Gait analysis in the science of rehabilitation. Veterans Health Administration

[20] Gage JR, Schwartz MH, Koop SE and Novacheck TF (2009) The Identification and Treatment of Gait Problems in Cerebral Palsy. Mac Keith Press

[21] Cockshott WP, Renfrew K (2004) SIMD programming manual for Linux and Windows. Springer London

[22] Hernandez PC (2006) Dual Bayesian and Morphology-based Approach for Markerless Human Motion Capture in Natural Interaction Environments. PhD Thesis

Journal Papers and Book Chapters

[23] He L, Chao Y, and Suzuki K (2008) A Run-Based Two-Scan Labeling Algorithm. In: IEEE Transactions on Image Processing, 7(5):749-756

[24] Sisto SS (1998) An overview of the value of information resulting from instrumented gait analysis for the physical therapist. In: DeLisa JA, Gait analysis in the science of rehabilitation, Veterans Health Administration, pp.76-84

[25] Baker R (2006) Gait analysis methods in rehabilitation. In: Journal of NeuroEngineering and Rehabilitation, 3:1-10

[26] Zhou H and Hu H (2008) Human motion tracking for rehabilitation – A survey. In: Biomedical Signal Processing and Control, 3:1–18

[27] Yániz C, Rocha J and Perales F (1998) 3D Part Recognition Method for Human Motion Analysis. In: Modelling and Motion Capture Techniques for Virtual Environments. Lecture Notes in Computer Science vol. 1537, pp 41-54

[28] Sezgin M and Sankur B (2004) Survey over image thresholding techniques and quantitative performance evaluation. In: Journal of Electronic Imaging 13(1):146-165

[29] Ristić D and Gräser A (2006) Performance measure as feedback variable in image processing. In: EURASIP Journal on Applied Signal Processing

[30] Shiratsu A and Coury HJCG (2003) Reliability and accuracy of different sensors of a flexible electrogoniometer. In: Clinical Biomechanics, vol. 18, pp. 682-684

[31] Huang Y, Fu S and Thompson C (2005) Stereovision-Based Object Segmentation for Automotive Applications. EURASIP Journal on Applied Signal Processing 14(1):2322–2329

[32] Preparata FP and Hong SJ (1977) Convex Hulls of Finite Sets of Points in Two and Three Dimensions, In: Communications of the ACM, vol. 20, no. 2, pp. 87–93

[33] Scharstein D and Szeliski R (2001) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In: International Journal of Computer Vision, 47(1/2/3):7-42, April-June 2002. Microsoft Research Technical Report MSR-TR-2001-81, November 2001

[34] Cousins S (2010) Welcome to ROS Topics. In: IEEE Robotics & Automation Magazine, vol. 17(1), pp. 13-14

[35] Chen S (2011) Kalman Filter for Robot Vision: a Survey. In: IEEE Transactions on Industrial Electronics, vol. 59(11), pp. 4409-4420

[36] Zhang GP (2000) Neural networks for classification: a survey. In: IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol.30(4), pp.451-462

[37] Bredensteiner EJ and Bennett KP (1999) Multicategory Classification by Support Vector Machines. In: Computational Optimization and Applications vol. 12, pp.53-79, Springer US

[38] Chang CC and Lin CJ (2011) LIBSVM: A Library for Support Vector Machines. In: ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27

[39] Hodgson ME (1988) Reducing the Computational Requirements of the Minimum-Distance Classifier. In: Remote Sensing of Environment vol. 25(1), pp. 117-128

[40] Hofmann M and Gavrila DM (2012) Multi-view 3D Human Pose Estimation in Complex Environment. In: International Journal of Computer Vision, vol. 96(1), pp. 103-124

[41] Van der Mark W and Gavrila DM (2006) Real-Time Dense Stereo for Intelligent Vehicles. In: IEEE Transactions on Intelligent Transportation Systems, vol. 7(1), pp. 38-50

[42] Ke SR, Thuc HLU, Lee YJ, Hwang JN, Yoo JH and Choi KH (2013) A Review on Video-Based Human Activity Recognition. In: Computers (Open Access), 2(2):88-131

Conference Papers

[43] Tsai RY (1987) An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1986, Miami Beach, FL, pp. 364-374

[44] Tao T, Koo J C and Choi H R (2008) A Fast Block Matching Algorithm for Stereo Correspondence. In: Proceedings of IEEE Conference on Cybernetics and Intelligent Systems, Sept. 21-24, 2008, Chengdu, pp. 38-41

[45] Bouchrika I and Nixon MS (2006) Markerless Feature Extraction for Gait Analysis. In: Proceedings of the 5th Chapter Conference on Advances in Cybernetic Systems, September 2006, Sheffield, United Kingdom

[46] Yi Z and Liangzhong F (2010) Moving object detection based on running average background and temporal difference. In: Proceedings of International Conference on Intelligent Systems and Knowledge Engineering, Nov 15-16, 2010, Hangzhou, China, pp. 270-272

[47] Alexander GL, Havens TC, Skubic M, Rantz M, Keller JM, and Abbott C (2008) Markerless Human Motion Capture-Based Exercise Feedback System to Increase Efficacy and Safety of Elder Exercise Routines. In: Proceedings of the Intl. Conf. of the Intl. Society for Gerontechnology, June, 2008

[48] Lee H, Guan L and Burne JA (2000) Human gait and posture analysis for diagnosing neurological disorders. In: Proceedings of the International Conference on Image Processing, Sept. 10-13, 2000, Vancouver, Canada, vol. 2, pp. 435-438

[49] Su H and Huang FG (2005) Human gait recognition based on motion analysis. In: Proceedings of the 4th International Conference on Machine Learning and Cybernetics, Aug. 18-21, 2005, Guangzhou, China

[50] Hasler N, Rosenhahn B, Thormählen T, Wand M, Gall J and Seidel HP (2009) Markerless Motion Capture with Unsynchronized Moving Cameras. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, June 20-25, 2009, Miami Beach, FL, pp. 224-231

[51] Martin V, Maillot N and Thonnat M (2006) A learning Approach for Adaptive Image Segmentation. In: Proceedings of the 4th IEEE International Conference on Computer Vision System, Jan 4-7, 2006

[52] Di Leo G, Liguori C and Paolillo A (2010) Propagation of uncertainty through stereo triangulation. In: Proceedings of IEEE Instrumentation and Measurement Technology Conference (I2MTC), 3-6 May 2010, Austin, Texas, pp. 12-17

[53] Cui J, Liu F, Li Z and Jia Z (2010) Vehicle localisation using a single camera. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), June 21-24, 2010, San Diego, California, pp. 871-876

[54] Nedevschi S, Vatavu A, Oniga F and Meinecke MM (2008) Forward collision detection using a Stereo Vision System. In: Proceedings of the International Conference on Intelligent Computer Communication and Processing (ICCP), August 28-30, 2008, Cluj-Napoca, România, pp. 115-122

[55] Mitzel D, Horbert E, Ess A and Leibe B (2010) Multi-Person Tracking with Sparse Detection and Continuous Segmentation. In: Proceedings of IEEE European Conference on Computer Vision, Sept. 5-11, 2010, Heraklion, Greece, pp. 397-410

[56] Fuerstenberg KC, Dietmayer KCJ and Willhoeft V (2002) Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), June17-21, 2002, vol.1 pp. 31-35

[57] Ewald A and Willhoeft V (2000) Laser scanners for obstacle detection in automotive applications. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), October 3-5, 2000, Dearborn, Michigan, pp. 682-687

[58] Wenger J and Hahn S (2007) Long Range and Ultra-Wideband Short Range Automotive Radar. In: Proceedings of the IEEE International Conference on Ultra-Wideband (ICUWB), September 24-26, 2007, Singapore, pp. 518-522

[59] Teutsch M, Heger T, Schamm T and Zöllner JM (2010) 3D-Segmentation of Traffic Environments with U/V-Disparity supported by Radar-given Masterpoints. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), June 21-24, 2010, pp. 787 – 792

[60] Soquet N, Aubert D and Hautiere N (2007) Road Segmentation Supervised by an Extended V-Disparity Algorithm for Autonomous Navigation. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), June 13-15, 2007, pp. 160 – 165

[61] Labayrade R, Aubert D and Tarel J (2002) Real Time Obstacle Detection in Stereovision on Non Flat Road Geometry Through V-disparity Representation. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), June17-21, 2002, pp. 646-651

[62] Einramhof P and Vincze M (2010) Stereo-based Real-time Scene Segmentation for a Home Robot, In: Proceedings of the 52nd IEEE Symposium Electronics in Marine (ELMAR), Zadar, Croatia, Sept. 15-17, 2010, pp. 455-458

[63] Li Z, Wang K, Li L and Wang FY (2006) A Review on Vision-Based Pedestrian Detection for Intelligent Vehicles. In: Proceedings of the IEEE International Conference on Vehicular Electronics and Safety(ICVES), Beijing, China, December 13-25, 2006, pp. 57-62

[64] Viola P and Jones M (2001) Rapid Object Detection using a Boosted Cascade of Simple Features. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, December 8-14, 2001, pp. 511-518

[65] Toshev A, Taskar B and Daniilidis K. (2010) Object detection via boundary structure segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 13-18, 2010, pp. 950-957

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

[67] Liem MC and Gavrila DM (2013) A comparative study on multi-person tracking using overlapping cameras. In: Proceedings of the International Conference on Computer Vision Systems, St.Petersburg, Russia, pp. 203-212

[68] Zaletelj J, Perhavc J and Tasic JF (2007) Vision-based human-computer interface using hand gestures. In: Eight International Workshop on Image Analysis for Multimedia Interactive Services(WIAMIS), Santorini, Greece, June 6-8, 2007

[69] Berci N and Szolgay P (2007) Vision Based Human-Machine Interface via Hand Gestures. In:18th European Conference on Circuit Theory and Design, (ECCTD), Seville, Spain, August 27-30, 2007, pp. 496-499

[70] Attard L and Farrugia RA (2011) Vision Based Surveillance System. In: Proceedings of the IEEE International Conference on Computer as a Tool (EUROCON), Lisbon, Portugal, April 27-29, 2011, pp. 1-4

Web References

[71] Vicon product description [online] Available at: <http://www.vicon.com/products/> [Accessed January 2013]

[72] Homepage of Orthopädie-Technik Team GmbH, Bremen [online] Available at: <http://www.ot-team.de> [Accessed Jannuary 2013]

[73] Homepage of Neurologisches Rehabilitationszentrum (NRZ) Friedehorst, Bremen [online] Available at: <http://www.friedehorst.de/nrz/> [Accessed Jannuary 2013]

[74] Biometrics SG150 product description [online] Available at: <http://www.biometricsltd.com/gonio.htm> [Accessed September 2011]

[75] Mesa imaging Time-of-Flight camera product description [online] Available at: <http://www.mesa-imaging.ch/> [Accessed January 2013]

[76] Microsoft Kinect product description [online] Available at: <http://www.microsoft.com/en-us/kinectforwindows/> [Accessed January 2013]

[77] Asus Xtion PRO LIVE prduct description [online] Available at: <http://www.asus.com/Multimedia/Xtion_PRO_LIVE/> [Accessed May 2013]

[78] Open NI homepage [online] Available at: <http://www.openni.org/> [Accessed May 2013]

[79] Point Grey Bumblebee product description [online] Available at: <http://www.ptgrey.com/products/bbxb3> [Accessed January 2013]

[80] PCO 1200 s product description [online] Available at: <http://www.pco.de/high-speed-cameras/pco1200-s/> [Accessed January 2013]

[81] Bosch PLR 50 product description [online] Available at: <http://www.bosch-do-it.de/boptocs2-de/Heimwerker/Werkzeuge/DE/de/hw/Entfernungsmesser/95299/PLR+50/24116/3165140532518/index.htm>[Accessed January 2013]

[82] ADAC Braking Distances Study [online] Available at: <http://www.adac.de/_mmm/pdf/Verkehr_und_Mathe_ Anhalteweg_45164.pdf> [Accessed June 2012]

[83] Nvidia Tesla C1060 product description [online] Available at: <http://www.nvidia.com/object/product_tesla_c1060_us.html> [Accessed March 2011]

[84] Nvidia CUDA description [online] Available at: <http://www.nvidia.com/object/cuda_home_new.html> [Accessed August 2013]

[85] Hog - A processing Library for pedestrian detection [online] Available at: <http://hogprocessing.altervista.org/> [Accessed February 2013]

[86] Nexcom NISE 3500 P2 product description [online] Available at: <http://www.nexcom.com/Products/industrial-computing-solutions/industrial-fanless-computer/core-i-performance/fanless-pc-fanless-computer-nise-3500p2> [Accessed August 2013]

List of figures

List of tables

List of abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
ANN	Artificial Neural Network
API	Application Programming Interface
CPU	Central Processing Unit
DCM	Direction Cosine Matrix
FPGA	Field Programmable Gate Array
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HMI	Human-Machine Interface
HOG	Histogram of Oriented Gradients
HRI	Human-Robot Interaction
MP	Multi-Processor
NN	Neural Network
PC	Personal Computer
RANSAC	RANdom SAmple Consensus
RGBD	Red Green Blue Depth
ROI	Region of Interest
ROS	Robot Operating System
SAD	Sum of Absolute Differences
SIMD	Single Instruction Multiple Data
SSD	Sum of Squared Differences
SVD	Singular Value Decomposition
SVM	Support Vector Machines

Index