Aus dem Institut für Neuro- und Bioinformatik
der Universität zu Lübeck
Direktor: Prof. Dr. rer. nat. Thomas Martinetz

Gesture-Based Interaction with Time-of-Flight Cameras

Inauguraldissertation
zur Erlangung der Doktorwürde
der Universität zu Lübeck
– Aus der Technisch-Naturwissenschaftlichen Fakultät –

vorgelegt von
Martin Haker
aus Reinbek

Lübeck 2010

Erster Berichterstatter:	Prof. Dr.-Ing. Erhardt Barth
Zweiter Berichterstatter:	Prof. Dr.-Ing. Achim Schweikard
Tag der mündlichen Prüfung:	20. Dezember 2010

Zum Druck genehmigt. Lübeck, den 10. Januar 2011

Contents

Acknowledgements

There are a number of people who have contributed to this thesis, both in form and content as well on a moral level, and I want to express my gratitude for this support.

First of all, I want to thank Erhardt Barth. I have been under his supervision for many years and I feel that I have received a lot more than supervision; that is the earnest support on both a technical and the above mentioned moral level. Thank you!

At the Institute for Neuro- and Bioinformatics I am grateful to Thomas Martinetz for giving me the opportunity to pursue my PhD. I thank Martin Böhme for a fruitful collaboration on the work presented in this thesis. Special gratitude goes to Michael Dorr, who was always around when advice or a motivating chat was needed. I also want to thank all my other colleagues for providing such a healthy research environment – not only on a scientific level but also at the foosball table.

I also need thank the undergraduate students who worked with me over the years. Special thanks go to Michael Glodek and Tiberiu Viulet who significantly contributed to the results of this thesis.

Major parts of this thesis were developed within the scope of the European project ARTTS (www.artts.eu). I thank both the ARTTS consortium for the collaboration as well as European Commission for funding the project (contract no. IST-34107) within the Information Society Technologies (IST) priority of the 6th Framework Programme. This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Part I

Introduction and Time-of-Flight Cameras

1 Introduction

Nowadays, computers play a dominant role in various aspects of our lives and it is difficult, especially for younger generations, to imagine a world without them. In their early days, computers were mainly used to perform the tedious work of solving complex mathematical evaluations. Only universities and large companies were able to afford computers and computing power was expensive.

This changed with the advent of personal computers in the 1970's, such as the *Altair* and the *Apple II*. These small, yet powerful, machines for individual usage brought the technology into both offices and homes. In this period, computers were also introduced to the gaming market.

As computers were intended for a wider range of users, they required suitable forms of user interaction. While *punched cards* provided a means for computer specialists to feed simple programs into early computer models even up to the 1960's, the general user required a graphical user interface and intuitive input devices, such as mouse and keyboard.

In the years that followed, the computer industry experienced an enormous technological progress; the computing power could be increased significantly while both the cost of production and the size of the devices were reduced. This development lead to the emergence of computers in more and more different aspects of our lives. Hand-held devices allow us to access the internet wherever we are. Complex multimedia systems for viewing digital media, such as movies or photos, appear in our living rooms. And computer-aided systems provide support in automobiles, production facilities, and hospitals.

These new appliances of computer technology require novel forms of human-computer interaction; mouse and keyboard are not always the ideal means for providing user input. Here, a lot of progress has been achieved in the areas of speech recognition systems and touch screens in recent years. In the former case, however, it has proven difficult to devise systems that perform with sufficient reliability and robustness, i.e. apart from correctly recognizing speech it is difficult to decide whether an incoming signal is intended for the system or whether it belongs to an arbitrary conversation. So far, this limited the widespread use of speech recognition systems. Touch screens, on the other hand, have experienced a broad acceptance in information terminals and hand-held devices, such as Apple's iPhone. Especially, in the latter case one can observe a new trend in human-machine interaction – the use of gestures.

The advantage of touch sensitive surfaces in combination with a gesture recognition system is that a touch onto the surface can have more meaning than a simple pointing action to select a certain item on the screen. For example, the iPhone can recognize a swiping gesture to navigate between so-called views, which are used to present different content to the user, and two finger gestures can be used to rotate and resize images.

Taking the concept of gestures one step further is to recognize and interpret human gestures independently of a touch sensitive surface. This has two major advantages: (i) It comes closer to our natural use of gestures as a body language and (ii) it opens a wider range of applications where users do not need to be in physical contact with the input device.

The success of such systems has first been shown in the gaming market by Sony's *EyeToy*. In case of the EyeToy, a camera is placed on top of the video display, which monitors the user. Movement in front of the camera triggers certain actions and, thus, allows the gamer to control the system.

A similar, yet more realistic, approach was taken by Nintendo to integrate gesture recognition into their gaming console *Wii*. Hand-held wireless motion sensors capture the movement of the hands and, thus, provide natural motion patterns to the system, which can be used to perform virtual actions, such as the swing of a virtual tennis racket.

The next consequent step is to perform complex action and gesture recognition in an entirely contact-less fashion - for example with a camera based system. This goal seems to be within reach since the appearance of low-cost 3D sensors, such as the time-of-flight (TOF) camera. A TOF camera is a new type of image sensor which provides both an intensity image and a range map, i.e. it can measure the distance to the object in front of the camera at each pixel of the image at a high temporal resolution of 30 or more frames per second. TOF cameras operate by actively illuminating the scene (typically with light near the infrared spectrum) and measuring the time the light takes to travel to the object and back into the camera.

Although TOF cameras emerged on the market only recently, they have been successfully applied to a number of image processing applications, such as shape from shading (Böhme et al., 2008b), people tracking (Hansen et al., 2008), gesture recognition (Kollorz et al., 2008), and stereo vision (Gudmundsson et al., 2007a). Kolb et al. (2010) give an overview of publications related to TOF cameras.

The importance of this technology with respect to commercial gesture recognition systems is reflected by the interest of major computer companies. To give an example, in the fourth quarter of 2009 Microsoft bought one of the major TOF camera manufacturers in the scope of the *Project Natal*, which is intended to integrate gesture recognition into Microsoft's gaming platform *XBox 360*.

This thesis will focus on the use of computer vision with TOF cameras for human-computer interaction. Specifically, I will give an overview on TOF cameras and will present both algorithms and applications based on this type of image sensor.

While Part I of this thesis will be devoted to introducing the TOF technology, Part II will focus on a number of different computer vision algorithms for TOF data. I will present an approach to improving the TOF range measurement by enforcing the shading constraint, i.e. the observation that a range map implies a certain intensity image if the reflectance properties of the surface are known allows us to correct measurement errors in the TOF range map.

The range information provided by a TOF camera enables a very simple and robust identification of objects in front of the camera. I will present two approaches that are intended to identify a person appearing in the field of view the camera. The identification of a person yields a first step towards recognizing human gestures.

Given the range information, it is possible to represent the posture of the person in 3D by inverting the perspective projection of the camera. Assuming that we have identified all pixels depicting the person in the image, we thus obtain a 3D point cloud sampling the surface of the person that is visible to the camera. I will present an algorithm that fits a simple model of the human body into this point cloud. This allows the estimation of the body pose, i.e. we can estimate the location of body parts, such as the hands, in 3D.

To obtain further information about the imaged object, one can compute features describing the local geometry of the range map. I will introduce a specific type of geometric features referred to as generalized eccentricities, which can be used to distinguish between different surface types. Furthermore, I will present an extension of these features to obtain scale-invariance by computing them on the 3D surface of the object instead of on the image grid.

An alternative approach to computing features describing the image is the sparse coding principle. Sparse codes have previoulsy been used for the representation of conventional image data. Here, I extend the concept to multimodal data, i.e. a sparse code is computed simultaneously for range and intensity data, thus yielding perfectly registered basis functions for both types of data.

Another type of features often used in image processing aims at describing the motion pattern visible in a sequence of images. For conventional image data one usually estimates the 2D projection of the true motion, generally referred to as optical flow. In case depth information is available, one can compute the so-called range flow to estimate object motion in 3D. I review the computation of range flow and will demonstrate how this feature can be used to reliably detect and recognize hand gestures.

Part III of this thesis will focus on applications for TOF cameras based on the described algorithms. The first application will be a system that tracks the human nose and thus allows a user to control the mouse cursor with the movement of his head. This system can be used for an alternative form of text input.

A second application focuses on the use of deictic gestures to control a slide show presentation. The user may point at the screen to initiate the appearance of a virtual laser pointer. Hand gestures made towards the left and the right of the screen enable the navigation between slides.

A third application targets digital photography. Based on the range map it is possible to create the effect of depth of field in digital photos in a post-processing step. Depth of field is one of the most important stylistic devices in photography and allows the artist to let parts of the image appear sharp while others become blurred, thus guiding the attention of the spectator towards the intended content of the scene. Although this topic does not fall into the scope of gesture-based interaction, it aptly demonstrates the diversity of applications in which TOF cameras can be employed and has thus been added to this work.

Many of the results in this thesis were obtained in collaboration with others. In these cases, I will identify which contributions are my own at the beginning of the corresponding chapter. To reflect this, I will use the personal pronoun "we" and will do so throughout the thesis for consistency, even when the results were obtained solely by myself.

2 Time-of-Flight Cameras

2.1 Introduction

The TOF camera is a combined range and image sensor that delivers both a distance and an intensity measurement at every pixel of the sensor array. The camera works by emitting light from an active illumination unit and measuring the time the light travels from the camera to an object and back to the sensor. Typically, the illumination unit operates near the infrared spectrum so that the active illumination does not interfere with the human visual system, which is relatively insensitive to infrared light.

There exist two main technical approaches to measuring the time the light takes to travel from the camera to the object and return to the sensor. The first approach emits pulsed light and uses an electronic shutter to control the exposure time of the sensor with high accuracy (Iddan and Yahav, 2001; Yahav et al., 2007). Depending on the distance of an object from the camera the amount of light that reaches a pixel before the shutter closes varies, i.e. objects close to the camera appear brighter than objects at larger distances. In order to be independent of the reflectivity of the object, the camera takes a second image as a reference for which the exposure time is extended such that the entire light pulse can return to the camera independently of the distance of the object.

The second approach modulates the intensity of the active illumination by a periodic signal over time (Schwarte et al., 1995; Lange, 2000). The signal that returns to the camera is a shifted version of the emitted signal, where the phase shift between the two signals depends linearly on the distance of the object. The sensor, which is synchronized with the illumination unit, integrates over a number of subintervals of the periodic signal and the phase shift can be reconstructed from these measurements. The duration of the integration time is related to the exposure time of a conventional camera. This second approach will be discussed in more detail throughout the remainder of this chapter.

An important property of TOF cameras that operate with modulated illumination is that they have a so-called *non-ambiguity range*. Beyond this range distances cannot be reconstructed unambiguously. The non-ambiguity range depends on the *modulation frequency* of the periodic signal. If the modulated signal passes through more than an entire period of the signal during the time the light requires to reach the object and return to the camera the distance measurement becomes ambiguous.

In the case of pulsed light the problem of the non-ambiguity range can be avoided. With these cameras, one can specify a certain range interval in which the camera is to deliver range estimates. The length of the pulse and the timing of the shutter determine this range interval in which the amount of sensed light corresponds to the distance. A drawback is however, that this interval has to be specified beforehand, i.e. if objects in the scene are very close the camera and the shutter closes only after the entire light pulse has returned to the sensor from every object in the scene all objects appear at the same distance to the camera.

As mentioned above, this chapter will give an overview of the measurement principles and properties of TOF cameras with a special focus on cameras using the approach of modulated illumination. For a more detailed review of the topic refer to the work of Lange (2000).

2.2 State-of-the-art TOF Sensors

Over the years, a number of manufacturers offering TOF cameras have emerged on the market. Initially, the market was mainly dominated by CSEM/MESA Imaging, PMD Technologies, Canesta, and 3DV Systems for several years. Only recently, other companies started to develop their own sensor technology or to integrate existing TOF sensors into their own cameras.

A typical state-of-the-art TOF camera is the SR4000 (see Figure 2.1a), which was developed by the Centre Swisse d'Electronique et de Microtechnique (CSEM) and is manufactured and distributed by the Swiss company MESA Imaging. It is based on the phase measurement principle (Oggier et al., 2004, 2005b) and the sensor has a resolution of 176 by 144 pixels. A typical modulation frequency is 30 MHz. With this modulation frequency the camera achieves a non-ambiguity range of 5 meters. The accuracy of the range measurement is given as roughly 0.5% at a distance of 2 meters from the camera. This corresponds to an accuracy of 5 millimeters. This error is with respect to the variation of the range measurement between two successive frames, i.e. it refers to the repeatability of the measurement. The absolute error is specified at 10 millimeters over the calibrated range of the camera. The SR4000 also supports modulation frequencies of 14.5 MHz, 15 MHz, 15.5 MHz, 29 MHz, and 31 MHz. This yields non-ambiguity ranges from roughly 5 to 10 meters. The camera achieves frame rates up to 54 fps. The view angle is specified at 43.6^o.

The CamCube (see Figure 2.1b) by the German company PMD Technologies offers a slightly higher resolution of 204 by 204 pixels. The CamCube is also based on the phase measurement principle (Schwarte et al., 1995; Xu et al., 1998). Furthermore, it is fitted with a CS-mount, which allows the use of different optics. The camera comes with two illumination units mounted on both sides of the sensor. Each illumination unit is equipped with a fan, which is required for cooling. In contrast to this, the SR4000 is cooled passively.

The ZCam (see Figure 2.1c) is a low-cost TOF camera developed by the Israeli company 3DV Systems and is based on the pulsed light approach using an electronic shutter (Yahav et al., 2007). It delivers both a range and a color image at a resolution of 320 by 280 pixels. Despite its higher resolution, it delivers range images of poorer quality than both the SR4000 and the CamCube. 3DV Systems was bought by Microsoft in the scope of Project Natal in the third quarter of 2009. It can be speculated that Microsoft sees the value of the TOF technology for the gaming market.

Canesta is a TOF camera manufacturer from the United States. Their TOF sensor technology CanestaVision (see Figure 2.1d) is also based on the phase measurement principle (Gokturk et al., 2004).

Other manufacturers include the companies ifm electronic GmbH from Germany (see Figure 2.1e), IEE S.A. from Luxembourg, Optrima from Belgium, and TriDiCam from Germany. The first two companies also employ the phase measurement principle (Nieuwenhove, 2009; Devarakota et al., 2005) whereas the latter bases their camera on the pulsed light approach (König et al., 2007).

The world's currently smallest TOF camera was developed by CSEM in the scope of the EU project ARTTS (see Figure 2.1f). Similarly to the SR4000, it comes with a resolution of 176 by 144 pixels but measures only 38 by 38 by 35.4 millimeters. Furthermore, the camera supports modulation frequencies of up to 80 MHz and has a low power consumption, which allows the camera to draw its power directly from the USB port, i.e. it does not require an additional external power supply.

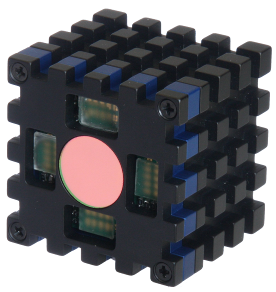
(a)

(b)

(c)

(d)

(e)



(f)

Figure 2.1: *Examples of current TOF cameras. Figure 2.1a shows the SR4000 by MESA Imaging. Figure 2.1b shows the CamCube by PMD Technologies. Figure 2.1c shows the ZCam by 3DV Systems. Figure 2.1d shows the DP200 development platform by Canesta. Figure 2.1e shows the efector PMD by ifm electronic. Figure 2.1f shows the ARTTS camera developed by CSEM.*

2.3 Alternative Optical Range Imaging Techniques

TOF cameras are the main focus of this work, however, there exist a number of alternative range imaging techniques. These techniques can be grouped into different categories with respect to the type of signal they detect, i.e. such systems can rely on microwaves, visible light, or ultrasonic waves. In the scope of this work, we will only focus on optical systems that operate by detecting electromagnetic radiation fields that lie near the spectrum of visible light. Furthermore, one can differentiate between systems that actively emit a signal into the scene or passively rely on an ambient signal present in the scene, such as daylight. To give a brief overview of existing technologies, we will now briefly discuss *triangulation*, *photometry*, *interferometry*, and *time-of-flight* methods.

2.3.1 Triangulation

Triangulation methods owe their name to the basic principle that the target point in the scene and two known points that are placed at a fixed baseline distance on the measurement device form a triangle. The range measurement is inferred from the angles of the triangle. A major restriction of triangulation methods is that the accuracy of the range measurement depends on the baseline distance, i.e. the accuracy is increased if the two known points are placed further apart which implies that there is a limit to the miniaturization of range imaging systems based on this approach. Generally, triangulation methods can be categorized into active and passive systems, and there exist different implementations for both categories:

Stereoscopy is an example of a passive system where the baseline is defined by two or more calibrated cameras that image the scene from different angles. The distance to a point in the scene is determined from the difference of the position in which the point appears in the individual images. This is achieved by establishing point-to-point correspondences in the individual images. Naturally, correct range estimates can only be computed if these point-to-point correspondences can be established, i.e. stereoscopy fails if the objects in the scene are not sufficiently textured. Apart from this restriction, the computational cost of the procedure must not be underestimated, even if the search space is restricted by taking the orientation of the baseline into account.

Laser scanners are a typical example of an active triangulation system. In such scanners an image sensor and a laser are placed on the baseline. Early scanners swept a laser beam across the scene which results in a dot on the image. Given the direction of the laser beam and the position in which the dot appears in the image, the distance to the target can be computed. More advanced scanners project a line (or a light sheet) onto the scene, which speeds up the acquisition process because the scene has to be scanned only in the direction orthogonal to the projected line. Nevertheless, laser scanners typically have a low temporal resolution in comparison to other range imaging techniques. On the other hand, they allow a robust range estimation with high accuracy.

Structured light refers to an extension of the laser scanning approach in the sense that a two-dimensional light pattern is projected onto the scene. Such patterns can e.g. be composed of parallel lines, random textures, or color-coded light. The appearance of the pattern is imaged using a camera and the shape of the scene can be reconstructed based on the distortions of the pattern. As laser scanners, the structured light approach is able to deliver range maps of high accuracy but usually at the cost of lower temporal resolution because often such systems use several different patterns to obtain a single range image.

2.3.2 Photometry

Photometric approaches aim at exploiting the knowledge of the image formation process to recover the imaged object surface. One approach is to invert the image formation process to recover the local surface orientation of the imaged objects. The shape of the object can then be reconstructed from these local estimates. The estimation of the local surface orientation is based on the fact that the intensity with which a surface point of an object appears in the image depends on the orientation of the surface normal towards the light source, the distance of the surface from the light source, and the reflectivity of the object. Under known lighting conditions and accurate assumptions on the reflectivity, the shape of an object can thus be reconstructed from its appearance in the image.

Shape from shading is a straightforward implementation of the above procedure that infers the shape of the object from a single image. Usually, shape from shading is restricted to a single point-light source and assumes a uniform reflectivity of the object. These strong constraints are generally fulfilled only under very well defined conditions, such as in microscopy, where shape from shading can yield highly accurate results.

Photometric stereo is a generalization of the shape from shading approach that uses several images of the scene taken from the same viewing angle but under different lighting conditions. This allows photometric stereo to infer the shape of objects with non-uniform reflectivity properties. However, the method is sensitive to shadows that are cast under the utilized lighting conditions.

Shape from focus is a third example of a photometric method. Here, the distance to the target is determined from the focal length of the optical system when the object appears sharp in the image. Intuitively speaking, an optical system with a large aperture has a small depth of field which in turn means that the optimal focal length in which the object appears sharp can be determined with higher accuracy. Similarly to stereoscopy, the shape from focus approach only works if the object is sufficiently textured. Otherwise, the blur caused by defocusing the image cannot be detected.

2.3.3 Interferometry

Interferometry is an active range imaging procedure where coherent monochromatic light is emitted towards the imaged object and the reflected signal is superimposed with a reference signal of the same light source. Due to the use of coherent monochromatic light both signals can be viewed as wavefronts of the same frequency. The distance of the object induces a phase shift between these two wavefronts, which results in constructive or destructive interference. Thus, the amplitude of the superimposed signal can be used to estimate the distance of the object.

The *Mitchelson interferometer* implements this approach using a beam splitter to split a laser beam of coherent monochromatic light into two beams. One beam is directed at the object whereas the other is targeted at a mirror at a fixed distance to generate the reference signal. Both reflected signals are superimposed by the beam splitter and projected onto an integrating detector. While this setup allows range measurements with an accuracy of fractions of the laser's wavelength, it is not possible to obtain absolute range estimates because the non-ambiguity range is only one half of a wavelength.

Multiple-wavelength interferometers overcome the limitation of low non-ambiguity ranges by using two laser beams of very similar wavelengths. This allows the generation of beat signals with high frequency which in turn yields absolute range measurements up to several tens of centimeters.

2.3.4 Time-of-Flight

The time-of-flight principle can be used for optical range measurement systems because we know the speed of light. A range estimate can thus be obtained by measuring the time the light requires travel from the imaging system to the object and back to the detector. This implies that time-of-flight systems fall into the category of active range imaging techniques.

Like TOF cameras, *laser scanners* can be used to implement this approach. A major disadvantage of laser scanners is, however, that they can only measure the distance to a single point in the scene at a time. Thus, they sweep the laser beam across the scene to obtain an entire range image, which reduces the temporal of resolution of the imager.

2.3.5 Summary

In contrast to the above mentioned procedures to measuring range, a number of advantages of the TOF sensor technology can be highlighted. These advantages can roughly be divided into three categories: the robustness of the measurement, the reliability of the hardware, and the cost of production.

As mentioned above, the most competitive alternative to TOF cameras are stereo systems. However, one has to note that stereo systems only work under very restrictive constraints on the scene, i.e. the surfaces of the measured objects have to provide sufficient texture for a robust estimate of the point-to-point correspondences. Here, TOF cameras are less dependent on the scene and their main limitations in this context are objects with specular surfaces. At the same time TOF cameras are widely independent of the ambient light conditions as they rely on their own active illumination.

Although TOF cameras cannot match the accuracy of laser scanners they offer the advantage that they use solid state sensor technology, i.e. the cameras are not composed of moving parts which makes them very reliable and reduces maintenance costs. Furthermore, they can achieve a higher temporal resolution as the range measurement is computed in hardware in each pixel for an entire image and instead of scanning the scene.

Finally, TOF cameras can be mass-produced at a very low cost because they are based on standard CMOS technology. Thus, the sensor itself lies in the same price range as regular 2D image sensors. Although, additional costs arise due to the illumination and control units, TOF systems are still cheaper than stereo cameras, which generally involve two high quality imagers. Here, they also have the advantage of being smaller in size.

2.4 Measurement Principle

The most common technical realization of current TOF cameras uses an illumination unit that emits light whose intensity is modulated by a periodic signal over time. The time the light requires to travel from the camera to the object and back to the sensor induces a delay between the emitted and received signal. Because of the periodicity of the signal, this delay corresponds to a phase shift, which encodes the distance of the object. The camera estimates this phase shift and thus determines the distance.

This measurement principle is illustrated in Figure 2.2. Here, the emitted signal $e(t)$ is an ideal sinusoidal function. The intensity of the active illumination is modulated according to such a periodic signal over time. The received signal $s(t)$ is composed of a shifted and attenuated version of $e(t)$ and an additive constant component. The phase shift between the signals $s(t)$ and $e(t)$ is denoted by φ. The distance to the object can be inferred from φ given the modulation frequency and the speed of light. The attenuation is due to the fact that only a small portion of the emitted light is actually reflected back towards the optical system of the camera while a large portion of the light is simply scattered into the scene. Hence, the amplitude A of the received signal is smaller than that of the emitted signal. The constant additive offset I_0 has its origin in ambient infrared light that is emitted by other light sources illuminating the scene, such as the sun. For convenience, we also introduce the DC component B of the received signal, which corresponds to $B = I_0 + A$. Thus the received signal takes the form

$$s(t) = B + A \cdot sin(\omega t - \varphi) \tag{2.1}$$

where ω denotes the angular frequency which relates to the modulation frequency f_{mod} by $\omega = 2\pi f_{\text{mod}}$.

As demonstrated by Lange (2000), the parameters ϕ, A, and B can be reconstructed by sampling the received signal $s(t)$ at four points over one modulation period $T_{\text{mod}} = \frac{1}{f_{\text{mod}}}$. This yields samples $A_0, \ldots, A_3$, where $A_k = s(\frac{k}{4} \cdot T_{\text{mod}}) = B + A \cdot \sin(k \cdot \frac{\pi}{2} - \varphi)$. Given these four samples, the parameters of the received

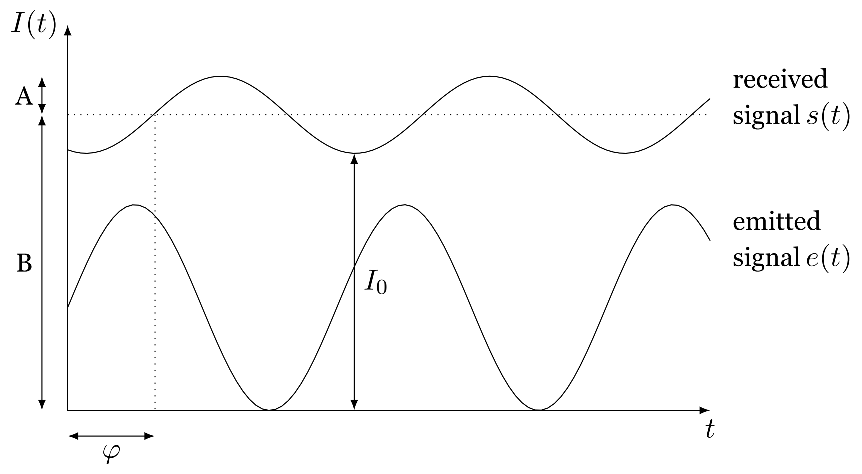


Figure 2.2: *Measurement principle of TOF cameras based on an intensity-modulated active illumination unit. The phase difference between the emitted signal $e(t)$ and the received signal $s(t)$ is denoted by φ, the amplitude of the received signal corresponds to A, the offset (or DC component) of received signal is given by B, and I_0 represents the ambient light present in the scene.*

signal can be reconstructed as follows

$$\text{Phase} \quad \varphi = \text{atan}\left(\frac{A_0 - A_2}{A_1 - A_3}\right) \tag{2.2}$$

$$\text{Amplitude} \quad A = \frac{\sqrt{(A_0 - A_2)^2 + (A_1 - A_3)^2}}{2} \tag{2.3}$$

$$\text{Offset} \quad B = \frac{A_0 + A_1 + A_2 + A_3}{4} \tag{2.4}$$

This concept is illustrated in Figure 2.3. Note that the sampling is done in synchrony with the modulation frequency of the illumination unit, i.e. the first sample is taken at the beginning of a new period.

Given the phase shift φ and the modulation frequency f_{mod} we can compute the distance of the object. To this end, we consider the time delay between the emitted and received signal which corresponds to $\frac{\varphi}{\omega}$ with $\omega = 2\pi f_{\text{mod}}$. During this time delay, the light travels a distance of $\frac{\varphi\, c}{\omega}$, where c denotes the speed of light. Because the light has to travel from the camera to the object and back to the sensor, the distance of the object is given by

$$R = \frac{\varphi\, c}{2\,\omega}. \tag{2.5}$$

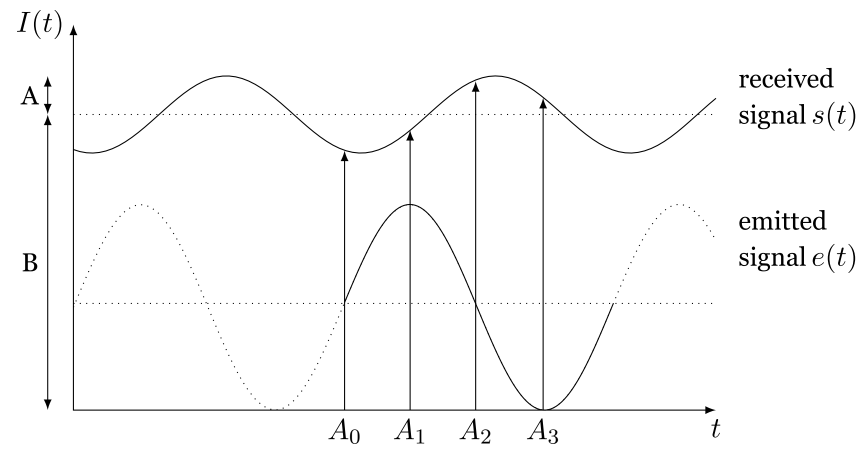


Figure 2.3: *The received signal $s(t)$ is sampled at four points over one modulation period of the emitted signal $e(t)$, i.e. the sampling is synchronized with the modulation of the emitted signal. This yields the samples $A_0, \ldots, A_3$ which enable the reconstruction of the parameters of the received signal.*

2.5 Technical Realization

In practice, an ideal sampling of the received signal cannot be realized. Instead, the signal has to be integrated over certain interleaving time intervals to approximate the samples $A_0, \ldots, A_3$. Before going into more detail on how this is done on the sensor, let us consider how the length of the integration interval Δt affects the reconstruction of the signal $s(t)$. Note that the integration theoretically equals a convolution of the signal $s(t)$ with a rect function followed by an ideal sampling under a signal processing point of view. Thus, it suffices here to investigate the effect of the convolution on the signal $s(t)$. Figure 2.4 visualizes this effect for three different lengths of the integration interval.

In the first case, an infinitesimal interval is used, which corresponds to the ideal sampling with a Dirac δ. When a signal is convolved with a Dirac in the time domain, each frequency component is multiplied with a constant one in the Fourier domain. Naturally, the frequency components of the sine function are not altered and the signal $s(t)$ can be reconstructed perfectly.

Extending the length of the integration interval Δt results in the convolution with a rect function $h(t) = \frac{1}{\Delta t}\text{rect}\left(\frac{t}{\Delta t}\right)$, which corresponds to a multiplication with the transfer function $H(f) = \text{sinc}(\pi f \Delta t)$ in the Fourier domain, where we define $\text{sinc}(x) = \frac{\sin(x)}{x}$. At the modulation frequency f_{mod}, the transfer function takes the

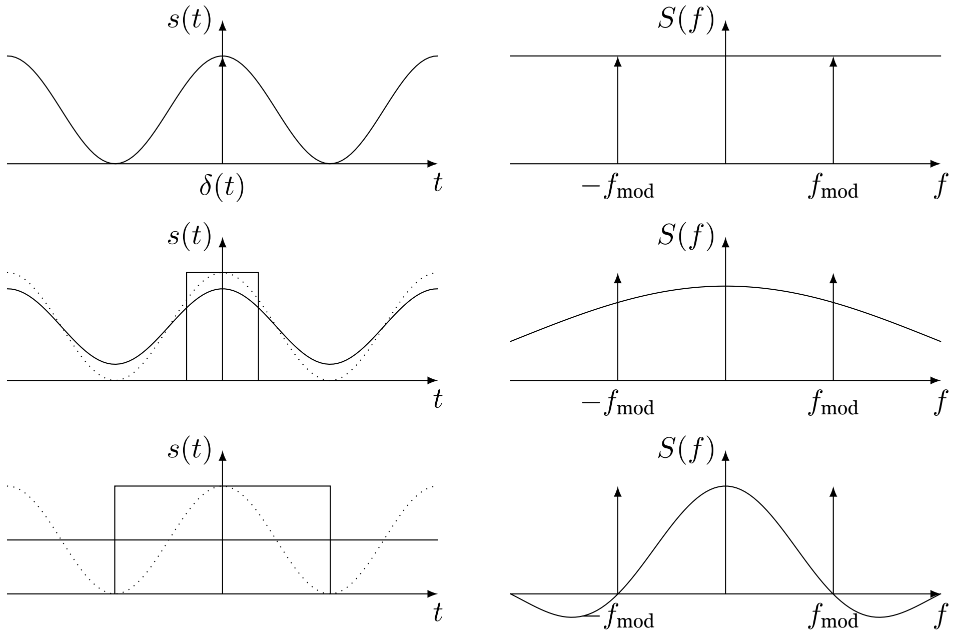


Figure 2.4: *Influence of the physical realization of an ideal sampling by integrating over a certain time interval Δt. The left column shows the integration in the time domain with intervals $\Delta t = 0$ (top), $\Delta t < T_{\text{mod}}$ (middle), and $\Delta t = T_{\text{mod}}$ (bottom). The right column shows the effect in the Fourier domain. The signal is either not affected (top), attenuated (middle), or lost (bottom).*

form $H(f_{\text{mod}}) = \text{sinc}(\pi \frac{\Delta t}{T_{\text{mod}}})$. Thus, the frequency components of $s(t)$ are attenuated as depicted in Figure 2.4. This results in an attenuation of the amplitude of the received signal, which has a negative effect on the reconstruction of the signal parameters ϕ, A, and B in the presence of noise.

In case we further elongate the integration interval to an entire period of the modulation frequency ($\Delta t = T_{\text{mod}}$), the zero-crossings of the transfer function $H(f)$ fall in place with the frequency components of $s(t)$, i.e. $H(f_{\text{mod}}) = 0$, and we lose all information on the signal.

This effect is leads us to the concept of the *demodulation contrast*. In the case that $\Delta t < T_{\text{mod}}$, the transfer function $H(f)$ does not affect the phase of the signal $s(t)$ and the phase shift φ can be estimated as defined in Equation (2.2). The illustration above indicates however that an increase of the integration interval Δt leads to an attenuation of the received signal's amplitude A relative to the amplitude A_{sig} of the emitted signal. The ratio between the received amplitude and the emitted amplitude is referred to as the demodulation contrast, which is defined as

$$c_{\text{demod}} = \frac{A}{A_{\text{sig}}}. \tag{2.6}$$

Thus, integrating over a time window of length Δt yields a demodulation contrast of $c_{\text{demod}} = \text{sinc}(\pi \frac{\Delta t}{T_{\text{mod}}})$. Again, we can observe that the demodulation contrast and thus the robustness of the reconstruction increases for small Δt.

We now consider the case where the length of the integration interval is one fourth of the modulation period ($\Delta t = \frac{T_{\text{mod}}}{4}$). In this case, we obtain a demodulation contrast of $c_{\text{demod}} = \frac{\sqrt{8}}{\pi} \approx 0.90$ and the samples $A_0, \dots, A_3$ can be estimated consecutively during one period, as illustrated in Figure 2.5. In practice, this is done by collecting and storing the photoelectrons generated on the sensor during one of these time intervals in corresponding potential wells - one for each sample A_k. This is achieved by applying an electric field over the light sensitive area of the pixel to guide the photoelectrons into one of the potential wells. This electric field is actuated in synchrony with the illumination unit so that the photoelectrons are driven cyclically into one of the four potential wells corresponding to $A_0, \dots, A_3$ during one cycle of the modulation period. This approach is known as the *4-tap pixel*.

In order to obtain reliable estimates A_k, the total exposure time for one image is extended over several periods of the modulation signal and the above procedure of sampling $s(t)$ is repeated for each period during the entire duration of the exposure time. Apparently, an increased exposure time reduces the frame rate of the camera

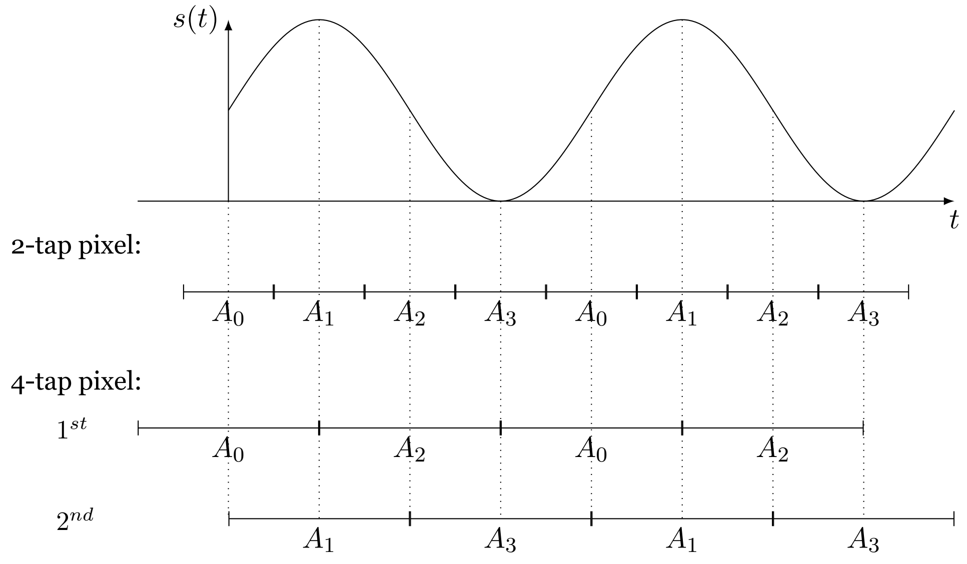


Figure 2.5: *Illustration of the measurement sequence for the samples $A_0, \dots A_3$ for both the 4-tap pixel and the 2-tap pixel.*

while increasing the signal-to-noise ratio of the measurement.

While the 4-tap pixel resembles a straightforward implementation of the measurement principle introduced in Section 2.4 it has a major disadvantage. The four potential wells for storing the photoelectrons take up a large portion of the total space of one pixel. Thus, the actual photo-sensitive area of a pixel is very small, i.e. the 4-tap pixel has a low so-called fill factor. As a result either the size of a pixel has to be very large or the exposure time of the camera has to be increased. Both approaches have their drawbacks. Large pixels reduce the effective spatial resolution of the camera. At the same time, the extension of a pixel is limited because the photoelectrons travel to the potential wells by diffusion and the expected amount of time for this diffusion process increases quadratically with the mean distance the photoelectrons have to travel before reaching the potential wells. Longer exposure times reduce the temporal resolution and also introduce so-called motion artefacts, which we will discuss in more detail at a later point.

Instead of implementing the 4-tap pixel, most recent cameras rely on the so-called *2-tap pixel*. Instead of having four potential wells for storing the photoelectrons during the estimation of $A_0, \dots, A_3$, the samples are estimated in two consecutive steps: First, the sensor integrates over two intervals of the periodic signal to estimate A_0 and A_2. For this operation only two potential wells are required; hence, the name 2-tap pixel. After A_0 and A_2 have been estimated over a number of periods of the modulation signal, the two values are read out and stored in an internal register of the camera. Once the readout is completed, the sensor shifts to measuring A_1 and A_3. In principle, the camera performs two independent measurements and the distance can only be computed after these two measurements have been completed. This approach is also depicted in Figure 2.5.

A main drawback of the 2-tap implementation originates from the fact that the sensor makes two independent measurements at different points in time to obtain one range estimate in a pixel. Imagine that an object boundary moves across the pixel in the moment the sensor switches between the two measurements. In that case, A_0 and A_2 correspond to a different phase shift than A_1 and A_3. Thus, both the computation of φ and the range value are erroneous.

Furthermore, we integrate over a time window $\Delta t = \frac{T_{\text{mod}}}{2}$ to estimate $A_0, \ldots, A_3$ in the 2-tap pixel. This results in a lower demodulation contrast of $c_{\text{demod}} = \frac{2}{\pi} \approx 0.64$ in comparison to $c_{\text{demod}} \approx 0.90$ for the 4-tap pixel. Although the 4-tap pixel seems to be the design of choice from this point of view, the 2-tap pixel has the major advantage that the fill factor is increased significantly. At the same time, the pixel is easily scalable, i.e. the potential wells are located on opposite sides of the pixel and the distance between the containers can be set to the optimal value. Here, the optimal value refers to a tradeoff between a low mean distance for the photoelectrons to travel to the potential wells and a large area of the light sensitive area. The scalability results from the fact that the pixels can be extended easily along the axis that is perpendicular to the potential wells to further increase the light-sensitive area of the pixel. This is the reason why the majority of TOF cameras are based on the 2-tap pixel.

2.6 Measurement Accuracy

The accuracy of the TOF range measurement depends mainly the signal-to-noise ratio of the samples $A_0, \ldots, A_3$. The signal-to-noise ratio is influenced by two different factors. The first is related to the conditions of the background illumination, i.e. the intensity of the ambient light illuminating the scene. The second depends on the actual process of generating the emitted signal $e(t)$ by the active illumination unit.

The influence of the ambient light can be reduced to a large extent through the use of a bandpass filter tuned to the wave length of the infrared light of the active illumination and a circuit that actively suppresses the contributions of the ambient light to the measurements of $A_0, \dots, A_3$,

In the second case, the generation of the signal $e(t)$, the noise sources can only be partially compensated. One deviation of a technical implementation from the measurement principle introduced in Section 2.4 is that the signal $e(t)$ will not be ideally sinusoidal. This has the effect that the range measurement deviates depending on the distance of the object from the camera. Since this error is systematic, it can be corrected (Lange, 2000; Rapp, 2007).

Another noise source related to the illumination unit is the so-called photon shot noise, which is a principal physical limitation of sensors based on active illumination. It refers to the fact that light sources do not emit a steady stream of photons at well defined time intervals, but that the generation of photons happens according to a Poisson distribution, i.e. the time interval between the emission of photons represents a Poisson-distributed random variable. Given that the standard deviation of such a Poisson-distributed random variable is the square root of the mean, the same is true for the standard deviations of the samples $A_0, \dots, A_3$, i.e. $\sigma_{A_i} = \text{sqrt}(A_i) \quad \forall i = 0, \dots, 3$.

According to Lange (2000) this yields a standard deviation of the phase measurement

$$\Delta\varphi = \frac{\sqrt{B}}{A\sqrt{2}} \tag{2.7}$$

and, hence, a standard deviation of the range measurement

$$\Delta R = \frac{c}{4\pi f_{\text{mod}}} \cdot \frac{\sqrt{B}}{c_{\text{demod}} A_{\text{sig}} \sqrt{2}}. \tag{2.8}$$

Equation (2.8) shows that the measurement accuracy is governed by three factors. As already mentioned above, a high amount of background illumination increases the measurement error. At the same time, the measurement error is reduced if we increased the optical power of the active illumination, i.e. if we increase the amplitude of the signal $e(t)$. Here, care has to be taken in situations where the TOF system has to comply with standards of eye safety. Finally, increasing the modulation frequency also reduces the measurement error. Note however, that a higher demodulation frequency reduces also the disambiguity range and that there is a physical limitation in terms of the circuitry on the sensor that toggles between the samples $A_0, \dots, A_3$.

Generally speaking, Equation (2.8) reflects a principal physical limitation of the measurement accuracy and TOF cameras currently on the market are already close to reaching this limit (Büttgen et al., 2005).

2.7 Limitations

As the previous sections have shown, TOF cameras constitute very appealing image sensors. Throughout this work we will see that they can ease numerous computer vision applications because they provide both range and intensity data. However, there also exist a number of limitations of the TOF sensor technology that need to be taken into account when designing applications for such cameras. Apart from the above mentioned *non-ambiguity range*, these limitations entail measurement errors which can be put into the categories of *systematic errors*, errors due to *multiple reflections* in the scene, wrong range estimates at jump edges (so-called *flying pixels*), *motion artefacts*, and problems that arise when *multiple cameras* are used together. In this section, we will review these sources of measurement errors and discuss solutions that avoid or overcome these limitations.

2.7.1 Non-Ambiguity Range

As mentioned earlier in this chapter, TOF cameras based on the modulation principle can only yield non-ambiguous range measurements within a certain distance from the camera. This effect depends on the modulation frequency of the active illumination unit.

To further investigate this effect, let us recall Equation (2.5) as it was introduced in Section 2.4:

$$R = \frac{\varphi\, c}{2\, \omega}. \tag{2.9}$$

As before, φ denotes the phase shift between the periodic signal that was emitted by the camera and the delayed version that is reflected back into the camera from the scene. The modulation frequency f_{mod} contributes to the denominator via the relation $\omega = 2\pi f_{\text{mod}}$.

A TOF camera can only disambiguate distances of objects when the signal does not pass through more than one period of the modulation signal while the light travels to the object and back to the camera. Thus, the non-ambiguity range R_{max} is defined as:

$$R_{\text{max}} = \frac{\varphi\, c}{2\, \omega}. \tag{2.10}$$

Given a typical modulation frequency of $f_{\text{mod}} = 30\text{MHz}$ the non-ambiguity range limits well-defined range measurements to up to 5 meters distance from the camera.

There exists, however, a simple approach to increase the non-ambiguity range. If two images are taken of the same scene with different modulation frequencies, one can reconstruct the distance unambiguously up to a range that corresponds to the least common multiple of the two corresponding non-ambiguity ranges. This procedure is limited to cases, where the objects undergo no or only limited motion in front of the camera. Otherwise, wrong distances will be estimated for pixels that move across object borders from one frame to the other and thus depict different objects at different distances in the two images.

2.7.2 Systematic Errors

As already mentioned in Section 2.6, TOF cameras can produce systematic errors, i.e. range measurement errors that are induced by deviations from an ideal sinusoidal signal $e(t)$ of the illumination unit. Theoretically, these errors can be compensated by using a higher number of samples A_k to estimate the phase shift φ (Lange, 2000; Rapp, 2007). On the contrary, this is impractical as it would introduce higher computational cost and abet motion artefacts (Kolb et al., 2009).

An alternative is to correct systematic errors in a post-processing step. This can, for example, be done using look-up tables (Kahlmann et al., 2007) or correction functions such as b-splines (Lindner and Kolb, 2006).

2.7.3 Multiple Reflections

The working principle of TOF cameras is based on the assumption that the modulated light that reaches a pixel is reflected from a single object at a well defined distance and that the light travels directly back into the camera. In this case, the camera receives a single signal $s(t)$ that has a phase shift φ with respect to the emitted signal $e(t)$ that corresponds to the distance of the object. This assumption may be violated due to multiple reflections that can occur both in the scene and inside the camera.

Multiple reflections in the scene refer to the situation when light of the active illumination travels not solely between the camera and the imaged object, but when it is reflected at additional objects in the scene along its path from the camera to the object and back to the sensor. In such a case, the distance travelled by the light is longer, which results in an increased time of flight and, thus, a false range estimate (Gudmundsson et al., 2007b).

Multiple reflections inside the camera have the effect that stray light from other objects than the imaged one reaches a pixel. This stray light is due to unwanted reflections in the lens and the body of the camera. Again, this results in a false range measurement (Falie and Buzuloiu, 2007).

While the effect of multiple reflections inside the camera can be attenuated by coating the inside of the camera with a non-reflective substance, the problem of multiple reflections in the scene is a fundamental one of the TOF principle. Currently, there exists no general solution to this problem other than arranging the scene in a way that avoids indirect reflections.

2.7.4 Flying Pixels

A related phenomenon is that of the so-called *flying pixels*. Again, the effect occurs because light from different objects at different distances reaches the same pixel which results in a false range measurement. In this case, the border between the objects runs through a single pixel. Thus, the range measurement is composed of the distances of both objects. The contribution of each object to the distance measurement depends on the relative amount of light it sends to the pixel. In case the scene is shifted slightly in front of the camera this relative amount changes and, thus, the corresponding pixel appears to be flying between the objects in the 3D scene.

2.7.5 Motion Artefacts

Motion artefacts occur mainly in implementations of the measurement principle described in Section 2.4 that are based on a 1-tap or a 2-tap pixel. These artefacts are due to the fact that the samples $A_0, \ldots, A_3$ are not taken simultaneously but subsequently. As described above, the 2-tap pixel first estimates A_0 and A_2 and then obtains A_1 and A_3 in a second measurement. In case an object moves between these two measurements, they will be inconsistent. As a result the range estimate will be erroneous. This effect is most severe if a jump edge moves across a pixel between the two measurements.

Note however, that the resulting range measurement does not reflect the mean between distances to the object seen during the two measurement. Instead, the estimated range value can be both smaller than the smallest and greater than the largest range value imaged by a pixel during the time all samples $A_0, \ldots, A_3$ are taken. For more detailed reading on this topic refer to Lindner and Kolb (2009).

2.7.6 Multi-Camera Setups

The use of TOF cameras operating at the same modulation frequency in a configuration where light from one camera's illumination unit can reach the sensor of the other camera corrupts the range measurement. This is due to the fact that the sensor cannot disambiguate between the phase shift induced by its own illumination unit and that of another camera.

A straight forward way of circumventing this problem is to run the cameras at different demodulation frequencies. The demodulation approach to measuring the phase shift responds only to a very narrow band of frequencies near the modulation frequency; all other frequencies will merely appear as an increase in background illumination (Lange, 2000).

A second solution to using several cameras in the same environment is to modulate the illumination by a pseudo-noise signal. In case each camera uses a different code, the sensor will only detect the phase shift corresponding to its own emitted signal (Heinol, 2001).

Part II

Algorithms

3 Introduction

The second part of this thesis is devoted to computer vision algorithms designed for TOF camera data. In this context, we will demonstrate the advantage of a combined image sensor that delivers both range and intensity, i.e. we will explicitly show how the combination of both types of data significantly improves the performance of algorithms in contrast to using either data alone. The first section of this chapter will focus on the improvement of the range measurement by exploiting the intensity image under the well-defined lighting conditions provided by the TOF camera illumination. Secondly, we will address the topic of image segmentation. Here, the goal is to identify connected image regions that depict an object or a person present in the scene. These results will then be used in an algorithm for the estimation of human pose that fits a simple model of the human body in 3D. Finally, we will turn to the discussion of suitable image features for encoding relevant properties of TOF images. In this context, we will first discuss geometrically motivated features that are related to the Gaussian curvature. We will then reformulate the computation of these features in 3D to achieve scale invariance. A third type of features is obtained using the sparse coding principle. These three types of features will be evaluated in the context of detecting facial features, such as the nose. Finally, we will turn to the computation of range flow and will use the resulting 3D motion vectors for the recognition of human gestures. In the following, we will give a brief overview of the four topics discussed in the scope of Part II of thesis within Chapter 4 through Chapter 7.

Shading Constraint Improves TOF Measurements

In Chapter 4, we describe a technique for improving the accuracy of range maps measured by a TOF camera. Our technique is based on the observation that the range map and intensity image are not independent but are linked by the *shading con-*

straint: If the reflectance properties of the surface are known, a certain range map implies a corresponding intensity image. In practice, a general reflectance model (such as Lambertian reflectance) provides a sufficient approximation for a wide range of surfaces. We impose the shading constraint using a probabilistic model of image formation and find a maximum a posteriori estimate for the true range map. We present results on both synthetic and real TOF camera images that demonstrate the robust shape estimates achieved by the algorithm. We also show how the reflectivity (or *albedo*) of the surface can be estimated, both globally for an entire object and locally for objects where albedo varies across the surface.

Image Segmentation

In image analysis, a very helpful step towards interpreting the content of an image is to assign the pixels of the image to different categories. Often, one uses a category for the background of the scene and one category for each object appearing in front of the background. In Chapter 5, we describe how the available range map can be used effectively in combination with the intensity data to devise two very efficient algorithms that reliably identify the pixels belonging to a person in front the camera. One method relies on a previously captured model of the background and determines the person as a deviation from this background model. The second method operates on a histogram of the range values to identify the object closest to the camera. In both cases, the range data of the TOF camera makes it possible to obtain very robust segmentation results even in complex scenes.

Pose Estimation

In Chapter 6, we describe a technique for estimating human pose from an image sequence captured by a TOF camera. The pose estimation is derived from a simple model of the human body that we fit to the data in 3D space. The model is represented by a graph consisting of 44 vertices for the upper torso, head, and arms. The anatomy of these body parts is encoded by the edges, i.e. an arm is represented by a chain of pairwise connected vertices whereas the torso consists of a 2-dimensional grid. The model can easily be extended to the representation of legs by adding further chains of pairwise connected vertices to the lower torso. The model is fit to the data in 3D space by employing an iterative update rule common to self-organizing maps. Despite the simplicity of the model, it captures the human pose robustly and can thus be used for tracking the major body parts, such as arms, hands, and head. The accuracy of the tracking is around 5–6 cm root mean square (RMS) for the head and shoulders and around 2 cm RMS for the head. The implementation of the procedure is straightforward and real-time capable.

Features

The discussion of TOF image features in Chapter 7 is divided into four individual parts. The first part in Section 7.1 will discuss the so-called *generalized eccentricities*, a kind of feature that can be used to distinguish between different surface types, i.e. one can for example distinguish between planar surface regions, edges, and corners in 3D. These features were employed for detecting the nose in frontal face images and we obtained an equal error rate of 3.0%. Section 7.2 will focus on a reformulation of the generalized eccentricities such that the resulting features become invariant towards scale. This is achieved by computing the features not on the image grid but on the sampled surface of the object in 3D. This becomes possible by using the range map to invert the perspective camera projection of the TOF camera. As a results, one obtains data that is irregularly sampled. Here, we propose the use of the Nonequispaced Fast Fourier Transform to compute the features. As a result, one can observe a significantly improved robustness of the nose detection when the person is moving towards and away from the camera. An error rate of zero is achieved on the test data.

The third category of image features is computed using the sparse coding principle, i.e. we learn an image basis for the simultaneous representation of TOF range and intensity data. We show in Section 7.3 that the resulting features outperform features obtained using Principal Component Analysis in the same nose detection task that was evaluated for the geometric features. In comparison to the generalized eccentricities we achieve a slightly reduced performance. On the other hand, in this scenario the features were simply obtained under the sparse coding principle without incorporating prior knowledge of the data or properties of the object to be detected.

The fourth type of features, presented in Section 7.4, aims at the extraction of the 3D motion of objects in the scene. To this end, we rely on the computation of range flow. The goal is the recognition of human gestures. We propose to combine the computation of range flow with the previously discussed estimation of human pose, i.e. we explicitly compute the 3D motion vectors for the hands of the person performing a gesture. These motion vectors are accumulated in 3D motion histograms. We then apply a learned decision rule to assign a gesture to each frame of a video sequence. Here, we specifically focus on the problem of detecting that no gesture was performed, i.e. each frame is either assigned to a one of the predefined gestures or to the class indicating that no gesture was performed. We achieve detection rates above 90% for three investigated hand gestures.

4 Shading Constraint Improves TOF Measurements

4.1 Introduction

In this chapter, we present a technique for improving the accuracy of the TOF camera's range measurements, based on the insight that the range and intensity measurements are not independent, but are linked by the *shading constraint*: Assuming that the reflectance properties of the object surface are known, we can deduce the intensity image that should be observed. In practice, a general reflectance model (such as Lambertian reflectance) will provide an acceptable approximation to the properties of a wide range of objects.

In theory, the shading constraint can be used to reconstruct the range map from an intensity image alone; this idea has been exploited in a wide range of *shape from shading* (SfS) algorithms (see (Zhang et al., 1999; Durou et al., 2008) for surveys). A principal limitation of these algorithms, however, is that they cannot determine whether intensity changes are caused by the object's shape or by changes in the object's reflectivity (or *albedo*). Because of this, the object is usually assumed to have constant albedo; this limits the applicability of SfS methods.

The range map measured by the TOF camera places a strong additional constraint on the shape of the object, allowing ambiguities that may exist in the pure SfS setting (Durou and Piau, 2000) to be resolved and enabling the albedo of the surface to be estimated, both globally for an entire object as well as locally for objects where albedo varies across the surface.

This chapter describes results obtained in collaboration with Martin Böhme, who formulated the probabilistic model of image formation used to enforce the shading constraint. Martin Böhme and I contributed approximately equally to refining and implementing the method. The work described here has previously been published in (Böhme et al., 2008b, 2010).

Besides the shading constraint, there are also other ways of fusing range and intensity data. A number of authors exploit the fact that an edge in the intensity data often co-occurs with an edge in the range data. Nadabar and Jain (1995) use a Markov random field (MRF) to identify different types of edges. Diebel and Thrun (2006) use edge strengths estimated on a high-resolution color image to increase the resolution of a low-resolution depth map.

The idea of integrating the shading constraint with other range information has been investigated by a number of researchers. Most of this work focuses on the integration of SfS with stereo. These two techniques complement each other well because SfS works well on uniformly colored areas whereas stereo requires surface texture to find stereo correspondences. Because of this, the fusion of SfS with stereo has a slightly different focus than the fusion of SfS with a TOF range map. In the stereo case, we only have range information in textured areas and need to rely on shading cues in untextured areas. In the TOF case, we have a dense range map and wish to fuse information from TOF and shading at the same pixel.

Many approaches to the fusion of SfS and stereo (see for example the work of Thompson (1993), Fua and Leclerc (1995), and Hartt and Carlotto (1989)) use an objective function that depends directly on the two or more images obtained from a multi-camera setup; for this reason, they do not generalize to settings where a range map has been obtained in some other way than through stereo. Samaras et al. (2000) combine stereo with SfS by using stereo in textured areas and SfS in untextured areas, but they do not perform a fusion of stereo and SfS at the same location. Haines and Wilson (2007, 2008) fuse stereo and SfS in a probabilistic approach based on a disparity map and the shading observed in one of the stereo images. Because there is a one-to-one correspondence between disparity and range, the approach could also be used with range maps obtained by arbitrary means. However, since color is used to segment areas of different albedo, the approach is not suitable for use with TOF cameras, which typically only deliver a grayscale image.

There are several other approaches that combine shading with a range map obtained by arbitrary means; stereo may be used, but it is not essential to the formulation of the algorithm. Leclerc and Bobick (1991) use a stereo range map to initialize an iterative SfS method. Cryer et al. (1995) use a heuristic that combines low-frequency components from the stereo range map with high-frequency components from the SfS range map. Mostafa et al. (1999) use a neural network to interpolate the difference between the SfS result and a more coarsely sampled range map from a range sensor; the SfS result is corrected using this error estimate. These approaches allow arbitrary range maps to be used, but they are all somewhat ad-hoc.

Our approach to improving the accuracy of the range map using the shading constraint is based on a probabilistic model of the image formation process. We obtain a maximum a posteriori estimate for the range map using a numerical minimization technique. The approach has a solid theoretical foundation and incorporates the sensor-based range information and the shading constraint in a single model; for details, see Section 4.2. The method delivers robust estimation results on both synthetic and natural images, as we show in Section 4.3.

4.2 Method

4.2.1 Probabilistic Image Formation Model

We seek to find the range map $\mathbf{R}$ that maximizes the posterior probability

$$p(\mathbf{X}^{\mathrm{R}}, \mathbf{X}^{\mathrm{I}} | \mathbf{R}, \mathbf{A}) \; p(\mathbf{R}) \; p(\mathbf{A}). \tag{4.1}$$

$p(\mathbf{X}^{\mathrm{R}}, \mathbf{X}^{\mathrm{I}} | \mathbf{R}, \mathbf{A})$ is the probability of observing a range map $\mathbf{X}^{\mathrm{R}}$ and an intensity image $\mathbf{X}^{\mathrm{I}}$ given that the true range map describing the shape of the imaged object is $\mathbf{R}$ and that the parameters of the reflectance model are $\mathbf{A}$. Typically, $\mathbf{A}$ is the albedo of the object – we will discuss this in more detail below. $p(\mathbf{R})$ is a prior on the range map, $p(\mathbf{A})$ is a prior on the reflectance model parameters.

The conditional probability $p(\mathbf{X}^{\mathrm{R}}, \mathbf{X}^{\mathrm{I}} | \mathbf{R}, \mathbf{A})$ is based on the following model of image formation: First of all, we assume that $p(\mathbf{X}^{\mathrm{R}}, \mathbf{X}^{\mathrm{I}} | \mathbf{R}, \mathbf{A})$ can be written as follows:

$$p(\mathbf{X}^{\mathrm{R}}, \mathbf{X}^{\mathrm{I}} | \mathbf{R}, \mathbf{A}) = p(\mathbf{X}^{\mathrm{R}} | \mathbf{R}, \mathbf{A}) \; p(\mathbf{X}^{\mathrm{I}} | \mathbf{R}, \mathbf{A}). \tag{4.2}$$

In other words, the observations $\mathbf{X}^{\mathrm{R}}$ and $\mathbf{X}^{\mathrm{I}}$ are conditionally independent given $\mathbf{R}$ and $\mathbf{A}$.

We now assume that the observed range map $\mathbf{X}^{\mathrm{R}}$ is simply the true range map $\mathbf{R}$ with additive Gaussian noise, i.e.

$$p(\mathbf{X}^{\mathrm{R}} | \mathbf{R}, \mathbf{A}) = \mathcal{N}(\mathbf{X}^{\mathrm{R}} - \mathbf{R} | \mu = 0, \sigma_{\mathrm{R}}(\mathbf{R}, \mathbf{A})). \tag{4.3}$$

Note that the standard deviation σ_{R} is not constant but can vary per pixel as a function of range and albedo. As we will see in Section 4.2.4, the noise in the range measurement of a TOF camera depends on the amount of light that returns to the camera.

The shading constraint postulates that a given range map $\mathbf{R}$ is associated with an intensity image $\mathbf{I}(\mathbf{R}, \mathbf{A})$, where the function expressed by $\mathbf{I}$ depends on the reflectance model. We generally use the Lambertian reflectance model, see Section 4.2.2; in this case, $\mathbf{A}$ is the albedo of the object, which may vary from pixel to pixel. Again, we assume that the intensity image is corrupted by additive Gaussian noise, i.e.

$$p(\mathbf{X}^{\mathrm{I}}|\mathbf{R}, \mathbf{A}) = \mathcal{N}(\mathbf{X}^{\mathrm{I}} - \mathbf{I}(\mathbf{R}, \mathbf{A})|\mu = 0, \sigma_{\mathrm{I}}). \tag{4.4}$$

For the range map prior $p(\mathbf{R})$, we use the shape prior introduced by Diebel et al. (2006), which favors surfaces with smoothly changing surface normals. We tessellate the range map into triangles (see Section 4.2.3) and compute the surface normal $\mathbf{n}_j$ for each triangle. The shape prior is then given by the energy function

$$E^{\mathrm{R}}(\mathbf{R}) = w_{\mathrm{R}} \sum_{\substack{\text{triangles } j,k \\ \text{adjacent}}} \|\mathbf{n}_j - \mathbf{n}_k\|_2, \tag{4.5}$$

which implies the distribution $p(\mathbf{R}) = \frac{1}{Z} \exp(-E^{\mathrm{R}}(\mathbf{R}))$, where Z is a normalization constant. w_{R} is a constant that controls the dispersion of the distribution.

We now turn to the prior $p(\mathbf{A})$ for the parameters $\mathbf{A}$ of the reflectance model. In the Lambertian reflectance model, these are the albedo values at each pixel location. We will investigate several alternatives for the prior $p(\mathbf{A})$: (i) "Fixed albedo": A single albedo value, specified beforehand, is used for all pixels. (ii) "Global albedo": The same global albedo is used for all pixels, but its value is allowed to vary; we assume a uniform distribution for this global albedo. (iii) "Local albedo": Each pixel location may have a different albedo, and the prior $p(\mathbf{A})$ favors smooth albedo changes. In this latter case, we use an energy function

$$E^{\mathrm{A}}(\mathbf{A}) = w_{\mathrm{A}} \sum_{\substack{\text{pixels } j,k \\ \text{adjacent}}} |a_j - a_k|, \tag{4.6}$$

which implies the prior $p(\mathbf{A}) = \frac{1}{Z} \exp(-E^{\mathrm{A}}(\mathbf{A}))$, in analogy to the shape prior defined above.

As usual, we take the negative logarithm of the posterior and eliminate constant additive terms to obtain an energy function

$$\begin{aligned} E(\mathbf{R},\mathbf{A}) = & \sum_j \frac{(X_j^{\mathrm{R}} - R_j)^2}{2\,\sigma_{\mathrm{R}}^2} + \\ & \sum_j \frac{(X_j^{\mathrm{I}} - I_j(\mathbf{R},\mathbf{A}))^2}{2\,\sigma_{\mathrm{I}}^2} + \\ & E^{\mathrm{R}}(\mathbf{R}) + E^{\mathrm{A}}(\mathbf{A}), \end{aligned} \tag{4.7}$$

where the index j runs over all pixels. (For the “fixed albedo” and “global albedo” models, the term $E^{\mathrm{A}}(\mathbf{A})$ is omitted.) Note that all the terms in the energy function are unitless due to multiplication or division by the constants σ_{R}, σ_{I}, w_{R} and w_{A}.

We find the maximum a posteriori estimate for the range map by minimizing $E(\mathbf{R},\mathbf{A})$ using the Polak-Ribière variant of the nonlinear conjugate gradient algorithm (see for example (Press et al., 1992)). As the starting point for the minimization, we use the observed range map $\mathbf{X}^{\mathrm{R}}$, smoothed using a median filter, and an albedo guess (see Section 4.2.4). The gradient of $E(\mathbf{R},\mathbf{A})$ is computed numerically using a finite differences approximation. The parameters σ_{R}, σ_{I}, w_{R} and w_{A} should be set to reflect the noise characteristics of the sensor and the statistical properties of the scene.

4.2.2 Lambertian Reflectance Model

Under the Lambertian model of diffuse reflection (Trucco and Verri, 1998), the intensity I with which a point on an object appears in the image is obtained as follows:

$$I = a\frac{\mathbf{n}\cdot\mathbf{l}}{r^2}, \tag{4.8}$$

where $\mathbf{n}$ is the surface normal, $\mathbf{l}$ is the unit vector from the surface point towards the light source, r is the distance of the surface point to the light source, and a is a constant that depends on the albedo of the surface, the intensity of the light source, and properties of the camera such as aperture and exposure time. For brevity, we will refer to a simply as the albedo, because any changes to a across the scene are due to albedo changes, while the properties of the light source and camera remain constant.

On a TOF camera, the light source can be assumed to be co-located with the camera, and so r is simply the range value for the surface point, and $\mathbf{l}$ is the unit vector from the surface point to the camera.

4.2.3 Computation of Surface Normals

Both the Lambertian reflectance model and the shape prior for smooth surfaces require the normals of the surface to be known; some care needs to be taken when computing these normals on a discretely sampled range map. An obvious way is to compute the cross product of two tangent vectors $\mathbf{p}(i+1,j) - \mathbf{p}(i-1,j)$ and $\mathbf{p}(i,j+1) - \mathbf{p}(i,j-1)$ (where $\mathbf{p}(i,j)$ are the three-dimensional coordinates of the point corresponding to pixel (i,j)), but surface normals calculated in this way can lead the minimizer astray: Because the normals of pixels with even indices depend only on the positions of pixels with odd indices, and vice versa, neighboring pixels are not constrained to have similar range values, and the minimizer may happily compute a surface with a "checkerboard" pattern, where neighboring pixels are alternately displaced upwards and downwards by a certain offset, instead of forming a smooth surface.

For this reason, care needs to be taken when evaluating the reflectance model and shape prior to obtain a formulation that is physically realistic and does not lead the minimizer astray. For any given pixel, the range and intensity measured by the camera are averages over the area covered by the pixel. Nevertheless, we will assume that these values correspond to the range and intensity of an individual point on the object's surface. To obtain a continuous surface between this grid of points, we tessellate the grid into triangles. Of the many possible tessellations, we choose one where the diagonals dividing a quad of pixels into two triangles run parallel.

To compute the intensity of a pixel, given a certain range map, we compute the average intensity over all triangles adjacent to it. (All triangles have the same projected area in the image, hence they are all weighted equally.) Because the triangles in the mesh are small compared to their distance from the camera, we can assume that intensity is constant across a triangle. The intensity I_j for pixel j is thus obtained as follows:

$$I_j = \frac{a_j \sum_{k \in N_j} \mathbf{n}_k \cdot \mathbf{l}_j}{R_j^2 \, |N_j|}, \tag{4.9}$$

where a_j is the albedo of the pixel, R_j is the range value of the pixel, $\mathbf{l}_j$ is the unit vector from the surface point to the light source, $\mathbf{n}_k$ is the surface normal of triangle k, and N_j is the set of triangles that are adjacent to pixel j; see Figure 4.1 for an illustration of the triangles that are adjacent to a pixel.

When computing the shape prior, we must take care to count each edge exactly once. We do this by iterating over all pixels and, for each pixel, evaluating the shape prior only for those edges in the tessellation that lie below and to the right of the pixel

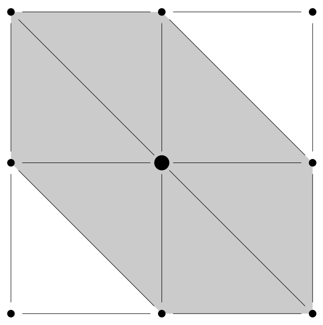


Figure 4.1: *The intensity of a pixel is computed by averaging over the intensity of all triangles that are adjacent to it. Triangles that are adjacent to the central pixel are shaded in gray.*

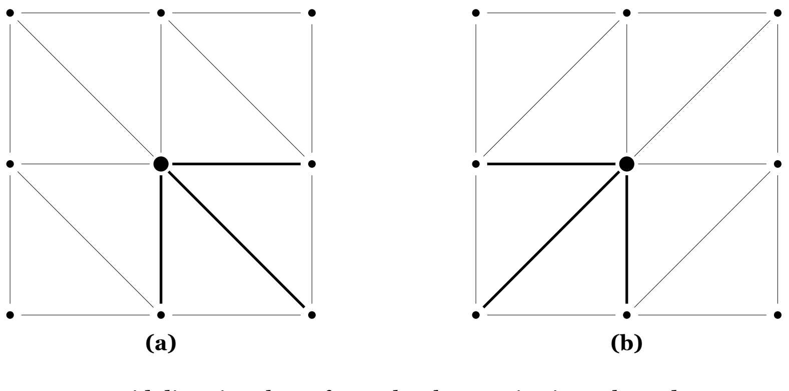


Figure 4.2: *To avoid directional artefacts, the shape prior is evaluated over two different tessellations, with the diagonals running in opposite directions. Figure 4.2a shows a tessellation with diagonals running top left to bottom right. For each pixel, the shape prior is evaluated for the edges (shown in bold) below and to the right of the pixel. Figure 4.2a shows a tessellation with diagonals running bottom left to top right. The shape prior is evaluated for the edges below and to the left of the pixel.*

in the grid. This is illustrated in Figure 4.2a, where the bold edges are the ones for which the shape prior is evaluated when considering the central pixel.

The direction in which the diagonals run in the tessellation introduces an asymmetry, and we have found that this asymmetry can cause the shape prior to generate directional artefacts. For this reason, we evaluate the shape prior for both of the two possible directions of the diagonal and sum over the result. Figure 4.2b shows which edges are evaluated for a given pixel on the tessellation with the alternative direction of the diagonal.

4.2.4 Application to Time-of-Flight Cameras

When applying the method to images recorded using a TOF camera, some particular characteristics of this sensor need to be taken into account to obtain optimal results.

First, the noise σ_R in the measured range map is not the same for all pixels but depends on the amount of light collected at each pixel – the more light, the more accurate the measurement. Hence, for each pixel, we set σ_R as a function of intensity. To estimate the functional relationship between intensity and σ_R, we recorded a sequence of images of a static scene and, at each pixel, calculated the standard deviation of the measured range values. We then fitted a power law function to the calculated standard deviations as a function of intensity and used this function to set σ_R in the surface reconstruction algorithm.

Another important point is that most TOF cameras do not illuminate the scene homogeneously; typically, the illumination falls off towards the edges of the field of view. To measure this effect, we recorded an image, averaged over 100 frames, of a planar object with constant albedo. By comparing the actual image $\mathbf{X}^\mathrm{I}_\mathrm{a}$ to the image predicted by our shading model $\mathbf{X}^\mathrm{I}_\mathrm{p}$ (which assumes homogeneous illumination), we were able to estimate the relative illumination strength at each pixel and use this to compensate for the effect in subsequent recordings $\mathbf{X}^\mathrm{I}$ via

$$\mathbf{X}^\mathrm{I}_\mathrm{corrected}(i,j) = \frac{\mathbf{X}^\mathrm{I}_\mathrm{p}(i,j) \cdot \mathbf{X}^\mathrm{I}(i,j)}{\mathbf{X}^\mathrm{I}_\mathrm{a}(i,j)}. \tag{4.10}$$

Finally, if the albedo of the measured surface is approximately constant, a good initial albedo estimate can be found as follows: We find the highest-intensity pixel in the image; generally, this pixel will correspond to a part of the object that is perpendicular to the incoming light, because such regions reflect the most light. Hence, at this location, equation (4.8) reduces to $I = \frac{a}{r^2}$, and we obtain the albedo as $a = I\,r^2$. A conventional camera cannot be used to estimate albedo in this way because there,

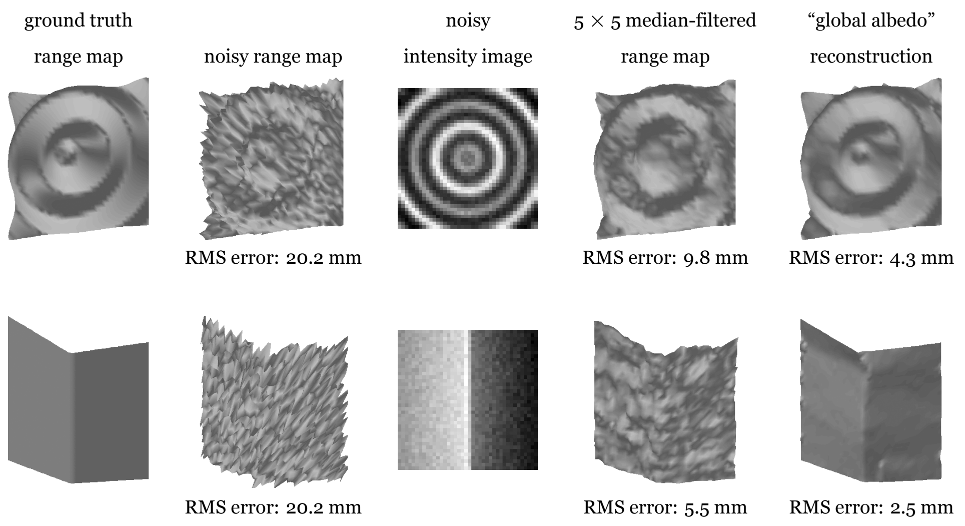


Figure 4.3: *Reconstruction results for two synthetic test objects ("wave", top, and "corner", bottom). Gaussian noise with a standard deviation of 20 mm was added to the range map; for comparison, the "wave" object has a depth of 100 mm, and the "corner" object has a depth of 120 mm. Gaussian noise was added to the intensity image at a signal-to-noise ratio of 36 dB; the maximum intensity in the images was 0.19 ("corner") and 0.22 ("wave").*

the range r is not known.

4.3 Results

4.3.1 Synthetic Data

To assess the accuracy of the method quantitatively, we first tested it on synthetic data with known ground truth: A rotationally symmetric sinusoid (the "wave" object) and an object composed of two planar surfaces that meet at a sharp edge (the "corner" object); see Figure 4.3. To simulate the measurement process of the TOF camera, we shaded the ground truth surface with a constant albedo, then added Gaussian noise; the observed range map was obtained by adding Gaussian noise to the ground truth surface. For all tests that follow, we set $w_\mathrm{R} = 1$ and $w_\mathrm{A} = 50$; σ_R and σ_I were set to the actual standard deviations of the noise that was added to the range map and intensity image.

Figure 4.3 shows the ground truth range maps for the "wave" and "corner" objects along with the noisy range map and intensity image that were used as input to the reconstruction algorithm, and the reconstruction result. For comparison, the figure also shows the result of filtering the range map with a 5×5 median filter.

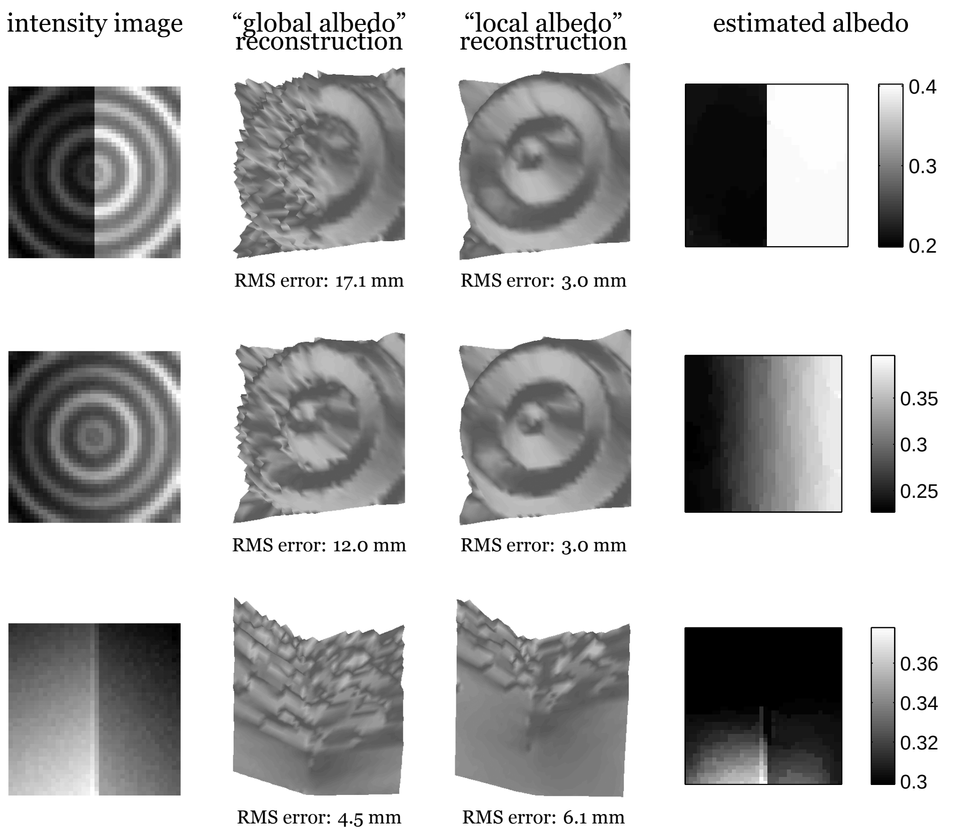


Figure 4.4: *Reconstruction results on synthetic objects with varying albedo. Top: "Wave" object with an albedo of 0.2 on the left half of the object and 0.4 on the right half. Middle: "Wave" object with albedo varying continuously from 0.2 at the left to 0.4 at the right. Bottom: "Corner" object with albedo varying continuously from 0.2 at the top to 0.4 at the bottom. In all cases, the noise in the range map had a standard deviation of 5 mm, and noise was added to the intensity image at a signal-to-noise ratio of 36 dB.*

The noise in the range map had a standard deviation of 20 mm; for comparison, the "wave" object has a depth of 100 mm, and the "corner" object has a depth of 120 mm. The intensity image noise was set to a signal-to-noise ratio of 36 dB; the maximum intensity in the images was 0.19 ("corner") and 0.22 ("wave"). The "global albedo" algorithm was used to reconstruct the surface; the initial albedo value for the minimization was set to twice the actual value that was used to produce the intensity image. The RMS error in the reconstructed surface is reduced by a factor of over 4 for the "wave" object and around 8 for the "corner" object.

Next, we examine the results of the algorithm on an object with varying albedo. First, we use the "wave" object with albedo set to 0.2 on the left half of the image and 0.4 on the right half (Figure 4.4, top); the noise in the range image was reduced to a standard deviation of 5 mm. Reconstructions were computed using the "global albedo" and "local albedo" algorithms; the initial albedo value for the minimization was set to 0.3. Note that the "global albedo" algorithm does not yield a satisfactory result, while the "local albedo" version does; local albedo is estimated almost perfectly. A second test (Figure 4.4, middle) used the same object but with albedo varying continuously from 0.2 at the left to 0.4 at the right. Albedo is overestimated slightly on the left side of the image, and the result is not quite as good as in the first case but still satisfactory. Finally, we show a case where the albedo estimation does not work properly (Figure 4.4, bottom): the "corner" object with albedo varying continuously between 0.2 at the top and 0.4 at the bottom. Here, the result of the "local albedo" algorithm is not satisfactory and, in fact, its RMS error is higher than that of the "global albedo" algorithm. We suspect the reason for the poor performance may be that the range map does not contain enough detail for the algorithm to "latch onto".

Finally, we measured the effect of the various components of the probabilistic model. Figure 4.5 shows the reconstruction error on the "wave" object as a function of the noise σ_{I} in the intensity image. We compare probabilistic models that use only the shading constraint $p(\mathbf{X}^{\mathrm{I}}|\mathbf{R},\mathbf{A})$, only the shape prior $p(\mathbf{R})$, or both together. (The term $p(\mathbf{X}^{\mathrm{R}}|\mathbf{R},\mathbf{A})$, which incorporates the information from the measured range map, was used in all cases. Because albedo did not vary across the image, the term $p(\mathbf{A})$ was omitted.)

Since σ_{I} appears in the denominator of the shading term in the energy function (Equation (4.7)), we need to treat $\sigma_{\mathrm{I}} = 0$ (no intensity noise) as a special case. Note that for $\sigma_{\mathrm{I}} \to 0$, the shading term dominates all the other terms; hence, if $\sigma_{\mathrm{I}} = 0$, we omit all other terms from the energy function. We can then avoid scaling by $\frac{1}{\sigma_{\mathrm{I}}{}^2}$.

The error for the reconstruction obtained using only the shape prior (along with the measured range map $p(\mathbf{X}^{\mathrm{R}}|\mathbf{R},\mathbf{A})$) is, of course, constant for all σ_{I} because it does not use the intensity image. The shape prior reduces the RMS error in the range map by around a factor of 2.

For the reconstructions obtained using only the shading constraint (along with the measured range map), the error in the reconstruction generally increases with the noise σ_{I} in the intensity image. However, it is notable that, even for $\sigma_{\mathrm{I}} = 0$, the range map is not reconstructed perfectly. Though, in this case, the ground truth range map is obviously the global minimum of the energy function, the algorithm appears to get stuck in a local minimum. Recall that in the case $\sigma_{\mathrm{I}} = 0$, the shading term dominates all other terms in the energy function. As σ_{I} increases, the energy function begins

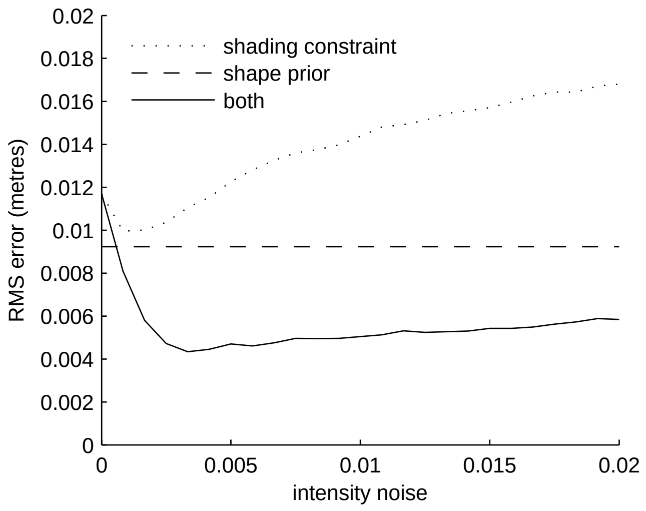


Figure 4.5: *Reconstruction error on the "wave" object as a function of noise in the intensity image, for different probabilistic models. (For comparison, the maximum intensity in the image was 0.22.) Range noise was fixed at a standard deviation of 20 mm.*

taking the measured range map into account, and this in fact leads to an initial slight reduction in the reconstruction error; we speculate that the additional constraint imposed by the measured range map makes it easier to minimize the energy function.

This effect is even more pronounced for the full model, which combines the shading constraint, the shape prior, and the measured range map. For $\sigma_I = 0$, the shading term again dominates all other terms in the energy function, and so we obtain the same result as for the shading constraint alone. As σ_I begins to increase, the reconstruction error decreases markedly as the shape prior and the measured range map come into play. After a certain point, the reconstruction error begins increasing again; for $\sigma_I \to \infty$, the reconstruction error will tend to that of the range prior because the shading term in Equation (4.7) tends to zero. Note that, except for very small σ_I, the combined model yields better results than either the shading constraint or the shape prior alone.

4.3.2 Real-World Data

We now apply the algorithm to data obtained using an SR3000 TOF camera (Oggier et al., 2005a), which has a resolution of 176 by 144 pixels. The parameters σ_R and σ_I (standard deviations of range and intensity) were set to values estimated on a sequence of images of a static scene; σ_R was set as a function of intensity for each pixel (see Section 4.2.4), while σ_I was constant across the whole scene, in accordance with the statistical properties of the sensor. The parameters for the shape and albedo prior were again set to $w_\mathrm{R} = 1$ and $w_\mathrm{A} = 50$.

We first demonstrate the algorithm on two terracotta objects, which fulfill the assumption of Lambertian reflectance quite well and can be assumed to have approximately constant albedo. Figure 4.6 shows the input data and reconstruction results for the two terracotta objects. To compare the results with the effect that a conventional filtering has on the range map, a 5×5 median-filtered version is also shown. The objects were segmented manually, and the reconstruction was performed using the “global albedo” algorithm. The subjective quality of the reconstruction is greatly improved compared to both the raw data and the median-filtered version; note, in particular, how the shading constraint allows us to reconstruct detail in the objects that was drowned out by noise in the measured range map.

Figure 4.7 shows the results of the algorithm on a human face. This is a more challenging object for the algorithm because the reflectance properties of skin are considerably more complex than the Lambertian reflectance assumed by the model; also, albedo variations occur in places such as the eyebrows and the lips.

We show the result of both the global and local albedo versions of the algorithm; again, a 5×5 median-filtered version is also shown. It is evident that local albedo estimation allows a much more faithful reconstruction than global albedo estimation in areas, such as the lips, where albedo variations occur. Also, despite the fact that skin is not a Lambertian reflector, the shape of the face is reconstructed quite accurately, demonstrating that the algorithm is not very sensitive to violations of the Lambertian reflectance assumption.

Finally, Figure 4.8 shows the results of the algorithm on the upper body of a person. Note how the shading constraint allows the cloth folds to be reconstructed faithfully. This example also illustrates the limitations of the algorithm: The head is reconstructed less well than in the previous example; we believe this is because there is too much albedo variation in an area of only a few pixels. The lowest part of the body is not reconstructed well either, and this is probably due to the low reflectivity of the material in this region, which leads to a large amount of noise in both the range map and the intensity image.

4.4 Discussion

As we have shown, enforcing the shading constraint can substantially improve the quality of range maps obtained using a TOF camera, both in terms of objective mea-

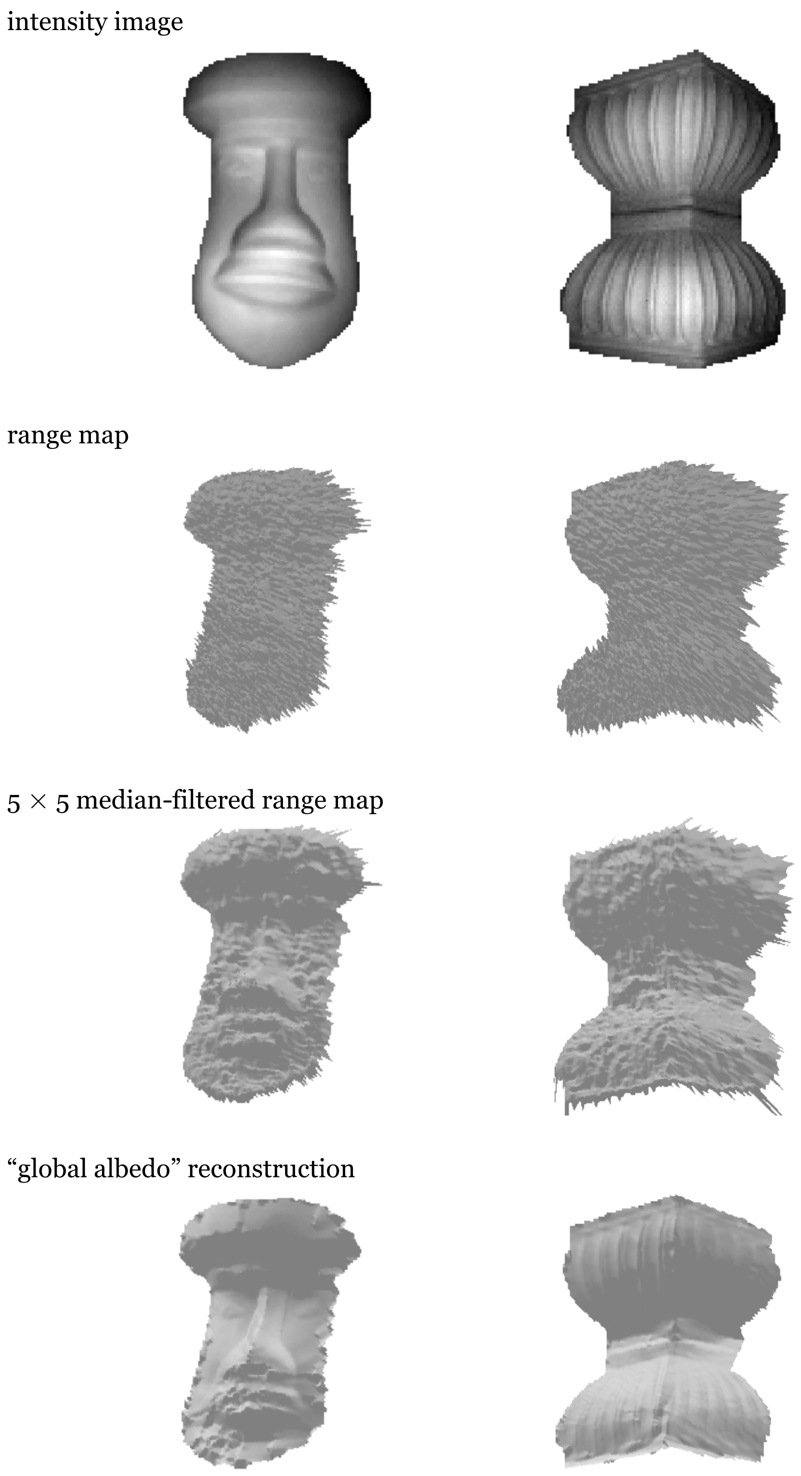

intensity image
range map
5 × 5 median-filtered range map
"global albedo" reconstruction

Figure 4.6: *Surface reconstructions of two terracotta objects, manually segmented in images taken with an SR3000 TOF camera. The renderings of the range maps are rotated 30 degrees around the vertical axis.*

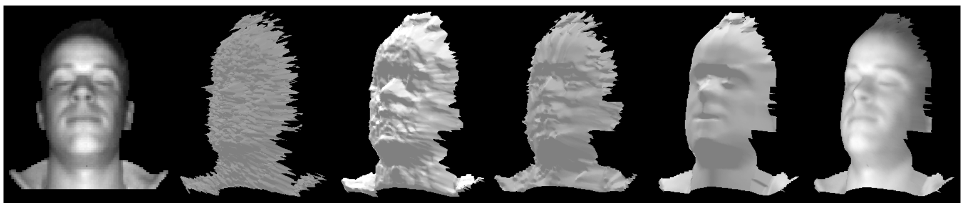


Figure 4.7: *3D reconstruction of a human face. (a) Manually segmented intensity image, (b) measured range map, (c) 5 × 5 median-filtered range map, (d) "global albedo" reconstruction, (e) "local albedo" reconstruction, (f) "local albedo" reconstruction textured with intensity image. The renderings of the range maps are rotated 30 degrees around the vertical axis.*

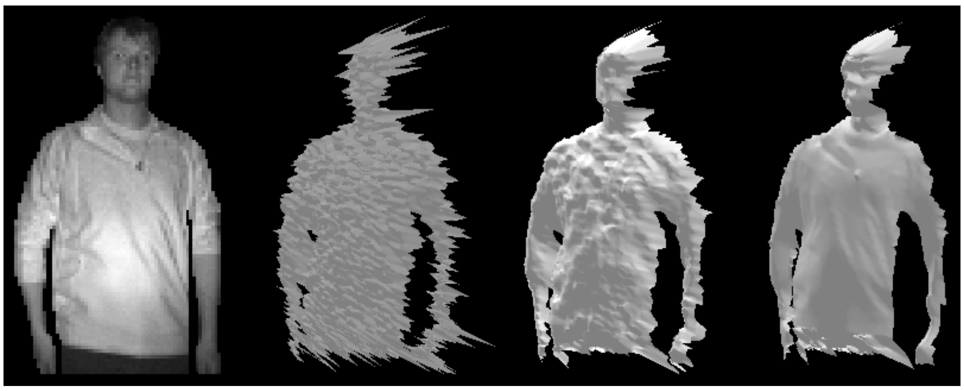


Figure 4.8: *3D reconstruction of a person's upper body. (a) Manually segmented intensity image, (b) measured range map, (c) 5 × 5 median-filtered range map, (d) "local albedo" reconstruction. The renderings of the range maps are rotated 30 degrees around the vertical axis.*

sures as well as subjectively perceived quality.

The TOF camera is particularly well suited for algorithms that incorporate SfS because it eliminates many sources of variability that are difficult to deal with in the general SfS setting: In the TOF camera, the position of the light source is known (it is co-located with the camera); the camera attenuates all other sources of light; and the albedo of the surface can be estimated robustly because its distance from the light source is known (see Section 4.2.4).

The main limitation of the current algorithm is that it does not cope well with range discontinuities, so-called *jump edges*. Because the reconstructed surface is always continuous, jump edges lead to surface normals that are almost perpendicular to the incoming light; hence, the corresponding regions are shaded with very low intensity. This disagrees with the observed image, so the algorithm will flatten the edge to compensate.

It should be possible to overcome this limitation by ignoring any mesh triangle that straddles a jump edge. Jump edges could be identified either by searching for large jumps in the measured range maps or by incorporating jump edges into the probabilistic image model, as in the work of Nadabar and Jain (1995).

It should also be noted that the algorithm is computationally fairly expensive; the current implementation takes several minutes to process an image on a contemporary PC. Since our main focus was correctness, not performance, we expect that optimization should yield a substantial speedup. However, optimizing the algorithm to the point that it could run at camera frame rates would present a major challenge and would probably require techniques such as computation on the graphics processing unit (GPU).

Even in its present form, though, the algorithm is suitable for the post-processing either of individual images or of recorded image sequences. Of course, other range sensors, such as laser range scanners, still provide far better accuracy than TOF camera data post-processed using our algorithm. The strength of the TOF camera, however, lies in its high temporal resolution and its potential to be manufactured at low cost for mass-market applications. Enforcing the shading constraint allows TOF cameras to provide range maps of considerably enhanced quality, opening up many new application fields.

5 Segmentation

5.1 Introduction

An important component of computer vision algorithms that aim at interpreting the scene in front of the camera is known as *segmentation*. It refers to segmenting the image into different regions, i.e. defining subsets of pixels in the image that share a common property, such as pixels depicting the same object.

One goal of segmentation in image analysis can be to identify the location of an object of interest and to separate it from the background of the scene. Taking gesture recognition as an example, one is interested in the pixels that show the visible surface of the person in front of the camera while the static background can be neglected if it has no meaning to the interpretation of the gesture. In the scope of this chapter, we will focus on exactly this problem, i.e. we want to assign all pixels in the image either to the background or to one of the persons in the field of view of the camera.

There exist two different approaches to this problem. In the simpler case, one has prior knowledge of the static background and can use this knowledge to determine deviations from the background model in the current image. However, this approach is limited to cases, where the assumption of static background holds. In the second case, where no knowledge about the scene is available beforehand, the segmentation must be based on information inherent in the image itself.

The problem of segmentation has been tackled in a number of ways for conventional 2D images (refer to Stockman and Shapiro (2001) for an overview). However, the task can become arbitrarily difficult based on the complexity of the scene. Fig-

The segmentation method using a background model was developed in collaboration with Ann-Kristin Grimm. Ann-Kristin Grimm and I contributed approximately equally to the development and implementation of the method. The histogram-based segmentation method has previously been described in part in (Haker et al., 2007, 2009a).

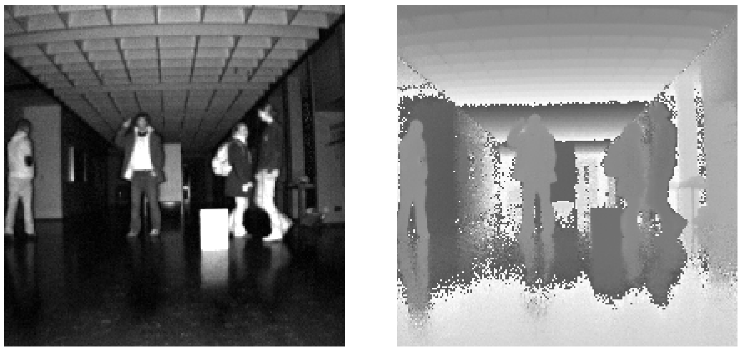


Figure 5.1: *Complex scene showing four persons in front of a cluttered background. Figure 5.1a shows the amplitude image captured with a PMD CamCube and Figure 5.1a shows the corresponding range map.*

ure 5.1 shows an example, where four persons stand in front of a complex background scene. On the basis of the intensity information a simple algorithm that operates on a per pixel basis will fail, because parts of the foreground objects have the same gray-values as the background. Here, the TOF camera facilitates the disambiguation due to the available range data, i.e. the persons appear at different distances to the camera than the background.

In the following, we will present solutions to both formulations of the problem. In Section 5.1.1, we will focus on the segmentation of the foreground in front of a known static background. Then, in Section 5.1.2, we will consider the case where we want to segment a single person standing in front of the camera before an unknown background.

5.1.1 Segmentation Based on a Background Model

In this section, we will discuss a procedure for segmenting the foreground based on an estimated model of the static background scene. The proposed procedure aims at identifying all pixels that belong to objects that differ from the static background model. Since we assume the background to be static, we can estimate the model beforehand.

The model is estimated by averaging over a fixed number of frames depicting the background scene. This is done for both amplitude and range data and we obtain the following figures for each pixel:

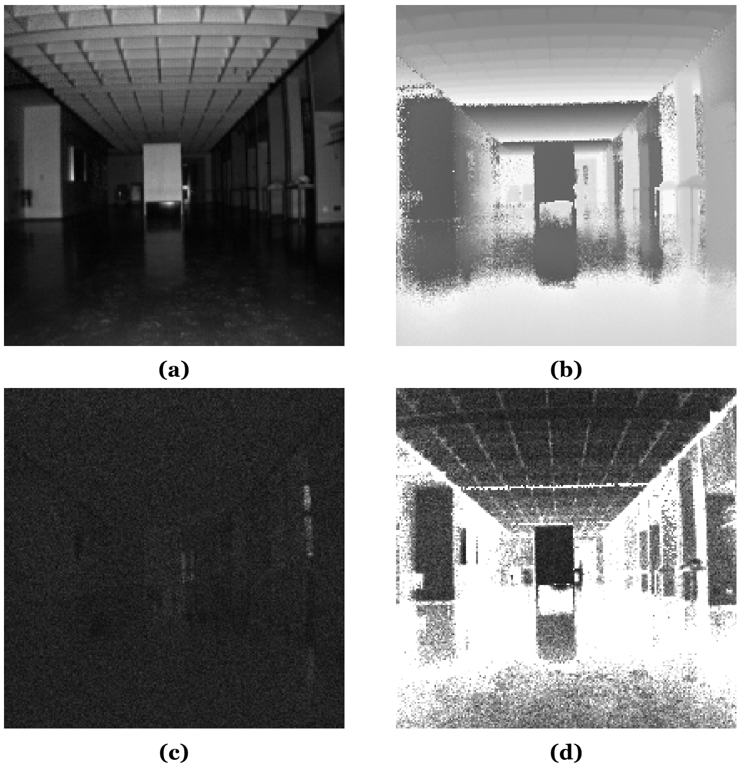


Figure 5.2: *Background model of the scene shown in Figure 5.1. The model was estimated over 20 frames of the empty scene captured with a PMD CamCube. Figure 5.2a shows the amplitude image and Figure 5.2b the range data. The corresponding standard deviations are given in Figure 5.2c and Figure 5.2d, respectively.*

$$\mu_{amp}(i,j) = E[x_{amp}(i,j)] \tag{5.1}$$

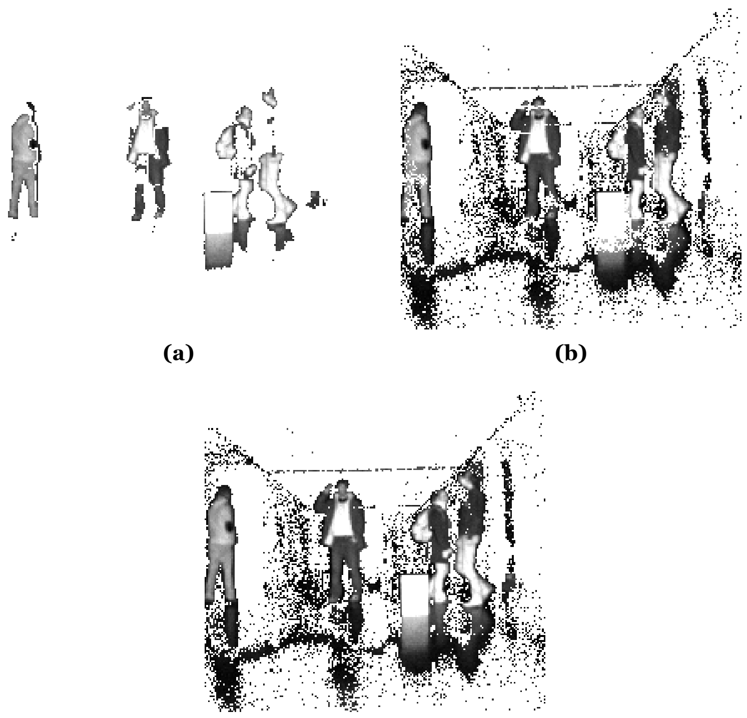
$$\mu_{rng}(i,j) = E[x_{rng}(i,j)] \tag{5.2}$$

$$\sigma_{amp}(i,j) = \sqrt{E[(x_{amp}(i,j) - \mu_{amp}(i,j))^2]} \tag{5.3}$$

$$\sigma_{rng}(i,j) = \sqrt{E[(x_{rng}(i,j) - \mu_{rng}(i,j))^2]} \tag{5.4}$$

Here, $\mu_{amp}(i,j)$ and $\mu_{rng}(i,j)$ refer to the mean values of amplitude and range while $\sigma_{amp}(i,j)$ and $\sigma_{rng}(i,j)$ denote the corresponding standard deviations. These values, estimated for the background of Figure 5.1, are presented in Figure 5.2.

Using these figures, we determine if a pixel $x(i,j)$ conforms with the background model using the following condition:



(a) (b) (c)

Figure 5.3: *Segmentation result using the rule for classifying background pixels given in Equation (5.5). Figure 5.3a uses only a threshold on the amplitude data. In analogy, a threshold is applied only to the range data in Figure 5.3b. The combination of amplitude and range as specified by Equation (5.5) is depicted in Figure 5.3a.*

$$\begin{aligned} \sqrt{(x_{amp}(i,j) - \mu_{amp}(i,j))^2} &< \theta_{amp} \cdot \sigma_{amp}(i,j) \ \wedge \\ \sqrt{(x_{rng}(i,j) - \mu_{rng}(i,j))^2} &< \theta_{rng} \cdot \sigma_{rng}(i,j) \end{aligned} \qquad (5.5)$$

Thus, a pixel is considered a foreground pixel if it deviates from the background either with respect to amplitude or range given the estimated noise level at that pixel. Here, θ_{amp} and θ_{rng} constitute parameters that can be used to control the sensitivity of the thresholding. The outcome of this approach is given in Figure 5.3 where we chose the parameter settings $\theta_{amp} = 0.05$ and $\theta_{rng} = 0.9$.

While the method performs well in image regions with low measurement noise, it yields random decisions in those areas of the image that are completely dominated by noise. These regions occur where the background cannot be sufficiently illuminated by the infrared LEDs of the TOF camera. Because the range data tends to take arbitrary values for those regions, background pixels often deviate significantly from the mean value $\mu_{rng}(i,j)$ and are thus classified as foreground.

This effect can only be tackled by ruling out those pixels with low confidence in the range measurement. In the case of the background model this can be achieved via the standard deviation of the range values. On the contrary, the standard deviation cannot be estimated for a single frame that we intend to segment. However, a good approximation can be achieved by thresholding the amplitude data, as it is related to the signal-to-noise ratio (see Chapter 2). Care has to be taken that this approach does not remove foreground objects that have low reflectivity or are at a large distance from the camera and thus appear with low intensity in the amplitude image.

Taking these considerations into account, we extend Equation (5.5) and obtain a rule that classifies a pixel as background if the following conditions are met:

$$\begin{array}{llll} \Big(\sqrt{(x_{amp}(i,j)-\mu_{amp}(i,j))^2} & < \theta_{amp} & \cdot \; \sigma_{amp}(i,j) & \wedge \\ \sqrt{(x_{rng}(i,j)-\mu_{rng}(i,j))^2} & < \theta_{rng} & \cdot \; \sigma_{rng}(i,j)\Big) & \vee \\ x_{amp}(i,j) & < \epsilon_{amp} & & \end{array} \tag{5.6}$$

Here, ϵ_{amp} denotes the threshold for the amplitude data, i.e. pixels with low confidence in the range measurement are automatically considered as background. Applying Equation (5.6) to the data of Figure 5.3 yields a significant improvement reducing the artefacts in the segmentation result, as Figure 5.4a demonstrates. Remaining false positives can easily treated by removing all connected components that are not sufficiently large to compose a person. The final outcome of the proposed segmentation procedure is depicted in Figure 5.4b.

5.1.2 Histogram-based Segmentation

In this section, we present a method to segment the person closest to the camera from the remainder of the scene. Following this approach we intend to devise systems for gesture-based single-user interaction using a TOF camera, as described in Chapter 6, Section 7.4, and Chapter 10. The proposed algorithm does not rely on a background model and can thus be used for any scene that meets certain contraints without prior calibration. The algorithm uses combined information from both the range and intensity data of the TOF camera. Previous work of Haker et al. (2007, 2009b) has already shown that the combined use of both range and intensity data can significantly improve results in a number of different computer vision tasks.

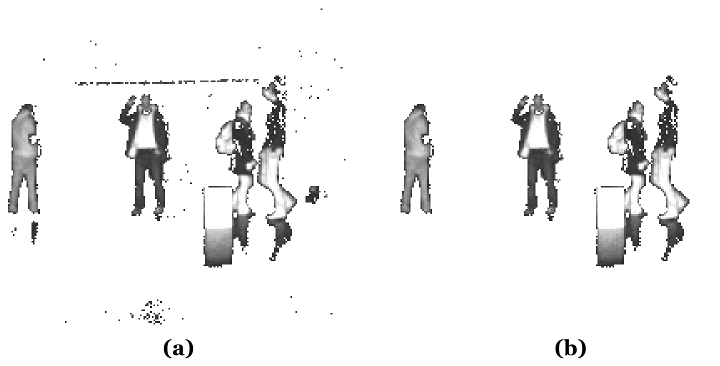


Figure 5.4: *The segmentation result using the rule for classifying background pixels given in Equation (5.6) is shown in Figure 5.4a. To remove remaining noise pixels, all connected components smaller than 100 pixels have been removed in Figure 5.4b.*

The proposed algorithm operates on a per pixel basis and determines adaptive thresholds for range and intensity based on histograms. These thresholds are then applied to decide for each pixel if it belongs to the foreground or background.

In case of the intensity data, the threshold discards dark pixels. This has two effects: Firstly, the amount of light that is reflected back into the camera decays proportionally to the squared distance of the object from the camera, thus the background generally appears significantly darker than foreground objects that are close to the camera. Secondly, this procedure discards unreliable pixels from the range measurement, because the intensity can be considered a confidence measure for the depth estimation as it is related to the signal-to-noise ratio (see Chapter 2). The value of the threshold is determined adaptively for each frame to increase the robustness of the procedure. For example, a lower threshold is required if the person is at a larger distance to the camera, whereas the person appears brighter in the image if it is closer to the camera. We employ the method of Otsu (1979), which assumes a bimodal distribution in the histogram of grayvalues. This simple assumption is usually sufficient because the amplitude decays quickly with increasing distance due to the above mentioned quadratic relation between distance and amplitude.

Figure 5.5 shows the amplitude image of a person in a cluttered scene. In addition we present the histogram of grayvalues and the estimated Otsu threshold. Note here, that we multiply the resulting Otsu threshold by a constant factor of 0.6 to allow more pixels to pass as foreground. This avoids that clothing with low reflectivity properties in the infrared spectrum are assigned to the background. Figure 5.5 also depicts the result of the segmentation when only the amplitude threshold is used. Note, that the result is not very accurate and many background pixels are still clas-

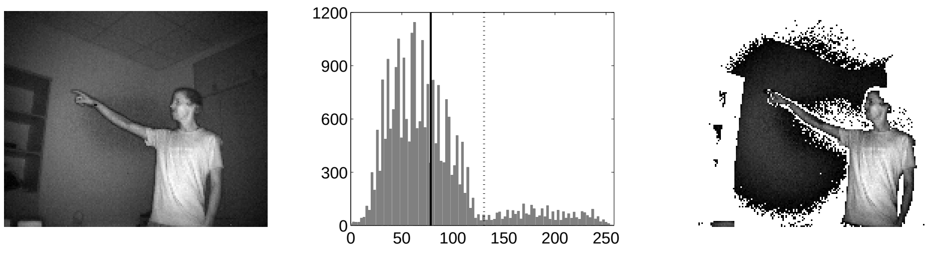


Figure 5.5: *Segmentation based on amplitude thresholding.* **Left:** *Amplitude image.* **Center:** *Histogram of amplitudes and estimated Otsu threshold (dotted line). Multiplication with a factor of 0.6 yields final threshold (solid line).* **Right:** *Resulting segmentation of pixels exceeding the threshold.*

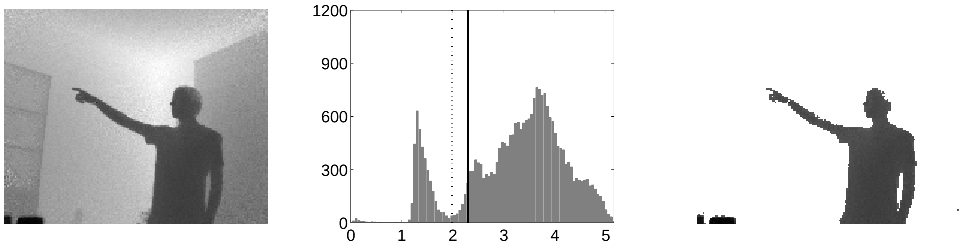


Figure 5.6: *Segmentation based on range thresholding.* **Left:** *Range image.* **Center:** *Histogram of range values and estimated threshold at minimum after first peak (dotted line). A shift by 0.3 meters yields final threshold (solid line).* **Right:** *Resulting segmentation of pixels that fall below the threshold.*

sified as foreground due to their bright appearance in the image.

In case of the range data, peaks in the histogram can be assumed to correspond to objects at different distances in front of the camera. Thus it is apparent, that the assumption of a bimodal distribution does not hold in the general case. The assumption is for example violated, if two people appear at different distances in the field of view before a constant background. In that case one would obtain three peaks, one for each person and a third for the background. To find the person closest to the camera, we determine the threshold as the one that separates the peak of the closest object from the remaining range values. This is achieved by applying a simple hill-climbing algorithm to the histogram. Once the first peak is detected, we apply the hill-climbing algorithm again to determine the local minimum that follows the peak. All pixels above this threshold should belong to the object closest to the camera. Because the estimation of this threshold is often too aggressive and partly discards pixels belonging to the foreground object, we add a constant offset of 0.3 meters to obtain the final threshold. This procedure is illustrated in Figure 5.6.

The final segmented image is composed of those pixels that were classified as

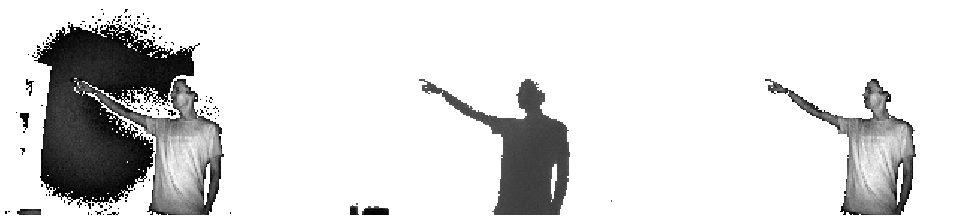


Figure 5.7: *Segmentation result using combined amplitude and range data.* **Left:** *Segmented amplitude image.* **Center:** *Segmented range map.* **Right:** *Combined segmentation result. Only pixels considered foreground in both amplitude and range data are classified as foreground pixels in the final segmentation result. Furthermore, only the largest connected component of foreground pixels is retained because we are interested in the closest foreground object.*

foreground pixels with respect to both types of data. To ensure that only a single object is considered, only the largest connected component of foreground pixels is retained, all other objects are considered background. The combined result of range- and amplitude-based thresholding is given in Figure 5.7.

A drawback of this simple approach is that the algorithm cannot distinguish between two people who stand side by side at the same distance from the camera. However, there exist applications (see Part III) in which one can assume that a single person stands close to the camera and in those cases the proposed procedure yields robust results at very low computational cost.

5.1.3 Summary

In this chapter we proposed two segmentation algorithms that aim at identifying connected components of pixels depicting a person in the field of view of the camera. The first algorithm uses a model of the background to determine foreground objects as deviations from this model. While this approach operates very robustly it has the main drawback that the method fails once the camera is moved or the static background changes. For scenarios in which this is likely to happen or when there are no means to acquire a model of the background, we proposed a second algorithm that does not require prior calibration. Instead, the foreground object is determined by applying adaptive thresholds to both the range and amplitude data. In both cases, we have not addressed the problem of telling multiple persons in the image apart.

An extension to the presented methods, which enables the segmentation and tracking of multiple persons in the field of view of the camera, was proposed by Hansen et al. (2008). The main idea is to project the 3D points corresponding to the

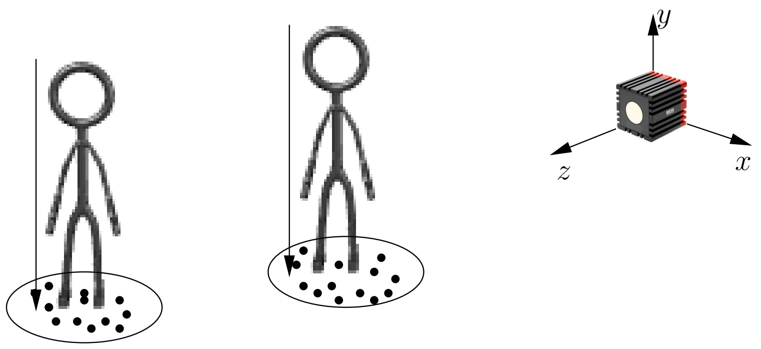


Figure 5.8: *An illustration of the procedure to identify and segment multiple persons standing in the field of view of a TOF camera. After inverting the perspective projection of the camera, the 3D points are projected onto the (x, z)-plane, where persons can be identified through a clustering algorithm.*

foreground objects onto a plane representing the floor. Thus, one will obtain clusters of points in areas where a person is standing. Two people standing in front of each other will form two clusters which appear at different distances on a line extending from the position of the camera. This situation is depicted in Figure 5.8.

Hansen et al. (2008) propose the use of an Expectation Maximization (EM) algorithm (Dempster et al., 1977) to identify the resulting clusters. An alternative is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm proposed by Ester et al. (1996). A major advantage of the DBSCAN algorithm is that it does not require the number of expected clusters as opposed to many other clustering algorithms, such as the EM algorithm and k-means (Lloyd, 1982). Hansen et al. (2008) avoid the problem of initializing the number of clusters by introducing a scheme that uses the clusters detected in the previous frame to create, delete, or merge existing clusters as required.

6 Pose Estimation

6.1 Introduction

In this section, we present a technique for estimating human pose in 3D based on a simple model of the human body. The model consists of a number of vertices that are connected by edges such that the resulting graph structure resembles the anatomy of the human body, i.e. the model represents the torso, the head, and the arms. The model is updated using an iterative learning rule common to self-organizing maps (SOMs) as proposed by Kohonen (1995). The position of certain body parts, such as the hands, can be obtained from the model as the 3D coordinates of the corresponding vertices, i.e. the position of the hands in 3D corresponds to the position of the vertex that terminates the chain representing an arm. Thus, body parts can be tracked in 3D space.

The estimation of 3D human pose has been addressed in a number of different publications. The majority of work focuses on the estimation of pose from single images taken with a regular 2D camera, and a number of different algorithmic approaches have been presented. In the work of Agarwal and Triggs (2006) the pose is recovered from shape descriptors of image silhouettes. Rosales and Sclaroff (2000) map low-level visual features of the segmented body shape to a number of body configurations and identify the pose as the one corresponding to the most likely body configuration given the visual features. An approach based on a large database of example images was presented by Shakhnarovich et al. (2003). The authors learn a set of parameter-sensitive hashing functions to retrieve the best match from the

Parts of this chapter are joint work with others. The devised idea of fitting a SOM to the data cloud in 3D to estimate the human pose and implemented the algorithm. Martin Böhme and I contributed approximately equally to evaluating the accuracy of the method. The work decried here has previously been published in (Haker et al., 2009a).

database in an efficient way. Rehg et al. (2003) describe a method for robustly tracking the human pose in single camera video sequences. They use both 2- and 3-dimensional models to overcome disambiguities when the body motion is directed along the viewing axis of the camera.

Human pose estimation using a multi-camera setup was proposed by Cheung et al. (2003). The motion, shape, and joints of an articulated model of the human body are recovered from a set of images based on the shape-from-silhouette method.

Very accurate 3D reconstruction of human motion from multi-view video sequences was published by Gall et al. (2010). Based on a segmentation of the subject, the authors use a multi-layer framework that combines stochastic optimization, filtering, and local optimization to estimate the pose using a detailed model of the human body. However, the computational cost is relatively high and the system does not operate at camera frame rates.

Pose estimation based on 3D data has been addressed by Weik and Liedtke (2001). The 3D volume of a person is estimated in a multi-camera setup using the shape-from-silhouette method. A skeleton model is then fit to a 2D projection of the volumetric data. The 2D projection is obtained by a virtual camera and the model is fit using certain features of the outer contour. The 3D coordinates of the model are finally reconstructed by inverting the 2D projection of the virtual camera, i.e. the vertices of the skeleton are projected back into 3D space using the intrinsic parameters of the virtual camera.

Another approach to obtaining a skeleton in 3D is to apply a thinning to volumetric data directly in 3D space (Palágyi and Kuba, 1999; Pudney, 1998). The human pose can then be estimated from the skeleton (Arata et al., 2006).

Two related methods based on stereo imaging were presented by Yang and Lee (2006) and Yang and Lee (2007). The authors introduce a hierarchical human body model database. For a given image the algorithm uses both silhouette and depth information to identify the model pose with the best match.

The work of Knoop et al. (2009) fuses 2D and 3D information obtained from a stereo rig and a TOF camera to fit a human body model composed of generalized cylinders. The system models body joints and uses kinematic constraints to reduce the degrees of freedom. The 3D data is obtained using a TOF camera and the system runs at frame rates of 10–14 frames per second.

Another recent approach using TOF cameras was presented by Zhu et al. (2008). The method tracks a number of anatomical landmarks in 3D over time and uses these to estimate the pose of an articulated human model. The model is in turn used to resolve disambiguities of the landmark detector and to provide estimates for undetected landmarks. The entire approach is very detailed and models constraints such as joint limit avoidance and self-penetration avoidance. Despite its complexity, the method runs at a frame rate of approximately 10 frames per second.

Our approach, in contrast, is a very simple one that demonstrates how effectively TOF cameras can be used to solve relatively complex computer vision tasks. A general advantage of TOF cameras is that they can provide both range and intensity images at high frame rates. The combined use of both types of data was already used for tracking (Böhme et al., 2008a; Haker et al., 2009a) and allows a robust segmentation of the human body in front of the camera. The range data, representing a $2\frac{1}{2}$D image, can then be used to obtain a point cloud in 3D representing the visible surface of the person. Thus, limbs extended towards the camera can still be easily identified while this proves to be a more difficult task in 2D projections of a scene.

Our approach takes advantage of this property and fits a simple model of the human body into the resulting point cloud in 3D. The model fitting algorithm is based on a SOM, can be implemented in a few lines of code, and the method runs at camera frame rates up to 25 frames per second on a 2 GHz Intel Core 2 Duo. The algorithmic approach to this procedure is discussed in Section 6.2. The method delivers a robust estimation of the human pose, as we show in Section 6.3 for image data that was acquired using a MESA SR4000 TOF camera.

6.2 Method

The first step of the proposed procedure is to segment the human body from the background of the image. We employ the histogram-based segmentation procedure described in Section 5.1.2. The method adaptively determines thresholds for both the range and intensity data. The final segmented image is obtained as the one where the foreground pixels have been classified as foreground pixels with respect to both types of data. Furthermore, the largest connected component of foreground pixels is identified and all remaining pixels are classified as background. Thus, we obtain a clear segmentation of a single person closest to the camera in most cases. A sample TOF image and the resulting segmented image is shown in Figure 6.1.

The identified foreground pixels can be assumed to sample the visible surface of the person in front of the camera. Since the intrinsic parameters of the camera, such as focal length and pixel size, are known, the surface pixels can be projected back into 3D space, i.e. one can invert the perspective projection of the camera. As a result

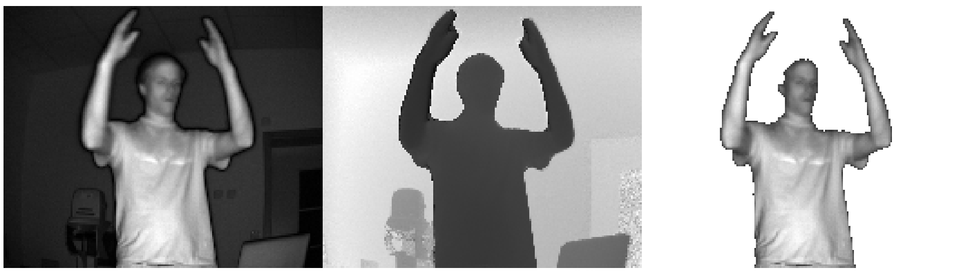


Figure 6.1: *Sample image taken with a MESA SR4000 TOF camera. The leftmost image shows the amplitude data. The range image is given in the center and the resulting segmentation is shown on the right.*

one obtains a point cloud in 3D that represents the 3-dimensional appearance of the person. This approach has two major advantages: (i) The representation is scale-invariant due to the fact that the size of the person in 3D space remains the same independently of the size of its image; (ii) body parts that are extended towards the camera in front of the torso can be easily identified due to the variation in distance, whereas this information is lost in 2D projections of the scene obtained with regular cameras.

Our method aims at fitting a simple graph model representing the anatomy of the human body into the resulting point cloud in 3D. To this end, we employ a SOM. We define a graph structure of vertices and edges that resembles a frontal view of the human body. Body parts, such as arms and torso, are modeled by explicitly defining the neighborhood structure of the graph, i.e. an arm is represented by a simple chain of pairwise connected vertices whereas vertices in the torso are connected to up to four neighbors forming a 2D grid. The resulting model structure is depicted in Figure 6.2.

The SOM is updated by an iterative learning rule for each consecutive frame of the video sequence. The first frame uses the body posture depicted in Figure 6.2 as an initialization of the model. During initialization the model is translated to the center of gravity of the 3D point cloud. The scale of the model is currently set manually to a fixed value that corresponds to an average-sized person. We can report that the scale is not a particularly critical parameter and that the same fixed scale works for adults of different height. Once the scale is set to an appropriate value, there is no need to adjust it during run-time due to the above mentioned scale-invariance of the method. The update of the model for each consecutive frame then depends on the model that was estimated for the previous frame.

The adaptation of the model to a new frame involves a complete training of the

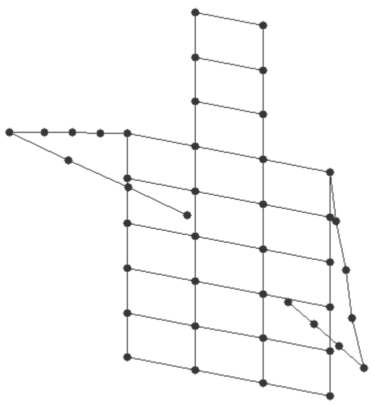


Figure 6.2: *Graph model of the human body. The edges define the neighborhood structure for the SOM.*

SOM, i.e. a pattern-by-pattern learning is performed using the data points of the 3D point cloud. This iterative procedure selects a sample vector $\vec{x}$ from the point cloud at random and updates the model according to the following learning rule:

$$\hat{\vec{v}}^{t+1} = \hat{\vec{v}}^{t} + \hat{\epsilon}^{t} \cdot (\vec{x} - \hat{\vec{v}}^{t}) \quad (6.1)$$
$$\tilde{\vec{v}}^{t+1} = \tilde{\vec{v}}^{t} + \tilde{\epsilon}^{t} \cdot (\vec{x} - \tilde{\vec{v}}^{t}). \quad (6.2)$$

Here, $\hat{\vec{v}}$ denotes the node that is closest to the sample $\vec{x}$ with respect to the distance measure $d(\vec{x}, \vec{v}) = \|\vec{x} - \vec{v}\|_2$. The nodes $\tilde{\vec{v}}$ are the neighbors of $\hat{\vec{v}}$ as defined by the model structure. The learning rates are denoted by $\hat{\epsilon}^t$ and $\tilde{\epsilon}^t$ for the closest node and its neighbors, respectively. The learning rate $\hat{\epsilon}^t$ was set to:

$$\hat{\epsilon}^t = \epsilon_i \cdot (\epsilon_f/\epsilon_i)^{t/t_{\text{max}}}. \quad (6.3)$$

Here, $t \in \{0, \ldots, t_{\text{max}}\}$ denotes the current adaptation step for this frame and t_{max} denotes the total number of adaptation steps performed for this frame. The initial learning rate ϵ_i and the final learning rate ϵ_f were set to 0.1 and 0.05. The learning rate for the neighbors was chosen to be $\tilde{\epsilon}^t = \hat{\epsilon}^t/2$. This choice of the learning rate was already proposed in previous work on self-organizing networks (Martinetz and Schulten, 1991). The initial and final learning rates were set to relatively high values in order to allow the network to handle fast movements of the person, i.e. if the limbs are moved quickly the correctional updates for the corresponding nodes have to be large so that the model can accurately follow.

This update rule does not always guarantee that the topology of the model is preserved. Here, we refer to topology with respect to the connectivity of the nodes within body parts such as the arm. Imagine the situation where the subject's hands touch in front of the torso. If the hands are separated again, it is possible that the model uses the last node of the left arm to represent samples that actually belong to the hand of the right arm. It can thus happen, that the last node of the left arm may continue to be attracted by the right hand although both hands have moved apart and, thus, the left arm will extend into empty space. In principle, the update rules resolve this problem over time. However, only a small number of updates are performed per frame and this may lead to a wrong estimation of the topology for a small number of frames.

To avoid this, we developed a modification of the update rule that speeds up the learning process by forcing neighboring vertices to stay close together. This is achieved by the following rule that is applied after the actual learning step if the distance $d(\hat{\vec{v}}, \tilde{\vec{v}}_a)$ exceeds a certain threshold θ:

$$\hat{\vec{v}} = \tilde{\vec{v}}_a + \theta \cdot \frac{(\hat{\vec{v}} - \tilde{\vec{v}}_a)}{\|\hat{\vec{v}} - \tilde{\vec{v}}_a\|_2}. \tag{6.4}$$

Here, $\tilde{\vec{v}}_a$ is a specific neighbor of $\hat{\vec{v}}$ referred to as an anchor. The rule enforces that the distance between the vertex $\hat{\vec{v}}$ and its anchor is always less than or equal to θ. The threshold θ depends on the scale of the model. The anchor of each vertex is defined as the neighbor that has minimal distance to the center of the torso with respect to the graph structure of the model, i.e. it is the vertex that is connected to the center of the torso by the smallest number of edges.

6.3 Results

6.3.1 Qualitative Evaluation

The proposed method was evaluated using a MESA SR4000 TOF camera. We operate the camera at a modulation frequency of 30 MHz for the active illumination. As a result the camera can disambiguate distances in the range of up to 5 meters. In the following sample images, the person was at a distance of roughly 2.5 meters from the camera. At that distance the range measurement has an accuracy of approximately 1 cm.

A sample result of the pose estimation is shown in Figure 6.3. The figure depicts the point cloud of samples in 3D that represent the visual surface of the person in

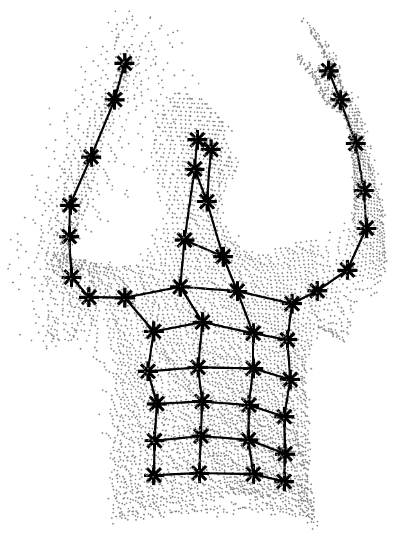


Figure 6.3: *Point cloud sampling the visible surface of a human upper torso in 3D. The graph represents the human model that was fitted to the data.*

front of the camera shown in Figure 6.1. The model that was fitted to the point cloud is imprinted into the data. One can observe that the model captures the anatomy of the person correctly, i.e. the torso is well covered by the 2-dimensional grid, a number of vertices extend into the head, and the 1-dimensional chains of vertices follow the arms. Thus, the position of the major body parts, such as the hands, can be taken directly from the corresponding vertices of the model in 3D.

The data from Figure 6.3 is taken from a sequence of images. Further sample images from this sequence are given in Figure 6.4. Each image shows the segmented amplitude image with the imprinted 2D projection of model. One can observe that the model follows the movement of the arms accurately, even in difficult situations where the arms cross closely in front of the torso. Note that the procedure does not lose the position of the head even though it is occluded to a large extent in some of the frames. The sample images are taken from a video sequence, which is available under http://www.artts.eu/demonstrations/.

It is important to point out that the method may misinterpret the pose. This can for example be the case if the arms come too close to the torso. In such a case the SOM cannot distinguish between points of the arm and the torso within the 3D point cloud. We can report, however, that in most cases the method can recover the true configuration within a few frames once the arms are extended again.

We assume that it is possible to detect and avoid such problems by imposing a number of constraints on the model, e.g. that the arms may only bend at the elbows

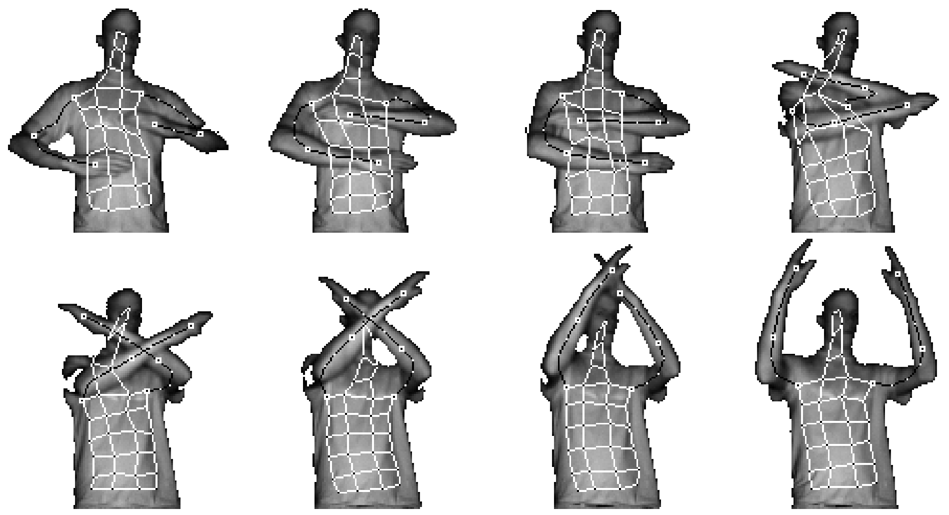


Figure 6.4: *A selection of frames from a video sequence showing a gesture. The model estimated by the pose estimation is imprinted in each frame. The edges belonging to torso and head are colored in white, whereas the arms are colored in black.*

and that the entire model should generally be oriented such that the head is pointing upwards. However, note that the current results were achieved without any such constraints.

6.3.2 Quantitative Evaluation of Tracking Accuracy

To evaluate the accuracy of the tracking quantitatively, we acquired sequences of 5 persons moving in front of the camera; each sequence was around 140 to 200 frames long. In each frame, we hand-labeled the positions of five parts of the body: the head, the shoulders, and the hands. To obtain three-dimensional ground-truth data, we looked up the distance of each labeled point in the range map and used this to compute the position of the point in space. This implies that the sequences could not contain poses where any of the body parts were occluded; however, many poses that are challenging to track, such as crossing the arms in front of the body, were still possible, and we included such poses in the sequences. (Note that the tracker itself can track poses where, for example, the head is occluded; see Figure 6.4.)

The labeled positions can now be compared with the positions of the corresponding nodes in the tracked model. However, when assessing the accuracy of the tracking in this way, we run into the problem that we never define explicitly which part of the body each node should track. For example, though the last node in each of the arms will typically be located on or near the hand, we do not know in advance exactly which part of the hand the node will track. This means that there may be a systematic offset between the position that is labeled as "hand" and the position that the hand node tracks. To give a realistic impression of tracking accuracy, we should eliminate these systematic offsets.

We do this by measuring the average offset between the tracked position and the labeled position on ten "training" frames; this offset is then used to correct the tracked position in the remaining "test" frames, on which the accuracy is measured. Because the orientation of the respective parts of the body can change, we need to measure the offsets not in the world coordinate system but in a local coordinate system. For the head and shoulders, we use a coordinate system where the x-axis points from the left shoulder (of the tracked model) to the right shoulder, the y-axis is defined so that the head lies in the x-y-plane, and the z-axis is perpendicular to the other two axes to form a right-handed coordinate system. For the hands, it is not as easy to define a full coordinate system because the model only measures the direction in which the forearm is pointing but not the orientation of the hand. For this reason, we estimate and correct the offset between tracked and labeled position only along the direction of the forearm, which we define by the last two nodes in the arm; this is the direction that accounts for most of the offset. Any offset perpendicular to the direction of the forearm is not corrected.

Once the tracked positions have been corrected in this way, we can measure the tracking error. Figure 6.5 shows a plot of tracking error over time for one of the recorded sequences. It is obvious that there is little to no systematic error remaining; instead, most of the error is due to tracking noise.

Table 6.1 shows the root mean square (RMS) tracking error, averaged over all frames and subjects. The average error is around 5 to 6 cm for the hands and shoulders and around 2 cm for the head. While this degree of accuracy is not sufficient for tracking very fine movements, it is more than adequate for determining overall body posture and for recognizing macroscopic gestures. Also, consider that no smoothing of the tracked positions over time was carried out.

A major advantage of the proposed method is that the training of the model converges very fast for each new frame. Thus, only a small number of the samples of the 3D cloud need actually be considered during the update even when the person performs very fast movements in front of the camera. The sample image in Figure 6.1 contains roughly 6500 foreground pixels. However, we use only 10% of these samples for updating the model, i.e. we select roughly 650 points in 3D in random order from the point cloud and use these for updating the model by pattern-by-pattern

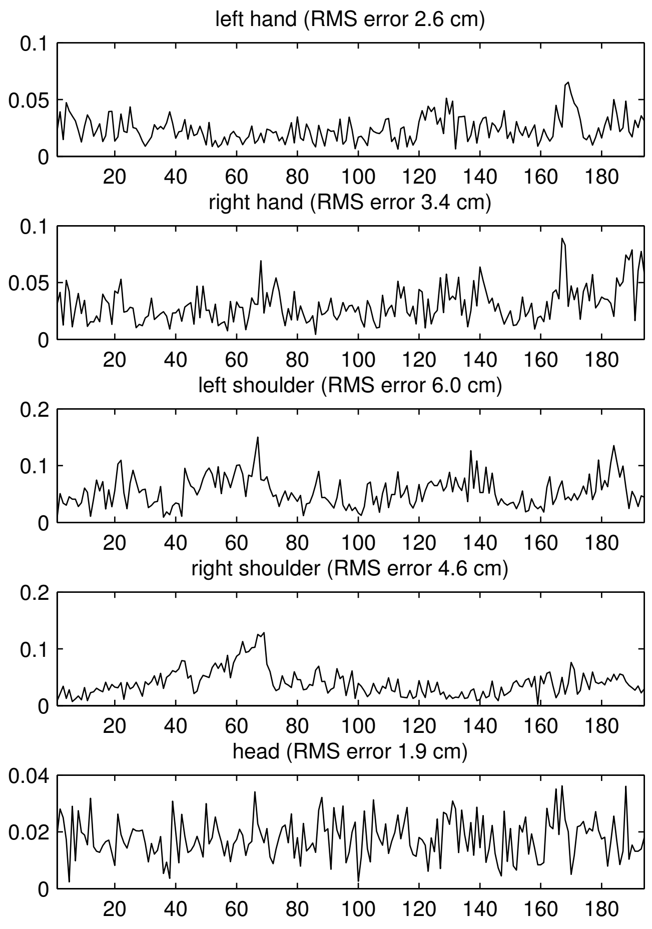


Figure 6.5: *Plots of tracking error over time for one of the sequences. The horizontal axis plots frame number, the vertical axis plots tracking error in meters.*

body part	RMS error
left hand	5.90 cm
right hand	5.29 cm
left shoulder	5.32 cm
right shoulder	5.15 cm
head	2.21 cm

Table 6.1: *Root mean square (RMS) error between tracked and labeled positions, averaged over all frames and subjects.*

learning. As a result the computational complexity is very low, and we achieve frame rates up to 25 frames per second on a 2 GHz PC while robustly tracking the human pose in scenarios such as the one depicted in Figure 6.4. The use of a higher number of samples for training will further increase the robustness while at the same time the frame rate will decrease.

6.4 Discussion

We have presented a simple procedure to estimate human pose from a sequence of range images. The procedure is especially suitable for TOF cameras as they can deliver range data in combination with intensity images at high frame rates. These cameras can be assumed to be available at relatively low costs in the near future.

The use of a SOM results in a very simple, yet very efficient implementation. In principle the procedure can be extended easily to any other kind of deformable object.

A major shortcoming of the current implementation is that the method cannot deal with multiple persons in front of the camera, i.e. the system always assumes that the segmented foreground pixels correspond to a single person. This approach fails for example if two people are at the same distance in front of the camera and very close together. In that case the segmented foreground pixels sample the visual surface of both persons. Since the SOM attempts to represent all samples equally the resulting pose estimation fails. Using the current approach, this problem must be solved by an improved method for segmentation that can handle multiple objects in front of the camera. Then, a SOM can be trained for each segmented object and thus multiple people can be tracked.

This in turn can lead to a related problem that occurs when the segmentation fails to detect parts of the body due to occlusion, e.g. when the lower part of an arm is occluded by a second person. In that case the SOM will use the entire chain of arm nodes to represent the upper part of the arm. Thus, the node for the hand will be misplaced. To tackle this problem the system needs to identify the presence of certain body parts based on pose estimates from previous frames. In case occluded body parts have been identified, the corresponding nodes of the SOM must be excluded from training. Instead their location could be predicted based on the posture of the remaining model. These two issues need to be addressed in future work.

There exist other approaches that compute a more accurate estimate of human pose but our goal within this work was to develop a simple method that gives a rough but robust estimate of human pose at high frame rates.

We use the proposed pose estimation method for action recognition and gesture-based man-machine interaction in Section 7.4. Generally, the evaluation of certain spatio-temporal features for the analysis of video sequences is computationally expensive. We argue that rough knowledge of the position of landmarks, such as the hands, can greatly improve the runtime of feature-based action recognition systems, because the features do not have to be evaluated over the entire video sequence but only at those locations where certain important landmarks have been detected. Furthermore, these features can be put into a larger context if their relative location to each other is known.

The proposed method of pose estimation using SOMs has been filed as a patent by the University of Lübeck in 2009 through the Patent- und Verwertungsagentur für die wissenschaftlichen Einrichtungen in Schleswig-Holstein GmbH. The patent is pending.

7 Features

In image processing, the data, i.e. the images, are generally represented by discrete samples of a function $f(x, y)$ where x and y denote the spatial coordinates of the image plane and $f(x, y)$ encodes properties of the imaged object. Such properties can be the object's irradiance in the form of image intensity or its distance from the image sensor in the form of a range map. In most cases, we use a regular sampling grid and x and y correspond to pixel coordinates of a digital image.

While the entirety of samples $(x, y, f(x, y))$ encodes the content of the image, one is generally interested in higher level features that repesent certain properties of the image when addressing computer vision tasks, such as object detection or image classification. An apparent reason for this is that neighboring pixels often encode redundant information as in regions of constant image intensity. Furthermore, the content of an image in terms of pixels is highly sensitive to simple image transformations, e.g. a slight translation of the camera can cause a change of $f(x, y)$ at every pixel location while the essential content of the image as an entity is only subject to a minor change.

Under this perspective, one is interested in features that encode structural information of the imaged scene while being invariant to a number of factors that may influence the imaging process. Such factors include translation, scale, and orientation, which depend on the relative position of the imaging device with respect to the scene. In this context, the features should tolerate shifts in local geometry, which arise from

Parts of this chapter are joint work with others. Martin Böhme and I contributed approximately equally to the implementation of the facial feature detector based on geometric eccentricities. It was my idea to compute the features in 3D to achieve scale-invariance. Martin Böhme proposed the use of the NFFT to obtain a fast and accurate implementation. Martin Böhme and I contributed approximately equally to implementing this concept. I implemented and evaluated the sparse coding principle with TOF image data. The computation of range flow was implemented by Tiberiu Viulet in the scope of his master's thesis (Viulet, 2009). I devised and implemented the gesture recognition framework. The work described here has previously been published in (Böhme et al., 2008a; Haker et al., 2008, 2009b).

the projection of the 3-dimensional scene to 2-dimensional image coordinates under different perspectives. Furthermore, it is desirable to achieve an invariance towards changes in illumination and contrast.

Considering the example of object detection, an image of a certain object would yield the same features even if the distance from the imaging device, viewing angle, and lighting conditions varied. Thus, the task of object detection would be invariant towards the process of imaging an object, which in turn eases the problem in many real-world scenarios.

There exist a number of approaches to object detection, that utilize the concept of invariance. Scale-invariance can be achieved by operating on an image pyramid (Koenderink, 1984), i.e. by creating scaled versions of the image and searching for the object in each level of the pyramid (Lowe, 1999). The same effect can be achieved by scaling the features (Viola and Jones, 2004; Lienhart et al., 2003). An invariance towards small translations can be obtained by applying a maximum operation over local neighborhoods (Serre et al., 2007; Labusch et al., 2008a). Rotation invariance can for example be based on rotated versions of the filters (Barczak et al., 2006; Serre et al., 2007) or by a transformation into a rotation invariant feature space (Weinland et al., 2006). Another approach is to assign location, scale, and orientation to a computed feature with respect to the image coordinates and to include this information in the detection algorithm (Lowe, 1999).

In the scope of this work we investigated four different kinds of features. Three of those features are static and computed on single image frames, while the fourth feature captures dynamic properties of an image sequence and estimates the 3D motion of objects in the scene.

The three static features are evaluated for a very simple feature-based nose detector in combined range and amplitude data obtained by a TOF camera. The image data shows frontal face images and the task is to identify the location of the nose. To find a nose in the image, the features are computed per pixel; pixels whose feature values lie inside a certain bounding box in feature space are classified as nose pixels, and all other pixels are classified as non-nose pixels. The extent of the bounding box is learned on a labeled training set. Despite its simplicity this procedure generalizes well, that is, a bounding box determined for one group of subjects accurately detects noses of other subjects. The performance of the detector is demonstrated by robustly identifying the nose of a person in a wide range of head orientations. An important result is that the combination of both range and amplitude data dramatically improves the accuracy in comparison to the use of a single type of data.

The first type of feature is related to the Gaussian curvature and is thus considered a *geometric feature*, known as the *generalized eccentricity*. As already mentioned, the feature achieves best results if it is evaluated for both range and intensity data, which is reflected in the equal error rates (EER) obtained on a database of head poses. Using only the range data, we detect noses with an EER of 66.0%. Results on the amplitude data are slightly better with an EER of 42.3%. The combination of both types of data yields a substantially improved EER of 3.1%.

The second type of feature is a generalization of the geometric features to compute *scale-invariant features* on range maps produced by a range sensor, such as a TOF camera. Scale invariance is achieved by computing the features on the reconstructed three-dimensional surface of the object. This technique is general and can be applied to a wide range of operators. Features are computed in the frequency domain; the transform from the irregularly sampled mesh to the frequency domain uses the Nonequispaced Fast Fourier Transform (NFFT). Here, we apply the procedure to the computation of the generalized eccentricities. On a dataset containing frontal faces at various distances from the camera the EER in the nose detection task is halved for the case of scale-invariant features compared to features computed on the range map in the conventional way. When the scale-invariant range features are combined with intensity features, the error rate on the test set reduces to zero.

In the case of the third type of feature the *sparse coding principle* is employed for the representation of *multimodal* image data, i.e. image intensity and range. We estimate an image basis for frontal face images taken with a TOF camera to obtain a sparse representation of facial features, such as the nose. These features are then evaluated in the nose detection task where we estimate the position of the nose by template matching and a subsequent application of appropriate thresholds that are estimated from a labeled training set. The main contribution of this work is to show that the templates can be learned simultaneously on both intensity and range data based on the sparse coding principle, and that these multimodal templates significantly outperform templates generated by averaging over a set of aligned image patches containing the facial feature of interest as well as multimodal templates computed via Principal Component Analysis (PCA). The system achieves an average EER of 3.7%.

Finally, we discuss a fourth type of feature that differs from the previously mentioned ones in the sense that it is not computed for static images but captures the motion patterns of an image sequence. In order to describe the 3D motion of objects in the scene we compute the so-called range flow which aims at estimating a 3D motion vector for each pixel of an image sequence composed of range maps. We follow an approach by Spies et al. (2000) that also exploits the available intensity data. We combine the range flow features with the estimation of human pose discussed in Chapter 6 in the way that we compute the range flow around local neighborhoods of the estimated positions of body parts. Thus, we can for example assign robust 3D motion to both hands. The motion patterns are accumulated in 3D motion histograms which we use to classify the performed gesture of each individual frame. Following this procedure, we achieve detection rates above 90% for three investigated hand gestures. We also consider the problem of detecting the case when no gesture was performed, i.e. we also define a fourth class representing arbitrary hand movements of the user which are not intended as a gesture.

In the following, we will first address the geometric features and evaluate their performance on a database of images in Section 7.1. Then, we will extend the results to obtain scale-invariant features using the NFFT in Section 7.2. The multimodal sparse features will be addressed in Section 7.3 and the features capturing 3D motion will be the subject of Section 7.4.

7.1 Geometric Invariants

7.1.1 Introduction

In this section we focus on object tracking, more precisely, on the task of nose and head tracking. The most common forms of digital images that are utilized in computer vision to solve such tasks are intensity and range images. The former type is by far the most popular, which is mainly due to the low cost of the corresponding image sensors.

However, within the last decade the TOF camera has been developed and fuses the acquisition of both intensity and range data into a single device at a relatively low cost. The future pricing of such cameras is expected to be comparable to a standard web-cam. In contrast to web-cams, the TOF camera simplifies the determination of geometrical properties of the 3D scene significantly, thus it is worth investigating methods that make explicit use of the available data.

We will discuss geometrically invariant measures that are suitable for identifying facial features in a TOF camera image. Based on these features, we construct a simple nose detector and test its performance on range and intensity data individually, as well as on the combination of these two types of data. And indeed, the detector performs better on TOF data than on intensity data alone. However, beyond this we have observed an interesting phenomenon in our results: It is not the range data by itself that improves the robustness of the results but rather the combination of range and intensity data; while both types of data perform roughly the same when used individually, the combination of the two yields substantially better results than either type of data alone. This underlines the potential of TOF cameras for machine vision applications.

Previous work has already identified the nose as an important facial feature for tracking, e.g. in (Gorodnichy, 2002) and (Yin and Basu, 2001). In the former approach the location of the nose is determined by template matching, under the assumption that the surface around the tip of the nose is a spherical Lambertian surface of constant albedo. This approach gives very robust results under fixed lighting conditions and at a fixed distance of the user from the camera. The latter approach is based on a geometrical model of the nose that is fitted to the image data.

We also consider the nose as being very well suited for head tracking, because the nose is obviously a distinctive characteristic of the human face. In terms of differential geometry, the tip of the nose is the point of maximal curvature on the object surface of a face. A benefit of analyzing the 3D surface in terms of differential geometry is that a major portion of differential geometry is concerned with the description of invariant properties of rigid objects. Although curved surface patches have been shown to be unique (Mota and Barth, 2000; Barth et al., 1993), Gaussian curvature is rarely used as a feature because its computation is based on first and second order derivatives, which are sensitive to noise. We propose alternative features that can be related to generalized differential operators. These features, which are computed per pixel of the input image, are used to decide for each input pixel if it corresponds to the tip of the nose based on the simple application of thresholds learned from a set of labeled training data.

We will first review and motivate the geometric features and evaluate them with respect to their suitability for the specific task of nose detection. Then, we will discuss the robustness of the method by presenting results on a database of head pose images acquired using a MESA SR3000 TOF camera (Oggier et al., 2005b).

7.1.2 Geometric Invariants

For the definition of invariant geometric features we will restrict ourselves to a special type of surface known as the *Monge* patch or the *2-1/2-D* image. Such surfaces are defined as a function $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^3$ in the following manner:

$$(x, y) \mapsto (x, y, f(x, y)). \tag{7.1}$$

This is the natural definition for digital intensity images, as each pixel value is bound to a fixed position on the image sensor without explicit reference to a coordinate representation in the 3D world. In the case of range data, however, each pixel is associated with explicit 3D coordinates via the geometry of the optical system of the camera. Nevertheless, we will assume a *weak-perspective* camera model for both range and amplitude data of the TOF camera, because we do not expect a great difference in range for the pixels of interest. (Within a frontal face we do not expect any range differences greater than 5 cm, which would roughly correspond to a relative shift of only 5% in the coordinates x and y at a distance of 1 meter from the camera if we assumed a *perspective* camera model.) Under the weak-perspective camera model we can treat both types of data as Monge patches with respect to the regular image grid, which results in a simplified mathematical formulation and comparable results for range and amplitude data. An approach that exploits a perspective camera model will be discussed in Section 7.2.

The features derived for the above data model are mainly due to a conceptual framework for image analysis which was introduced by Zetzsche and Barth (1990a,b). Within this framework, image regions are associated hierarchically, with respect to their information content, with 0D (planar), 1D (parabolic), and 2D (elliptic/hyperbolic) regions of a Monge patch. Naturally, the concept of curvature is fundamental to this representation. Within this framework, the authors proposed a set of measures that provide basic and reliable alternatives to the Gaussian curvature K and the mean curvature H for the purpose of surface classification.

Let us first recall the definition of Gaussian curvature for a Monge patch:

$$K = \frac{f_{xx} f_{yy} - f_{xy}^2}{(1 + f_x^2 + f_y^2)^2}. \tag{7.2}$$

In case only the sign of the curvature is relevant, one can rely on the DET-operator D, which can be formulated in terms of the determinant of the Hessian

$$(h_{ij}) = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix}. \tag{7.3}$$

This amounts to the numerator of Equation (7.2). Thus the DET-operator takes the following form:

$$D = f_{xx} f_{yy} - f_{xy}^2 = \text{det}(h_{ij}) = d_1 d_2. \tag{7.4}$$

Here, d_1 and d_2 denote the eigenvalues of the Hessian. Rearranging the first part of the formula (Barth et al., 1993) yields

$$\begin{aligned} D &= \tfrac{1}{4}(f_{xx} + f_{yy})^2 - \tfrac{1}{4}(f_{xx} - f_{yy})^2 - f_{xy}^2 \\ &= (\Delta f)^2 - \epsilon^2, \end{aligned} \tag{7.5}$$

where Δf denotes the Laplacian and ϵ is referred to as the eccentricity, which is defined as

$$\epsilon^2 = \frac{1}{4}(f_{xx} - f_{yy})^2 + f_{xy}^2. \tag{7.6}$$

The above formulation yields a relationship of the curvature to the Laplacian and the eccentricity. A generalized representation of the operators Δf and ϵ can be achieved in the Fourier domain by defining the generalized eccentricity ϵ_n via the following filter functions in polar coordinates ρ and θ, where $A(\rho)$ represents the radial filter

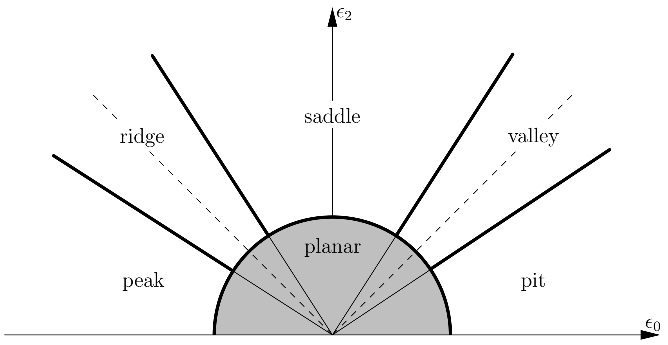


Figure 7.1: *Discrimination of the six surface types* pit, peak, saddle, valley, ridge, *and* planar *within the feature space spanned by* $\epsilon_0 \ (\Delta f)$ *and* $\epsilon_2 \ (\epsilon)$.

tuning function:

$$\begin{aligned} C_n &= \mathrm{i}^n A(\rho)\cos(n\theta), \\ S_n &= \mathrm{i}^n A(\rho)\sin(n\theta). \end{aligned} \tag{7.7}$$

Recall that the transfer functions of partial derivatives are of the form $(\mathrm{i}f_x)^n$ and $(\mathrm{i}f_y)^n$, where f_x and f_y represent the spatial frequencies and n denotes the order of differentiation. Even-order partial derivatives correspond to real transfer functions, whereas odd-order partial derivatives correspond to imaginary transfer functions.

The two transfer functions in Equation (7.7) correspond to convolution kernels $c_n(x,y)$ and $s_n(x,y)$ in the image domain. Using these, we obtain the generalized eccentricity

$$\epsilon_n^2 = (c_n(x,y) * l(x,y))^2 + (s_n(x,y) * l(x,y))^2 \tag{7.8}$$

for $n = 0, 1, 2, \ldots$, which corresponds to $|\Delta f|$ for $n = 0$ and to the eccentricity ϵ for $n = 2$, when $A(\rho) = (2\pi\rho)^2$. The gradient is defined by ϵ_n for $n = 1$ and $A(\rho) = 2\pi\rho$. In a purely geometrical interpretation, all measures ϵ_n are positive, and as a result one cannot distinguish between convex and concave curvature using ϵ_0 and ϵ_2. An extension to this formulation in (Barth et al., 1993) justifies the use of ϵ_0 with the sign of Δf, i.e. $\epsilon_0 = -c_0 * l$.

For practical applications, the radial filter tuning function $A(\rho)$ can be combined with a low-pass filter, e.g. Gaussian blurring of the form $G(\rho,\sigma) = \exp(-\pi\rho^2/4\sigma^2)$. Ideally, the low-pass filter should be adapted to the distribution of noise inherent in the data.

The measures ϵ_n for $n = 0$ and $n = 2$ can be used to distinguish between the six well-known surface types in the feature space spanned by ϵ_0 and ϵ_2. Figure 7.1

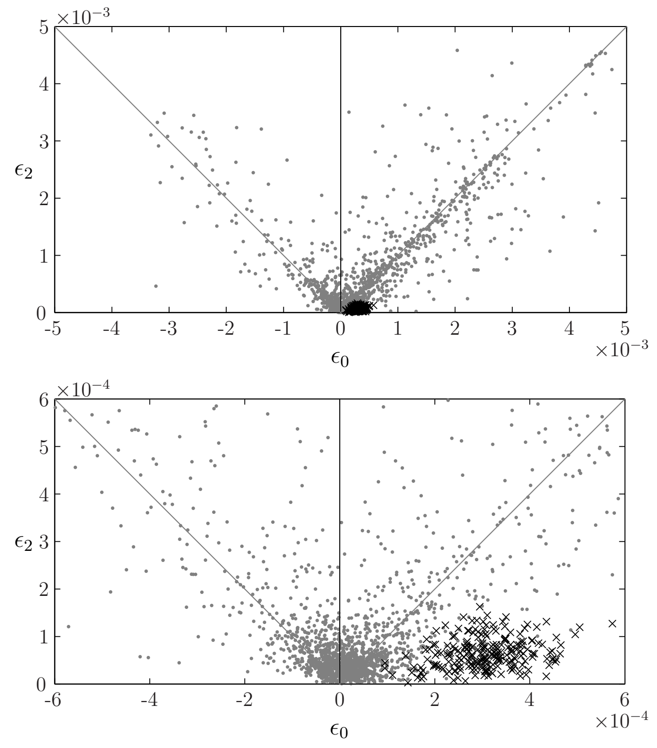


Figure 7.2: (Top) *Distribution of feature points for pixels taken from range data of the SR3000 camera projected into the 2D feature space spanned by* ϵ_0 (Δf) *and* ϵ_2 (ϵ). **(Bottom)** *The feature space around the origin at a higher resolution. The black crosses represent feature points corresponding to the nose tip of various subjects and clearly cluster in the region associated with the surface type* pit *as expected. The grey dots represent randomly chosen non-nose pixels.*

shows where the different surface types lie in feature space. Because the nose is a local minimum in the range data, we would expect the corresponding pixels to lie in the region labeled *pit*. Conversely, since the nose tends to be a local maximum in the intensity data, we would expect to find the corresponding pixels in the region labeled *peak*.

7.1.3 Feature Selection

The interpretation of Figure 7.1 not only demonstrates how the features ϵ_n can be interpreted intuitively, but it also shows how the interpretation can be used to select meaningful features for the task at hand. In other words, the goal of feature extraction should be that all data points belonging to the same class, e.g. pixels corresponding to the tip of the nose, form clusters in the feature space while being easily separable from points of other classes.

The top plot in Figure 7.2 displays feature points for pixels computed on range data in the feature space spanned by ϵ_0 and ϵ_2. Only pixels with high amplitude, i.e. pixels belonging to an object that is close to the camera, were considered. First, it is noticeable that the frequency of occurrence is ordered with respect to 0D structures around the origin, 1D structures along the diagonals, and 2D structures. Thus, only a small portion of pixels belongs to curved regions. However, in the bottom plot in Figure 7.2, one can observe that the feature points associated with the tip of the nose cluster nicely and correspond to the surface type *pit* as one would expect. Qualitatively similar results can be observed on the amplitude data, the major difference being that nose pixels cluster in the region associated with the surface type *peak*.

7.1.4 Nose Detector

We use the geometric invariants ϵ_0 and ϵ_2, as introduced in Section 7.1.2, to construct a nose detector. It decides per pixel of the input image if it corresponds to the tip of a nose or not. In other words, each pixel of the input image is mapped to a d-dimensional feature space, where d is the number of features considered. Within this feature space, we estimate a bounding box that encloses all pixels associated with the tip of a nose. It is important to mention that the bounding box should be estimated in polar coordinates due to the interpretation of the feature space (see Figure 7.1).

The estimation is done in a supervised fashion based on a set of feature points computed for pixels that were hand-labeled as belonging to the tip of the nose. The extent of the bounding box within each dimension of the feature space is simply taken as the minimum and the maximum value of the corresponding feature with respect to all training samples. A softness parameter can be introduced to control the sensitivity of the detector.

To detect a nose within an image, the features are computed for each pixel, and a pixel is classified as the tip of a nose if its feature point lies within the estimated bounding box in the feature space. Despite the simplicity of this approach, we obtain very accurate and robust results, as we will show below.

The input for each feature computation was either the range or the amplitude data of an image from the SR3000 camera. The raw camera data was preprocessed by scaling it to the interval $[0, 1]$. Then, a threshold computed using the method by Otsu (1979) was applied to the amplitude data to separate the foreground from the background. The background was set to a fixed value in both range and amplitude data. This was mainly done to avoid unwanted spatial filter responses on the range

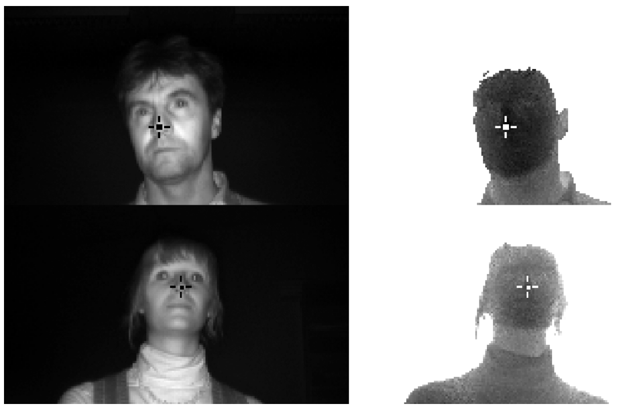


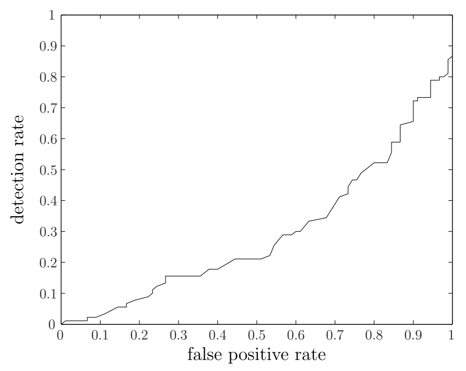
Figure 7.3: *Sample images from the database of head poses. The amplitude data (left column) and the range data (right column) are given for four subjects. All pixels identified as nose pixels by our detector are marked in each image, the cross simply highlighting the locations.*

data due to high levels of noise in regions with low confidence. The radial filter tuning function was set to $A(\rho) = (2\pi\rho)^2 \cdot \exp(-\pi\rho^2/4\sigma^2)$ with $\sigma = 0.3$ for all feature computations. We expect that filter optimization will further improve the results.

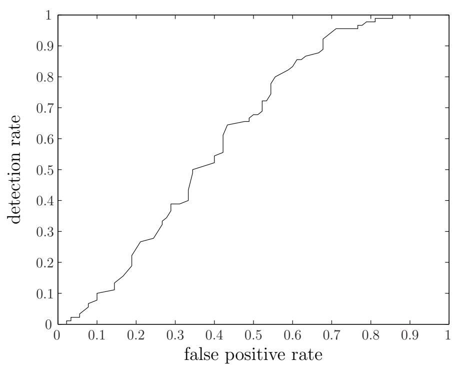
7.1.5 Results

The procedure was evaluated on a database of images taken of people at different head poses. A sample of such images is shown in Figure 7.3, where both amplitude and range data are given for four subjects. Our database consists of a total of 13 subjects; for each subject, nine images were taken at roughly the same distance from the camera for different orientations of the head. The extent of the bounding box was estimated on a training set of three subjects, and the method was evaluated on the remaining ten subjects. The results presented in the following show that the method generalizes very well when using the combination of range and amplitude data.

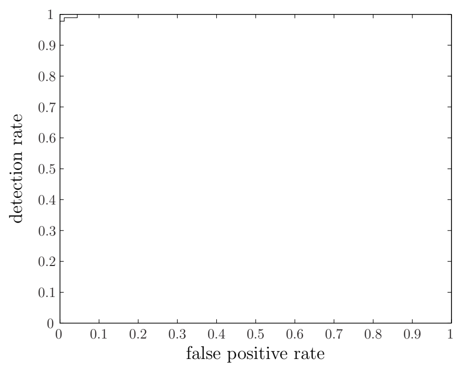
Figure 7.4 shows the ROC curves for different combinations of input data. For Figure 7.4a only the range data was used from each image, whereas Figure 7.4b shows the results for the amplitude data. The features ϵ_0 and ϵ_2 were used in both cases. The method achieves an equal error rate (EER) of 66.0% on the range data and 42.3% on the amplitude data. Even though the range data seems to be better suited for the proposed geometric features the amplitude data gives slightly better results. We cannot give a final explanation for this effect, but we assume that it is due to a higher level of noise in the range data. Although the EER is not satisfying in both cases, the results are quite good considering the simplicity of the classifier.



(a) Range data.



(b) Amplitude data.



(c) Range and amplitude data.

Figure 7.4: *ROC curve of detection rate vs. false positive rate on range data (a), amplitude data (b), and the combination of both (c). The detection rate gives the percentage of images in which the nose has been identified correctly, whereas the false positive rate denotes the percentage of images where at least one non-nose pixel has been misclassified. Thus, strictly speaking, the curves do not represent ROC curves in the standard format, but they convey exactly the information one is interested in for this application, that is, the accuracy with which the detector gives the correct response per image.*

We were able to improve the performance dramatically by using a combination of features on range and amplitude data. We used the two features ϵ_0 and ϵ_2 for both types of data, which amounts to a total of four features. The corresponding ROC curve is shown in Figure 7.4c, and we can report an EER of 3.1% for this method.

7.1.6 Discussion

In this section, we presented a very simple detector for the human nose based on the TOF camera SR3000. The method yields very accurate and robust detection rates. However, we can point out three aspects that have potential to increase the performance the detector: (1) The use of a more sophisticated classifier, (2) an adaptation of the low-pass filter to the noise distribution, and (3) the use of additional features.

Also, we point out that when the detector is used to track the nose over several frames, as opposed to performing detection on individual frames, robustness can be improved by exploiting the fact that, in general, the position of the nose does not change much from frame to frame. In Chapter 9 we show that a very robust tracking of the nose can be achieved using this feature based approach.

7.2 Scale Invariant Features

7.2.1 Introduction

The fact that the apparent size of an object in a camera image changes with the distance of the object from the camera leads to one of the fundamental problems in computer vision: Finding scale-invariant image features, i.e. features that, by their mathematical formulation, are unaffected by image scale (for an example of a recent approach, see (Lowe, 1999)). Achieving scale invariance usually requires increased algorithmic complexity and additional computation. For example, the image can either be scanned for objects of different sizes, or it can be transformed into scale-space (Koenderink, 1984), where the feature extraction is computed individually at different levels of scaling. In both cases, the treatment of objects at different scales has to be made explicit within the algorithm.

In this section, we suggest a novel approach to the problem of scale-invariance: If we use a range sensor – such as a TOF camera – to image the object, we can compute features directly on the reconstructed surface of the object in 3D space. In this way, the features become scale-invariant, because the 3D reconstruction – unlike the image of the object – does not undergo scale changes as the object moves towards or away from the camera.

We are particularly interested in the TOF camera as a basis for this type of approach because it provides a range map that is perfectly registered with an intensity image in a single device, making it easy to create detectors based on a combination of range and intensity features.

Naturally, one can compute image features on the regular image grid of both range and amplitude images directly as we have demonstrated in the previous section (Haker et al., 2007). Note, however, that interpreting the range map as an array of height values measured over a regular grid is equivalent to the *weak-perspective* assumption, i.e. to assuming that the total depth variation within the object is small compared to the distance of the object from the camera. If this assumption is violated, the geometry of the surface reconstructed using weak perspective will differ markedly from the true geometry of the object. Furthermore, the size of an object in the image changes with its distance to the camera.

Alternatively, if the intrinsic camera parameters are known, we can apply the inverse of the camera projection to the range data, thus obtaining an actual sampling of the object's surface in three-dimensional Cartesian coordinates. Obviously, the measured object does not undergo any scaling in the 3D scene if it is moved towards or away from the camera; instead, only the spatial sampling frequency decreases as the distance to the camera is increased (see Figure 7.5). It is important to note, however, that the sampling grid in this representation is no longer regular; the spacing between two samples depends on the distance of the relevant part of the object to the camera. Many techniques for extracting image features, such as convolution with a filter kernel, require a regular sampling and can thus no longer be used.

To overcome this problem, we compute the features not in the spatial domain, but in the frequency domain. We transform the sampled object shape to a frequency domain representation using the Nonequispaced Fast Fourier Transform (NFFT, see Section 7.2.2), an efficient algorithm for computing the Fourier transform of a signal sampled on an irregular grid. Thus, any feature computation that can be performed in the Fourier domain can now be evaluated efficiently. The main advantage of this approach is that any filter operation has, in theory, the same effect on an object independently of the distance of the object to the camera. The NFFT has been used for image processing tasks such as CT and MRI reconstruction (see e.g. (Potts and Steidl, 2001)), and it has also been used to extract features from vector fields for visualization (Schlemmer et al., 2005). However, to our knowledge, the approach of using the NFFT to compute scale-invariant features for classification is novel.

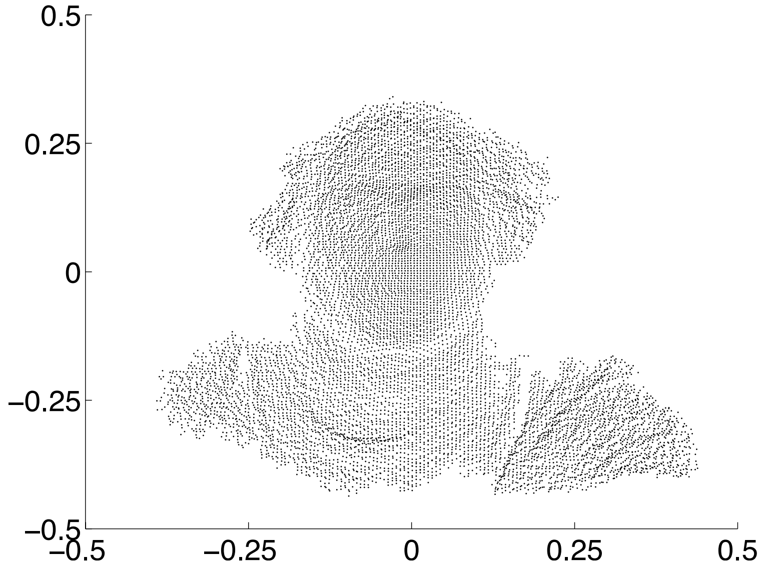
We demonstrate this approach using the set of geometric features introduced in Section7.1, which are related to mean and Gaussian curvature. When evaluated on the image in the conventional way, these features are sensitive to scale changes; however, when evaluated on the object surface using the NFFT, the features are invariant to scale. We verify this using a synthetic test object in Section 7.2.4.

Finally, we use these features for a facial feature tracking problem. In previous work by Böhme et al. (2008a) (as described in Section 7.1), we tackled this problem using features evaluated conventionally on the camera image; this solution had limited robustness towards scale variations. The new scale-invariant features yield greatly improved detection at varying distances from the camera, as we show in Section 7.2.4.

7.2.2 Nonequispaced Fast Fourier Transform (NFFT)

Definition

As noted in the introduction, we need to compute the Fourier transform of a function sampled on a nonequispaced grid. To do this, we use the NFFT (Potts et al., 2001),



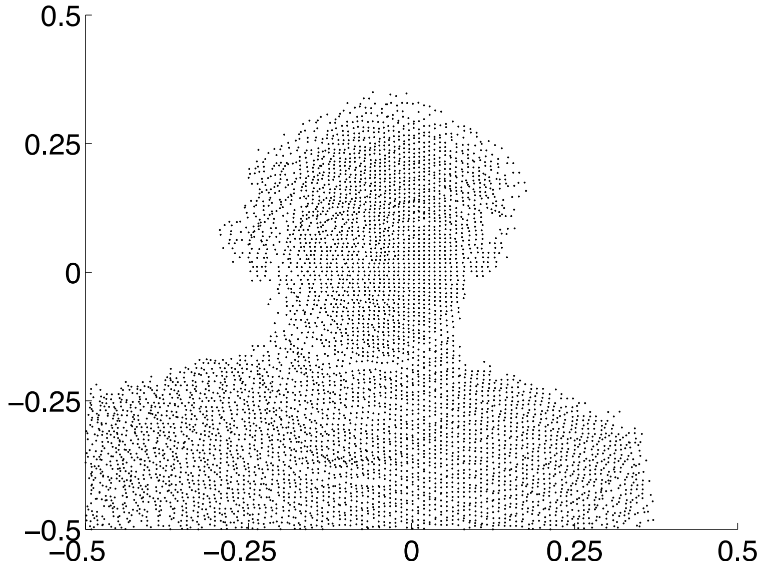


Figure 7.5: Top: *Sampling of a face 35 cm from the camera.* **Bottom:** *Sampling of a face at 65 cm from the camera. Note that the spatial sampling frequency is significantly lower compared to the face at 35 cm, but the physical size of the face in 3D space is still the same.*

an algorithm for the fast evaluation of sums of the form

$$f(\mathbf{x}_j) = \sum_{\mathbf{k} \in I_\mathbf{N}} \hat{f}_\mathbf{k} \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{x}_j} \quad , \tag{7.9}$$

where the $\mathbf{x}_j \in [-\frac{1}{2}, \frac{1}{2})^d, j = 1, \ldots, M$ are arbitrary nodes in the spatial domain (of dimension d), the $\mathbf{k}$ are frequencies on an equispaced grid, and the $\hat{f}_\mathbf{k}$ are the corresponding Fourier coefficients. The equispaced frequency grid $I_\mathbf{N}$ is defined as

$$I_\mathbf{N} := \Big\{ \mathbf{k} = (k_t)_{t=1,\ldots,d} \in \mathbb{Z}^d : \\ -\frac{N_t}{2} \le k_t < \frac{N_t}{2}, t = 1, \ldots, d \Big\} \quad , \tag{7.10}$$

where $\mathbf{N} = (N_1, \ldots, N_d)$ is the so-called *multibandlimit*, which specifies the band limit along each dimension. (Note that all N_t must be even.)

We refer to Equation (7.9) as the Nonequispaced Discrete Fourier Transform (NDFT). From this equation the Discrete Fourier Transform (DFT) (with equispaced nodes in the spatial domain) can be obtained by setting the $\mathbf{x}_j$ to the nodes of the grid $\mathbf{x} = (\frac{k_t}{N_t})_{t=1,\ldots,d}$, $k_t \in \{-\frac{N_t}{2}, \ldots, \frac{N_t}{2} - 1\}$.

Equation (7.9) describes the transform from the frequency domain to the spatial domain. In the case of the equispaced DFT, because the matrix that describes the transform is unitary, the same algorithm can be used for the opposite transform (from the spatial to the frequency domain). This is not true in the nonequispaced case; here, to transform from the spatial to the frequency domain, i.e. to find Fourier coefficients $\hat{f}_\mathbf{k}$ such that evaluation of Equation (7.9) will yield certain given values $f(\mathbf{x}_j)$, we need a second algorithm (see (Kunis and Potts, 2007)), which is based on a combination of the conjugate gradient method with the NFFT. Note that this algorithm (which transforms from the spatial to the frequency domain) is sometimes referred to as the "inverse NFFT", whereas the term "inverse" is otherwise usually applied to a transform from the frequency to the spatial domain.

Applying the NFFT to Range Data

We assume that the object surface, reconstructed from the range data by inverting the camera projection, is given in Cartesian coordinates x, y, z, where the x-y-plane is parallel to the image plane and the z-axis is parallel to the camera's optical axis. To apply the NFFT to this data, we interpret the z-coordinate as a function $z(x, y)$ of the x-y-coordinates; hence, the x-y-coordinates define the grid nodes for the NFFT.

As noted in the previous section, these nodes need to lie in the interval $[-\frac{1}{2}, \frac{1}{2}) \times [-\frac{1}{2}, \frac{1}{2})$. Generally, this means that the x-y-coordinates of the surface points need to be scaled to this interval. We wish to use the same scaling for all images so that the interpretation of the Fourier domain remains the same.

To define this scaling, we will introduce the concept of an *equivalence range*; we want to choose such a scaling that, for an object at the equivalence range e, the effect of applying a particular transfer function to the FFT spectrum and to the NFFT spectrum is the same. The correct scaling is computed by intersecting the camera's field of view with a plane perpendicular to the view direction at distance e from the camera, yielding a rectangle; the x-y-plane is then scaled such that this rectangle fits exactly within the interval $[-\frac{1}{2}, \frac{1}{2}) \times [-\frac{1}{2}, \frac{1}{2})$.

Note that the x-y-coordinates of points beyond the equivalence range may lie outside the interval $[-\frac{1}{2}, \frac{1}{2}) \times [-\frac{1}{2}, \frac{1}{2})$; these points are discarded. The equivalence range thus needs to be chosen such that the resulting clipping volume is large enough to contain the objects of interest. The centroid of these objects should be shifted to $x = 0, y = 0$ to ensure they are not clipped.

Another point of note is that it is advisable, if possible, to segment the foreground object of interest and apply the NFFT only to the points belonging to that object. There are various reasons for doing this: (i) Steep edges between the foreground and background can lead to ringing artefacts. (ii) The grid nodes in the background region are spaced further apart; the greater the spacing between grid nodes, the lower the frequency where aliasing begins. (iii) Passing fewer points to the NFFT reduces the computational requirements.

Finally, note that the transform from the spatial domain to the frequency domain is often an underdetermined operation. In this case, the NFFT computes the solution with minimal energy, meaning that the background region, where there are no grid nodes, is implicitly set to zero. To avoid steep edges between the foreground and the background, we subtract a constant offset from the z values so that the maximum z value becomes zero.

7.2.3 Nose Detection

In Section 7.1, we have used the geometric features for the task of nose detection and tracking. Here we will evaluate the benefit of the scale-invariant feature computation by comparing the results in the nose detection task to those obtained using the feature computation in image coordinates. As described in the previous section, the geometric features can be computed conveniently in the Fourier domain. In case of the orthographic projection, the spectrum can simply be computed using the FFT. However, we leverage the depth information using the perspective camera model to obtain a 3D object representation that is independent of the object's distance to the camera. As a consequence, we have to rely on the NFFT for the computation of the spectrum. There exist a number of technical details regarding the application of the NFFT, which will be the focus of this section.

Some of the details are best explained with respect to the nose detector, thus, we will briefly review the algorithm as described in 7.1.4. For each pixel in an image we compute the generalized eccentricities ϵ_0 and ϵ_2 and, thus, map it to the feature space given in Figure 7.1. Within feature space, noses are characterized by a bounding box which is learned from a set of labeled training data. During classification, a given pixel is said to belong to the tip of a nose iff it is mapped into this bounding in feature space.

In correspondence to Section 7.1, we segmented the foreground from the background in the TOF range images to discard noise in image regions with low signal to noise ratio. All background pixels were set to the maximum distance in the foreground region to obtain a smooth continuation from the foreground to the constant background region. To further improve the smoothness, we also discarded 20% of the foreground pixels with largest distance to the camera. As a result, the border of the foreground is roughly at the same distance to the camera and filling the background with a constant does not induce any discontinuities.

In the orthographic case, the resulting range image was then passed directly to the FFT. In case of the NFFT based algorithm, we first inverted the perspective projection to obtain xyz-coordinates. Note, that the background noise should be removed prior to this step, because noise pixels having very low distance values will project to points that lie between the object in the foreground and the camera resulting in spurious peaks in the Monge patch. In a second step, the coordinates x and y were shifted so that the foreground object was centered on the optical axis of the camera. This was achieved by a simple subtraction of the mean of all xy-coordinates of foreground pixels.

The final step in preparing the data for the NFFT is to select the relevant samples for processing and to scale the sampling nodes to the correct interval. Apparently, the sampling nodes have to lie within a constant interval for all images so that results relate to the same scale. Thus, we determined upper and lower bounds for x and y which correspond to a square frame of 50 cm side length in the 3D scenery. The resulting interval is scaled to $(-\frac{1}{2}, \frac{1}{2}) \times (-\frac{1}{2}, \frac{1}{2})$ for processing with the NFFT.

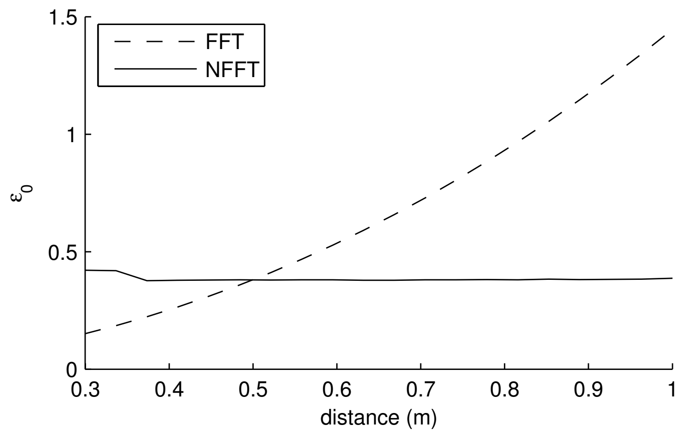


Figure 7.6: *Generalized eccentricity ϵ_0, computed on a synthetic test image of a sphere, as a function of distance from the camera.*

Furthermore, only foreground samples were passed to the NFFT to reduce the computational cost. Note, that it makes sense omit too distant background pixels because, intuitively speaking, the Nyquist frequency depends on the largest sampling rate within the data which is inversely proportional to the distance. Finally, the NFFT assumes the input function to be zero in all regions in $(-\frac{1}{2}, \frac{1}{2}) \times (-\frac{1}{2}, \frac{1}{2})$ where no samples are provided. As a consequence, the maximum input value was subtracted from all input samples to avoid discontinuities. An example of the samples passed to the NFFT for two specific images is given in Figure 7.5.

7.2.4 Experimental Results

The algorithms were implemented in Matlab; the NFFT 3.0 library (Keiner et al., 2006), which is implemented in C, was used to compute the NFFT.

Synthetic Data

To begin, we will examine the feature values computed on a synthetic test object using both the classical scale-dependent FFT-based approach and the scale-independent NFFT-based approach. We synthesized range images of a sphere at various distances from the virtual camera and computed the generalized eccentricity ϵ_0 using the FFT- and NFFT-based approaches.

Figure 7.6 shows the value of ϵ_0 at the apex of the sphere as a function of distance. (ϵ_2 is not shown because it is identically zero for objects that, like the sphere, exhibit the same curvature in all directions.) It is apparent that while the feature value changes noticeably with distance for the FFT-based approach, it remains essentially constant for the NFFT-based approach. Note also that at the equivalence range of $e = 0.5$ m, both approaches compute the same feature value.

Real-World Data

To evaluate the performance of the features on a real-world problem, we compare the detection rates of the nose detector (Haker et al., 2007) on images of an SR3000 TOF camera using the NFFT-based algorithm and the original FFT-based version, respectively. In both cases, the training was done on a database of three subjects who were imaged at a fixed distance of 60 cm from the camera. During evaluation, the detector had to generalize to a dataset of 87 face images showing a different subject. The test images were taken at distances ranging from 35 to 70 cm. Figure 7.7 shows two examples of range maps from the test set along with the scale-invariant features.

In the case where only the range data of the TOF images is considered, the results are given in Figure 7.8. Here, the NFFT-based algorithm achieves an EER of 20% in comparison to 39% in case of the FFT-based version and, thus, clearly yields a significant improvement in detection performance.

We have shown the detection results on features extracted from the range data alone to make the effect more clearly visible; for optimal detection performance, we would additionally use the same type of features computed on the intensity data. As we will discuss in Section 7.2.5, we still compute the intensity features in the conventional way using the FFT, and they are thus not scale-invariant. Nevertheless, when they are combined with the FFT-based range features, we obtain an EER of 4%; when NFFT-based range features are used, the EER drops to 0%, i.e. there are no errors on the test set – a larger test set would be required to measure a more meaningful EER. (As a point of note, the EER on the intensity features was 78%; it is only the combination of range and intensity features that yields low error rates.)

It should be mentioned that, while the NFFT has the same asymptotic running time as the FFT, it is slower by a relatively large constant factor. Our Matlab implementation, running on a 2.66 GHz E6750 Intel CPU, requires 0.1 s to compute the FFT-based features, versus 5 s for the NFFT-based features. A C implementation of the FFT-based detector runs at camera frame rates (Böhme et al., 2008a); the NFFT-based detector is currently too slow for this type of application. However, we believe there are ways of achieving the same effect at lower computational cost; in the meantime, we see NFFT-based scale-invariant features as an attractive technique

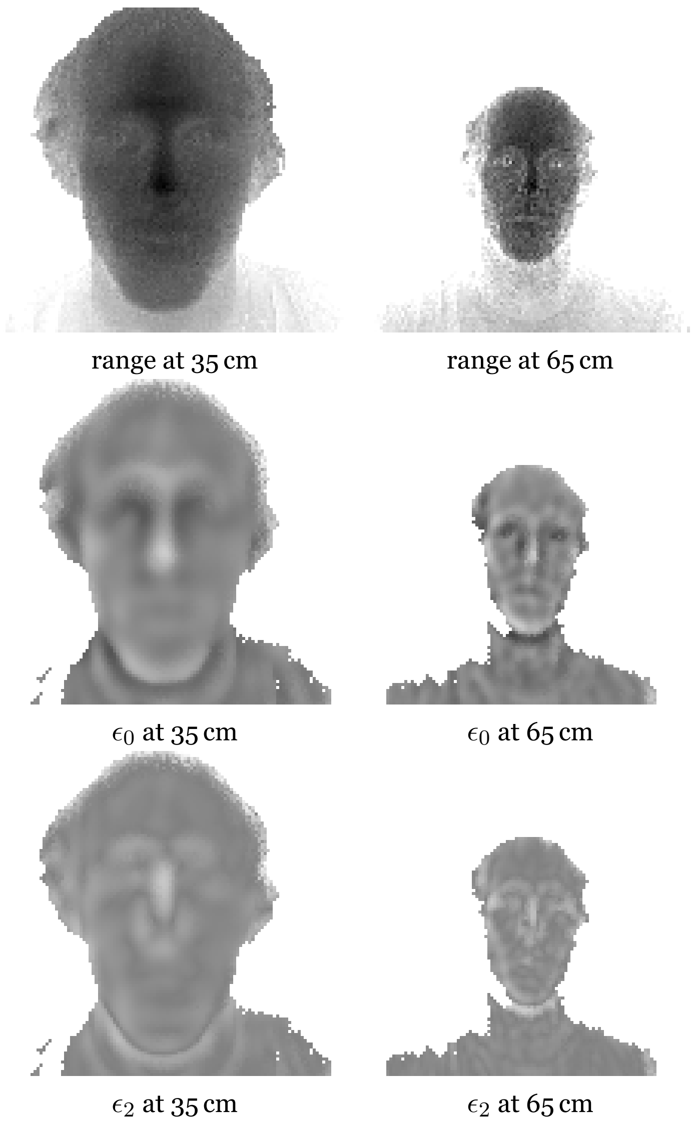
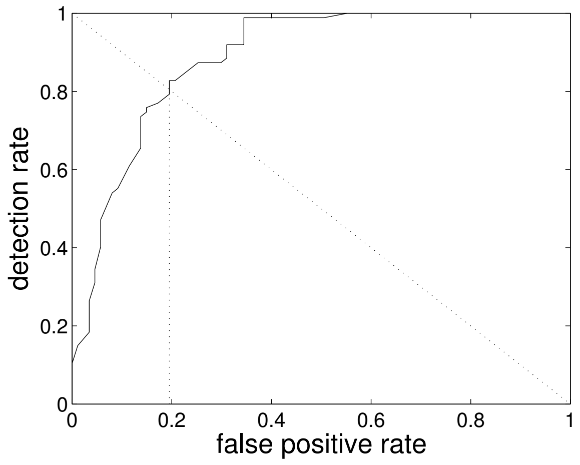


Figure 7.7: *Two different range image samples from the test set, taken at 35 cm (left column) and 65 cm distance (right column). The corresponding features ϵ_0 and ϵ_2 computed at foreground pixels via the NFFT are shown, respectively.*



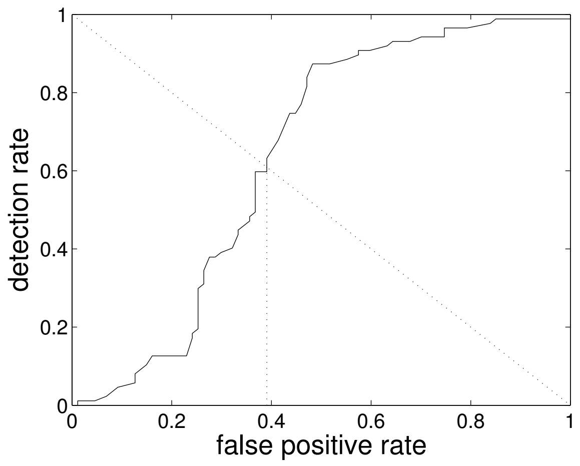


Figure 7.8: Top: *ROC curve showing detection rate vs. false positive rate for the nose detection task using the NFFT.* **Bottom:** *The ROC curve obtained on the same database using the FFT. The detection rate indicates the percentage of images in which the nose has been identified correctly, whereas the false positive rate denotes the percentage of images where at least one non-nose pixel has been misclassified. Thus, strictly speaking, the curves do not represent ROC curves in the standard format, but they convey exactly the information one is interested in for this application, that is, the accuracy with which the detector gives the correct response per image.*

for non-interactive applications.

7.2.5 Discussion

Features computed directly on the three-dimensional geometry of the object are, by their nature, scale-invariant. As we have shown, this allows for a more robust classification than when the same features are computed directly on the range map, where they are sensitive to scale variations. We have demonstrated this using a specific set of features, the generalized eccentricities, but the method itself is very general and can be applied to a wide range of operators.

We have used our technique to implement a facial feature detector using a TOF camera. The detector generalizes from faces presented at a fixed training distance to a test set containing different faces at different distances. It achieves good detection performance, making no errors on the test set. Two important factors that help us achieve this result are the scale-invariant features and the fact that the TOF camera provides a perfectly registered intensity image in addition to the range map. The combination of range and intensity data yields substantially better classification results than either type of data alone.

Currently, we still compute the intensity features on the image, where they are sensitive to scale variations. Ideally, we would like to compute these features on the object surface, too. This is, however, a slightly more complicated problem, because intensity is a function of the position on a two-dimensional sub-manifold (the object surface) in three-dimensional space; the geometry of this sub-manifold must be taken into account when computing the intensity features. This is an avenue for future work.

On a general level, our key point is this: The perspective transformation that is inherent in the image formation process causes scale variations, which present additional difficulties in many computer vision tasks. This is why, in our view, the TOF camera is an attractive tool for computer vision: In effect, it can act as a digital orthographic camera, thereby simply eliminating the problem of scale variations.

7.3 Multimodal Sparse Features for Object Detection

7.3.1 Introduction

In recent years there has been a lot of interest in learning sparse codes for data representation, and favorable properties of sparse codes with respect to noise resistance have been investigated (Donoho et al., 2006). Olshausen and Field (1997) applied sparse coding to natural images and showed that the resulting features resemble receptive fields of neurons in the visual cortex. Thus, it stands to reason that the basis functions computed by sparse coding can be used effectively in pattern recognition tasks in the fashion introduced by Serre et al. (2007), who model a recognition system that uses cortex-like mechanisms. They use Gabor jets, which also roughly correspond to receptive fields of V1 simple cells, to extract images features for object recognition.

Sparse coding has also been successfully applied to the recognition of handwritten digits. Labusch et al. (2008a) learn basis functions for representing patches of handwritten digits and use these to extract local features for classification.

In this work, we aim to learn a sparse code for multimodal image data, i.e. we simultaneously learn basis functions for representing corresponding intensity and range image patches. As a result, we obtain aligned pairs of basis functions that encode prominent features that co-occur consistently in both types of data. Thus, a corresponding pair of basis functions can be used to consistently extract features from intensity and range data. To our knowledge, sparse representations have not yet been learned for multimodal signals. The considered image data was obtained by a TOF camera, which provides a range map that is perfectly registered with an intensity image.

We already showed in Section 7.1 and Section 7.2 that using both intensity and range data of a TOF camera in an object detection task can significantly improve performance in comparison to using either data alone (Haker et al., 2007, 2008). The fact, that a sparse code learned simultaneously on both intensity and range data yields perfectly aligned basis functions, allows us to extract relevant features from both types of data.

Here, we aim to learn a set of basis functions that encode structural information of frontal face images in a component-based fashion. As a result, the basis functions estimated by sparse coding can be regarded as templates for facial features, such as the nose. We evaluate the resulting templates on a database of TOF images and use simple template matching to identify the presence and position of the nose in frontal face images. The importance of the nose as a facial feature for problems such as head tracking was already mentioned by Yin and Basu (2001) and Gorodnichy (2002). The fact that we use a single basis function for template matching yields a very efficient algorithm for detecting the nose that can be implemented at camera frame rates.

The common alternative to using only a single basis function in the nose detector would be to compute a sparse coefficient vector for every image patch that may contain a nose based on the estimated basis functions. The coefficient vector can then serve as an input to a high-dimensional classification algorithm, such as a support vector machine (SVM). While this procedure is computationally more expensive than template matching and cannot be implemented at camera frame rates, it yields improved detection rates.

Section 7.3.2 will discuss the computation of a set of basis functions under the constraint of the sparse coding principle. In Section 7.3.3 we discuss the procedure of determining the basis function that yields the optimal equal error rate (EER) in the template matching approach for the nose detection task. Furthermore, we discuss the training of an SVM on the sparse coefficient vectors. Section 7.3.4 presents the results and shows that templates generated via sparse coding yield significantly better detection rates than templates obtained by PCA or by computing an average nose from a set of aligned nose samples. In addition, we compare the results of an SVM classifier trained on coefficient vectors computed under the sparse coding principle in comparison to coefficient vectors obtained via PCA.

7.3.2 Sparse Features

The investigated database of frontal face images (art) was obtained using an SR3000 TOF camera (Oggier et al., 2005a). The subjects were seated at a distance of about 60 cm from the camera and were facing the camera with a maximum horizontal and/or vertical head rotation of approximately 10 degrees. As a result, the facial feature of interest, i.e. the nose, appears at a size of roughly 10×10 pixels in the image. A number of sample images are given in Figure 7.9.

As a TOF camera provides a range map that is perfectly registered with an intensity image, we aim to learn an image basis for intensity and range simultaneously. To this end, the input data for the sparse coding algorithm are vectors whose first half is composed of intensity data and the second half of range data, i.e. in case we consider image patches of size 13×13, each patch is represented by a 338-dimensional vector ($d = 338 = 2 \cdot 13 \cdot 13$) where the first 169 dimensions encode intensity and

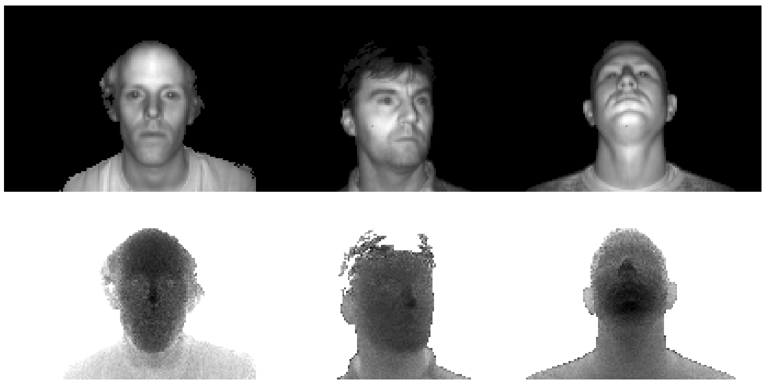


Figure 7.9: *Three sample images of frontal face images taken by an SR3000 TOF camera. The top row shows the intensity and the bottom row the range data.*

the remaining 169 dimensions encode range.

In order to speed up the training process, we only considered training data that originated from an area of 40×40 pixels centered around the position of the nose. The position of the nose was annotated manually beforehand. By this procedure we prevent the basis functions from being attracted by irrelevant image features, and a number of 72 basis functions proved to be sufficient to represent the dominant facial features, such as the nose or the eyes.

A common difficulty with TOF images is that the range data is relatively noisy and that both intensity and range can contain large outliers due to reflections of the active illumination (e.g. if subjects wear glasses). These outliers violate the assumed level of Gaussian additive noise in the data and can lead the sparse coding algorithm astray. To compensate for this effect, we applied a 5×5 median filter to both types of data. To ensure the conservation of detailed image information while effectively removing only outliers, pixel values in the original image I_o were only substituted by values of the median filtered image I_f if the absolute difference between the values exceeded a certain threshold:

$$I_o(i,j) = \begin{cases} I_f(i,j) & \text{if } |I_o(i,j) - I_f(i,j)| \geq \theta \\ I_o(i,j) & \text{otherwise} \end{cases} .$$

There exist a number of different sparse coding approaches, see for example (Olshausen and Field, 1997; Lewicki et al., 2000; Labusch et al., 2009). We employed the Sparsenet algorithm proposed by Olshausen and Field (1997) for learning the sparse code. The basic principle aims at finding a basis W for representing vectors $\vec{x}$ as a linear combination of the basis vectors using coefficients $\vec{a}$ under the assumption of Gaussian additive noise: $\vec{x} = W\vec{a} + \vec{\epsilon}$. To minimize the reconstruction error while enforcing sparseness, i.e. the property that the majority of coefficients a_i are zero, the Sparsenet algorithm solves the following optimization problem:

$$\min_W E\left(\min_{\vec{a}} \left(\|\vec{x} - W\vec{a}\| + \lambda S(\vec{a})\right)\right) \ . \tag{7.11}$$

Here, E denotes the expectation and $S(\vec{a})$ is an additive regularization term that favors model parameters W that lead to sparse coefficients $\vec{a}$. The parameter λ balances the reconstruction error $\vec{\epsilon}$ against the sparseness of the coefficients.

In order to apply the method, the input data has to be whitened beforehand as indicated by Olshausen and Field (1997). We applied the whitening to both types of data individually. Only after this preprocessing step, the training data was generated by selecting random image patches of the template size, i.e. for a patch in a given image the corresponding intensity and range data were assembled in a single vector.

The resulting features for 19×19 image patches are given in Figure 7.10. Facial features, e.g. nose, eyes, and mouth, can readily be distinguished. We set the parameter λ to a relatively high value ($\lambda = 0.1$), i.e. we enforce high sparseness, in order to obtain this component-based representation. However, we can report that the results are not particularly sensitive to minor changes of this parameter.

7.3.3 Nose Detection

Template Matching Approach

Since the basis functions computed by sparse coding in Section 7.3.2 represent facial features, it stands to reason that they can be used for object detection via template matching. At this point two questions arise: (i) Which basis function represents the best template, and (ii) what is the actual position of the facial feature with respect to the center of the image patch corresponding to this basis function. A straightforward solution would be to select the most promising feature by visual inspection and to annotate the position of the facial feature within the image patch manually. Obviously though, this procedure is not generally applicable and is likely to yield suboptimal results.

Thus, we decided to follow a computationally more intensive procedure that, in contrast, is fully automatic and operates in a purely data-driven fashion. For each of the 72 basis functions we trained and evaluated a nose detector for every possible position of the nose in a certain neighborhood around the center of the image patch.

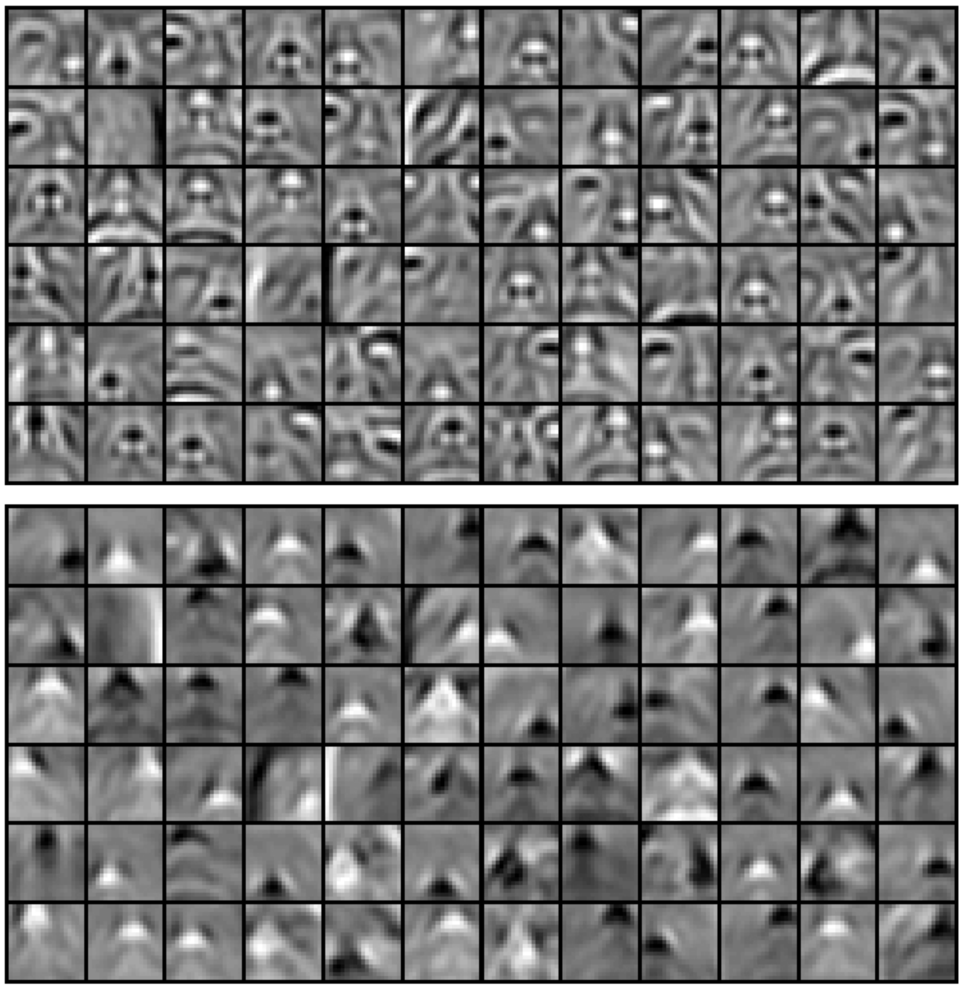


Figure 7.10: *Basis functions learned for frontal face images via the Sparsenet algorithm. The upper and lower part of the figure show the basis functions representing intensity data and range data, respectively. The basis functions for both types of data were learned simultaneously and correspond pairwise, i.e. the top-left intensity feature is aligned with the top-left range feature.*

In the case of 13×13 image patches we chose this neighborhood to be 11×11. As a result, a total of $8712 = 72 \cdot 11 \cdot 11$ detectors were trained. The final detector uses the basis function and the position of the nose out of the 8712 configurations that produced the best EER on the training set.

The thresholds of each detector were simply determined by taking the minimum and the maximum of the filter responses at the annotated positions of the nose on a set of training images, i.e. upper and lower bounds for the filter responses that identify a nose were determined for both intensity and range data. In order to identify a nose in a new image, both intensity and range were filtered with the corresponding template images and each pixel whose filter responses complied with the identified bounds was classified as a nose pixel. To obtain an EER, these bounds were relaxed or tightened.

The procedure was evaluated on a data set of 120 TOF images of frontal faces taken from 15 different subjects. To double the amount of data, mirrored versions of the images were also added to the data set. From the total of 240 images one half was chosen at random as a training set to determine the bounds of each classifier. These bounds were then adjusted to yield an EER on the training set. Finally, the optimal classifier, i.e. the one out of the 8712 candidates yielding the best EER, was evaluated on the remaining 120 images that were not used during the training process.

In order to assess the performance of the learned templates, we also evaluated two other types of templates – "average" and "eigen" templates. The former were generated by computing an average nose from a set of image patches containing aligned sample noses. The latter were obtained as the principal components of these aligned image patches. Again, we generated corresponding pairs of templates for both intensity and range data. The same training images, including the preprocessing, were used as in Section 7.3.2 for the Sparsenet algorithm.

A fundamental difference between these two approaches to generating the average and eigen templates and the sparse coding method is, that the former only yield templates in which the nose is centered in the image patch whereas the latter also produces translated versions of the nose (see Figure 7.10). To guarantee a fair comparison between the different templates we applied the following procedure: Since the optimal position of the nose within the template is not known a priori, we generated a total of 121 13×13 templates centered at all possible positions in a 11×11 neighborhood around the true position of the nose, i.e. the templates were shifted so that the nose was not positioned in the center of the template. In correspondence to the procedure described above for the sparse-coding templates, each shifted template was then evaluated for every possible position of the nose in a 11×11 neighborhood around the center of the image patch. For the average templates the resulting number of possible detectors amounts to 14641. In the case of the eigen templates, it is not apparent which principal component should be used as a template. To constrain the computational complexity, we considered only the first three principal components. Nevertheless, this resulted in 43923 possible detectors. Again, the optimal average and eigen templates were determined as the ones yielding the best EER on the training set according to the procedure described above.

Coefficient Vector Approach

In contrast to the procedure described above, where only a single basis function was used for template matching to detect the nose, we will now discuss the procedure that utilizes the entire set of basis functions W. Again, the basis functions W were computed according to Equation (7.11) using the Sparsenet algorithm. In this setting, we compute a coefficient vector $\vec{a}$ for each image patch of a new image under the sparse coding principle:

$$\vec{a} = \min_{\vec{a}} \left(\|\vec{x} - W\vec{a}\| + \lambda S(\vec{a}) \right) \ . \tag{7.12}$$

As before, the vector $\vec{x}$ is composed of both the range and intensity data of an image path. Based on the resulting vector $\vec{a}$, a classifier decides for each image patch if it depicts a nose or not.

Again, we rely on the annotated data set of frontal face images described above to extract training vectors representing human noses. As we intend to train an SVM we also require samples from the non-nose class, which we extracted randomly from the training images.

Based on the resulting vectors we train an SVM using the iterative SoftDoubleMaxMinOver algorithm proposed by Martinetz et al. (2009). The hyper-parameters are optimized via 10-fold cross-validation.

To evaluate the detection performance of this approach, we compare it to basis functions U computed via PCA. To obtain the coefficient vector $\vec{a}$ with respect to the PCA basis functions, we followed the standard procedure of projecting the vector $\vec{x}$ onto the orthonormal basis U:

$$\vec{a} = U^T \vec{x} \ . \tag{7.13}$$

For the results to be comparable, we used the same number of basis functions for both approaches. The number equals the dimensionality of the input vectors $\vec{x}$

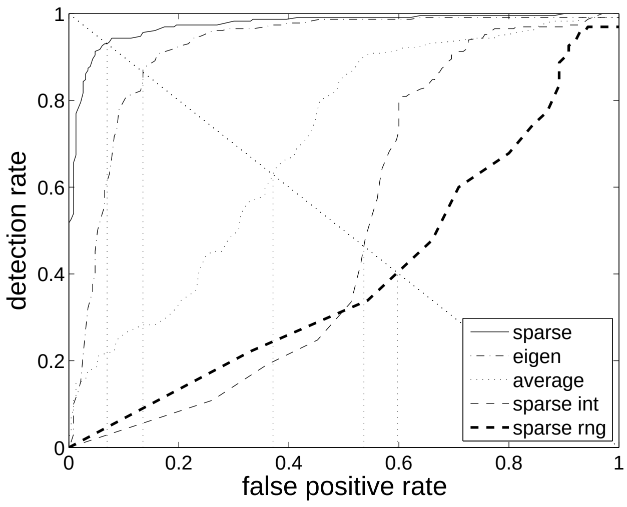


Figure 7.11: *ROC curves of detection rate vs. false positive rate. The curves were generated using the* **sparse***-coding templates, the* **eigen** *templates, the* **average** *templates, and the sparse-coding templates using only the intensity data (***sparse int***) or only the range data (***sparse rng***). The detection rate gives the percentage of images in which the nose was identified correctly, whereas the false positive rate denotes the percentage of images where at least one non-nose pixel was misclassified. Thus, strictly speaking, the curves do not represent ROC curves in the standard format, but they convey exactly the information one is interested in for this application, that is, the accuracy with which the detector gives the correct response per image.*

representing an image patch. Thus, for a patch size of 13×13 both $\vec{x}$ and $\vec{a}$ were of dimension 338.

7.3.4 Results

Template Matching Approach

The results of the training for the nose detection task using 13×13 templates are given in Figure 7.11. The EER on the training set using the sparse-coding templates is 3.9%. The eigen templates achieve an EER of 6.6%, and the average templates yield an EER of 22.5%, i.e. the EERs for these two procedures are higher by roughly 50% and 500%, respectively. The EERs prove to be largely independent of the training set. We ran 100 evaluations of the procedure with random configurations of the training set and recomputed both the templates and the classifiers in each run. The standard deviations for the three EERs over the 100 evaluations were $\sigma = 0.9\%$, $\sigma = 1.6\%$, and $\sigma = 2.3\%$, respectively.

Figure 7.11 also shows the ROC curves for detectors that use the sparse-coding templates computed on either intensity or range data of the TOF images alone. Note that the EERs are dramatically higher in comparison to the detector that uses both types of data together. This confirms results reported by Haker et al. (2007), where the combination of intensity and range data also yielded markedly better results in the detection of the nose based on geometrical features.

The error rates on the test set are only slightly higher than the EERs on the training set, which shows that the method generalizes well to new data. The false positive rates (FPR) amount to 5.3%, 9.3%, and 24.4% for the sparse-coding, eigen, and average templates.

The evaluation above considered only templates of a fixed size of 13×13 pixels. However, varying the template size reveals some interesting properties of the different approaches. To this end, we computed templates of size $n \times n$, where $n = 1, 3, \ldots, 19$, for each approach and estimated the optimal detector according to the same procedure outlined above. To reduce the number of possible detectors to be evaluated, the neighborhood sizes for positioning the nose and shifting the template were reduced to 7×7 pixels for templates with size n smaller than 13. Again, we considered 100 random configurations of training and test set.

Figure 7.12 shows the configurations of template and position of the nose within the template that yielded the best EERs on the training set for each approach with respect to the different template sizes.

Note that the sparse-coding templates (first two rows) exhibit a more appropriate structure and convey higher frequency information in comparison to the average templates (rows three and four), especially for larger sizes of the template. This explains the bad performance of the average templates, because an increase in size does not add more information to the template. This effect becomes clearly visible in Figure 7.13. The plot shows the FPRs on the test set for the different approaches over varying template sizes. One can observe that the FPR of the average template starts to increase for templates of size five. In comparison, the FPR of the sparse-coding templates continues to decrease up to a template size of 19.

A decrease of the FPR can also be observed for the eigen templates up to size 11 of the template. For larger template sizes the FPR also starts to increase, whereas the sparse-coding templates continue to achieve low FPRs, as already mentioned above. It seems that sparse coding can exploit further reaching dependencies.

A comparison of the FPRs with respect to the optimal template size for each method reveals that the average template achieves the worst overall performance with an FPR of 9.6% ($\sigma = 3.5$) for a 7×7 template. The best results for the eigen

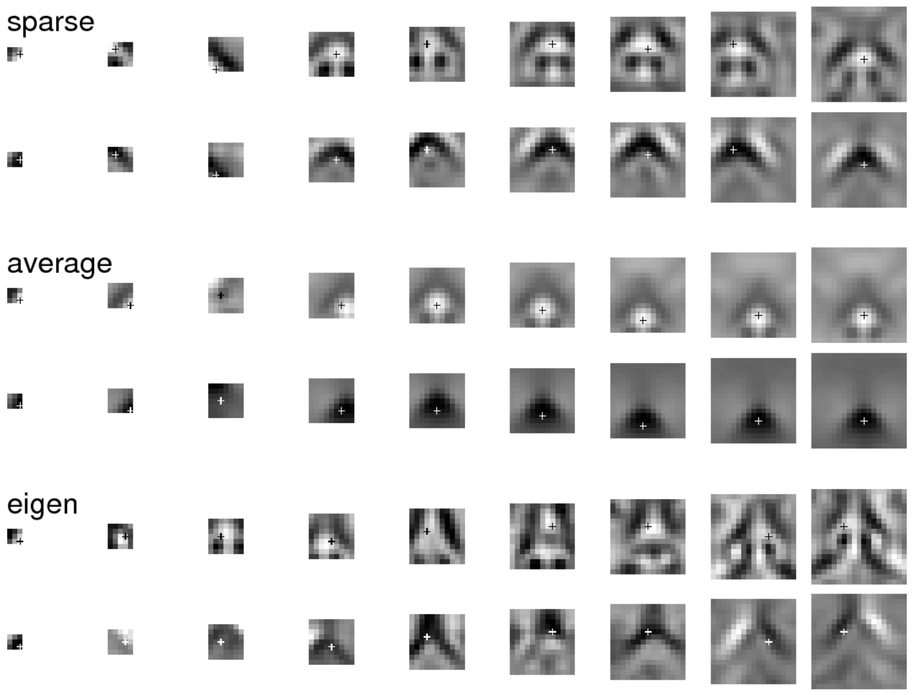


Figure 7.12: *Optimal templates for different template sizes, where each column shows templates of odd pixel sizes ranging from 3 to 19. The first row shows the sparse-coding templates for intensity data and the second row shows the corresponding features for range data. Rows three and four give the average templates and rows five and six show eigen templates. The crosses mark the estimated position of the nose within the templates.*

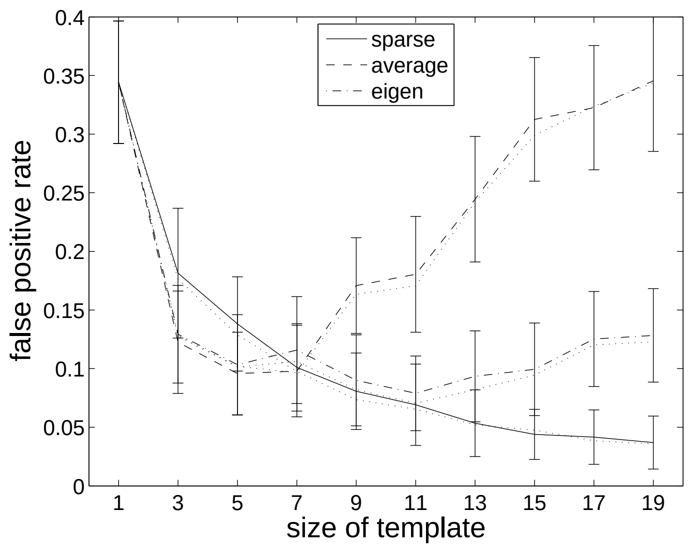


Figure 7.13: *The graph shows the FPRs and standard deviations on the different test sets for the different templates at different template sizes. The dotted lines show the corresponding false negative rates.*

templates were obtained with templates of size 11 yielding an FPR of 7.9% ($\sigma = 3.2$). The sparse coding templates of size 19 had the best overall performance (FPR 3.7%, $\sigma = 2.3$), and the FPR improved roughly by a factor of 2.5 in comparison to the best eigen template.

Note that the false negative rate for the different approaches lies well within the error bars of the FPR in Figure 7.13, as one would expect, since the classifier was set to achieve an EER during training.

Coefficient Vector Approach

The results of the classification with an SVM trained on the full coefficient vectors $\vec{a}$ are presented in Figure 7.14 and Figure 7.15. Again, we evaluated the classification of image patches of different sizes. Here, we considered patches of size $n \times n$, where $n = 7, 9, \ldots, 19$. We ran ten different configurations of training and test set to obtain representative results.

Both Figure 7.14 and Figure 7.15 show an effect that is similar to the one observed previously for the template matching approach. For small sizes of the image patch the PCA yields better results in comparison to sparse coding. However, for large template sizes the sparse coding method achieves similar FPRs at slightly better FNRs. Both approaches perform at an average FPR of 0.3% and an FNR of 2.0%. While

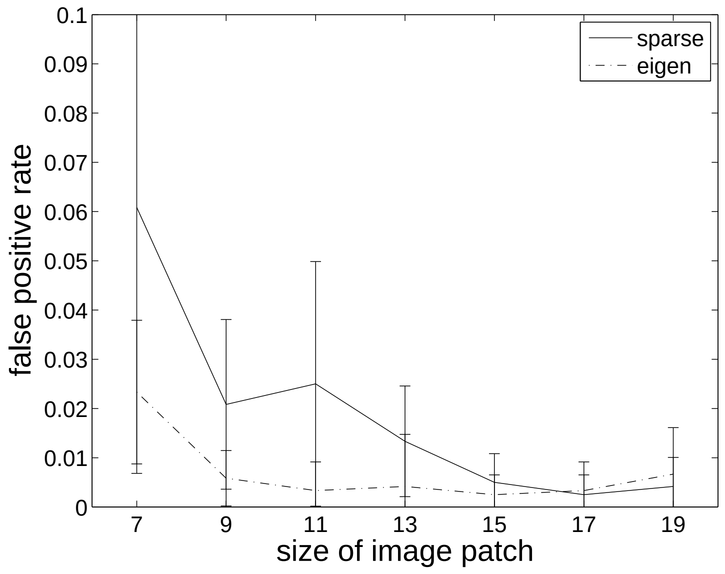


Figure 7.14: *The graph shows the FPRs and standard deviations on the different test sets for the different sizes of the classified image patches.*

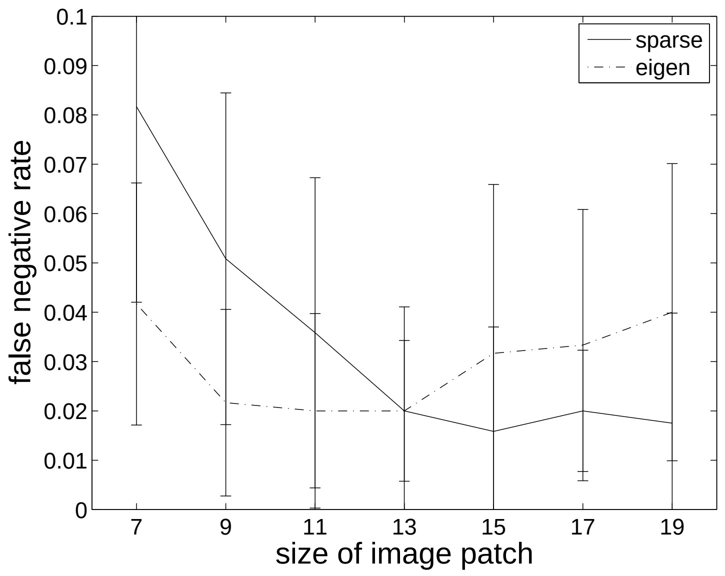


Figure 7.15: *The graph shows the FNRs and standard deviations on the different test sets for the different sizes of the classified image patches.*

this result is obtained at a patch size of 11×11 in case of the PCA, the sparse coding approach achieves these error rates at a patch size of 19×19. Note, that the differences between both methods are not significant. We point out that the classifier was not adjusted to an EER in this setting. The disparity between the FPRs and FNRs is due to the fact that the two classes of nose pixels and non-nose pixels were not represented equally among the training set, i.e. we extracted approximately seven times more non-nose samples than nose samples for training to achieve the best overall performance. It remains a task for future work to retrain the classifier at an EER.

7.3.5 Discussion

We have demonstrated how a sparse code can be learned for multimodal image data, i.e. we computed an image basis for the simultaneous representation of both range and intensity data of a TOF camera. As a result the basis functions capture prominent characteristics of the training images that consistently co-occur in both types of data.

We have shown how the resulting basis functions can be used effectively for template matching in detecting the nose in frontal face images. On a labeled set of TOF images we obtained a FPR of 3.7%. The sparse-coding templates yield significantly better results in comparison to templates obtained by averaging over a number of aligned sample images of noses. Templates resembling the principal components of these aligned sample images were also outperformed, especially for large sizes of the template. We could also confirm previous observations (Haker et al., 2007, 2008) that the combination of intensity and range data yields a greatly improved detector compared to using either type of data alone.

The use of the complete set of basis functions for the feature representation of an image patch yields significantly improved results in comparison to using a single basis function as a template, lowering the FPR to 0.3% at an FNR of 2.0%. However, the computational complexity of this approach prevents an implementation at interactive frame rates.

7.4 Local Range Flow for Human Gesture Recognition

The features introduced in Section 7.1, 7.2, and 7.3 where all computed with respect to a single image, i.e. they capture static properties of the image data. For a number of computer vision problems it is desirable to also consider dynamic properties of a sequence of images. For example it is possible to learn a sparse code in analogy to Section 7.3 for the representation of motion patterns (Olshausen, 2003). However, the most commonly used representative of dynamic features in image analysis is the optical flow.

A typical computer vision problem that requires the use of dynamic features is human action and gesture recognition. It has been tackled by a number of different approaches in recent years. Approaches that are based on the estimation of optical flow were for example proposed by Polana and Nelson (1994) and Efros et al. (2003).

Other, approaches aim at extracting meaningful features that describe the characteristic evolvement of the human pose over time for a specific action. Shechtman and Irani (2007) compute a correlation measure for a space-time patch showing the performed action with templates of the actions to be recognized. A differential geometric approach by Yilmaz and Shah (2008) computes features on a 3D volume in spatio-temporal space that is composed of the 2D contours of the performing person. Other approaches, such as the one by Sun et al. (2009), determine trajectories of keypoints and use them to recognize actions.

The general interest in the topic of action and gesture recognition is also reflected by an increasing number of companies that are active in this area. The Belgian company Softkinetic S.A. provides a middleware for human pose tracking and gesture recognition based on 3D range sensors. The Israeli company 3DV Systems Ltd. developed a similar system that was mainly targeted at the gaming market. In late 2009, 3DV Systems was purchased by Microsoft in the scope of the Project Natal, which aims at providing a gesture interface for the XBox 360.

In this work, we propose a gesture recognition system that decides on a per frame basis whether a gesture out of a set of predefined gestures was performed by a user within a video stream captured by a TOF camera. The gestures investigated here are performed with the users hands. The challenges in this scenario are twofold, i.e. the system is not only required to distinguish between the predefined gestures, but it also needs to decide whether the user is actually performing a gesture or if the movements of the hands are arbitrary and not intended for the gesture recognition system.

We propose a method that combines the pose estimation procedure presented in Chapter 6 with 3D motion features, i.e. we compute the features only at keypoints identified by the estimated pose model. In this way, we can directly assign the motion vectors to certain parts of the body, such as the hands. The utilized features are referred to as range flow. For their computation we use the algorithm proposed by Spies et al. (2000, 2002). The resulting 3D motion vectors from local space-time neighborhoods around the keypoints are accumulated in 3D motion histograms. These histograms are used as input for a classification algorithm that decides on a per frame basis if one of the predefined gestures was performed.

The proposed feature vectors will then be evaluated in a gesture recognition task, i.e. we train and test a gesture recognition system on features extracted from a labeled database of video sequences. We demonstrate the benefit of using the proposed range flow features by comparing them against motion vectors computed directly from the estimated models of human pose of two successive frames.

In the Section 7.4.1 we will briefly review the computation of range flow. Section 7.4.2 will be devoted to the computation of suitable feature vectors using range flow histograms that yield a distinctive representation of the local motion patterns. We will then present the results of the gesture recognition task in Section 7.4.3.

7.4.1 Range Flow

The goal of estimating the range flow for an image sequence of range maps is to estimate a 3D motion vector $(u, v, w)^T \in \mathbb{R}^3$ at every pixel to characterize the 3D motion of the observed surface.

To this end, one considers the range map as a function $Z = Z(x, y, t) \in \mathbb{R}$ of space and time. Based one this function one can formulate the *range flow motion constraint* equation (RFMC) (Horn and Harris, 1991; Yamamoto et al., 1993) as follows:

$$Z_x u + Z_y v + w + Z_t = 0 \; . \tag{7.14}$$

A similar equation, generally used for the estimation of optial flow, can be formulated for the intensity data of the TOF camera. Assuming intensity as a function $I(x, y, t) \in \mathbb{R}$ of space and time, the well-known *brightness change constraint equation* (BCCE) (Horn and Schunck, 1981) takes the following form:

$$I_x u + I_y v + I_t = 0 \; . \tag{7.15}$$

As the TOF camera delivers both range and intensity data, we can use both the RFMC and the BCCE in combination to estimate the range flow. Naturally, the BCCE can only contribute to the components u and v of the motion vector. When combining the two constraints, care has to be taken that not one of the terms dominates the numerical calculation due to a different scaling of the two types of data. Spies et al. (2000) propose to use a scaling factor β^2 based on the average gradient magnitudes computed on a representative data set of range and intensity data:

$$\beta^2 = \frac{< \|\nabla Z\|^2 >}{< \|\nabla I\|^2 >} \, . \tag{7.16}$$

According to Spies et al. (1999), the RFMC and BCCE lead to a total least squares framework of the form:

$$u_1 Z_x + u_2 Z_y + u_3 + u_4 Z_t = 0 \tag{7.17}$$

$$u_1 I_x + u_2 I_y + u_4 I_t = 0 \, . \tag{7.18}$$

Solving this system yields the desired range flow in the form:

$$(u, v, w)^T = \frac{1}{u_4}(u_1, u_2, u_3)^T \, . \tag{7.19}$$

In order to obtain enough constraints to solve the system of equations, one assumes constant range flow in a local neighborhood of the considered pixel, often referred to as an aperture. As a result, one obtains an overdetermined system of equations consisting of Equation (7.17) and Equation (7.18) for every pixel in this local neighborhood. It can be shown that the solution in a total least squares sense corresponds to the eigenvector to the smallest eigenvalue of the extended structure tensor:

$$\begin{bmatrix} ZI_{xx} & ZI_{xy} & < Z_x > & ZI_{xt} \\ ZI_{yx} & ZI_{yy} & < Z_y > & ZI_{yt} \\ < Z_x > & < Z_y > & < 1 > & < Z_t > \\ ZI_{tx} & ZI_{ty} & < Z_t > & ZI_{tt} \end{bmatrix} \, . \tag{7.20}$$

Here, the operator $< \cdot >= \int \cdot$ denotes the integration over the aperture. We introduced the abbreviation $ZI_{ab} =< Z_a Z_b > +\beta^2 < I_a I_b >$ for a more compact notation.

Nevertheless, this approach cannot always retrieve the full range flow if the aperture does not provide sufficient information to uniquely determine the underlying 3D motion vector. This circumstance is referred to as the aperture problem. To reflect this situation one distinguishes between *full flow*, *line flow*, and *plane flow*.

In case three linear independent constraints are available, all but the smallest eigenvalue differ from zero and one can compute the full flow as described above. This is for example the case if the aperture encompasses a 3D corner. In case of linear structures, such as a 3D edge, one obtains two independent constraints and can determine the so-called line flow as the minimum norm solution. Plane flow refers to the minimum norm solution of depth planar structures, which merely yield a single constraint. Note, that one obtains a full-rank tensor if the assumption of locally constant range flow is violated. In such a case the range flow cannot be estimated.

7.4.2 Human Gesture Recognition

Given the range flow computed on a sequence of TOF images, we aim at recognizing human gestures. To this end, we need to integrate the estimated motion vectors into an appropriate feature representation, which can serve as an input to a decision rule that classifies gestures. For the practical applicability of such a gesture recognition system, special attention has to be paid to the problem that unintentional hand movements should not be interpreted as gestures. Thus, the extracted features need to be highly distinctive of the corresponding gesture. In the following, we will first discuss the procedure of extracting the features and will then turn to the utilized classification procedure for recognizing the gestures.

Feature Extraction

In this work, we investigate gestures performed with the users hands. Thus, the considered feature representation aims at characterizing the 3D motion of the subjects hands. To this end, we employ the pose estimation procedure described in Chapter 6, i.e. we compute the range flow for every pixel in a local neighborhood around the estimated positions of the hands according to the procedure described in Section 7.4.1. For the hand position within a specific frame, this neighborhood includes every pixel in a 11×11 window centered around the position of the hand. In order to capture only the motion of the subject, we first segment the person and compute the range flow only for those pixels within the window that belong to the person. For the utilized segmentation algorithm refer to Section 5.1.2. An illustration of this procedure is given in Figure 7.16.

To obtain a compact representation of the local motion field around a hand, we accumulate the resulting 3D motion vectors in a 3D motion histogram, i.e. we trans-

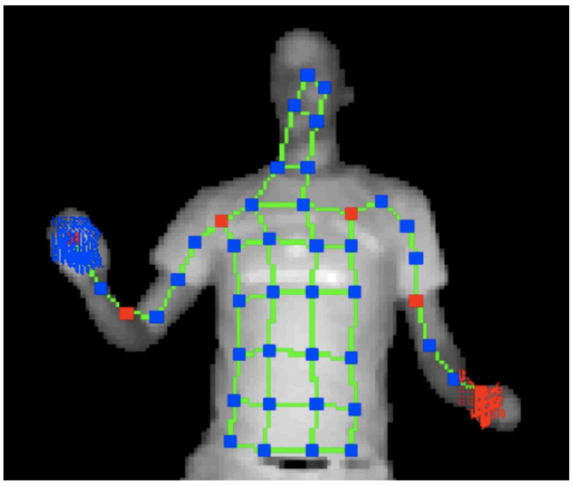


Figure 7.16: *Sample image of a person performing a gesture. The estimated model of the human pose is imprinted into the image. Furthermore, the range flow estimated in a local neighborhood around the hands is depicted.*

form the motion vectors $(u, v, w)^T$ into spherical coordinates $(\theta, \phi, r)^T$ and estimate the frequency of the vectors in different sectors of the sphere around the considered pixel. We use three bins for each of the spherical coordinates which yields a total number of 27 bins for the 3D motion histogram. This concept is depicted in Figure 7.17.

The above procedure yields a 3D motion histogram describing the range flow in a local spatial neighborhood around the estimated positions of both hands for every frame of the video sequence. These two histograms are composed into a feature vector $\vec{x}_i \in \mathbb{R}^{54}$ to characterize the motion of both hands with respect to a given frame i.

Our goal is to use these feature vectors to classify the performed gesture on a per frame basis. Note, however, that a complete gesture is typically performed over a number of successive frames. To pay respect to this observation, the final feature vector $\hat{\vec{x}}_i$ for a given frame i is obtained by averaging the feature vectors of the frames $j \in i-N, \ldots, i+N$:

$$\hat{\vec{x}}_i = \frac{1}{2N+1} \sum_{j=i-N}^{i+N} \vec{x}_j \; . \tag{7.21}$$

Here, the temporal extension of a gesture in terms of frames is defined by $2N + 1$, i.e. we assume that it takes a subject a certain number of frames to perform a gesture and we aim at detecting the frame that lies in the middle between the first and last frame of the gesture.

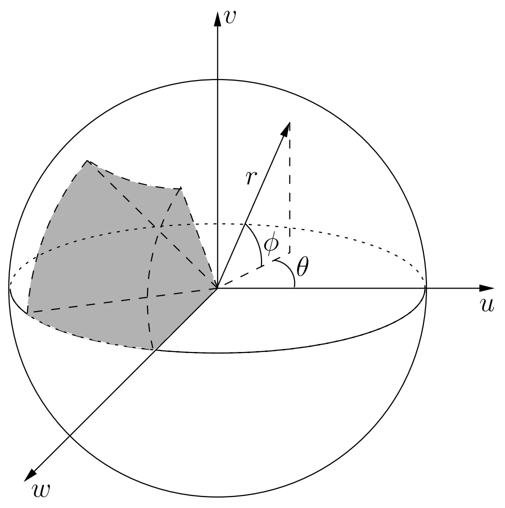


Figure 7.17: *Illustration of a 3D motion histogram. The motion vectors of a local neighborhood are transformed into spherical coordinates $(\theta, \phi, r)^T$. The vectors are then accumulated in a motion histogram. Intuitively speaking, the sphere is divided into sectors and we estimated the frequency of the motion vectors within each sector. An example of such a sector is depicted as a shaded volume.*

Training and Classification

Given the feature vectors described above, we trained a classifier based on a labeled training set. To this end, we recorded a number of video sequences in which a subject performed ten instances of a predefined gesture. Between each instance the subject was asked to return to a normal standing position, however, the length of the interval between different instances of the gesture was arbitrary. We then labeled each frame of the video sequence, i.e. we assigned each frame either to one of the corresponding gestures or it was labeled as belonging to a non-gesture sequence.

We investigated a total of three different gestures: (i) A *push* gesture where a subject extends the palm in front of the body, (ii) a *left-right* gesture where the subject sweeps a hand in front of his body from the left to the right, and (iii) a corresponding *right-left* gesture. Sample frames of these gestures are depicted in Figure 7.18.

The data was recorded using an SR4000 TOF camera from MESA Imaging and we recorded at a framerate of approximately ten frames per second. It took a subject less than a second to perform the core part of the gestures we aim to recognize. Thus, a typical gesture extends over roughly seven successive frames. By core part of the gesture, we refer to those frames that actually identify the gesture, excluding the frames that the user requires to move the hands to the starting position of the gesture.

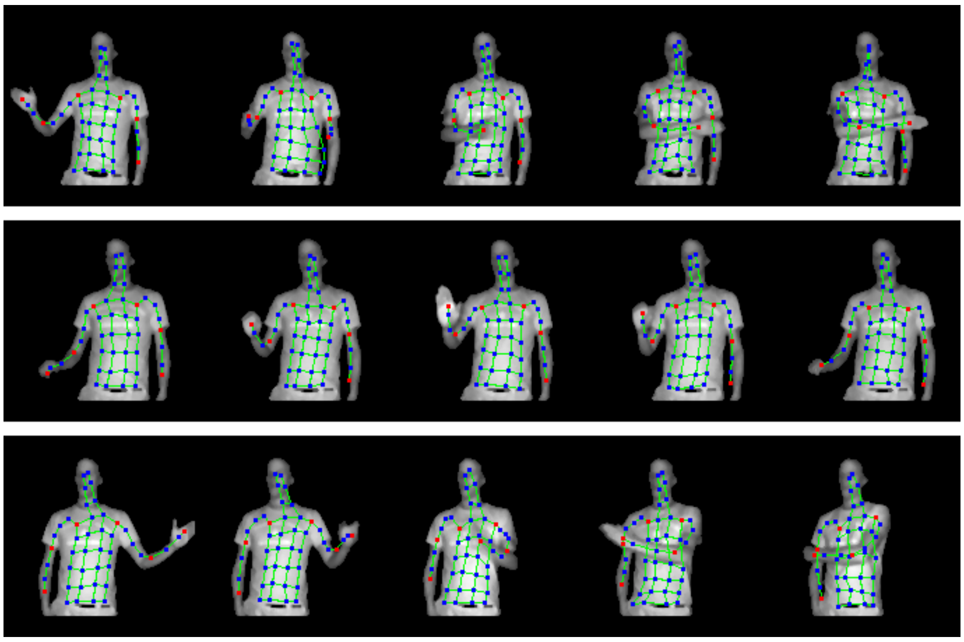


Figure 7.18: *Sample frames of the three investigated gestures. The top row shows the gesture where the subject moves a hand from right to left. The center row shows a push gesture. The bottom row depicts the user sweeping a hand from left to right. In each frame the estimated model of the pose is imprinted into the image.*

Taking the left-right gesture as an example, we consider only those frames as the core part in which the movement from the left to the right takes place. Given this observation, we evaluated Equation (7.21) for $N = 3$ to obtain a feature vector for every frame.

To increase the number of training samples, we extracted five feature vectors for each performed instance of a gesture from the training sequences – the second through the sixth frame of the core part of an instance. An efficient way to avoid high false positive rates was to also extract samples from those frames were no gesture was performed. These frames were selected at random. The resulting feature vectors were use to train a classification algorithm. After training, each frame of a new video sequence was classified by computing the corresponding feature vector and applying the previously learned decision rule.

In this context, we evaluated three different classification algorithms: (i) The k-Nearest-Neighbor classifier (kNN), (ii) the Nearest Mean classifier (NM), and (iii) the One-Class MaxMinOver classifier (OMMO) (Labusch et al., 2008b).

In case of the kNN, we assigned a frame to the gesture with the largest number of feature vectors in the neighborhood of the feature vector to be classified. As we are dealing with a multi-class problem, it is possible that two classes perform equally well. In such a case, the frame was classified as a no-gesture frame. As mentioned above, we also sampled the no-gesture class to reduce the number of false positives. However, this class is rather difficult to sample. Thus, we introduced another rule to improve the identification of frames in which no gesture was performed. To each class c we assigned a margin m_c in terms of a minimum distance to the closest feature vector among the nearest neighbors of that class, i.e. let $\vec{x}$ be a feature vector to be classified and $\vec{f_c}$ the closest feature vector of the winning class, then

$$\|\vec{x} - \vec{f_c}\| < m_c \tag{7.22}$$

must hold for $\vec{x}$ to be assigned to c. Otherwise, $\vec{x}$ is assigned to the no-gesture class. Intuitively speaking, we avoid assigning a feature vector to a gesture class if it differs too much from the representatives of that class. The margins can be viewed as hyper parameters of the kNN algorithm. We adjusted these parameters to achieve 100% detection rate on the training set while maintaining the lowest possible false positive rate. The same procedure was also applied to the NM classifier.

In case of the third type of decision rule, we trained an OMMO for each class including the no-gesture class. For a new frame, the OMMO classifiers for each class were evaluated. The frame was only assigned to a gesture class, if the classifiers of the other gesture classes rejected the frame. Otherwise, the frame was assigned to the no gesture class to avoid false positives. We trained a hard-margin classifier with a Gaussian kernel using the OMMO algorithm and thus had to adjust the kernel parameter σ. Again, the parameter was chosen to achieve high detection rates on the training set at the lowest possible false positive rate.

7.4.3 Results

In total, we recorded five data sets with different persons and evaluated three different gestures; the push gesture, the left-right gesture, and the right-left gesture. Each person performed each gesture ten times. Thus, the data set consisted of 150 instances of gestures – 50 per individual gesture.

For the evaluation of the method we applied the leave-one-out procedure with respect to the subjects, i.e. we trained a classifier for each gesture on the feature vectors computed for four subjects and did the evaluation on the data set of the fifth person. The hyper parameters of the considered classification algorithms (k for the kNN, the margins in case of both the kNN and the NM, and the kernel parameter

	PU	**RL**	**LR**	**NG**
PU	90.48	4.54	2.81	9.52
RL	0.00	95.65	0.00	4.35
LR	0.20	0.00	95.83	4.17
NG	0.77	0.73	0.17	97.68

Table 7.1: *Confusion matrix for the kNN classifier. The gesture classes are denoted as* **PU** *for the push gesture,* **RL** *for the right-left gesture,* **LR** *for the left-right gesture, and* **NG** *for the no-gesture class. Note that the entries on the diagonal and the last column, excluding the bottom right entry, were computed with respect to instances of a gesture while all other entries were computed with respect to individual frames. By this procedure we emphasize that we are interested in two different measures: (i) That each performed gesture is detected and (ii) that each individual occurrence of an erroneously classified frame counts as a false positive. Note, that this implies that the rows of the matrix do not automatically sum to one.*

	PU	**RL**	**LR**	**NG**
PU	76.19	4.10	0.43	23.81
RL	5.63	100.00	0.00	0.00
LR	0.00	0.00	91.67	8.33
NG	1.50	0.73	0.27	96.52

Table 7.2: *Confusion matrix for the NM classifier.*

σ for the OMMO) were adjusted on one of the leave-one-out configurations. This configuration was excluded from the test results presented here, i.e. the confusion matrices were averaged over the remaining four leave-one-out configurations.

The results of the three classification procedures in the proposed hand gesture recognition task are presented in Table 7.1 through Table 7.3. One can observe that the method detects the predefined gestures with high accuracy in most cases, typically with detection rates above 90%. At the same time, the procedure successfully distinguishes between those frames in which a gesture occurred from those were the user did not perform a gesture. Here, detection rates are above 95% for the kNN and the NM classifier and above 90% for the OMMO classifier. The overall performance appears to be best for the kNN classifier, i.e. it achieves the highest detection rates

	PU	**RL**	**LR**	**NG**
PU	63.64	7.83	1.30	36.36
RL	1.92	91.67	0.00	8.33
LR	0.00	0.00	100.00	0.00
NG	1.38	4.00	0.67	91.58

Table 7.3: *Confusion matrix for the OMMO classifier.*

	PU	**RL**	**LR**	**NG**
PU	61.90	0.00	0.00	38.10
RL	0.26	95.65	0.00	4.35
LR	0.40	0.00	83.33	16.67
NG	2.52	4.00	0.19	96.14

Table 7.4: *Confusion matrix for the kNN classifier based on 3D motion vectors extracted directly from human pose model.*

in most cases and especially for the push gesture while maintaining low confusion rates between the individual gestures.

In order to assess the advantage of using range flow as feature for 3D motion, we also used the estimated model of the human pose to directly extract the motion vectors of the hands by considering their 3D coordinates from two successive frames. We followed the same procedure as above and accumulated the motion vectors from seven successive frames in a 3D motion histogram to obtain a feature vector. The classification results using the kNN classifier are given in Table 7.4. One can observe, that the overall performance is significantly reduced in comparison to the results based on range flow presented above. Thus, we can deduce that range flow serves as a robust feature for characterizing object motion in TOF image data.

7.4.4 Discussion

In this section, we presented an approach to the classification of hand gestures. The main goal was to achieve high detection rates while also handling the situation, where a user is not performing any gesture at all. To this end, we proposed highly discriminative features that combine knowledge of the human pose with 3D motion estimation, i.e. we extracted 3D motion histograms based on the computation of range flow at the priorly estimated image locations of the hands. This approach has two advantages. First, the computationally intensive estimation of range flow can be kept to a minimum for each frame because we require the range flow only in local neighborhoods around the estimated positions of the hands. Furthermore, this approach allows us to assign the motion information directly to body parts instead of to an entire image or fixed image region.

Although the database for evaluating the proposed procedure is very limited and this section presents only preliminary work, we obtained quite promising results. We evaluated three different classification algorithms. The best performance was achieved by the kNN algorithm, i.e. it yielded detection rates ranging from 90.48% to

95.83% while the situation where no gesture was performed was detected with an accuracy of 97.68%. This was achieved at confusion rates with the other gestures ranging from 0.00% to 4.54%. Future work should aim at investigating a larger database because one may expect the decision rules to be trained with higher accuracy. This is especially the case for the OMMO classifier where we assume the poor performance to be due to the limited amount of training data. At the same time, the results may be evaluated in a more representative fashion. Furthermore, we expect an increase in detection performance if one does not classify single frames. Instead, the method should be extended to consecutive sequences of frames in which a gesture must be detected before it is recognized.

Part III

Applications

8 Introduction

In Part II we have introduced a collection of algorithms for TOF camera data. These include the segmentation of foreground objects (see Section 5), the estimation of human pose (see Section 6), and the tracking of facial features (see Section 7). In the scope of the third part of this thesis, we will discuss two applications for human-computer-interaction that make use of these results. First we will investigate how the nose can be used to control a mouse cursor for an alternative text input application. Then we will discuss the use of deictic gestures for the control of a slide show presentation. Finally, we will turn to the topic of digital photography, where we will demonstrate how a TOF camera can be used to generate depth of field in digital images in a post-processing stage.

Facial Feature Tracker

We describe a facial feature tracker based on the combined range and amplitude data provided by a TOF camera. We use this tracker to implement a head mouse, an alternative input device for people who have limited use of their hands.

The facial feature tracker is based on the geometric features introduced in Section 7.1 that are related to the intrinsic dimensionality of multidimensional signals. In Section 7.1 we also showed how the position of the nose in the image can be determined robustly using a very simple bounding-box classifier, trained on a set of labelled sample images. Despite its simplicity, the classifier generalizes well to subjects that it was not trained on. An important result is that the combination of range and amplitude data dramatically improves robustness compared to a single type of data.

The resulting facial feature tracker runs in real time at around 30 frames per second. We demonstrate its potential as an input device by using it to control Dasher, an alternative text input tool.

Deictic Gestures

We present a robust detector for deictic gestures based on a TOF camera. Pointing direction is used to determine whether the gesture is intended for the system at all and to assign different meanings to the same gesture depending on pointing direction. We use the gestures to control a slideshow presentation: Making a “thumbs-up” gesture while pointing to the left or right of the screen switches to the previous or next slide. Pointing at the screen causes a “virtual laser pointer” to appear. Since the pointing direction is estimated in 3D, the user can move freely within the field of view of the camera after the system was calibrated. Near the center of the screen the pointing direction is measured with an absolute accuracy of 0.6 degrees and a measurement noise of 0.9 degrees.

Depth of Field Based on Range Maps

Depth of field is one of the most important stylistic devices in photography. It allows the photographer to guide the spectators attention to certain features in the image by tuning the camera parameters in such a way that only the important portions of scene at the focused distance appear sharp while other objects appear blurred.

This effect requires certain features of the optical system, such as a large aperture. On the contrary, modern digital camera become smaller and smaller in size and, thus, incapable of creating the effect of depth of field.

We propose an algorithm based on Distributed Raytracing that uses a range map acquired with a TOF camera to simulate the effect of depth of field in a post-processing step. We compare our approach, referred to as $2^1/_2$D Distributed Raytracing, to previously published approaches and apply the algorithm to both synthetic and real image data.

9 Facial Feature Tracking

9.1 Introduction

In this chapter, we use a TOF camera to implement a facial feature tracker and show how this tracker can be used as an alternative means of controlling a mouse cursor.

In contrast to conventional cameras, TOF cameras have the obvious benefit of providing information about the three-dimensional structure of the scene. Computer vision applications that use this information can reasonably be expected to be more robust than if they used only a conventional intensity image, because geometric structure is typically more invariant than appearance.

Previous work has already identified the nose as an important facial feature for tracking, see for example (Gorodnichy, 2002) and (Yin and Basu, 2001). The former approach determines the location of the nose by template matching and gives very robust results under fixed lighting conditions and at a fixed distance of the user from the camera. The latter approach works by fitting a geometrical model of the nose to the image data.

We also consider the nose as being well suited for head tracking because, in terms of differential geometry, the tip of the nose is the point of maximal curvature on the surface of the face. This observation allows the nose to be tracked robustly in range maps provided by a TOF camera. In the scope of differential geometry, the Gaussian curvature would seem to be a good choice as an image feature for tracking the nose, but it has the disadvantage that it is based on first and second order derivatives, which are sensitive to noise. In Section 7.1 we proposed alternative features that can be related to generalized differential operators. In Section 7.1.4, we described a

Parts of this chapter are joint work with others. Martin Böhme and I contributed approximately equally to the implementation of the nose tracker. The work described here has previously been published in (Böhme et al., 2008a).

nose detector based on these features that achieves good error rates using a simple threshold-based classifier.

The nose detector described previously in Section 7.1 was a Matlab implementation that did not run in real time. In this chapter, we describe a real-time C++ implementation of the nose detector, which we then use as the basis for a nose tracker. Because of the low error rate of the detector, the tracker requires only simple post-processing and tracking algorithms. We envisage that this type of tracker can be used for human-computer interaction and, particularly, for Augmentative and Alternative Communication. To demonstrate this type of application, we use the tracker to control the alternative text entry tool Dasher proposed by Ward and MacKay (2002).

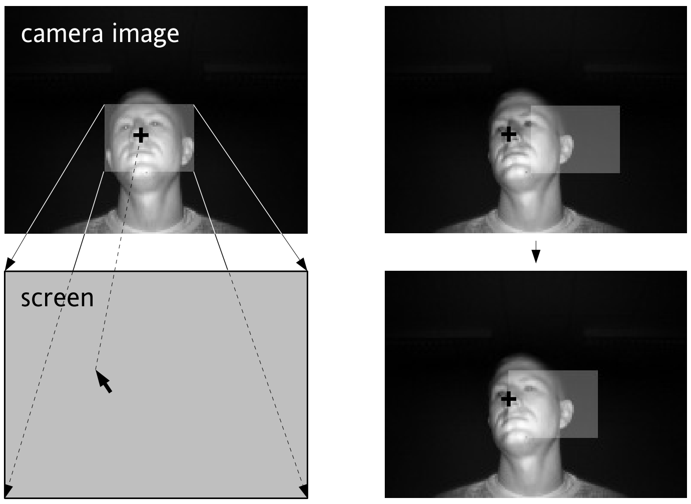
9.2 Nose Tracking

To track the nose and use it to control the position of the mouse pointer, we need to choose one of the potentially several nose locations that were detected, and we need to translate this nose location into a mouse cursor position.

First, we find all of the connected components in the image; each connected component yields one candidate location for the position of the nose, which we determine as follows: We first find the pixel whose feature vector is closest to the center of the bounding box, i.e. for which $||\mathrm{F} - \mathrm{F}_{\mathrm{centre}}||_2$ is minimal. We then refine this location with subpixel precision by taking a small window around the candidate location and computing a weighted centroid of the pixels in the window, where the weight of a pixel depends on the distance of its feature vector to $\mathrm{F}_{\mathrm{centre}}$; specifically, the weight function is a Gaussian centered on $\mathrm{F}_{\mathrm{centre}}$.

Since the raw results delivered by the nose detector are already quite good (see Section 9.3), most camera frames contain only a single connected component and thus a single candidate location, placed correctly on the nose. If the current frame contains more than one candidate location, we choose the candidate that is closest to the position of the nose in the previous frame. For the very first frame, or if no nose was found in the previous frame, we simply choose one of the candidates arbitrarily. Even if we choose the wrong candidate, it is quite likely that the next frame will not contain any misdetections, and from that point on, we will be tracking the correct position. This is a simple strategy for eliminating misdetections, but it works quite well in practice.

Once the position of the nose has been identified, we need to convert it into a position on the screen. We define a rectangular "active region" in the camera image,



a) b)

Figure 9.1: *a) Illustration of the mapping between the position of the nose in the camera image and the position of the mouse cursor on the screen. The shaded rectangle is the active region, the cross marks the position of the nose, and the arrow symbolizes the mouse cursor. For clarity, the horizontal flip contained in the mapping is not shown. b) Adjustment made when the nose moves outside the active region. Top: Before the adjustment of the active region. Bottom: After the adjustment.*

sized so that when the user rotates his or her head, they are able to place their nose anywhere within this region. We then convert the nose position to a mouse cursor position by defining a mapping from camera image coordinates to screen coordinates such that the borders of the active region are mapped to the borders of the screen (see Figure 9.1a). The mapping has to flip the horizontal axis (for clarity, this is not shown in the figure) because a head movement to the left causes the position of the nose in the camera image to move to the right.

Two questions remain to be solved: Where in the image should we place the active region? And what should be done if the user's seating position changes such that the nose leaves the active region? We solve these problems using an implicit calibration procedure that continually adjusts the position of the active region. For example, if the nose moves beyond the left border of the active region, we shift the active region to the left until the nose is just inside the active region again (see Figure 9.1b). Effectively, users "drag" the active region along as they change their seating position. This procedure works equally well for setting the initial position of the active region: A quick rotation of the head left, right, up, and down is sufficient to put the active region in a suitable position. If desired, the size and initial position of the active region can also be determined during an explicit calibration phase.

9.3 Results

The algorithms were implemented in GNU C++ and tested on a 2.8 GHz Pentium 4 under Linux 2.6. The FFTW library (Frigo and Johnson, 2005) was used for the FFT transforms needed to calculate the generalized eccentricity measures. Images were acquired using a MESA SR3000 camera (Oggier et al., 2005a), which has a resolution of 176 by 144 pixels.

To evaluate the error rate of the nose detector, we first tested it on a set of static images, consisting of images from 13 subjects, with nine head orientations per subject. The position of the nose was hand-labelled in each image. The data from three subjects was used to estimate the bounding box for the nose detector, which was then tested on the remaining ten subjects. We calculated the detection rate as the percentage of images in which the nose was identified correctly (to within five pixels distance from the hand-labelled position) and the false positive rate as the percentage of images where at least one non-nose pixel was falsely classified as “nose”. We adjusted the softness parameter until the false negative rate (one minus the detection rate) and the false positive rate were equal, thus obtaining an equal error rate (EER). When nose detection was performed on the range and amplitude data individually, we obtained EERs of 0.64 and 0.42, respectively, When nose detection was performed on the combination of both types of data, we obtained an EER of 0.03 (see (Haker et al., 2007) for details). Figure 9.2 shows some examples of detection results.

The softness parameter that corresponded to the EER was chosen to set the bounding box for the nose tracker. On the combined range and amplitude data, the tracker ran at a rate of 27 frames per second. This is adequate for the frame rate we run the camera at but could be improved further (for example, we currently use a complex-to-complex FFT for simplicity of implementation, which could be replaced with a real-to-complex FFT). A video showing the nose tracker in action is available on the web at `http://www.artts.eu/demonstrations`.

To evaluate the usefulness of the nose tracker for interaction tasks, we used it to control the alternative text input tool Dasher proposed by Ward and MacKay (2002). Dasher can be controlled using a variety of pointing devices, which are used to “steer” towards the letters to be input. A test subject using the nose tracker with Dasher to input text from an English language novel achieved 12 words per minute (wpm). For

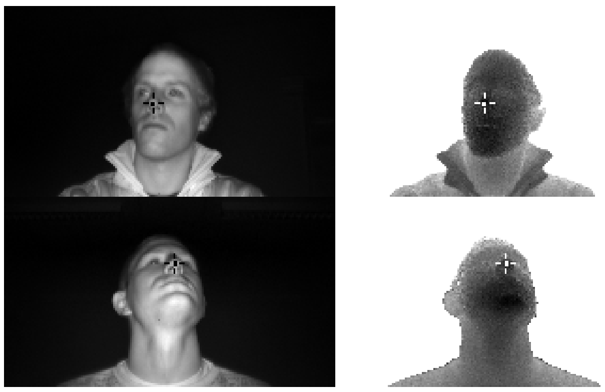


Figure 9.2: *Examples of detection results. The amplitude data (left column) and the range data (right column) are given for four subjects. All pixels identified as nose pixels by our detector are marked in each image, the cross simply highlighting the locations.*

comparison, the rate that can be achieved using an eye tracker is between 15 and 25 wpm (Ward and MacKay, 2002).

9.4 Conclusions

We have demonstrated how a very simple classifier based on geometric features can be used to detect a user's nose robustly in combined range and amplitude data obtained using a TOF camera. A particularly interesting result is that markedly better results were obtained on the combination of range and amplitude data than on either type of data alone.

Based on this classifier, we implemented a real-time nose tracker. To demonstrate the usefulness of this tracker for human-computer interaction, we have shown that it can be used effectively to control the alternative text entry tool Dasher.

10 Gesture-Based Interaction

10.1 Introduction

In this chapter, we use gestures recognized using a TOF camera to control a slideshow presentation, similar to (Baudel and Beaudouin-Lafon, 1993) where, however, a data glove was used to recognize the gestures. Another idea we adapt from Baudel and Beaudouin-Lafon (1993) is to recognize only gestures made towards an "active area"; valid gestures made with the hand pointing elsewhere are ignored. This solves the problem (also known as the "immersion syndrome") that unintentional hand movements or gestures made towards other people may erroneously be interpreted as commands.

We expand this idea by allowing the same gesture to mean different things when made towards different active areas. Specifically, the slideshow is controlled in the following way: To go to the next slide, point to the right of the screen and make a thumbs-up gesture with the hand; to go to the previous slide, point to the left of the screen and make a thumbs-up gesture. Point at the screen and a dot appears at the location you are pointing to, allowing you to highlight certain elements of the slide. This scenario is depicted in Figure 10.1.

To determine where the user is pointing on the screen, we need to know its position relative to the camera. This is determined in a calibration procedure where the user points at the four corners of the screen from two different locations; this information is sufficient to compute the position of the screen. After calibration the user is allowed to move freely within the field of view of the camera, as the system estimates both the screen position and the pointing direction in 3D with respect to

Parts of this chapter are joint work with others. I devised and implemented the concept of deictic gestures. Martin Böhme and I contributed approximately equally to the evaluation of the accuracy of the pointing gesture. The work described here has previously been published in (Haker et al., 2009a).

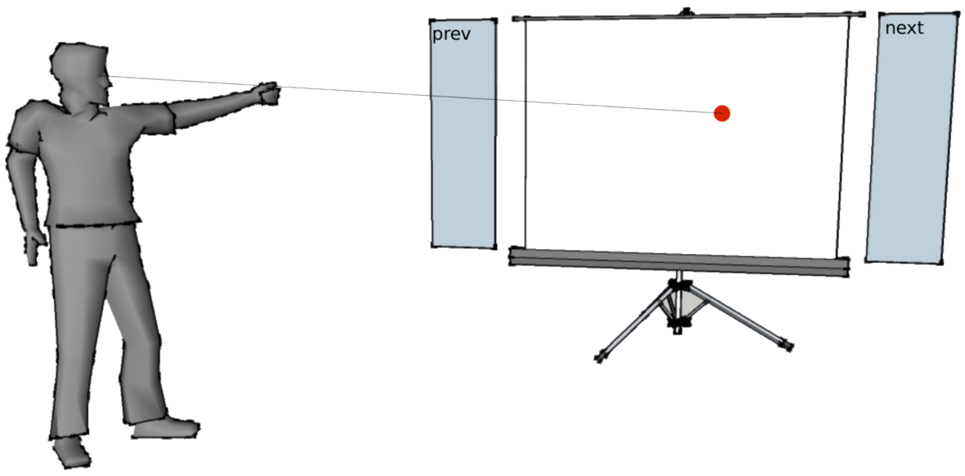


Figure 10.1: *The application scenario where a user controls a slideshow presentation using deictic gestures. The gestures include switching between the slides and pointing at the screen using a virtual laser pointer.*

the camera coordinate system.

The "thumbs-up" gesture is recognized using a simple heuristic on the silhouette of the hand. This simple technique is sufficient because hand gestures are only recognized when the user is pointing at one of the two active regions; when pointing elsewhere, the user need not be concerned that hand movements might be misinterpreted as gestures.

One important advantage of the TOF camera in this setting is that it directly measures the three-dimensional position of objects in space, so that the pointing direction can easily and robustly be obtained as a vector in space. This is much more difficult for approaches that attempt to infer pointing direction using a single conventional camera. One solution is to restrict oneself to pointing directions within the camera plane (see e.g. (Hofemann et al., 2004; Moeslund and Nøregaard, 2006)), but this places restrictions on the camera position and type of gestures that can be recognized. A physical arm model with kinematic constraints (see e.g. (Moeslund and Granum, 2003)) allows arm pose to be estimated from a single camera image, but the depth estimation can be unreliable for some poses of the arm. In contrast, the approach we will present here is simple but at the same time accurate and robust.

In the remainder of the chapter we will first discuss the detection of the pointing and the "thumbs-up" gesture. We will then describe the calibration of the system. Finally, the accuracy of the virtual laser pointer will be evaluated in an experimental setup where users had to point at given targets. This evaluation is conducted in two scenarios: One, where the user does not receive visual feedback, and another, where the estimated pointing position is indicated by the virtual laser pointer.

10.2 Method

Our method can be divided into three individual components: (i) the detection and interpretation of the pointing gesture, (ii) the recognition of the thumbs-up gesture used for navigating between slides, and (iii) the calibration of the system. This section will cover each component in the order mentioned above. For simplicity, we assume that the user always points towards the left as seen from the camera throughout this section although this is not a restriction of the system.

10.2.1 Pointing Gesture

The algorithm for detecting pointing gestures can be subdivided into four main stages. The first stage segments the person in front of the camera from the background. The second stage uses the segmented image to identify both the head and the extended hand that is used for pointing. During the third stage, the 3D coordinates of head and hand in space are estimated, which are then used to determine the location on the screen the user is pointing to during the fourth stage. In the following, we will discuss each step of this procedure individually in more detail.

Stage 1: The segmentation of the person in front of the camera uses combined information from both the range and intensity data of the TOF camera. Previous work by Haker et al. (2007, 2009b) has shown that the combined use of both range and intensity data can significantly improve results in a number of different computer vision tasks. We determine adaptive thresholds for range and intensity based on histograms as described in Section 5.1.2. The final segmented image is composed of those pixels that were classified as foreground pixels with respect to both types of data. To ensure that only a single object is considered, only the largest connected component of foreground pixels is retained, all other objects are considered background. A sample TOF image showing both range and intensity with the resulting segmented foreground is given in Figure 10.2.

Stage 2: In the second stage, the segmented image is used to determine an initial guess for the location of the head and hand in the image. We employ a simple heuristic based on the number of foreground pixels in each column of the segmented image.

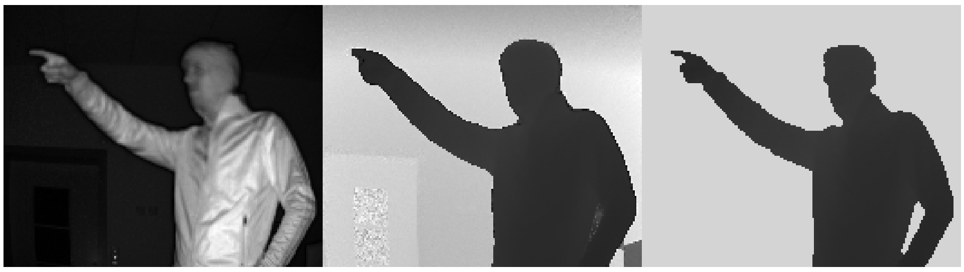


Figure 10.2: *Sample image taken with a MESA SR4000 TOF camera. The leftmost image shows the intensity data. The range image is given in the center, and the resulting segmentation is shown on the right.*

The initial guess for the hand is the topmost pixel in the leftmost pixel column of the silhouette; the head is the topmost pixel in the tallest pixel column. This procedure is extremely simple to implement, yet reliable. We use a single parameter θ to determine whether we have a valid initial estimate, i.e. whether the hand is actually extended and the person is performing a pointing gesture:

$$|i_{\text{head}} - i_{\text{hand}}| \geq \theta \; . \tag{10.1}$$

Here, i_{head} and i_{hand} denote the indices of the pixel columns corresponding to the initial guess for the head and hand, respectively, where indices of pixel columns increase from left to right.

Stage 3: During the third stage of the method, the initial guesses are refined to more accurate pixel positions in the image. Once these positions are determined, the corresponding range values are estimated, and finally the coordinates of both the head and hand can be computed in 3D by inverting the perspective camera projection using the known intrinsic camera parameters.

In order to refine the pixel positions of the head and hand in the image, we define rectangular regions of interest (ROIs) around their estimated locations from the previous stage and compute the centroids of the foreground pixels in the ROIs to find the centers of the head and hand blobs; these refined positions are marked by crosses in Figure 10.3.

To invert the camera projection we require the actual distance of the head and hand from the camera. Again, we define ROIs around the estimated pixel coordinates and take the average range value of the foreground pixels within the ROI to obtain estimates for the two range values. Finally, from the pixel coordinates (x, y), the

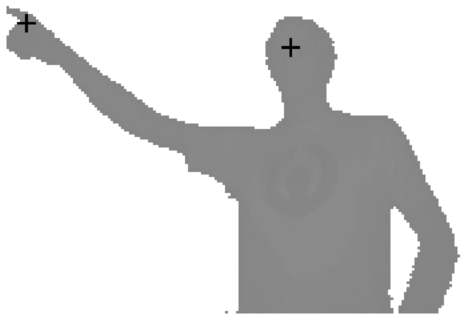


Figure 10.3: *Segmented image of the user with the detected locations of the head and hand marked by crosses. The time-of-flight camera measures the three-dimensional positions of these points, which are then used to compute the pointing direction.*

distance from the camera r, and the intrinsic parameters of the camera one can infer the 3D coordinates of the corresponding point $\vec{x}$ in camera coordinates using the following formula:

$$\vec{x} = r \frac{((c_x - x) \cdot s_x, (c_y - y) \cdot s_y, f)^T}{\|((c_x - x) \cdot s_x, (c_y - y) \cdot s_y, f)^T\|_2} . \tag{10.2}$$

Here, (c_x, c_y) denotes the principal point, i.e. the pixel coordinates of the point where the optical axis intersects the image sensor. The width and height of a pixel are defined by s_x and s_y, and the focal length is given by f. To obtain a more stable estimate, a Kalman filter (Kalman, 1960) tracks the 3D coordinates of the head and hand from frame to frame.

Stage 4: Because the TOF camera allows us to determine the position of the head and hand in space, we directly obtain an estimate for the pointing direction from the ray that emanates from the head and passes through the hand. As Nickel and Stiefelhagen (2003) show, the line connecting the head and hand is a good estimate for pointing direction. This ray can be represented in camera coordinates by the following line equation:

$$\vec{r} = \vec{o} + \lambda \vec{d} . \tag{10.3}$$

Here, $\vec{o}$ denotes the origin of the ray and corresponds to the 3D position of the head. The direction of the ray is given by $\vec{d} = \vec{p} - \vec{o}$, where $\vec{p}$ denotes the position of the hand. The parameter $\lambda \geq 0$ defines a point $\vec{r}$ in front of the person along the pointing direction.

We now intersect this ray with the screen used for projecting the slides. To this end we represent the screen by its center $\vec{c}$ and the normal $\vec{n}$ of the screen plane. Assuming that both $\vec{c}$ and $\vec{n}$ are also given in camera coordinates, the intersection $\vec{x}$ of the ray and the screen is given by:

$$\vec{x} = \vec{o} + \frac{\langle \vec{c} - \vec{o},\ \vec{n} \rangle}{\langle \vec{d},\ \vec{n} \rangle} \vec{d} \quad . \tag{10.4}$$

The intersection is only valid if the scalar product $\langle \vec{c} - \vec{o},\ \vec{n} \rangle$ is positive, otherwise the user is most likely pointing away from the screen plane.

What remains to be determined is if the intersection lies within the limits of the screen. In that case, the intersection can be converted to pixel coordinates on the screen in order to display the virtual laser pointer.

The location and size of the screen are determined by the calibration procedure introduced in Section 10.2.3. Since the procedure determines the 3D position of the four corners of the screen independently, the screen is generally not represented by a perfect rectangle. Thus, we determine the intersection by considering two triangles that are obtained by dividing the screen diagonally along the line from the bottom left corner to the top right corner. Assume that the triangles are defined by their three corners $\vec{a}$, $\vec{b}$, and $\vec{c}$ in counter-clockwise order such that either the top left or the bottom right corner are represented by $\vec{a}$. For both triangles one can solve the following equation under the constraint that $d_1 = 1$:

$$\vec{x} = \begin{pmatrix} a_1 & b_1 - a_1 & c_1 - a_1 \\ a_2 & b_2 - a_2 & c_2 - a_2 \\ a_3 & b_3 - a_3 & c_3 - a_3 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \quad . \tag{10.5}$$

Intuitively, we check if the intersection $\vec{x}$, represented as a linear combination of the two sides of the triangle given by $\vec{b} - \vec{a}$ and $\vec{c} - \vec{a}$, lies within the bounds of the triangle. Thus, if $d_2 + d_3 \leq 1$ holds for the upper triangle, the intersection $\vec{x}$ lies above the diagonal through the screen. Correspondingly, $\vec{x}$ lies below the diagonal if $d_2 + d_3 \leq 1$ holds for the lower triangle.

We now convert the coefficients d_2 and d_3 to coordinates x and y on the screen in such a way that the top left corner corresponds to $(x, y) = (0, 0)$ and the bottom right corner corresponds to $(x, y) = (1, 1)$. This is achieved by setting $(x, y) = (d_2, d_3)$ if $\vec{x}$ was above the diagonal through the screen and setting $(x, y) = (1 - d_2, 1 - d_3)$ otherwise. As a result one obtains for example the four different interpretations of the pointing gesture listed in Table 1. These interpretations correspond

Table 10.1: *Interpretation of pointing gesture*

on screen	0.0	$\leq x \leq$	1.0	$\wedge$	$0.0 \leq y \leq 1.0$
left of screen	-0.05	$\leq x <$	0.0	$\wedge$	$0.0 \leq y \leq 1.0$
right of screen	1.0	$< x \leq$	1.05	$\wedge$	$0.0 \leq y \leq 1.0$
off screen	otherwise				

to the scenario depicted in Figure 10.1.

In the “on screen” case, the virtual laser pointer is displayed on the screen at the location (x, y) the user is pointing to. If the user is pointing to one of the two active areas “left of screen” or “right of screen”, a small triangle is displayed at the corresponding edge of the screen to indicate that the system is now expecting input in form of the “thumbs-up” gesture to navigate between the slides of the presentation. In all other cases, any detected pointing gesture is ignored, which avoids the so-called immersion syndrome (Baudel and Beaudouin-Lafon, 1993).

Despite the fact that the estimation of the head and hand is quite robust and we apply a Kalman filter to the approximated 3D coordinates for temporal smoothing, the estimated intersection of the pointing direction and the screen in the “on screen” case is not entirely free of noise. This is dealt with by applying a smoothing filter with an exponential impulse response. The strength of the smoothing is adaptive and depends on the amount by which the pointing position changed: The greater the change, the less smoothing is applied. In this way, we suppress “jitter” in the virtual laser pointer when the user’s hand is stationary but allow the pointer to follow large hand movements without the lag that would be caused by a non-adaptive smoothing filter.

10.2.2 Thumbs-up Gesture

The detection of the thumbs-up gesture is only triggered when a pointing gesture made towards one of the two active areas was detected for the current frame according to the procedure described above.

The thumbs-up detector uses the segmented image and the pixel coordinates that were estimated for the hand. The main idea of the algorithm is that the silhouette of an extended thumb that points upwards is significantly narrower along the horizontal axis than a fist.

Thus, we define an ROI around the position of the hand and count the number of foreground pixels in each row. Next, we estimate w_{fist}, which denotes the width of the fist, by taking the maximum number of foreground pixels counted per row. The parameter w_{fist} is then used to determine the presence of both the fist and the thumb. We count the number c_{fist} of rows containing at least $0.8 \cdot w_{\text{fist}}$ foreground pixels and the number c_{thumb} of rows containing at least one and at most $0.3 \cdot w_{\text{fist}}$ foreground pixels. A thumb is detected in the current frame if both c_{fist} and c_{thumb} exceed a threshold of two. Due to the fact that the thresholds for detecting the fist and thumb depend on the estimated width of the fist w_{fist} in the current image, the procedure is relatively independent of the distance of the user from the camera, i.e. the algorithm is scale-invariant.

To avoid misdetections due to noise, we keep track of the detections of the thumb per frame over a certain time window, i.e. the command for switching to the next slide is only issued if the thumbs-up gesture was detected in four out of six consecutive frames. At the same time, we want to avoid multiple activations of the command for switching to the next slide if the above criterion is fulfilled in a number of consecutive frames. Otherwise, the user would not be able to go from one slide to the next in a controlled fashion without unintentionally skipping slides. Thus, we ignore any detections of the gesture for a total of 50 frames once a switch-to-next-slide command was issued. Since our system operates at roughly 25 Hz, a new command can only be issued every two seconds. This gives the user sufficient time to end the thumbs-up gesture once the gesture takes effect in order to prevent the system from switching directly to the next slide.

10.2.3 System Calibration

The third component of our method deals with the calibration of the system. To determine where the user is pointing on the screen, we need to know its position relative to the camera. This is determined in a calibration procedure where the user points at the four corners of the screen from two different locations; this information is sufficient to compute the position of the screen, as we will demonstrate in more detail in the following.

During calibration the user is asked to point continuously at one of the four corners of the screen for a total of 50 frames. This allows us to obtain a robust estimate for the position of the head and hand for the given pointing direction. Again, the pointing direction can be represented by a ray $\vec{r} = \vec{o} + \lambda \vec{d}$ that emanates from the head and passes through the hand. We can assume that this ray passes through the corner the user was pointing at. However, we do not know the exact location of the corner along the ray.

To obtain this information, the user is asked to move to a different position in the field of view of the camera and to point again at the same corner for a total of 50 frames. By this procedure we estimate a second ray that should also pass through the corner of the screen. Ideally, the two rays intersect at the position of the corner; however, this assumption does generally not hold due to measurement noise. Nevertheless, a good estimate for the position of the corner can be obtained from the point that is closest to both rays in 3D space.

Assuming that the two estimated pointing directions are represented by rays $\vec{r_i} = \vec{o_i} + \lambda_i \vec{d_i}$ where $i \in \{1, 2\}$, one can obtain this point by minimizing the squared distance $\|\vec{o}_1 + \lambda_1 \vec{d}_1 - \vec{o}_2 - \lambda_2 \vec{d}_2\|_2^2$ between the two rays with respect to λ_1 and λ_2. This leads to the following linear system of equations where we assume $\|\vec{d_i}\|_2 = 1$ without loss of generality:

$$\begin{pmatrix} 1 & \langle \vec{d}_1, \vec{d}_2 \rangle \\ -\langle \vec{d}_1, \vec{d}_2 \rangle & -1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} -\langle \vec{o}_1 - \vec{o}_2, \vec{d}_1 \rangle \\ -\langle \vec{o}_1 - \vec{o}_2, \vec{d}_2 \rangle \end{pmatrix} \tag{10.6}$$

Solving Equation (10.6) yields the parameters λ_1 and λ_2, which specify the closest point on one ray with respect to the other ray, respectively. Taking the arithmetic mean of both solutions as specified by Equation (10.7) yields the approximation of the intersection of both rays and, thus, an estimate for the position of the corner in camera coordinates:

$$\vec{x} = 0.5 \cdot (\vec{o}_1 + \lambda_1 \vec{d}_1 + \vec{o}_2 + \lambda_2 \vec{d}_2) \tag{10.7}$$

This procedure can be repeated for the remaining three corners of the screen. The approach does not guarantee, however, that all four corners lie in one plane. Thus, we fit a plane through the four corners by least squares and project the corners onto this plane to obtain their final estimates. The normal to this plane and the four corners are used to determine where the user is pointing on the screen, as described in Section 10.2.1. Obviously, this calibration procedure does not generally yield a screen that resembles a perfect rectangle in 3D space. How this problem can be treated by dividing the screen into two triangles along its diagonal was also discussed in Section 10.2.1.

We consider the procedure of not enforcing the screen to be rectangular as an advantage, because it provides an implicit way of correcting systematic errors. Such errors may for example be caused by measurement errors or a simplified approximation of the camera parameters. The former can e.g. be due to multiple reflections of the scene (Gudmundsson et al., 2007b). The latter can occur if the optical system is not modelled accurately in the process of inverting the camera projection, e.g. if radial distortions of the lens or other effects are not taken into account.

10.3 Results

The method was implemented in C++ under the Windows operating system. On a 2 GHz Intel Core 2 Duo, it requires 40 ms per frame, achieving a frame rate of 25 frames per second.

To assess the accuracy with which the pointing direction is measured, we performed a test with 10 users. Each user was first given a few minutes to practice using the system. We then presented a sequence of nine targets at predefined positions on the screen; users were instructed to point at a target as soon as it appeared. Once a pointing gesture towards the screen was detected, each target was presented for a total of 50 frames, which corresponds to a time interval of roughly two seconds, before it disappeared. Users were asked to return to a normal standing position after the target had disappeared. Before presenting the next target, the system waited for four seconds to allow the user to rest the arm. The order in which the targets appeared was chosen in such a way that the average distance between successive targets on the screen was maximized.

For each user, we performed this test under two different conditions: Under the first condition, the virtual laser pointer was switched off, i.e. the users did not receive any feedback about the measured pointing direction. This gives an impression of the overall accuracy of the system. For the second test condition, the virtual laser pointer was switched on, allowing users to compensate for systematic calibration and measurement errors. This test condition therefore gives an impression of the residual measurement noise after the temporal smoothing described in Section 10.2.1.

Figure 10.4a shows the results of the first test condition (without visual feedback) to assess the overall accuracy of the system. Here, measured error in pointing direction can have two sources: (i) Systematic errors due to measurement noise and inaccurate calibration and (ii) errors induced by the assumption that the ray emanating from the head across the hand corresponds to the natural human pointing direction (Kranstedt et al., 2006). The horizontal axis plots the frame number after the pointing gesture was detected, and the vertical axis plots the distance between the target and the measured pointing position in pixels. In the test setup the screen

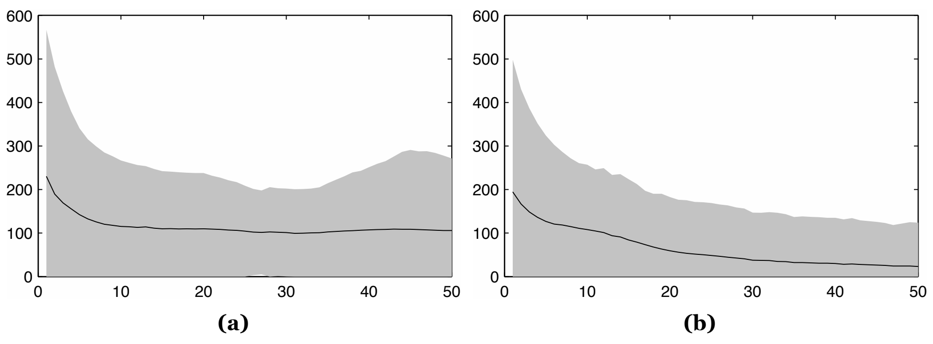


Figure 10.4: *Measurement error in pointing direction (a) without visual feedback (the virtual laser pointer was switched off) and (b) with visual feedback (virtual laser pointer switched on). The horizontal axis plots the time in seconds after the target appeared, and the vertical axis plots the distance between the target and the measured pointing position in pixels. In our setup, an offset of 20 pixels corresponds to approximately one degree.*

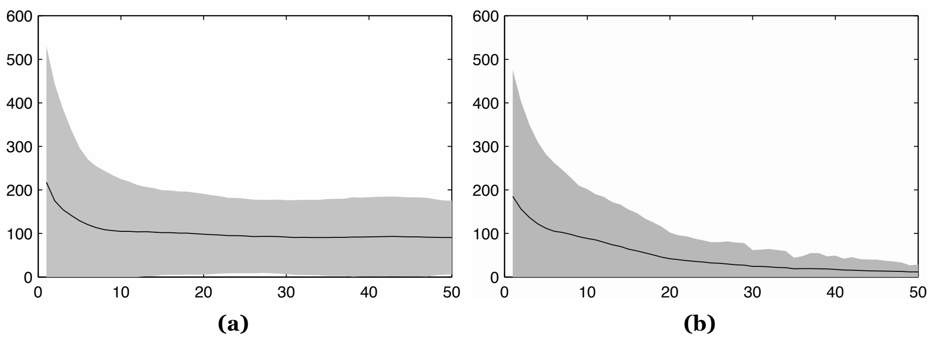


Figure 10.5: *Measurement error in pointing direction (a) without visual feedback (the virtual laser pointer was switched off) and (b) with visual feedback (virtual laser pointer switched on). Only targets near the center of the screen where considered.*

had a size of $1.71\text{ m} \times 1.29\text{ m}$ and the user was standing at a distance of roughly 3.2 m from the screen. As a result, an offset of 20 pixels corresponds to approximately one degree. The solid line gives the mean distance, averaged over all users and pointing targets, and the shaded area indicates an interval of two standard deviations above and below the mean, i.e. 95% of the errors fell within this range.

From the plot, we can see that users took around 10 frames or 400 ms to point at a target; after this time, the average error stabilizes at around 106 pixels (or 5.3 degrees), with 95% of error falling between 0 and 271 pixels.

Figure 10.4b shows the results of the second test condition (with visual feedback). As expected, the error stabilizes at a lower value of 23 pixels (or 1.2 degrees) on average but also takes a longer time to do so – around 1600 ms. This is because, after pointing at the target as in the first test, users need to correct their hand position to compensate for the systematic measurement error and bring the virtual laser pointer onto the target.

A closer look at the data reveals that the largest measurement errors occur for targets that are close to the corners of the screen. This is mainly due to shortcomings of the rather simple calibration procedure. However, the system is well calibrated for the center of the screen where most of the content of a presentation is likely to be placed. Thus, the impact of calibration errors near the corners on the usability is rather low. This becomes clear by looking at Figure 10.5a and Figure 10.5b. Again, the plots show the distance between the target and the measured pointing position without and with visual feedback, respectively. This time however, only targets near the center were considered, i.e. targets with an offset from the center of the screen corresponding to less than 15 degrees. In case of the first test condition without visual feedback the average error amounts to 91 pixels (4.6 degrees). For the second test condition with visual feedback the average error was halved to 12 pixels, which corresponds to 0.6 degrees. Note also that the standard deviation decreased significantly to 17 pixels (0.9 degrees) in the test with visual feedback, which indicates that the system is very robust and hence intuitive to use near the center of the screen.

10.4 Discussion

We have presented a framework that implements simple and robust gesture recognition in the context of a slideshow presentation. The system is based on a TOF camera that allows us to detect and interpret pointing gestures in an intuitive and effective way, because the provided range data facilitates the localization of the user in front of the camera and allows the estimation of the pointing direction in 3D space once the head and hand have been identified.

We believe pointing is a powerful way to determine whether a gesture is intended for the system at all and to assign different meanings to the same gesture depending on where the user is pointing. Because the meaning of the gesture is strongly tied to the direction it is made in, we avoid the immersion syndrome and thus greatly increase the usability of the system.

Thus, we have developed an intuitive system that allows the user to control a slideshow by switching between slides through a simple thumbs-up gesture that is made towards one of the sides of the screen. Alternatively, we could have implemented an additional thumbs-down gesture using a single active area, but our intention here was to demonstrate the use of multiple active areas, i.e. the number of active areas multiply the number of actions that can be triggered with a given set of gestures. Finally, the user may highlight certain details on the slides simply by pointing at them; a virtual laser pointer is displayed at the location the user is pointing to. This virtual laser pointer has three advantages: (i) The user is not required to hold an additional physical device, (ii) the size and appearance can be chosen depending on the context, and (iii) the build-in smoothing of the pointer can veil a tremor of the users hand that may originate from an impairment or excitement.

11 Generation of Depth of Field Based on Range Maps

11.1 Introduction

In this chapter, we present a method to synthesize the effect of depth of field in digital color images. In photography, depth of field describes the portion of the scene that appears sharp in the image. The extent of the depth of field depends on the optics of the imaging system. While a lens can focus objects precisely only at a specific distance, the sharpness will decrease gradually for objects that are moved away from the focused distance. The amount of decrease in sharpness depends on a number of parameters, such as the aperture of the lens. For example, a pinhole camera has infinite depth of field.

Depth of field is one of the most important stylistic devices in photography. As objects within the depth of field appear sharp in the image while objects at other distances become blurred, the objects in focus can be accentuated in the image and attention can thus be directed towards them.

However, we can observe a trend that digital compact cameras become smaller and smaller in size. From a technical perspective, this miniaturization implies that the depth of field for such cameras becomes larger and larger and images often appear completely sharp. Thus, the stylistic device of depth of field is available in a rather limited form to photographers using compact cameras.

The proposed method is targeted at photographers who want to add the effect of depth of field in a post-processing step. A digital compact camera that is also

Parts of this chapter are joint work with Michael Glodek, who conducted his diploma thesis (Glodek, 2009) under my supervision. In the scope of his thesis Michael Glodek implemented and evaluated the proposed method.

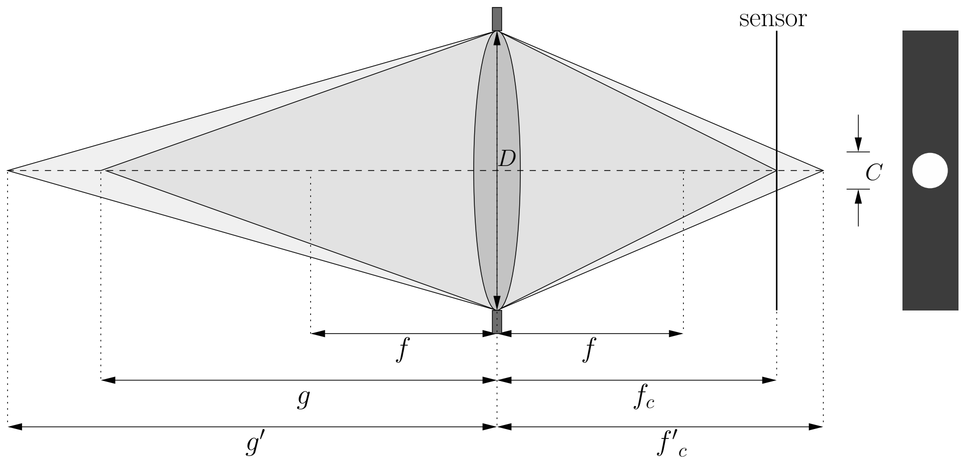


Figure 11.1: *The circle of confusion. An optical system with a focal length f and an aperture D images a point at distance g. The camera constant f_c is chosen such that the point appears in focus on the sensor. A second point at a distance g' is also in the field of view but its image is spread over the circle of confusion with diameter C. The depth of field is characterized by the interval of distances g for which the circle of confusion is sufficiently small for objects to be perceived sharp in the image. Apparently, for large apertures this interval becomes smaller and the effect stronger.*

equipped with a range sensor, such as a TOF sensor, can capture a range map of the imaged scene. The resulting range map can then be used to generate the effect of depth of field.

This task has first been addressed by Potmesil and Chakravarty (1981, 1982) who proposed the algorithm known as the *Forward-Mapped Z-Buffer*. The main concept is to use the range map to estimate the so-called circle of confusion for each pixel. The concept of the circle of confusion is depicted in Figure 11.1.

In analogy to the circle of confusion, one can define the set of points on the object surface that are mapped by the optical system in such a way that they contribute to a certain pixel. Then, the pixel value of the post-processed image is computed as a linear mixture of pixel values belonging to this set. For a given pixel, Potmesil et al. approximate this set using the circle of confusion of the pixel itself. Here, they always assume a circular circle of confusion where its size is chosen based on the range value of the considered pixel. This assumption is violated if partial occlusions occur, i.e. at borders of an object.

The theoretical solution to this problem was introduced by Cook et al. (1984) under the name of *Distributed Raytracing*. For each pixel a number of rays are sent into the scene to sample the circle of confusion. The algorithm determines where these rays intersect with the scene and the post-processed pixel is composed of a linear mixture of intensities at these intersections. This approach is only applicable if a complete 3D model of the scene is available, which is rarely the case in real-life scenarios.

Next to Potmesil, a number of authors proposed approximations of Distributed Raytracing. Haeberli and Akeley (1990) suggest the so-called *Accumulation Buffer* to simulate depth of field. They use a model of a pinhole camera to render the scene from different perspectives and generate the final image as a mixture of the different views. While results are similar to Distributed Raytracing the method can be implemented more efficiently.

Variations of the Forward-Mapped Z-Buffer algorithm by Potmesil et el. were proposed by Chen (1987) and Rokita (1993, 1996). These methods aim at a reduction of the processing time. A variant of the algorithm that addresses the problem of partial occlusions was proposed by Shinya (1994).

A very different approach is known as *Layered Depth of Field* (Scofield, 1992). The method creates different depth layers using the range map and each layer is blurred according to its distance from the camera. The final image is created by combining the different layers. A disadvantage of the method is that the amount of blur introduced is not continuous but is discretized by the number of utilized layers. Propositions to circumvent the resulting artefacts where made by Barsky et al. (2005) and Kraus and Strengert (2007).

The *Reverse-Mapped Z-Buffer Technique* was proposed by Demers (2003). The algorithm is optimized for an implementation on a graphics card and yields results that are similar to the Forward-Mapped Z-Buffer and are obtained at very high frame rates.

An approach based on the heat equation computes the depth of field using *Anisotrophic Diffusion* (Bertalmío et al., 2004). The heat equation is used to avoid artefacts at discontinuities in the range map that are due to partial occlusions. While the algorithm can be computed efficiently, it is not very realistic because the circle of confusion is approximated by a Gaussian and the depth of field cannot be tuned with respect to a specific aperture.

In this work, we introduce an algorithm called $2^1/_2$D Distributed Raytracing that uses the TOF range map to simulate the depth of field. Based on the information of the distance of the object that is depicted at a pixel we determine how sharp the image should be at that pixel location with respect to the desired depth of field. As TOF cameras usually have a significantly lower resolution than standard color imaging sensors, we also rely on a method that aligns the high resolution color image with the low resolution range map based on a Markov random field.

Cook et al. (1984) already noticed that a camera with a small aperture, which is generally the case for compact cameras and the reason why they lack the feature of depth of field, cannot capture all information that is required to generate realistic depth of field. Going by the example of the pinhole camera, the reason for this is that the scene is rendered from only a single view point. Cameras with larger apertures have more view points depending on where the corresponding light ray from the object enters the lens.

This lack of information leads to artefacts in the synthesis of depth of field. We propose a heuristic that extends the $2^1/_2$D Distributed Raytracing algorithm to attenuate these artefacts.

The remainder of this chapter is organized as follows: In Section 11.2.1 we will discuss the procedure to upsample the range map and align it with the high resolution color image. Section 11.2.2 will discuss preprocessing steps of the image data to obtain more natural results. The $2^1/_2$D Distributed Raytracing algorithm and a heuristic to improve the visual outcome will be introduced in Section 11.2.3 and Section 11.2.4. Finally, we will demonstrate results on both artificial and real data in Section 11.3.

11.2 Method

11.2.1 Upsampling the TOF range map

Before the generation of depth of field in a color image, it is important to preprocess the range map. This has two objectives: (i) We require a range value for every pixel in the color image to determine the extension of the circle of confusion and (ii) the range map must be properly aligned with the color image, i.e. edges in the range map must coincide with edges in the color image. Otherwise, parts of the image that should be in focus appear blurred or vice verse. This can lead to disturbing artefacts; especially near object borders.

An approach to this problem was introduced by Andreasson et al. (2006) and is referred to as *Nearest Range Reading Considering Color*. For a given pixel in the color image, one selects the range value in a corresponding neighborhood that maximizes a certain likelihood function.

An alternative procedure is the *Multi-Linear Interpolation Considering Color* which was also suggested by Andreasson et al. (2006). The interpolated range value is a linear combination of known range values in its neighborhood where the coefficients depend on both the range values and the color information.

We employ a method by Diebel and Thrun (2006) that is based on a Markov random field. The Markov random field is organized in three layers; two layers represent the color image and the measured range map. A third layer represents the upsampled range map and it is optimized with respect to the known data.

The Markov Random Field is composed of two potentials. The first potential – the *depth measurement potential* – aims at reconstructing the measured range sample and takes the following form:

$$\Psi = \sum_{i \in L} k(y_i - z_i)^2 \ . \tag{11.1}$$

Here, the set L denotes the indices of pixels for which a range measurement z_i is provided by the TOF range map. y_i denotes a range value in the upsampled range map. The parameter k depends on the variance of z_i with respect to the true 3D scene.

The second potential – the *depth smoothness prior* – aims at creating a smooth range map:

$$\Phi = \sum_i \sum_{j \in N(i)} w_{ij}(y_i - y_j)^2 \ . \tag{11.2}$$

Here, $N(i)$ denotes the set of direct neighbors of the pixel i and the deviation of neighboring range values is penalized. The weights w_{ij} are influenced only by the color image:

$$w_{ij} = \exp(-c\, u_{ij}) \tag{11.3}$$

$$u_{ij} = \|\vec{x}_i - \vec{x}_j\|_2^2 \ . \tag{11.4}$$

Here, $\vec{x}_i$ and $\vec{x}_j$ denote the RGB color values of neighboring pixels. Deviations in the range map get a lower weight if the neighboring pixels differ with respect to color, i.e. if the two pixels lie across an edge in the color image. The parameter c denotes a constant that quantifies the amount of smooting we allow across edges in the image.

Given these two potentials we can now the define the Markov Random Field. The conditional distribution over the target range values y is given by:

$$p(y \,|\, \vec{x}, z) = \frac{1}{Z} \exp\left(-\frac{1}{2}(\Psi + \Phi)\right) \ , \tag{11.5}$$

where Z is a normalization constant.

The computation of the full posterior is impossible for high resolution images. Thus, Diebel and Thrun (2006) propose to compute the solution via a conjugate gradient descent of the log-posterior.

11.2.2 Preprocessing the Image Data

Before we can synthesize the effect of depth of field, two preprocessing steps are required to obtain optimal results: (i) The inversion of the gamma correction of the utilized file format and (ii) an increase of the contrast ratio.

An inversion of the gamma correction is required for certain file formats, such as JPEG. As almost all displays have a non-linear tonal response curve, such file formats store the gamma-corrected image data with standard correction parameters to supersede a correction before the image is displayed. Since we assume a linear relation between pixel intensities, we need to invert the gamma correction prior to the synthesis of depth of field.

The second preprocessing step is due to the fact the most formats store *low dynamic range* images, i.e. the maximal contrast between low and high pixel intensities is limited to optimize data compression. As a result, high intensity values are often clipped. The blur induced by the depth of field effect spreads the intensity of pixels that are out of focus over a larger area in the output image. Thus, clipped pixels are likely to appear darker than they should because their intensities were already artificially reduced through low dynamic range. The optimal solution would be to use *high dynamic range* images. While these are often not available, one can simulate high dynamic range by increasing pixel values with maximal intensity values.

The effect of these two preprocessing steps is demonstrated in Figure 11.2. The figure shows a treetop where the sunlight is falling through the leafs. In Figure 11.2a and 11.2b the depicted tree is once in focus and once defocused. Both images were taken with a real camera. The remaining images show the result of simulating depth of field without preprocessing (Figure 11.2c), with gamma correction (Figure 11.2d), and with additional simulation of high dynamic range (Figure 11.2e). Apparently, the version that uses both preprocessing methods appears to be more lively and comes closest to the natural effect in Figure 11.2b.

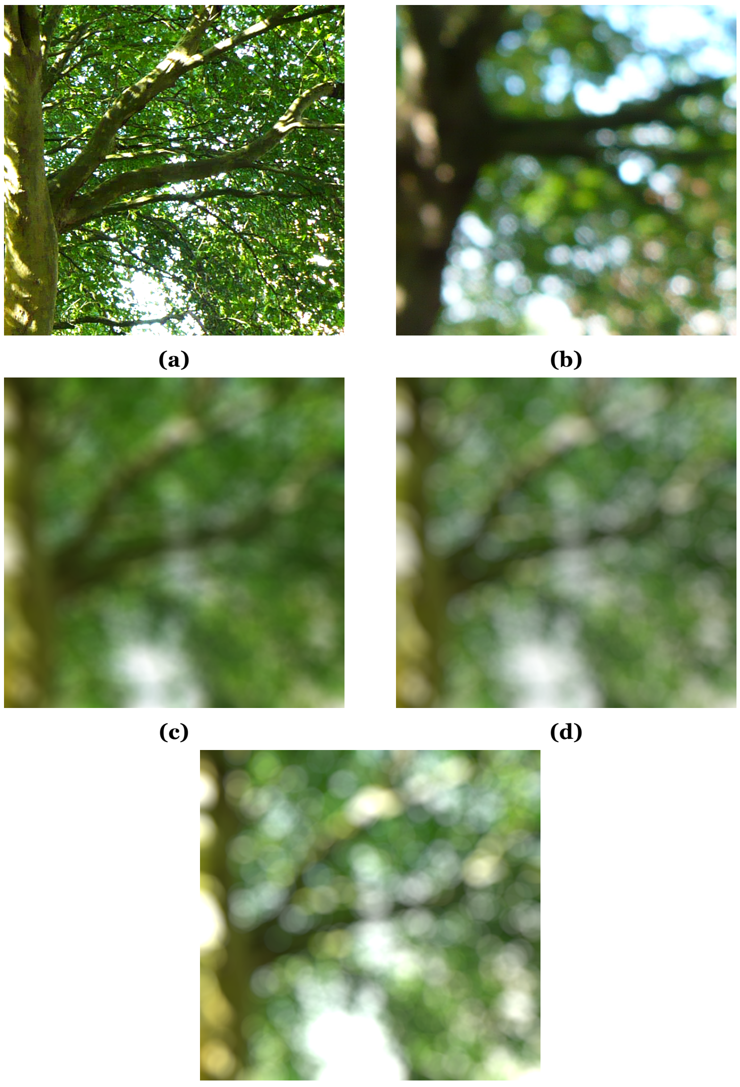


Figure 11.2: *The effect of preprocessing for the synthesis of depth of field. A treetop with sunlight glittering through the leafs in focus is shown in 11.2a. The defocused treetop imaged with a real camera is given 11.2b. The remaining three figures simulate the effect of depth of field on 11.2a with a simple blur of* 30×30 *pixels. The effect on the unprocessed JPEG image is depicted in 11.2c. In 11.2d a gamma correction was applied and in 11.2e high dynamic range was simulated by increasing the maximum pixel values by a factor of three.*

11.2.3 Synthesis of Depth of Field

The goal of this work is to devise an algorithm for the synthesis of depth of field that comes as close as possible to the natural effect. Thus, we decided to follow the approach by Cook et al. (1984) and use Distributed Raytracing. Since a TOF camera does not provide a complete model of the 3D scene and instead only gives range measurements at those points that are directly visible to the camera, we refer to our algorithm as $2^1/_2$D Distributed Raytracing.

The main idea of Distributed Raytracing is to send a number of rays from each pixel into the scene and to determine the color value of the pixel as a mixture of the color values that occur at intersections of the rays with the scene. In the simplest case of a pinhole camera model only a single ray is considered for each pixel.

If a camera with a lens is simulated, one considers a distribution of sample points $\vec{u}_l$ over the aperture of the lens and sends a ray from a given pixel (i, j) to each sample. Suitable distributions can for example be found in the book of Pharr and Humphreys (2004). The effective size of the aperture can be controlled by scaling the distribution. The shape of the utilized diaphragm can be modelled by discarding those sample points that are obscured by it. In the following, we will restrict ourselves to a circular diaphragm and thin lenses.

In this case, all rays through the sample points $\vec{u}_l$ will intersect at a 3D point $\vec{q}_{i,j}$ in the scene. We will now determine this point $\vec{q}_{i,j}$. Let us assume a camera with a fixed focal length f. Furthermore, we want to focus the camera such that objects at a distance g appear sharp in the image. Given these parameters, we can determine the camera constant f_c, i.e. the distance at which the image sensor must be placed behind the lens for objects at distance g to be in focus, as:

$$f_c = \frac{f \cdot g}{g - f} \quad . \tag{11.6}$$

Given the camera constant f_c, we can send a ray from each pixel (i, j) through the center of projection, which will pass the lens unrefracted. Following this ray for the distance g will yield the 3D point $\vec{q}_{i,j}$, where all rays from the pixel (i, j) passing through the lens at sample points $\vec{u}_l$ will intersect:

$$\vec{q}_{i,j} = \frac{((c_x - i) \cdot s_x, (c_y - j) \cdot s_y, f_c)^T}{\|((c_x - i) \cdot s_x, (c_y - j) \cdot s_y, f_c)^T\|_2} \cdot g \quad . \tag{11.7}$$

Here, (c_x, c_y) denotes the principal point, i.e. the pixel coordinates of the point where the optical axis intersects the image sensor. The width and height of a pixel are defined by s_x and s_y.

Thus, a ray R_l passing from pixel (i, j) through the lens at $\vec{u}_l$ can be expressed as follows:

$$R_l = \vec{u}_l + t \cdot \vec{r}_{i,j,l} \tag{11.8}$$

$$\vec{r}_{i,j,l} = \vec{q}_{i,j} - \vec{u}_l \quad . \tag{11.9}$$

The next step is to determine, where the ray R_l intersects with the surface of the scene characterized by the range map of the TOF camera. To this end, we invert the perspective projection of the TOF camera and obtain the 3D coordinates $\vec{p}$ of each pixel as a sample on the surface of the scene. We now tessellate the range map into triangles to obtain a smooth surface and compute the surface normal $\vec{n}$ and the bias d for each triangle. Assuming that a triangle is characterized by its three corners $P_1 = \vec{p}_1$, $P_2 = \vec{p}_2$, and $P_3 = \vec{p}_3$ we obtain:

$$\tilde{\vec{n}} = (P_2 - P_1) \times (P_3 - P_1) \tag{11.10}$$

$$\vec{n} = \alpha \cdot \frac{\tilde{\vec{n}}}{\|\tilde{\vec{n}}\|} \tag{11.11}$$

$$d = -\langle\, P_1,\, \vec{n} \,\rangle \ , \tag{11.12}$$

where $\alpha = \mathrm{sign}(\langle\, P_1,\, \tilde{\vec{n}} \,\rangle)$, i.e. α enforces that the normal $\vec{n}$ always points towards the camera. Using Equation (11.8), we can now determine the intersection $\vec{x}$ of the ray R_l with the plane defined by the triangle. To this end we determine t as:

$$t = -\frac{\langle\, \vec{u}_l,\, \vec{n} \,\rangle + d}{\langle\, \vec{r}_{i,j,l},\, \vec{n} \,\rangle} \tag{11.13}$$

and evaluate Equation (11.8).

Finally, we need to determine whether the intersection $\vec{x}$ lies within the triangle defined by P_1, P_2, and P_3. To this end we compute the barycentric coordinates u, v, and w that represent $\vec{x}$ as a linear combination of the corners of the triangle:

$$\vec{x} = u \cdot P_1 + v \cdot P_2 + w \cdot P_3 \ . \tag{11.14}$$

In case the four constraints $w = 1 - (u + v), \quad 0 \leq u \leq 1, \quad 0 \leq v \leq 1$, and

$u + v \leq 1$ hold, the point $\vec{x}$ lies within the triangle and the barycentric coordinates fulfil $u + v + w = 1$.

Now, that we have determined where a ray hits the scene, we can compute its contribution to the pixel (i, j) using the coefficients u, v, w and the color values $\vec{x}_{P_k}$ corresponding to the corners of the triangle. Thus, we compute the contribution of ray R_l to the color value $\vec{z}_{i,j,l}$ of pixel (i, j) in the output image by:

$$\vec{z}_{i,j,l} = u \cdot \vec{x}_{P_1} + v \cdot \vec{x}_{P_2} + w \cdot \vec{x}_{P_3} \,. \tag{11.15}$$

The color value of pixel (i, j) is a linear mixture of all rays R_l defined by the distribution of samples $\vec{u}_l$ over the aperture. In the simplest case, all rays get equal weights ω_l. Generally, we can compute the final pixel value $\vec{z}_{i,j}$ as follows:

$$\vec{z}_{i,j} = \frac{\sum_l \omega_l \cdot \vec{z}_{i,j,l}}{\sum_l \omega_l} \,. \tag{11.16}$$

11.2.4 Dealing with Missing Data

Choosing equal weights for all rays R_l does not always yield the optimal visual experience because it can produce artefacts especially near the borders of objects. This is due to the fact that the proposed method cannot access the complete 3D model of the scene. Instead we must rely on the range map provided by the TOF camera, i.e. we have to deal with missing data where an object that is close to camera partially occludes another object. Hence, we refer to our algorithm as $2^1/_2$D Distributed Raytracing.

The reason why this situation causes problems is depicted in Figure 11.3. The first observation we have to make is that the effect of depth of field is strongest for cameras with a large aperture, i.e. the circle of confusion for a defocused point in the scene grows with the diameter of the aperture (see Figure 11.1). For simplicity, let us assume that the TOF sensor is based on a pinhole camera. In order to create the effect of depth of field, we have to simulate an optical system with a large aperture – larger than the one used for acquiring the range map. As a result, the simulated camera has different view angles on the scene for rays emanating from the boundary of the aperture and is able see parts of objects that are occluded to the TOF sensor.

Figure 11.3 shows two rays the intersect with triangles that do not exist in the 3D scene. Nevertheless, the algorithm described in the previous section assigns a color value to these rays based on the barycentric coordinates. This causes two kinds of problems: (i) In case of the ray that passes between the objects at distance g and

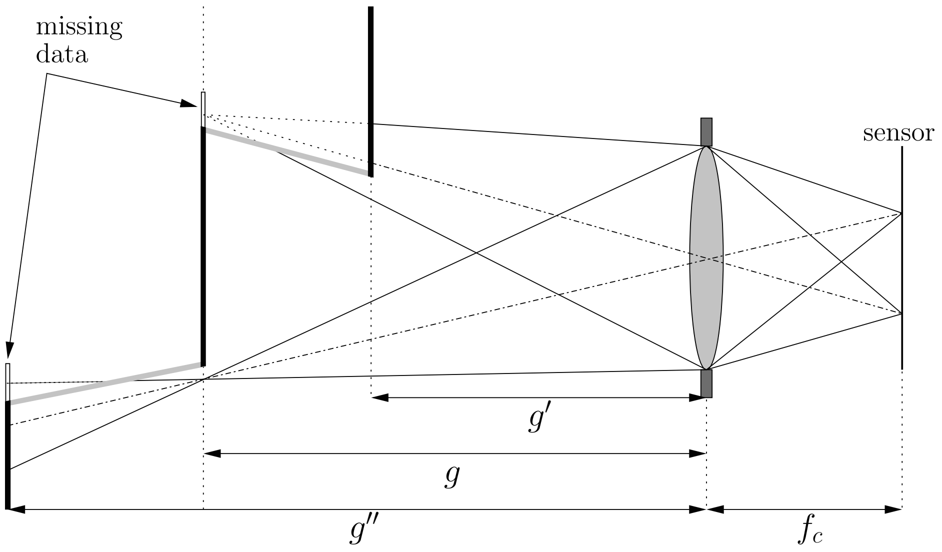


Figure 11.3: $2^1/_2$*D Distributed Raytracing at the borders of objects that are not at the focused distance g. For both objects at distances g' and g'' some of the rays intersect with a triangle that does not represent the true scene, i.e. we have an incomplete model of the 3D scene with missing data.*

g', the true color value should correspond to the color value of the object at distance g. This would cause the focused object to shine through the object in front, i.e. the object at g' appears strongly defocused. However, this effect is attenuated because the ray also gets a color contribution from the object at distance g'. (ii) In the second case, the ray passing between the focused object and the object at g'' gets a color contribution from the former one. This causes the focused object to blur although it should be in focus.

We propose an approach to avoid this problem and devise a strategy to attenuate rays with erroneous color values. This comes at the disadvantage of discarding information for rays that have already been computed, however, we avoid disturbing artefacts. We refer to this procedure as *angle-of-incidence-based weight adaptation*.

Missing data often co-occurs with jump edges in the range data. Thus, the angle between the normals of the corresponding triangles and the incoming rays is large for missing data and we can define weights ω_l as follows:

$$\omega_l = (h-1) \cdot \frac{\cos(\langle \vec{n}, \vec{r}_{i,j,l} \rangle) + (\beta - 1)}{\beta} + h \quad , \tag{11.17}$$

where $$h = \frac{\min(|g - y_{P_1}|, |g - y_{P_2}|, |g - y_{P_3}|)}{\Delta_{zg}} \quad . \tag{11.18}$$

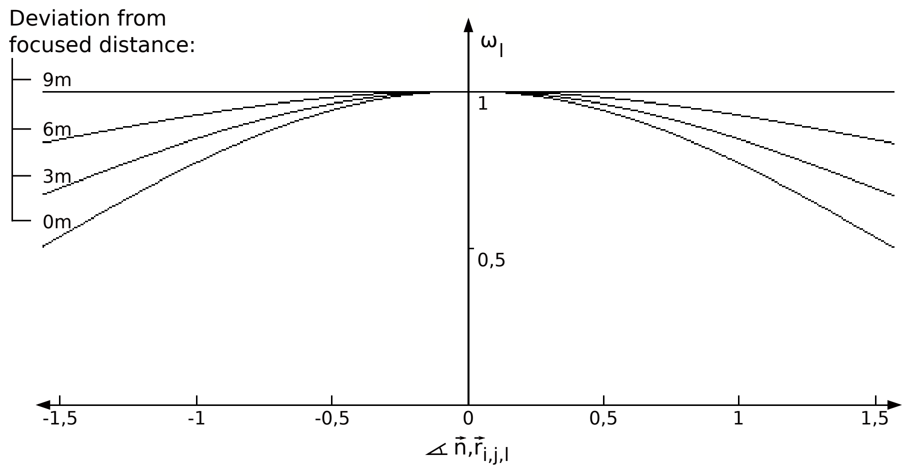


Figure 11.4: *Behavior of the weight ω_l for different angles $\measuredangle(\vec{n}, \vec{r}_{i,j,l})$ and deviations from the focused distance g.*

We take the cosine of the measured angle between the normal $\vec{n}$ and the incoming ray R_l. We introduce the parameter h to ensure that the weight adaptation takes effect only if one of the corners of the triangle is near the focused distance g. Here, Δ_{zg} is a normalization constant representing the largest possible range value in the image to enforce $h \in [0, 1]$. The additional parameter β controls how much influence h has on the weights ω_l. Figure 11.4 shows the behavior of ω_l with respect to different distances and angles.

11.3 Results

We have evaluated the proposed method using both synthetic and real data. The basis for the evaluation with synthetic data is an image rendered by PBRT (Pharr and Humphreys, 2004) which is shown in Figure 11.5. The image shows three spheres which partially occlude each other, i.e. the blue sphere on the right occludes the green sphere in the center, which in turn occludes the red sphere on the left.

We synthesized the effect of depth of field using a number of different methods for a qualitative comparison in Figure 11.6. In all images the focus is on the green sphere in the center of the image. The image in Figure 11.6a was generated by PBRT using the Distributed Raytracing algorithm by Cook et al. (1984). Since the algorithm has access to the full 3D model of the scene in PBRT it can be considered ground truth and serves as a comparison for the other methods.

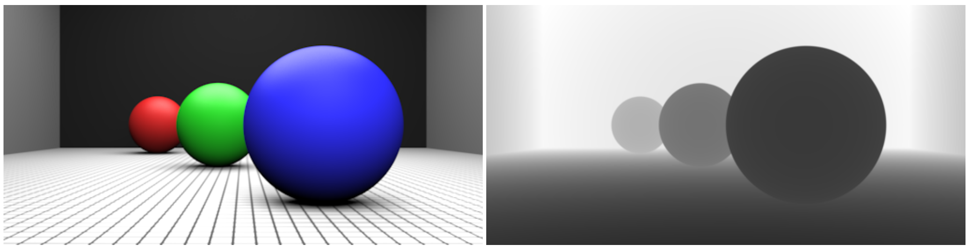


Figure 11.5: *Synthetic raw data showing three spheres at different distances. The images were rendered by PBRT at a resolution of* 512×256 *pixels. The range map has values in the interval [7.8m, 43.4m]. A camera with a focal length of 10mm, an opening angle of* $30°$*, and a sensor of size* $3.2\text{mm} \times 1.7\text{mm}$ *was simulated.*

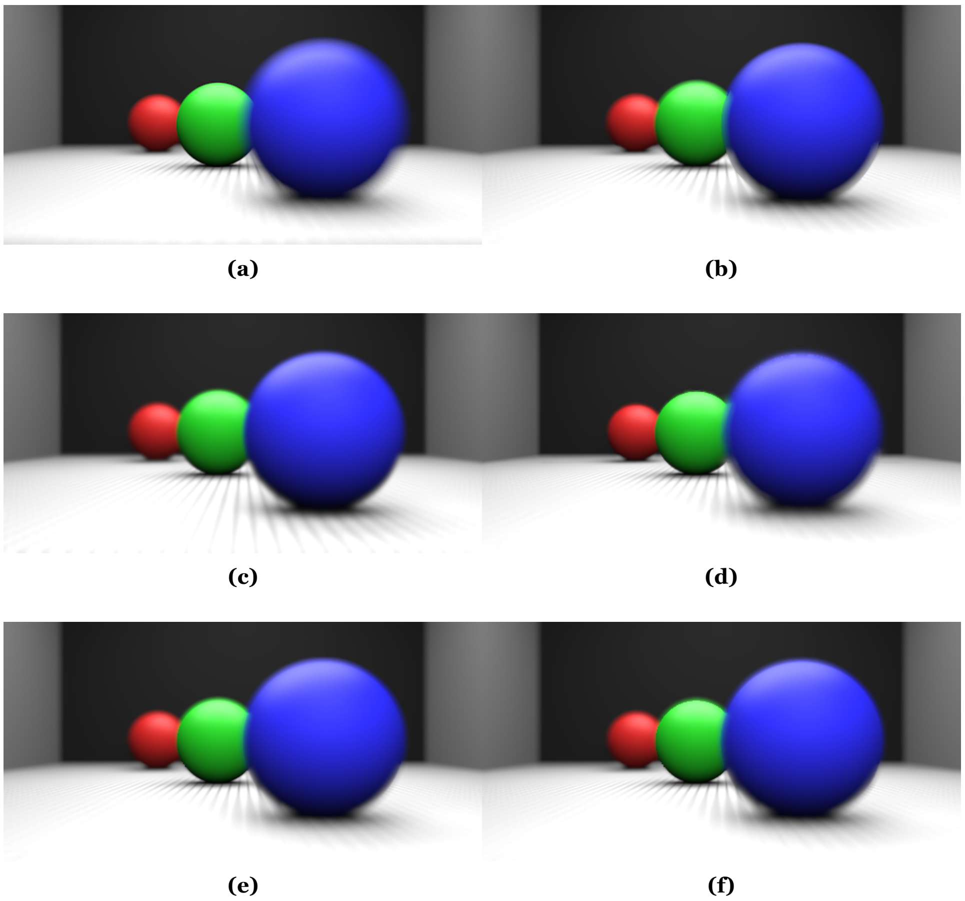


Figure 11.6: *Synthesis of depth of field on the data of Figure 11.5 using different algorithms. The following algorithms where used for the generation: 11.6a Distributed Raytracing using the full 3D model of the scene. 11.6b Forward-Mapped Z-Buffer. 11.6c Anisotropic Diffusion. 11.6d Layered depth of field. 11.6e* $2^1/_2$*D Distributed Raytracing. 11.6f* $2^1/_2$*D Distributed Raytracing with angle-of-incidence-based weight adaptation.*

The Forward-Mapped Z-Buffer algorithm using the implementation proposed by Chen (1987) was used to generate the image in Figure 11.6b. One can observe that the method fails to maintain sharp edges around the green sphere and the blurred region around the blue sphere shows a strong contour.

Figure 11.6c shows the results of the anisotropic diffusion approach by Bertalmío et al. (2004). While the green sphere is still blurred, the sharp contour around the blue sphere disappeared.

The same is true for Layered depth of field approach by Scofield (1992), which is depicted in Figure 11.6d. One can observe, however, the effect of depth of field is stronger than it is the case for the anisotropic diffusion method.

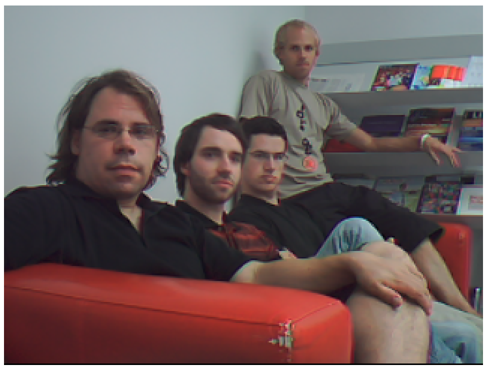
Figure 11.6e and Figure 11.6f show the results obtained with our method. In the latter case, the angle-of-incidence-based weight adaptation was used, which visibly improves the sharpness of the green sphere.

Results for a real image captured with the ZCam by 3DV Systems is given in Figure 11.7. The original images are available under `http://inb.uni-luebeck.de/mitarbeiter/haker/dof`. The image in Figure 11.7a shows the color image captured by the ZCam. The corresponding range map is given in Figure 11.7b. One can observe, that both the color image and the range map have limited quality. The synthesis of depth of field with $2^1/_2$D Distributed Raytracing and angle-of-incidence-based weight adaptation is shown in Figure 11.7c. The simulated camera focuses on the third person from the left and the effect of depth of field is clearly visible. However, there occur a number of disturbing artefacts that are mainly due to the flawed range map. Notice for example the hair of the foremost person; the strands that stand out have erroneous range values and are thus not sufficiently blurred.

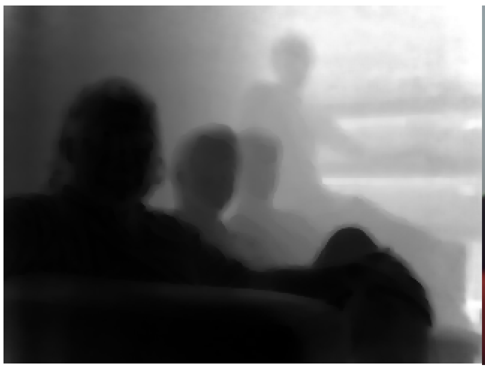
To demonstrate the capabilities of the algorithm, we manually created a range map that aims at assigning correct range values to the pixels at borders of objects. The resulting image is shown in Figure 11.7e and one can observe a significant reduction of artefacts and a generally more consistent experience of the effect. From this we can conduct that it is highly important to have range maps that are consistent with the content of the color image, especially at the border of objects.

11.4 Discussion

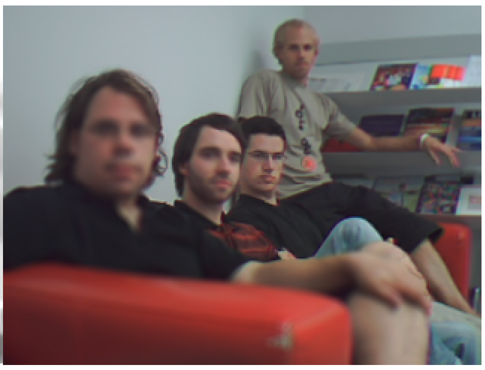
In this section we have presented an algorithm that can be applied as a post-processing method to create the effect of depth of field in digital images. The basis of this approach is a range map of the scene depicted by the color image. Such a range map



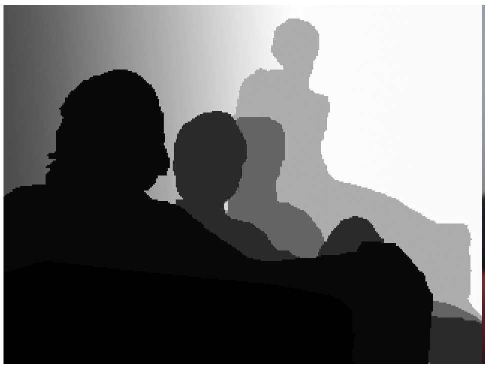
(a)



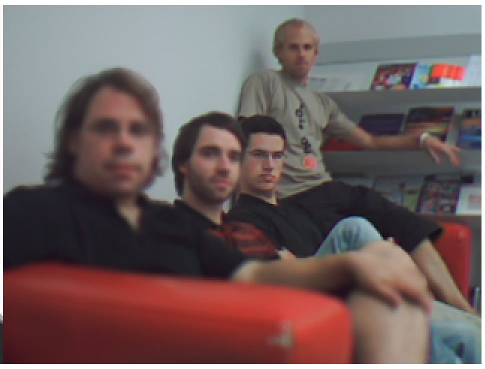
(b)



(c)



(d)



(e)

Figure 11.7: *Synthesis of depth of field on the data captured by a ZCam by 3DV Systems. 11.7a Shows the color image at a resolution of* 320×240 *pixels. The range map captured by the ZCam and the resulting synthesis of depth of field is given in 11.7b and 11.7c. An artificially created range map and the resulting output image are depicted in 11.7d and 11.7e.*

can for example be obtained using a TOF camera. We have presented a technique for upsampling and smoothing the range map such that is has the same resolution as the color image.

Using the range data, we create a mesh of triangles representing the surface of the scene and use Distributed Raytracing to assign a color value to each pixel. The color value is computed as a linear mixture of color values that are computed for rays intersecting the mesh. Since the camera setup does not acquire a complete model of the scene, we have to deal with missing data. We have proposed a heuristic that avoids artefacts that are induced by missing data.

Finally, we have compared the method, which we refer to as $2^1/_2$D Distributed Raytracing, to a number of alternative algorithms on synthetic data. The algorithm was also tested on a real image captured by a TOF camera.

We can conclude that the method achieves realistic results if the range map is consistent with the color image. Thus, future work should focus on the improvement of the procedure for upsampling and aligning the range map with the color image. Otherwise, artefacts can impede the visual experience.

12 Conclusion

The results presented in this thesis were already discussed individually at the end of the corresponding chapters. The main achievements on the algorithmic side include the exploitation of the shading constraint to dramatically improve the range maps measured by a TOF camera, a robust algorithm for the estimation of human pose that is based on self-organizing maps, and a variety of different image features that can be employed for feature tracking and action recognition. These algorithms play an important role in three potential applications for TOF cameras: A nose tracker which enables an alternative form of text input, and framework for controlling a slideshow presentation with pointing gestures, and a method for photographers to generate the stylistic device of depth-of-field in digital images. I will now conclude this thesis, by discussing where I see the potential of computer vision systems in the context of human-computer interfaces at a more general scope.

The IT sector has seen a fascinating development since the invention of the first computers. We can observe that the role of computer systems is changing in the sense that computers are no longer only stationary terminals with the standard technology of display, mouse, and keyboard for user interaction. On the contrary, powerful computer systems become integrated into various devices we continuously use throughout the day, such as mobile phones. These devices often provide some form of assistance while the user does not even notice that he is dealing with a computer in the conventional manner.

This concept of the embedded computer was already envisioned by Mark Weiser in the early 90's (Weiser, 1991). In this context, Weiser formed the term *ubiquitous computing* and recast the role of computers in human-machine interaction: "*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.*" The technical realization of this vision is commonly known by the term *information appliance*, which was coined by Jef Raskin in the late 70's. In the scope of information appliance, Donald A. Norman sees computers as enhancements of very common devices, which are designated to certain very specific tasks (Norman, 1998). While the user interface of such devices is kept simple and easy to use even for untrained people, the computer operates invisbily in the background providing additional functionality; possibly through an exchange of information with other information appliances.

While some of these computer systems will simply provide information or operate without intentional input of the user, the total number of human-computer interfaces will nevertheless increase. And for most systems in the category of ubiquitous computing conventional user interfaces consisting of display, mouse, and keyboard will not be an option. Thus, we will require new means of user interaction.

Under this perspective, I see a lot of potential for gesture-based interaction. Currently, the most dominant market for gesture-based interaction is the gaming market. An obvious reason for this development is that the targeted consumers are generally young and open to new concepts involving computer technology. New forms of human-computer interaction can be explored in a playful manner and the level of accuracy and reliability required in games is not critical, i.e. a failure of immature technology does not endanger anyone.

However, I can also imagine a number of appliances for human action and gesture recognition as well as gesture-based interaction beyond the gaming market. An important sector is the health industry. Imagine a medical doctor in an operating room filled with high-tech equipment ranging from life sustaining devices to systems providing the clinician with important patient data. Gestures may enable doctors to have all these devices within reach while adhering to hygiene standards. Conventionally, doctors have to access this information through the assistance of a surgical nurse or they have to abandon their operating equipment while actuating the device.

Another field of application is health monitoring; either at home or in the hospital. Disabled people can be monitored continuously through a camera based system which signals an alarm when an unexpected event occurrs, for example when an elderly person has an accident at home and is unable to reach for the phone to call for assistance.

The use of gestures and action recognition can also be applied to the automotive sector. Navigation systems are available in almost every high-class car. The use of simple gestures for the control of such systems would both ease the use and increase the security while driving. In addition, camera based systems can be used to monitor the traffic. Thus, potential hazards can be detected and the driver can be warned or precrash counter measures can be activated.

The techology and algorithms described in this thesis provide a steping stone towards the realization of the above mentioned scenarios. During the years of my research in this area and numerous discussions with camera manufacturers and companies interested in gesture-based interaction, I also began to see the potential of commercializing the acquired knowledge. The first serious footstep in this direction was taken in late 2009, when we applied for funding in the scope of the EXIST Forschungstransfer at the German Federal Ministry of Economics and Technology. And since the beginning of 2010 we are heading straight towards realizing this idea, preparing a start-up company that targets human action and gesture recognition by means of computer vision based on 3D imaging sensors.

Bibliography

ARTTS 3D TOF Database. `http://www.artts.eu/publications/3d_tof_db`.

Ankur Agarwal and Bill Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

Henrik Andreasson, Rudolph Triebel, and Achim J. Lilienthal. Vision-based interpolation of 3D laser scans. In *Proceedings of the International Conference on Autonomous Robots and Agents (ICARA)*, 2006.

Miyamoto Arata, Sumi Kazuhiko, and Matsuyama Takashi. Human pose estimation from 3D object skeleton using articulated cylindrical human model. *IPSJ SIG Technical Reports*, 51:133–144, 2006.

Andre L. C. Barczak, Martin J. Johnson, and Chris H. Messom. Real-time computation of Haar-like features at generic angles for detection algorithms. *Research Letters in the Information and Mathematical Sciences*, 9:98–111, 2006.

Brian A. Barsky, Michael J. Tobias, Derrick P. Chu, and Daniel R. Horn. Elimination of artifacts due to occlusion and discretization problems in image space blurring techniques. *Graph. Models*, 67(6):584–599, 2005.

Erhardt Barth, Terry Caelli, and Christoph Zetzsche. Image encoding, labeling, and reconstruction from differential geometry. *CVGIP: Graphical Models and Image Processing*, 55(6):428–46, November 1993.

Thomas Baudel and Michel Beaudouin-Lafon. CHARADE: Remote control of objects using free-hand gestures. *Communications of the ACM*, 36:28–35, 1993.

Marcelo Bertalmío, Pere Fort, and Daniel Sánchez-Crespo. Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. *3D Data Processing, Visualization, and Transmission (3DPVT)*, pages 767–773, 2004.

Martin Böhme, Martin Haker, Thomas Martinetz, and Erhardt Barth. A facial feature tracker for human-computer interaction based on 3D Time-of-Flight cameras. *International Journal of Intelligent Systems Technologies and Applications*, 5(3/4): 264–273, 2008a.

Martin Böhme, Martin Haker, Thomas Martinetz, and Erhardt Barth. Shading constraint improves accuracy of time-of-flight measurements. In *CVPR 2008 Workshop on Time-of-Flight-based Computer Vision (TOF-CV)*, 2008b.

Martin Böhme, Martin Haker, Thomas Martinetz, and Erhardt Barth. Shading constraint improves accuracy of time-of-flight measurements. In *CVIU - International Journal of Computer Vision and Image Understanding*, 2010.

Bernhard Büttgen, Thierry Oggier, Michael Lehmann, Rolf Kaufmann, and Felix Lustenberger. CCD/CMOS lock-in pixel for range imaging: Challenges, limitations, and state-of-the-art. In *1st Range Imaging Research Day*, pages 21–32, ETH Zürich, Switzerland, 2005.

Yong C. Chen. Lens effect on synthetic image generation based on light particle theory. In *CG International '87 on Computer graphics 1987*, pages 347–366, Springer-Verlag New York, Inc., 1987.

German K.M. Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:77–84, 2003.

Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *SIGGRAPH Computer Graphics*, volume 18, pages 137–145, ACM, 1984.

James Edwin Cryer, Ping-Sing Tsai, and Mubarak Shah. Integration of shape from shading and stereo. *Pattern Recognition*, 28(7):1033–1043, 1995.

Joe Demers. Depth of field in the 'toys' demo. In *Ogres and Fairies: Secrets of the NVIDIA Demo Team*. GDC 2003, 2003.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

Pandu R. Rao Devarakota, Bruno Mirbach, Marta Castillo-Franco, and Bjorn Ottersten. 3-D vision technology for occupant detection and classification. In *3DIM '05: Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*, pages 72–79, IEEE Computer Society, 2005.

James R. Diebel and Sebastian Thrun. An application of Markov random fields to range sensing. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 291–298. MIT Press, 2006.

James R. Diebel, Sebastian Thrun, and Michael Brünig. A Bayesian method for probable surface reconstruction and decimation. *ACM Transactions on Graphics*, 25 (1):39–59, 2006.

David L. Donoho, Michael Elad, and Vladimir N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2006.

Jean-Denis Durou and Didier Piau. Ambiguous shape from shading with critical points. *Journal of Mathematical Imaging and Vision*, 12(2):99–108, 2000.

Jean-Denis Durou, Maurizio Falcone, and Manuela Sagona. Numerical methods for shape-from-shading: A new survey with benchmarks. *Computer Vision and Image Understanding*, 109(1):22–43, 2008.

Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 726–733, IEEE Computer Society, 2003.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press, 1996.

Dragos Falie and Vasile Buzuloiu. Noise characteristics of 3D time-of-flight cameras. In *Proceedings of the IEEE International Symposium on Signals, Circuits & Systems (ISSCS)*, volume 1, pages 229–232, Iasi, Romania, 2007.

Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW. *Proceedings of the IEEE*, 93(2):216–231, 2005.

Pascal V. Fua and Yvan G. Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–56, 1995.

Jürgen Gall, Bodo Rosenhahn, Thomas Brox, and Hans-Peter Seidel. Optimization and filtering for human motion capture. *International Journal of Computer Vision*, 87(1):75–92, 2010.

Michael Glodek. Modellierung von Schärfentiefe anhand von Tiefenkarten. Diploma thesis, Universität zu Lübeck, 2009.

S. Burak Gokturk, Hakan Yalcin, and Cyrus Bamji. A time-of-flight depth sensor - system description, issues and solutions. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 3*, page 35, IEEE Computer Society, 2004.

Dimitry O. Gorodnichy. On importance of nose for face tracking. In *Proc. IEEE Intern. Conf. on Automatic Face and Gesture Recognition (FG'2002)*, Washington, D.C., 2002.

Sigurjón Árni Gudmundsson, Henrik Aanæs, and Rasmus Larsen. Fusion of stereo vision and time-of-flight imaging for improved 3D estimation. In *Dynamic 3D Imaging – Workshop in Conjunction with DAGM*, volume 5, pages 425–433, Inderscience Publishers, 2007a.

Sigurjón Árni Gudmundsson, Henrik Aanæs, and Rasmus Larsen. Effects on measurement uncertainties of time-of-flight cameras. In *Proceedings of the IEEE International Symposium on Signals, Circuits & Systems (ISSCS)*, volume 1, pages 1–4, 2007b.

Paul Haeberli and Kurt Akeley. The accumulation buffer: hardware support for high-quality rendering. In *SIGGRAPH Computer graphics*, pages 309–318, ACM, 1990.

Tom S. F. Haines and Richard C. Wilson. Integrating stereo with shape-from-shading derived orientation information. In *British Machine Vision Conference*, volume 2, pages 910–919, 2007.

Tom S. F. Haines and Richard C. Wilson. Combining shape-from-shading and stereo using Gaussian-Markov random fields. In *International Conference on Pattern Recognition*, pages 1–4, 2008.

Martin Haker, Martin Böhme, Thomas Martinetz, and Erhardt Barth. Geometric invariants for facial feature tracking with 3D TOF cameras. In *Proceedings of the IEEE International Symposium on Signals, Circuits & Systems (ISSCS)*, volume 1, pages 109–112, Iasi, Romania, 2007.

Martin Haker, Martin Böhme, Thomas Martinetz, and Erhardt Barth. Scale-invariant range features for time-of-flight camera applications. In *IEEE Confer-*

ence on Computer Vision and Pattern Recognition (CVPR), Workshop on Time-of-Flight-based Computer Vision (TOF-CV), 2008.

Martin Haker, Martin Böhme, Thomas Martinetz, and Erhardt Barth. Deictic gestures with a time-of-flight camera. In Stefan Kopp and Ipke Wachsmuth, editors, *Gesture in Embodied Communication and Human-Computer Interaction – International Gesture Workshop GW 2009*, volume 5934 of *LNAI*, pages 110–121. Springer, 2009a.

Martin Haker, Thomas Martinetz, and Erhardt Barth. Multimodal sparse features for object detection. In *Artificial Neural Networks - ICANN 2009, 19th International Conference, Limassol, Cyprus, September 14–17, 2009, Proceedings*, volume 5769 of *Lecture Notes in Computer Science*, pages 923–932. Springer, 2009b.

Dan Witzner Hansen, Mads Hansen, Martin Kirschmeyer, Rasmus Larsen, and Davide Silvestre. Cluster tracking with time-of-flight cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Time-of-Flight-based Computer Vision (TOF-CV)*, 2008.

Keith Hartt and Mark Carlotto. A method for shape-from-shading using multiple images acquired under different viewing and lighting conditions. In *Proceedings of Computer Vision and Pattern Recognition*, pages 53–60, 1989.

Horst G. Heinol. *Untersuchung und Entwicklung von modulationslaufzeitbasierten 3D-Sichtsystemen*. PhD thesis, University of Siegen, Germany, 2001.

Nils Hofemann, Jannik Fritsch, and Gerhard Sagerer. Recognition of deictic gestures with context. In *Proceedings of the 26th Annual Symposium of the German Association for Pattern Recognition (DAGM 2004)*, volume 3175 of *Lecture Notes in Computer Science*, pages 334–341, 2004.

Berthold K. P. Horn and John G. Harris. Rigid body motion from range image sequences. *CVGIP: Image Understanding*, 53(1):1–13, 1991.

Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

Gabi J. Iddan and Giora Yahav. 3D imaging in the studio. In *Proceedings of SPIE*, volume 4298, pages 48–56, 2001.

Timo Kahlmann, Fabio Remondino, and Sébastien Guillaume. Range imaging technology: new developments and applications for people identification and tracking. In *Proc. of Videometrics IX - SPIE-IS&T Electronic Imaging*, volume 6491, 2007.

Rudolf E Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Series D, Journal of Basic Engineering*, 82:35–45, 1960.

Jens Keiner, Stefan Kunis, and Daniel Potts. NFFT 3.0, C subroutine library. `http://www.tu-chemnitz.de/~potts/nfft`, 2006.

Steffen Knoop, Stefan Vacek, and Rüdiger Dillmann. Fusion of 2D and 3D sensor data for articulated body tracking. *Robotics and Autonomous Systems*, 57(3): 321–329, 2009.

Jan J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–70, 1984.

Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, Heidelberg, 1995.

Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. Time-of-Flight Sensors in Computer Graphics. *Eurographics State of the Art Reports*, pages 119–134, 2009.

Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. Time-of-flight cameras in computer graphics. *Computer Graphics Forum*, 29(1):141–159, 2010.

Eva Kollorz, Jochen Penne, Joachim Hornegger, and Alexander Barke. Gesture recognition with a Time-Of-Flight camera. *International Journal of Intelligent Systems Technologies and Applications*, 5(3/4):334–343, 2008.

Bernhard König, Bedrich J. Hosticka, Peter Mengel, and Ludwig Listl. Advanced time-of-flight range camera with novel real-time 3d image processing. *Optics and Photonics for Information Processing - SPIE*, 2007.

Alfred Kranstedt, Andy Lücking, Thies Pfeiffer, Hannes Rieser, and Ipke Wachsmuth. Deixis: How to Determine Demonstrated Objects Using a Pointing Cone. In S. Gibet, N. Courty, and J.-F. Kamp, editors, *Gesture in Human-Computer Interaction and Simulation*, LNAI 3881, pages 300–311, Springer, 2006.

Martin Kraus and Magnus Strengert. Depth-of-field rendering by pyramidal image processing. In *Proceedings Eurographics 2007*, pages 645–654, 2007.

Stefan Kunis and Daniel Potts. Stability results for scattered data interpolation by trigonometric polynomials. SIAM *Journal on Scientific Computing*, 29: 1403–1419, 2007.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Simple Method for High-Performance Digit Recognition Based on Sparse Coding. *IEEE Transactions on Neural Networks*, 19(11):1985–1989, 2008a.

Kai Labusch, Fabian Timm, and Thomas Martinetz. Simple incremental one-class Support Vector classification. In Gerhard Rigoll, editor, *Pattern Recognition - Proceedings of the DAGM*, Lecture Notes in Computer Science, pages 21–30, 2008b.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas: Learning of Overcomplete Data Representations. *Neurocomputing*, 72(7–9): 1547–1555, 2009.

Robert Lange. *3D Time-of-Flight Distance Measurement with Custom Solid-State Sensors in CMOS/CCD-Technology*. PhD thesis, University of Siegen, Germany, 2000.

Yvan G. Leclerc and Aaron F. Bobick. The direct computation of height from shading. In *Computer Vision and Pattern Recognition (CVPR '91)*, pages 552–558, 1991.

Michael S. Lewicki, Terrence J. Sejnowski, and Howard Hughes. Learning overcomplete representations. *Neural Computation*, 12:337–365, 2000.

Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Proceedings of the 25th Annual Symposium of the German Association for Pattern Recognition (DAGM)*, pages 297–304, 2003.

M. Lindner and A. Kolb. Compensation of Motion Artifacts for Time-of-Flight Cameras. In *Proc. Dynamic 3D Imaging*, LNCS, pages 16–27. Springer, 2009.

Marvin Lindner and Andreas Kolb. Lateral and depth calibration of PMD-distance sensors. In *Proc. Int. Symp. on Visual Computing*, LNCS, pages 524–533. Springer, 2006.

Stuart P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the IEEE International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

Thomas Martinetz and Klaus Schulten. A "Neural-Gas" Network Learns Topologies. *Artificial Neural Networks*, I:397–402, 1991.

Thomas Martinetz, Kai Labusch, and Daniel Schneegaß. SoftDoubleMaxMinOver: Perceptron-like Training of Support Vector Machines. *IEEE Transactions on Neural Networks*, 20(7):1061–1072, 2009.

Thomas B. Moeslund and Erik Granum. Modelling and estimating the pose of a human arm. *Machine Vision and Applications*, 14:237–247, 2003.

Thomas B. Moeslund and Lau Nøregaard. Recognition of deictic gestures for wearable computing. In *6th International Gesture Workshop, GW 2005, Revised Selected Papers*, volume 3881 of *Lecture Notes in Computer Science*, pages 112–123, 2006.

Mostafa G.-H. Mostafa, Sameh M. Yamany, and Aly A. Farag. Integrating shape from shading and range data using neural networks. In *Computer Vision and Pattern Recognition (CVPR '99)*, volume 2, page 2015, 1999.

Cicero Mota and Erhardt Barth. On the uniqueness of curvature features. In G. Baratoff and H. Neumann, editors, *Dynamische Perzeption*, volume 9 of *Proceedings in Artificial Intelligence*, pages 175–178, Infix Verlag, 2000.

Sateesha G. Nadabar and Anil K. Jain. Fusion of range and intensity images on a Connection Machine (CM-2). *Pattern Recognition*, 28(1):11–26, 1995.

Kai Nickel and Rainer Stiefelhagen. Pointing gesture recognition based on 3D-tracking of face, hands and head orientation. In *International Conference on Multimodal Interfaces*, pages 140–146, 2003.

Daniël Van Nieuwenhove. *CMOS Circuits and Devices for 3D Time-of-Flight Cameras*. PhD thesis, Vrije Universiteit Brussel, Belgium, 2009.

Donald A. Norman. *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. The MIT Press, Cambridge, Massachusetts, 1998.

Thierry Oggier, Michael Lehmann, Rolf Kaufmann, Matthias Schweizer, Michael Richter, Peter Metzler, Graham Lang, Felix Lustenberger, and Nicolas Blanc. An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRanger). In *Proceedings of SPIE*, volume 5249, pages 534–545, 2004.

Thierry Oggier, Bernhard Büttgen, Felix Lustenberger, Guido Becker, Björn Rüegg, and Agathe Hodac. SwissRanger™ SR3000 and first experiences based on miniaturized 3D-TOF cameras. In *Proceedings of the 1st Range Imaging Research Day*, pages 97–108, Zürich, Switzerland, 2005a.

Thierry Oggier, Bernhard Büttgen, Felix Lustenberger, Guido Becker, Björn Rüegg, and Agathe Hodac. SwissRanger™ SR3000 and first experiences based on miniaturized 3D-TOF cameras. In Kahlmann Ingensand, editor, *Proc.* 1^{st} *Range Imaging Research Day*, pages 97–108, Zurich, 2005b.

Bruno A. Olshausen. Learning sparse, overcomplete representations of time-varying natural images. In *Proc IEEE International Conference on Image Processing*, volume 1, pages 41–44. IEEE Computer Society, 2003.

Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

Kálmán Palágyi and Attila Kuba. A parallel 3D 12-subiteration thinning algorithm. *Graphical Models and Image Processing*, 61(4):199–221, 1999.

Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

Ramprasad Polana and Al Nelson. Recognizing activities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–818, 1994.

Michael Potmesil and Indranil Chakravarty. A lens and aperture camera model for synthetic image generation. In *SIGGRAPH Computer graphics*, pages 297–305, ACM, 1981.

Michael Potmesil and Indranil Chakravarty. Synthetic image generation with a lens and aperture camera model. *ACM Transactions on Graphics*, 1(2):85–108, 1982.

Daniel Potts and Gabriele Steidl. A new linogram algorithm for computerized tomography. *IMA Journal of Numerical Analysis*, 21(3):769–782, 2001.

Daniel Potts, Gabriele Steidl, and Manfred Tasche. Fast Fourier transforms for nonequispaced data: A tutorial. In John J. Benedetto and Paulo J. S. G. Ferreira, editors, *Modern Sampling Theory: Mathematics and Applications*, pages 247–270. Birkhäuser, Boston, 2001.

William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, second edition, 1992.

Chris Pudney. Distance-ordered homotopic thinning: A skeletonization algorithm for 3D digital images. *Computer Vision and Image Understanding*, 72:404–413, 1998.

Holger Rapp. Experimental and theoretical investigation of correlating ToF-camera systems. Master's thesis, University of Heidelberg, Germany, 2007.

Jim Rehg, Daniel D. Morris, and Takeo Kanade. Ambiguities in visual tracking of articulated objects using two- and three-dimensional models. *International Journal of Robotics Research*, 22(1):393–418, June 2003.

Przemyslaw Rokita. Fast generation of depth of field effects in computer graphics. *Computers and Graphics*, 17(5):593–595, 1993.

Przemyslaw Rokita. Generating depth-of-field effects in virtual reality applications. *IEEE Computer Graphics and Applications*, 16(2):18–21, 1996.

Rómer Rosales and Stan Sclaroff. Inferring body pose without tracking body parts. In *Proceedings of Computer Vision and Pattern Recognition*, pages 721–727, 2000.

Dimitrios Samaras, Dimitris Metaxas, Pascal V. Fua, and Yvan G. Leclerc. Variable albedo surface reconstruction from stereo and shape from shading. In *Proceedings of Computer Vision and Pattern Recognition*, volume 1, pages 480–487, 2000.

Michael Schlemmer, Ingrid Hotz, Vijay Natarajan, Bernd Hamann, and Hans Hagen. Fast Clifford Fourier transformation for unstructured vector field data. In *Proceedings of the Ninth International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 101–110, 2005.

Rudolf Schwarte, Horst G. Heinol, Zhanping Xu, and Klaus Hartmann. New active 3D vision system based on rf-modulation interferometry of incoherent light. In *Intelligent Robots and Computer Vision XIV*, volume 2588 of *Proceedings of SPIE*, pages 126–134, 1995.

Cary Scofield. $2^1/_2$d depth-of-field simulation for computer animation. In *Graphics Gems III*, pages 36–38. Academic Press Professional, Inc., San Diego, CA, USA, 1992.

Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.

Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings of International Conference on Computer Vision*, pages 750–757, 2003.

Eli Shechtman and Michal Irani. Space-time behavior-based correlation—or—how to tell if two underlying motion fields are similar without computing them? *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):2045–2056, 2007.

Mikio Shinya. Post-filtering for depth of field simulation with ray distribution buffer. In *Graphics Interface*, pages 59–66. Graphics Interface '94, 1994.

Hagen Spies, Horst Haußecker, Bernd Jähne, and John L. Barron. Differential range flow estimation. In *Mustererkennung 1999, 21. DAGM-Symposium*, pages 309–316, London, UK, 1999. Springer-Verlag. ISBN 3-540-66381-9.

Hagen Spies, Bernd Jähne, and John L. Barron. Dense range flow from depth and intensity data. In *ICPR00*, volume 1, pages 131–134, 2000.

Hagen Spies, Bernd Jähne, and John L. Barron. Range flow estimation. *Computer Vision and Image Understanding*, 85(3):209–231, 2002.

George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 0: 2004–2011, 2009.

Clay Matthew Thompson. Robust photo-topography by fusing shape-from-shading and stereo. AI Technical Report 1411, Massachusetts Institute of Technology, 1993.

Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

Tiberiu Viulet. Range flow computation using time-of-flight cameras. Diploma thesis, Universitatea Politehnica Bucuresti, 2009.

David J. Ward and David J.C. MacKay. Fast hands-free writing by gaze direction. *Nature*, 418(6900):838, 2002.

Sebastian Weik and C.-E. Liedtke. Hierarchical 3D pose estimation for articulated human body models from a sequence of volume data. In *Robot Vision*, pages 27–34, 2001.

Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Comput. Vis. Image Underst.*, 104(2): 249–257, 2006.

Mark Weiser. The computer for the 21st century. *Scientific American*, 1991.

Zhanping Xu, Rudolf Schwarte, Horst Heinol, Bernd Buxbaum, and Thorsten Ringbeck. Smart pixel – photonic mixer device (PMD). In *Proceedings of the International Conference on Mechatronics and Machine Vision in Practice*, pages 259–264, Nanjing, China, 1998.

Giora Yahav, Gabi J. Iddan, and David Mandelboum. 3D imaging camera for gaming application. In *International Conference on Consumer Electronics*, pages 1–2, 2007.

M. Yamamoto, P. Boulanger, J. A. Beraldin, and M. Rioux. Direct estimation of range flow on deformable shape from a video rate range camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):82–89, 1993.

Hee-Deok Yang and Seong-Whan Lee. Reconstructing 3D human body pose from stereo image sequences using hierarchical human body model learning. In *ICPR*

'06: Proceedings of the 18th International Conference on Pattern Recognition, pages 1004–1007, IEEE Computer Society, 2006.

Hee-Deok Yang and Seong-Whan Lee. Reconstruction of 3D human body pose from stereo image sequences based on top-down learning. *Pattern Recognition*, 40(11): 3120–3131, 2007.

Alper Yilmaz and Mubarak Shah. A differential geometric approach to representing the human actions. *CVIU - International Journal of Computer Vision and Image Understanding*, 109(3):335–351, 2008.

Lijun Yin and Anup Basu. Nose shape estimation and tracking for model-based coding. In *Proceedings of IEEE International Conference on Acoustics, Speech, Signal Processing*, pages 1477–1480, 2001.

Christoph Zetzsche and Erhardt Barth. Fundamental limits of linear filters in the visual processing of two-dimensional signals. *Vision Research*, 30:1111–1117, 1990a.

Christoph Zetzsche and Erhardt Barth. Image surface predicates and the neural encoding of two-dimensional signal variation. In Bernice E Rogowitz, editor, *Human Vision and Electronic Imaging: Models, Methods, and Applications*, volume SPIE 1249, pages 160–177, 1990b.

Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

Youding Zhu, Behzad Dariush, and Kikuo Fujimura. Controlled human pose estimation from depth image streams. *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–8, 2008.

Index