Universität Bielefeld

Vision-based Posture Detection and Tracking for Interactive Scenarios

Dissertation zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

der Technischen Fakultät der Universität Bielefeld

vorgelegt von
Joachim Schmidt

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

Contents

1 Introduction

"Man has learned much from studies of natural systems, using what has been learned to develop new algorithmic models to solve complex problems. [...] A major thrust in algorithmic development is the design of algorithmic models to solve increasingly complex problems. Enormous successes have been achieved through the modelling of biological and natural intelligence, resulting in so-called 'intelligent systems'."

Andries P. Engelbrecht (2007) [43]

"At the basic level, the name given to the science dedicated to the broad area of human movement is kinesiology. It is an emerging discipline blending aspects of psychology, motor learning, and exercise physiology as well as biomechanics. Biomechanics, as an outgrowth of both life and physical sciences, is built on the basic body of knowledge of physics, chemistry, mathematics, physiology, and anatomy. It is amazing to note that the first real 'biomechanicians' date back to Leonardo DaVinci, Galileo, Lagrange, Bernoulli, Euler, and Young. All these scientists had primary interests in the application of mechanics to biological problems."

David A. Winter (1990) [171]

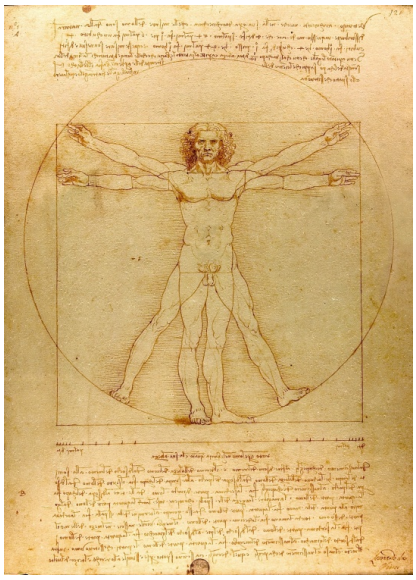


Figure 1.1: *Vitruvian Man*. Painting by Leonardo Da Vinci (1485/90, Venedig, Galleria dell' Accademia), Photo by Luc Viatour.

As Engelbrecht and Winter mention, it has often been nature that inspired man to develop new ideas and that encouraged us to use these ideas for applications that can affect our daily life. For any scientist, curiosity and amazement are two substantial characteristics. For me this has often shown in amazement about the solutions that nature provides for many big and small problems and in curiosity, how theories, concepts and finally algorithms and systems could eventually be derived from that. These are the kind of thoughts that have driven my research for the last years.

The thesis here present is about the perception of the human body and the environment with means of computer vision and the analysis of these information for applications in the field of human robot interaction. The discussion will mostly be about real-world scenarios involving the observation of real humans; that means we will have to deal with an ever-changing and dynamic environment and possibly large variations in the appearance of an object to be observed. This poses a huge challenge to automated vision techniques. Additional constraints can ease the problem, but also make the resulting system less flexible. The presented work combines various techniques from computer vision and optimization theory. The scenarios that are addressed are wide spread: worker safety in an industry environment, interacting with a mobile robot and even understanding the relevance of gestures for learning in children. The common ground for all these scenarios is the fact that methods from computer vision are applied to enable or to understand an interaction between humans among themselves and humans and machines. The best way to outline the scope of this thesis is to describe the topics covered.

Computer vision is a broad discipline, as are the applications where automated vision techniques are applied. To get a better focus on the relevant topics, the Chapter (2) gives an overview of related approaches and techniques that are of special importance for this thesis. The basic step for any of the presented approaches is to find the human in the scene. For camera images, humans can be found based on their appearance. More detailed methods are able to find individual body parts and can put them in relation with each other to reconstruct the pose of the human. Besides working with the 2D information from a single image, using volumetric data has become more and more common with the availability of affordable sensors and fast but reliable algorithms. Such data can significantly improve the performance for localizing persons and objects in a scene, especially when incorporating motion information. The application of such techniques to human robot interaction has led to some remarkable systems that can handle challenges like ambiguities in the appearance, a changing environment and the variability of the objects and persons observed.

During the work on this thesis, optimization techniques have consistently been a central part of the algorithms and methods developed. Chapter (3) describes the theoretical background of optimization. Optimization means two things here. First, a theory to find a mathematical formulation for a given problem such that a solution can be found. Secondly, it means a technique or an algorithm that realizes a search process for this solution given the constraints of the specific scenario. It is also important to mention that the general term optimization is always meant here in the context of an application as we aim at finding an optimal solution to solve a given task. Presenting the algorithms

in a chapter on their own provides the opportunity for a better comparison between the individual algorithms, actually to find that there are more similarities then differences, without being too much diverted by the concrete application.

The primary step for a system trying to interact with a human as a potential interaction partner must be to localize the human in the environment. The main task is therefore to find its position in space, which can be achieved by using volumetric data originating from a stereo camera system or a time-of-flight sensor. There is even a market for industrial applications of such localization systems. The SafetyEYE[1], produced and sold by Pilz and developed in cooperation with Daimler, is a camera-based system that can detect if a person enters a potentially hazardous area, for instance the operational range of an industry robot. Some of the processing steps this device uses to analyze the image data have a substantial similarity with the algorithms presented in this thesis. Going further, additional velocity information can not only help to improve the segmentation, it can also be used to predict the motion of the human and other objects in the scene. In Chapter (4), the principle of abstracting from the raw data in multiple steps is introduced. As a first level of abstraction, locally dense sets of points exhibiting similar velocity annotations are summarized to form clusters. They serve as the basis for generating person hypothesis using cylinders as a weak object model. It is presented how these hypotheses can be tracked over time using particle filtering framework.

The principle of matching a parameterized model representation with the image data is further exploited in Chapter (5). While the previous chapter is oriented more on large-scale scenarios, we now aim at observing the motions of an individual person trying to interact with a mobile robot using a single monocular camera. A clear focus on needs of the proposed scenario helps to restrict the variability in possible situations the system should be able to cope with. The goal of the studies is to develop a system that is able to track the gestures of a human in 3D, focusing on the arms, while posing no restrictions on the type of motions performed. In particular, the system should be able to track prior unseen motions. This can be achieved by using an articulated 3D upper body model that describes the physical properties and also serves as a model for the appearance of the human. An inference process rates each configuration of the model on its agreement with the image data and provides a pose likelihood by fusing information from multiple cues, which is a special challenge when using monocular images only. The space of possible configurations of the model is defined by the 14 joint angles. The task of finding the best-fitting pose is now to locate the point in the parameter space which represents this pose. The ambiguity, nonlinearity, and non-observability during the inference process make the posterior likelihood in the space of the body configurations multi-modal and unpredictable. Probabilistic search processes have shown their ability to efficiently explore such highdimensional spaces. The proposed system therefore combines the kernel particle filtering technique with intermediary mean shift optimization steps that help to better exploit the number of particles available. A combination of motion models, including priors for modeling the motion of an individual joint, are employed to narrow the search space. To allow a self-starting tracking, an automatic initialization routine is proposed that builds up on

[1] `http://www.pilz.de/products/sensors/camera/f/safetyeye/index.jsp`

the detections of a face recognition module to obtain a rough guess on the initial pose and to learn a color appearance model of the observed person.

To summarize the work presented in these two chapters, the goal of my thesis consists in providing methods that facilitate the automatic localization and tracking of humans in interaction scenarios. This includes the development of novel pose detection methods as well as the investigation of mechanisms allowing for an analysis of high dimensional and multi-modal feature spaces. Furthermore, the application of these techniques for various applications constitutes an innovative contribution to the current research in robotics and human machine interaction.

Eventually, the most important results of my work are summarized in Chapter (6), which presents the evaluations that have been carried out to examine the accuracy of the previously presented approaches for person localization and body pose tracking. A leading idea for the development of the person localization system has been to use it as a basic component for a safety system in an industrial workspace. The feasability of the approach for such a setting can only be assured if the localization is both precise and robust. Therefore, the system has been applied to a number of ground truth annotated image sequences, measuring the algorithm's ability to detect static and moving objects. Similarly, ground truth data can be used to measure the reconstruction accuracy of the body pose tracking. As the level of detail of the generated results is much higher here, this also calls for a more elaborate evaluation method. For comparing the estimated model pose with the actual pose of the human, a measure based on the positioning of the person's individual body parts is motivated. Recording the according ground truth corpus is made possible by calibrating and synchronizing an active infrared marker tracking system with a standard camera. The results allow a detailed inspection of the behavior of the algorithm under different parameterizations and for different scenarios. Going even further, a method for an automatic optimization of the system's parameters that makes use of the same ground truth corpus is presented and evaluated. Using an evolutionary computation approach, an optimized set of parameters is automatically generated to enhance the performance of the body tracking system for a given task. This approach is based on a genetic algorithm which tests differently parameterized instances of the body pose tracking system regarding their tracking accuracy and robustness. As a tremendous amount of computational power is needed for this kind of evaluation, the proposed approach offers a distributed computing framework to combine the computers available in a local network.

Chapter (7) addresses the applications that the previously presented approaches have been used for so far. These are in particular the reconstruction of static scenes using a mobile robot, a fast and reliable hand gesture detection system, understanding the importance of gestures for early-childhood learning of actions and finally a gesture recognition and object attention system for a mobile robot. The systems designed for these tasks are usually not based on a single algorithm, rather the presented approaches are combined in a new fashion for each task.

This thesis ends with a discussion on the benefit of the presented work for potential users in Chapter (8). Furthermore, possible extensions are addressed, which could be of interest in the future.

2 Related Approaches for Recognizing Humans

In the following chapter, related approaches for recognizing humans using computer vision systems are presented. While this is a huge field to cover in general, we will rather focus on approaches that are of special significance for developing interactive systems. As an example, the topic of surveillance will be addressed from the viewpoint of recognizing and observing one or few persons instead of unserstanding the behaviour of large groups of people. Also, we are most interested in recognizing the human as a whole up to the detail of individual body parts as seen from a distance of several meters. Detecting fine details, like the movements of the fingers, or a precise reconstruction of the surface can be used to understand the motions of an arm, but this level of detail is not what this thesis aims at.

The main topics to be discussed in the following are localizing persons and objects in the environment and tracking their motions, reconstructing the pose of an individual human and tracking his motions and discussing suitable modelling approaches including topics like the initialization of tracking systems and the recovery from failues. Works on gesture detection and action recognition present options to understand the meaning of the recognized motions in the context of a specific scenario. The chapter concludes with a presentation of approaches that employ the discussed techniques in human robot interaction scenarios.

2.1 Person Localization

Within the last years, lower production costs lead to a big increase in the number of video surveillance cameras at public places. Even for the current number of deployed cameras, an analysis of the images by a human observer is virtually impossible, and their number is still growing. This is why an automated analysis is seen by many as the only possible way to handle the big amount of data recorded. This need is also reflected in the steadily growing activity of the computer vision community concerning this topic. Concerns about pervasive surveillance and the consequences for a society are not new [118] but arguing this topic will be left for others. Here, we will rather focus on the benefits of these works for interactive vision systems.

2D Approaches

A first step to understand what happens in a scene is to analyze the presence of humans in static images. If a human has been detected, consecutive methods can extract more detailed information, for instance the path the human is walking and his interactions with other humans or objects in the scene.

Surveillance cameras are typically set up to observe a specific location, like public places. For such setups, humans are typically far away and show up quite small in the image. As they are usually passing by, the motion to be observed most commonly will therefore be walking. To robustly detect humans in images, an algorithm can make use of the

Figure 2.1: *Wavelet descriptors for pedestrian detection.* Not all features are equally important for the task of detecting a human. The image shows the activation for three coefficients resembling different filter directions at two different scales. The average human shape is clearly visible. (Image found in [119])

constraints of the scenario. Pedestrians, for example, can be robulstly detected [119] by applying a multi scale search and using a support vector machine (SVM) classifier with wavelet descriptors for detection, cf. Fig. (2.1). Additionally, motion information can be taken into account, as it is presented by Viola et al. [164]. The combined motion and appearance descriptor can be efficiently trained using AdaBoost [49]. Dalal and Triggs [33] proposed a 2D global detector using histograms of oriented gradients (HoG) as descriptors which can be calculated very efficienty. The classification is based on a linear SVM for best runtime efficiency. A big step forward is also that fact that their system tolerates different poses, clothing, lighting and background much more than previous approaches. But it currently works for fully visible upright persons only. More general features, namely HoGs of variable-size blocks and a rejection cascade for improved performance are presented by Zhu et al. [176] as an extension of the former approach.

Figure 2.2: *Scene geometry reconstruction.* After estimating planar structures in the image, the search pattern of the detector is adapted to handle the perspective transformations. (Image found in [71])

Apart from the detection of learned patterns, the structure of the image helps to understand it [71]. From estimating planar structures such as walls and the floor the camera viewpoint can be derived as well. Given that, the 3D relationships of the camera, the surfaces and the objects in the scene can be reconstructed and can be used in a further step to refine the search process as depicted in Fig. (2.2). Contextual information can also be exploited using a bilattice-based logical reasoning approach [140] that integrates knowledge about interactions between humans and can also deal with uncertainties from detections and even from logical rules. If multiple cameras are avaiable, information about the observed persons and objects can be interrelated. Such a system is able to track multiple targets even in crowded environments [124].

3D Approaches

Three-dimensional vision plays an important role in human perception, especially for the recognition of motion and the ability to track objects over time. It is advantageous for vision systems to make use of this very basic information in order to perceive and interprete the environment. Multiple cameras in a surveillance scenario can be used to

Figure 2.3: *Kernel-based 3D person tracking.* A 3D Point cloud is generated using multiple camera views. Tracking robustness also benefits from fusing multiple appearance features. (Image found in [162])

generate 3D point clouds of the moving objects. The localization and tracking of objects is achieved by mean shift clustering of the point cloud [87]. Extending this approach by combining evidence from multiple calibrated cameras allows more robust tracking [162]. Here, the appearance features from all cameras are fused during the detection step, see Fig. (2.3) for typical results. In addition to the 3D point cloud, contour and flow detection in the image plane yields motion information that can be used for person tracking [139]. For mobile robots, mapping of the environment and localizing and tracking moving objects can be combined into a single framework [113, 167]. The authors show that neglecting moving objects during mapping yields more accurate results.

2.2 Pose Reconstruction and Motion Tracking

Analyzing humans in motion has a long history, driven by the wish to understand the functionality of the human body. Many different techniques to retrieve the pose of a human have been developed, each serving a specific purpose and considering constraints as for example the need for realtime analysis, an accurate reconstruction or versatility and simplicity of use.

The Role of Markers

Marey used photos with multiple exposures at a fixed interval of time to record a moving person [107]. For his studies, the subjects had to wear black suits with reflecting pieces of metal and white lines, see Fig. (2.4). This is one of the first marker-based approaches to record human motions. In the composite image, the markers form trajectories depicting the movement of the different body parts, see Fig. (2.4). Even today, more than 120 years later, the state of the art for recording human full body motions are visual systems using active or passive markers and multiple cameras to simplify the correspondence problem. Such systems are commercially avaiable, e.g., from VICON[1] or

[1] www.vicon.com

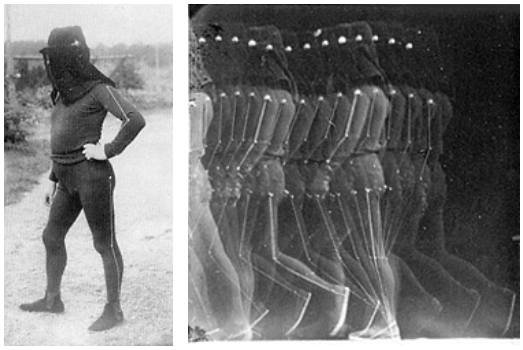


Figure 2.4: *Marey's "Motion Capture" suit.* During his studies of the human in motion he created characteristic multi-exposure pictures showing subjects wearing black suits with reflecting markers or white lines.

LUKOTRONIC[2], but their applicability is limited due to the heavily instrumented setups and the demands made on the environment. Some of these restrictions can be overcome by using other sensors that do not rely on visual information, e.g., accelerometers or electromagnetic tags that can be exactly localized in a confined space. For recording the motions of a human the direct measuring of joint angles is also an option if an exact kinematic model has been derived beforehand. A goniometer, attached to a limb, can provide the relative position with respect to the adjacent limb by measuring the bending of a sensor bar.

Vision for Markerless Tracking

The major challenge of moving from the laboratory to the real world remains. Many recent works deal with the task of reconstructing the pose of the observed human from visual input only, without the need for special markers or posing strong restrictions on other factors like clothing, lighting conditions or the background. The surveys of Gavrila [54], and Moeslund et al. [111, 112] provide a good overview on the topic of tracking the human body.

Tracking the pose of a human over time is a problem of inference under uncertainty and ambiguity. Powerful inference and learning algorithms are needed for an effective solution as well as complex models for the appearance and motions.

Appearance-based Approaches

The methods presented earlier for detecting a person as a whole work well for pedestrians or distant detections. For complex human motions, however, they may not scale well enough as the appearance changes drastically when considering all possible articulations.

One way to overcome this is to extended the known approaches for localizing the human by detecting simple parts like the face and the limbs individually. Such pictorial structures generalize well and show robust performance concerning cluttered background or varying appearence [46]. Ramanan and Forsyth represented the human as an articulated

[2] `www.lukotronic.com`

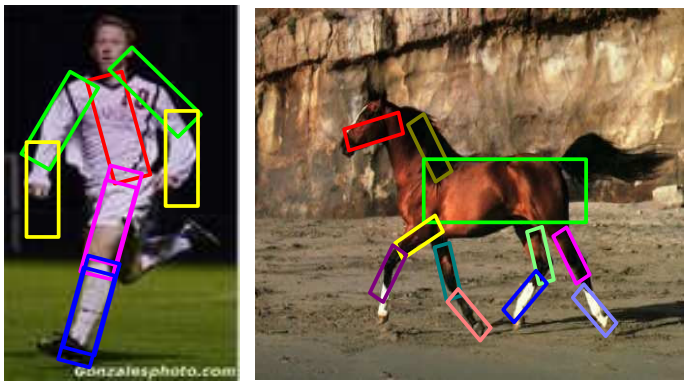


Figure 2.5: *Articulated part-based 2D model.* The generic model structure allows efficient training and tracking for miscellaneous creatures, e.g. humans and horses. (Image found in [128])

structure and presented a 2D part-based model detection system that finds typical poses using a generic pictorial model [126]. An efficient training and inference scheme for such models can be achieved using dynamic programming [128]. The technique works well for different articulated objects, like humans and other animals as shown in Fig. (2.5) The individual body parts are often organized in a tree structure that models the correlations between the limbs and enforces consistency. Choosing a different representation can halp to avoid expensive inference due to large cliques, as Lan and Huttenlocher [96] showed for their common-factor models.

Introducing 3D Models

By deriving 2D joint positions from the individual parts and incorporating knowledge about the camera projection and the limb lengths, the amount of ambiguity can be reduced far enough to allow a reconstruction of the 3D joint positions [97]. A full 3D body model can be constructed in a probabilistic way out of body parts detected in the image [125, 143]. Besides the ability to initialize and recover from almost arbitrary poses, the advantage of such systems is that they are usually more robust against occlusions of individual body parts. As a drawback, the body part detection is slow and the generation of a coherent body model proves to be prone to confusions and ambiguities and therefore often provides unpredictable results.

Multiocular Pose Reconstruction

Approaches for pose tracking need to solve a complex inference task, especially for the monocular case. Monocular algorithms achieving a high accuracy are often computationally intensive, which prohibits their use for realtime human-machine interaction. Depending on the aimed scenario, however, different techniques to lower the complexity of the inference task can be applied. With multiple cameras or a multiocular camera systems, images from different viewpoints or the additional depth information can be used to resolve ambiguities. Posing restrictions on the environment or the appearance of the tracked persons helps to narrow the search space, as also restricting the number of detectable motions does.

Deutscher [37] employs an articulated 3D body model based on truncated cones and a cost function based on edge and silhouette information. The images have been acquired using 3 calibrated cameras in scenarios with a black background. Tracking has been performed using the annealed particle filter. The authors report very good tracking results for general unconstrained motions, but the computation times are far from real-time. Multiple cameras are used in many systems to resolve ambiguities and to cope with self-occlusions [39, 88, 132], but such approaches either require camera systems with a wide baseline that are of no interest for our proposed scenarios or require too large an mount of computational resources. Tracking of a human in 3D with limited computational resources on a mobile robot was already described in 1996 by Kortenkamp et al. [91]. This approach uses depth information from a stereo camera to track a coarse 3D model of a single human arm but is restricted to slow motions and uses a small repertoire of static gestures. Even for stereo approaches, simple image features like skin color are often integrated as an additional cue to get the 3D hand position and its pointing direction more accurately [115].

Other 3D Approaches

The use of multiple cameras in a task consisting in tracking the human body from a mobile robot is technically difficult. In contrast to stereo camera systems, time-of-flight sensors provide a denser depth map but with a relatively low spatial resolution. Tracking only few anatomical landmarks in these distance maps is already sufficient for estimating a pose in quasi realtime, although the spatial accuracy does not match camera-based approaches [177]. Time constraints and also difficulties in recognizing ambiguous poses can be overcome when 3D information acquired from time-of-flight sensors are avaiable, as shown by [133].

Monocular Pose Reconstruction

Only few authors have addressed the problem of 3D full-body tracking using a single uncalibrated camera. One such approach for tracking a detailed 3D human body model was proposed by Sidenbladh [141, 142]. It is based on a variety of gray-level image cues and a particle filter for tracking the human motions. To cope with the huge search space, motion priors are used to predict the 3D body configuration prohibiting the tracking of unconstrained motions. Learned image statictics help to interpret the likelihood of body parts matching with image features.

Following Sidenbladh's work, Sminchisescu used a more precise modeling of the 3D body model and a complex parameter space exploration [148], but the computational time required prohibits its use for real-time tracking. To cope with a large parameter space, kernel-based Bayesian filtering has been proposed to track objects or isolated body parts in the 2D image space [23, 63]. The problem dimensionality can be reduced by performing standard dimensionality reduction techniques, as shown by Zhao and Liu [175]. The pose is then reconstructed using an annealed genetic algorithm that exploits the hierarchical structure of the solution space and the local characteristics of the

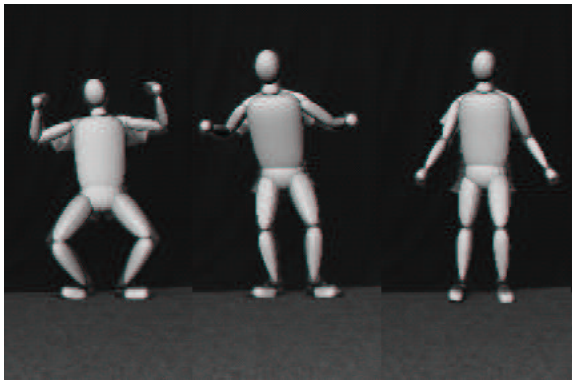
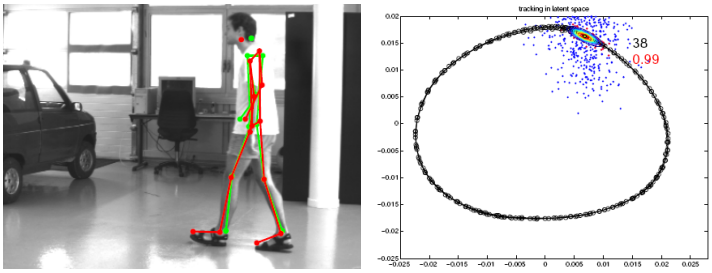


Figure 2.6: *Kinematic full body model.* The 'flesh' is modeled by superquadric ellipsoids, the whole model has 30 degrees of freedom. (Image found in [147])

fitness function. In addition to a kinematic body model which describes the appearance of the body, bringing in prior knowledge about familiar body configurations [16] can help to constrain the search process and prevent the production of unrealistic pose estimates.

The advantages of appearance-based and model-based approaches can be combined to capture the multimodal and nonlinear relationships more reliably. Sigal and Black [145] propose a multi-stage approach utilizing 2D bottom-up body part detectors for initialization and non-parametric belief propagation for pose estimation in combination with a learned mapping from 2D to 3D poses to estimate the full 3D body pose even from monocular images. Jaeggli et al. [81] propose to learn a joint probability distribution of the appearance and the body pose using a mixture of view-dependent models. They formulate inference algorithms that are based on generative models but also exploit the advantages of a learned model. Prior information about likely body poses and a motion model is taken into account. Sminchisescu et al. [146] show how to learn a

Figure 2.7: *Recover human pose from learned examples.* A prior for tracking is constructed utilizing the Laplacian Eigenmaps Latent Variable Model (LELVM) for dimensionality reduction. The walk cycle is projected and then tracked within a low dimensional manifold. (Image found in [103])



generative recognition model that combines top-down and bottom-up processing for monocular 3D human motion reconstruction. The recognition model is used to scan the image and to predict 3D human poses, the inference scheme confirms the detections. The framework covers detection of human body poses as well as initialization and recovery from failures. Lu et al. [103] present a method to reduce the dimensionality of the reconstruction problem and bias the estimates towards typical human poses and motions even for missing, noisy and ambiguous image measurements. The method is computationally efficient and requires only few training data. Unknown poses or motions, however, can be covered only to a cretain degree, depending on the similarity to the already seen examples.

2.3 Model Acquisition, Initialization and Error Recovery

For bottom-up approaches, the appearance model of the human is typically learned from training examples, a well known example is the approach proposed by Viola et al. [164]. Top-down approaches, however, use a given model for the inference process. The idea is to integrate common knowledge about the human body and its characteristics into the model that can not – or not that easily – be learned from training images. The more complex the model is, the more detailed can it represent the human body and the more can it exert influence during the inference process.

But then, the challenge of how to acuire such a detailed model remains. Furtunately, the human body has long been subject to scientific research. Already Leonardo da Vinci was inpired by the idea to define rules that can describe the proportions of a perfect man, see Fig. (1.1). It finally turned out that he managed to give a good average of all men living at that time. These formal descriptions of the human body are still used for various purposes, e.g.in industry when designing cars and airplanes or in biomechanics for analyzing the performance of competitive athlets [168]. In these cases, the model is used as an abstract representation that describe the properties of the body, like forces during movements or the shape in different poses. The human body is often represented as a set of geometric components, such as spheres, cylinders and cones, resembling the different body segments. A well known realization of a segment model was introduced 1964 by Hanavan, see Fig. (2.8). There exist many modifications derived from this representation, many systems still base on this type of model. Segment models require only few simple anthropometric measurements, e.g., segment lenths and circumferences. These can be measured individually for each person to be analyzed or they can be chosen from a generic model representing the mean measures of a human. In his report, Bjørnstrup gives a chronological overview of the methods used for estimation of body segment parameter from the 17th century to 1995 [12].

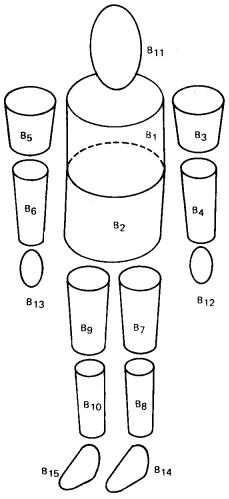


Figure 2.8: *Hanavan's model of the human body.* A 15-body 34-degrees-of-freedom, finite-segment model of the human body. The 15 bodies B_k (k= 1, ..., 15) are connected through hinges and ball-and-socket joints modeling the human limb connections. (Image found in [77])

Some of the works mentioned above also include a learning part for the structure or the appearance of the model. A generic approach for learning an appearance model

from examples has been presented by Oechsle [116]. His approach is based on genetic programming and uses sub-goals during learnig similar to the boosting technique. Pictorial structures make use of the spatial relations between multiple image parts. These part-based models can also be learned from training examples, including articulations to a certain degree, but as 2D planar models they remain in the image plane [46, 126]. Adding additional constraints about the connections between body parts, full 3D body models can be constructed out of body parts detected in the image [125, 127, 143].

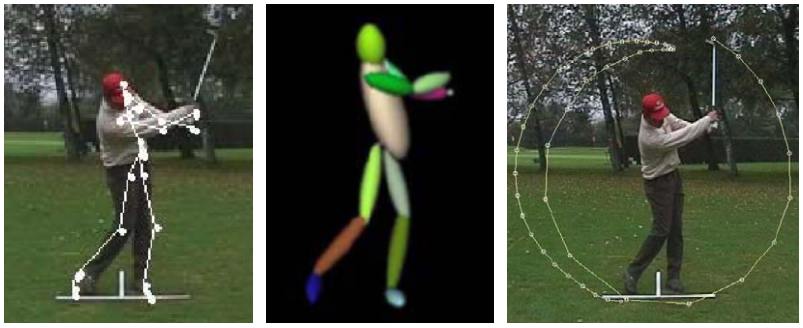


Figure 2.9: *Task specific initialization and tracking.* The golf club is tracked to find key postures for initialization, tracking utilizes local 2D appearance models and a strong motion prior from a database of learned golf swings. (Image found in [163])

The algorithms discussed above provide an estimate of the human's pose simultaneously with the acquisition of the structural model. However, there is still a gap between tracking algorithms and systems working in the real world, mainly due to the fact that for most tracking approaches the challenges of automatic initialization and error recovery are not addressed. For most of the presented approaches, the question of initialization stays open or is subject to manual or semi-automatic procedures.

Urtason et al. [163] present a system that relies on stereotyped poses and can track complex motions with strong self occlusions from monocular video. Interestingly, they use context knowledge to initialize the tracking in this very restricted scenario of tracking a golf swing. They first track the golf club and from that detect key postures to initialize the tracking process, see also Fig. (2.9). As the scenario is very restricted, a deterministic search scheme and a database of learned motions are enough to enable robust tracking.

Other tracking systems incorporating automatic or semi-automatic initialization procedures have been presented recently [11, 161]. They usually rely on learned appearance models or models of typical human motion patterns, e.g. walking. Knowledge about the projection from the 3D world to the 2D camera image can be automatically acquired by using a multi-ocular system for tracking and then mapping the 2D appearance as seen from a single camera to the 3D body pose [145].

For such systems, initialization can also be formulated as the problem of pose estimation or object reconstruction from a single image using strong models [98, 175]. Recovering from tracking errors can then be refered as a different formulation of the initialization problem.

2.4 Vision for Human Robot Interaction

As stated by McNeill gestures play an important role in human communication [109]. According to this, a social robot requires the abilities to localize, track and interpret human behavior during an interaction session. Since the early work of Crowley and Coutaz [31], hands and the vision-based detection of their motions is known as ways to interact with computer systems. A simple 3D body model based on joints and their motion is sufficient for humans to perform action recognition [84]. Having this fact in mind, 3D body acquisition and tracking provides the data source necessary to accomplish such a task, offering a great domain of applications: surveillance, activity recognition, human computer and human robot interaction, mobile robotics, etc.

Using body tracking in human robot interaction often comes with strong restrictions, e.g., the number and type of cameras available or the gesture repertoire to be observed. Pursuing this challenge is an active field in computer vision research, due to the restrictions imposed by approaches developed so far for this purpose.

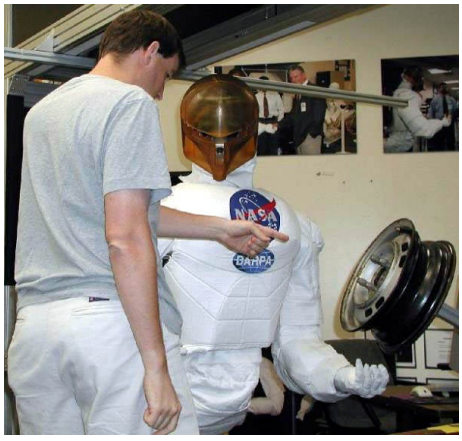


Figure 2.10: *Interaction scenario with the Robonaut humanoid robot.* The human indicates the sequence in which the nuts have to be tightened with pointing gestures, the robot is able to match the visual information and the recognized gestures with the posed task. (Image found in [15])

Especially the combination of several modalities enables robots to become aware of interactions partners in the real world and to achieve situated and human-like interactions with them. A robot system heading for this goal is presented by Stiefelhagen et al. [153]. In detail they utilize the temporal synchrony of speech and hand trajectories for recognizing pointing actions to objects. However, a specialized depth sensor (stereo or time-of-flight) is needed for tracking the human's face and hands in 3D.

But even a monocular camera can be enough for realizing visual recognition and tracking of people and hand gestures, as Germa et al. [55] show with their tour-guide robot. The fusion of multiple visual cues is achieved by a particle filtering framework, command gestures towards the robot are recognized from hand configuration templates. As Germa et al. neither apply 3D body tracking nor hand trajectory recognition, their scenario does not involve interaction with real-world objects. Jenkins et al. [83] presented a kinematic pose estimation and action recognition system for that also monocular vision is sufficient. To interact with the robot, they propose a motion vocabulary comprised of learned primitives that describe human movement dynamics. In the CoSy project, Kruijff et al. [95] work on aspects of anaphoric and exophoric references to objects. They

present a framework to manage the problems of dynamic changes of the environment typical for human-robot interactions. Combining speech and object recognition they achieve learning and detecting of objects. However, they do not dwell on gestural input as an additional cue.

For humans and robots interacting multimodally in order to work together on realworld spatial tasks involving objects, a robust joint visual attention is necessary for achieving a common frame of reference, as proposed by Brooks and Brazeal [15], cf. Fig. (2.10). They present a spatial reasoning framework that is able to determine an object from deictic reference including pointing gestures and speech.

Even if each, object recognition, visual attention, language and action processing yield useful results as individual algorithms, Fay et al. [45] prove that integrating sensory data from different modalities is essential for human-like performance in dealing with ambiguities. Within the last years, this has developed into complex integrated vision systems, integrating gesture recognition, object detection as well as dialog systems to build up multimodal and interactive systems [7].

3 Optimization Techniques

"Nothing is less productive than to make more efficient what should not be done at all."
PETER DRUCKER

Optimization is a term commonly used for describing the act of searching for the best solution to a given problem. Finding the best solution is often understood as finding a sufficiently good one selected from a set of possible solutions based on a quality measure. The nature of the problem determines the attributes of the resulting error function defined through the measure. The amount of possible solutions can be very large, even infinite, which brings the time for finding the solution into play. For optimization problems from the real world the amount of time available for calculation is often limited. Hence, it is favorable to choose a good result that is fast to compute instead of searching for the best result but not knowing when or if the computation will succeed.

Although this chapter is meant to give a general introduction to optimization theory and optimization algorithms, the selection of topics covered here is strongly biased by the needs of the different projects that are part of this thesis. Similarly, we will allow simplifications to general statements where the theory conforms to the task constraints.

This chapter commences with a formal definition of optimization problems. Several techniques to solve such problems are presented thereafter, in particular approaches which have been utilized throughout the thesis. Here, deterministic and probabilistic approaches are distinguished.

3.1 Optimization Problems

The following section gives a formal definition on optimization problems with a special focus on the subject of this thesis. A categorization into different types of problems helps for a given problem to judge its complexity and thereby alleviate the choice of an appropriate optimization method.

3.1.1 Definition of an Optimization Problem

As presented by Engelbrecht [42] a classification problem consists of three parts: An objective function, a set of variables, and a set of constraints.

- **Objective Function:** The objective function f, also called *observation function*, *cost function*, or *error measure*, represents the value to be optimized. It describes the

task to achieve in such a way that its result yields maximum values for the optimal configuration. Ideally, the objective function is expected to be linear and unimodal to guide an optimization process to its maximum. Depending on the formulation of the problem, the aim can also be to minimize f, but this can be transfered into the former formulation by maximizing $-f$. In the following, we assume that f is to be maximized.

Implementations of the objective function can be a model describing an object or a process observed in the real world. The desire for linearity and unimodality often cannot be satisfied but is instead fulfilled locally. The formulation of the objective function is the most critical part when designing an algorithm, as the subsequent optimization techniques solely obtain their knowledge about the problem from observing the values of the objective function.

The general constraints on the observation function are almost negligible. Common restrictions are that it needs to be defined for all feasible parameterizations, some optimization techniques may require additional properties, e.g. that the derivation can be calculated for any given point or that the function is continuous differentiable.

- **Set of Variables:** The parameter vector $\underline{x} \in \mathcal{S}$ represents a set of J variables $\underline{x}^{(j)}$, with $1 \leq j \leq, J$, therefore J denoting the dimension of $\underline{x}$: $\dim(\underline{x}) = J$. To achieve a mathematically correct definition, the variables are assumed to be independent, although in practice they are often not independent or only partly independent. The violation of this condition and its implications have to be considered for each individual implementation. We will refer to the set of variables as the unknown *parameters* that affect the value of the objective function, with the parameter space $\mathcal{S}$ being the domain of all possible parameter combinations. A vector $\underline{x}$ represents a single candidate parameterization. Given this, the objective function $f(\underline{x})$ can be calculated, its result y quantifies the quality of the solution. The vector $\widehat{\underline{x}}$ yielding the highest result is the outcome of the optimization process and thus the estimated optimal configuration, while the true optimum is expressed as $\underline{x}^*$. The value $y^* = f(\underline{x}^*)$ is then the global maximum value of the objective function.

- **Set of Constraints:** Predefined constraints allow the process to exclude certain combinations of values from being considered as a solution. The result is set of allowed solutions $\mathcal{F} \subseteq \mathcal{S}$, called feasible space. For a parameter vector $\underline{x} \in \mathcal{F}$, all constraints are satisfied. During the optimization process, however, it may be beneficial to allow violations of the constraints as long as the final result emanates from the feasible space.

The aim of the optimization is to assign values from the allowed domain to the unknowns, such that the objective function is optimized and all constraints are satisfied. The optimization process searches for a solution in the search space $\mathcal{S}$ while constraints restrict the result $\underline{x} \in \mathcal{F}$ to the feasible space.

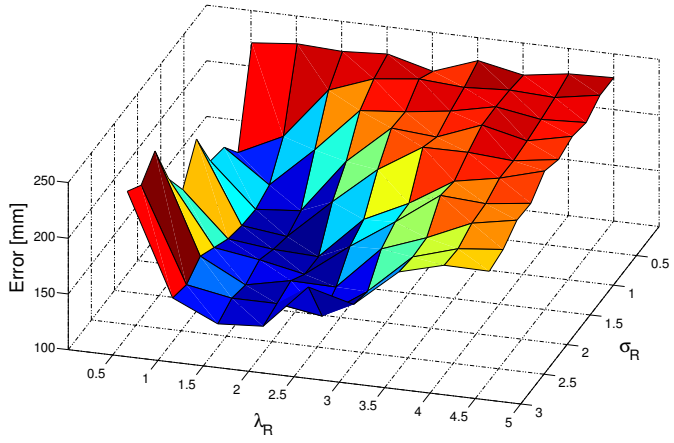


Figure 3.1: *Example of a nonlinear objective function.* The function defined by the two parameters λ_R and σ_R exhibits a nonlinear behaviour with an elongated valley-like area for which the error is minimized.

3.1.2 Problem Classification

Knowledge about the nature of a given optimization problem can help choosing the best suited method. Different types of problems can be categorized based on their characteristics [42].

- **Number of Variables:** The number of variables J defines the dimensionality of the search space $\mathcal{S}$, as each variable corresponds to one dimension, that is $\dim(\mathcal{S}) = J$. If only a single variable is to be optimized ($J = 1$), the problem is referred as univariate. For multiple variables, it is named a multivariate problem.
- **Type of Variables:** Given that all variables of the parameter vector $\underline{x}$ are real, i.e. $\underline{x}^j \in \mathbb{R}$ for each $j \in 1, \ldots, J$, the problem is referred to as a continuous problem. For all variables in $\underline{x}$ being integer valued, i.e. $\underline{x}^j \in \mathbb{Z}$ for each $j \in 1, \ldots, J$, the problem is named discrete. If the parameter vector is composed of both real and integer valued entries, the problem is referred to as a mixed problem.
- **Degree of Nonlinearity of the Objective Function:** Problems differ according to the behavior of the objective function, that can be categorized as linear or nonlinear functions. For linear problems, the value of the objective function is linear in the parameter values. Other dependencies can be stated, e.g., quadratic problems, but generally the problem is called nonlinear for all other behaviors of the objective function. For real world applications, it is often suitable to approximate the objective function to be at least locally linear, although the global behavior with respect to a parameter may be highly nonlinear.
- **Type of Constraints:** It is often reasonable to define boundary constraints that restrict the domain for each variable, e.g., $\underline{x}^1 > 0$ for nonnegative parameters. Such a problem with boundary constraints only is referred to as unconstrained.

  For more complex methods, however, constraints can also be used to consider dependencies between the variables by expressing them as additional equality and/or inequality conditions. The problem is then called a constrained problem.
- **Number of Optima:** If there exists only one clear solution to the optimization problem, it is called unimodal; that is, the objective function has a single distinctive

global maximum.

If more than one solution exists, the problem is called multimodal. Then, the objective function exhibits multiple optima. It can even exhibit false optima, for example due to ambiguities during the observation, making the problem deceptive. Sadly, this is the common case for many real world applications dealing with ambiguities and uncertainties.

- **Number of optimization criteria:** A problem is called uni-objective (or single-objective) if the quantity to be optimized can be expressed using only one objective function. A multi-objective problem specifies more than one sub-objective which need to be simultaneously optimized, e.g., for finding an accurate and at the same time fast parameterization of an algorithm.

The optimization methods used to solve the above problems differ significantly, as will be illustrated in the parts that follow.

3.1.3 Optimality Conditions

The solutions found by an optimization algorithm can be classified concerning their quality. The main types of solutions are local optima and global optima. Knowing about the quality of a solution is of particular importance when observing the real world. In the ideal case, the best solution to the problem is also the global maximum of the observation function. Due to sensor noise and uncertain measurements the resulting error function is more likely to show false local optima and is often deformed. The optimization algorithm has to account for these conditions and, for example, needs to be able to escape from false local maxima.

A global optimum can be defined as follows:
The solution $\underline{\widehat{x}} \in \mathcal{F}$, is a global optimum of the objective function f if

$$f(\underline{\widehat{x}}) > f(\underline{x}), \forall \underline{x} \in \mathcal{F} \tag{3.1}$$

where $\mathcal{F} \subseteq \mathcal{S}$. Thus, $\underline{\widehat{x}}$ is the only point in the feasible space for which the objective function becomes maximized.

The local optima can be divided in strong and weak local optima as follows:
The solution $\underline{\widehat{x}}_{\mathcal{N}} \in \mathcal{F}$, is a strong local maximum of the objective function f if

$$f(\underline{\widehat{x}}_{\mathcal{N}}) > f(\underline{x}), \forall \underline{x} \in \mathcal{N} \tag{3.2}$$

where $\mathcal{N} \subseteq \mathcal{F}$ is a set of feasible points in the neighborhood of $\underline{\widehat{x}}_{\mathcal{N}}$. Thus, there is no point close to $\underline{\widehat{x}}_{\mathcal{N}}$ for which the objective function results in bigger values. Outside $\mathcal{N}$, however, there may well exist such points.

The solution $\underline{\widehat{x}}_{\mathcal{N}} \in \mathcal{F}$, is a weak local maximum of the objective function f if

$$f(\underline{\widehat{x}}_{\mathcal{N}}) \geq f(\underline{x}), \forall \underline{x} \in \mathcal{N} \tag{3.3}$$

where $\mathcal{N} \subseteq \mathcal{F}$ is a set of feasible points in the neighborhood of $\underline{\widehat{x}}_{\mathcal{N}}$. Thus, there is no point close to $\underline{\widehat{x}}_{\mathcal{N}}$ for which the objective function results in bigger values but there may

be any number of points resulting in the same value. This happens for example for regions in the parameter space, where no measurements were taken and the resulting objective function is just flat.

Note that all optima can well be located at the borders of the feasible space. Imagine an objective function with a slope, e.g. $f(\underline{x}) = 0.5x$, then it is obvious that selecting $\underline{x}$ as large as possible yields a maximum result. Thus, $\underline{x}^*$ will lie at the border of the feasible space $\mathcal{F}$.

Although the definition of the objective function is very specific to the posed optimization problem, its interpretation may be very different depending on the given context. For measuring the quality of a process, it may represent an error term, with the aim to find the lowest possible error in the search space. The objective function is then often called *error function*. For detection and tracking purposes in computer vision, the definition of the objective function is usually based on an observation process. An example for such a process is an appearance-based detector evaluated in the 2D image plane providing a similarity measure for each pixel. The value of the observation function can then be calculated as a function of the two parameters for the position in the image. As an other example a model representation can be used to describe the appearance of an object in the image. The variables are then the free parameters of the model, e.g. the position in the world and additional deformations. Depending on the nomber of free parameters, the underlying parameter space can easily become highdimensional. An observation process is used to match the model with the image and provides a rating of how good the model under the current parameterization fits the observation. The response from the observation process can be utilized for optimization as is or can be re-formulated as a *probability density function* (PDF) of the parameter space. For both interpretations, the observation function is equally important, as it resembles the interface between the real-world observations and the optimization process.

In the following, we will discuss different deterministic and probabilistic techniques for solving optimization problems. As already mentioned, this is a broad topic that is difficult to cover in general. The focus will therefore be on those methods that have been used for various purposes within this thesis.

3.2 Deterministic Optimization Algorithms

The common aim for any optimization process is to find an optimum of the objective function, or put in other words: to find the parameter combination, a point in the search space that solves a given problem best. This process will also be referred as *mode search*, as the aim is to find the most dominant modes of the observation function.

As already discussed above, there can either be only one optimal solution or multiple local optima in the – often highdimensional – parameter space. Achieving an effective search in many dimensions is a challenge in itself, but the true difficulty becomes obvious by realizing that the examination of the observation function is the most costly part of the whole process. An exhausting search is therefore not feasible, especially

for problems defined for a large number of dimensions which are quite common in this thesis. For the topics covered here, definition of the observation function is usually related to some kind of measurement in the image. This can be a filter or a detector, as an example, but our main concern is that it takes time to calculate the response for a given parameter vector. Therefore, each run of the observation process is expensive, which makes it important to make good decisions where to look. For the algorithms presented in the following, let us keep in mind that we aim at useing as few as evaluations of the observation function as possible to reach the maximum but still need to be general enough to avoid local suboptimal maxima in the search space.

3.2.1 The Simplex Algorithm

The simplex algorithm as described by Danzig [34] uses fairly simple linear algebra to find a local maximum. It does so by examining a sequence of points in the feasible space. The algorithm is based on the fact that a solution that maximizes the objective function will always occur at an extremum or at the borders of the feasible space. The runtime of the algorithm depends on the properties of the observation function. It works best for linear problems and solves general problems quickly in practice, although it can require exponential time for some peculiar shaped functions [30].

The most commonly used implementation of a simplex algorithm was presented 1965 by Nelder and Mead [114], also called the Nelder-Mead-Algorithm.

Like many other multidimensional optimization algorithms for general purpose [30], it tends to get stuck in local maxima. One way to avoid this is to select a starting point that is already close to the optimal solution. In practice, the observation function is often not sufficiently predictable and the algorithm is extended to circumvent these problems. A common extension is to start the optimization process multiple times with random starting points, in the hope that the real optimum is reached at least once. But running the whole optimization many times is very time consuming and thus rarely used. An other option is to extend the algorithm with an annealing technique to escape local maxima. The difficulty that arises for any unsupervised annealing process is the choice of the annealing parameters, as they have to reflect the properties of the observation function, e.g. the size and the number of local maxima, which are often unknown in advance.

Despite all shortcomings of deterministic optimization methods, they play an important role in system design. They follow strict procedures and produce coherent results. If the optimization task can be restricted to a local search and computation time is not a key issue, they can play to their strength.

3.2.2 The Mean Shift Algorithm

The mean shift algorithm is an effective technique for localizing modes of the objective function. Proposed 1975 by Fukunaga and Hostetler [52], it has been used in many applications since then. Comaniciu and Meer reviewed the properties of the algorithm,

especially the convergence behavior [28]. The Mean Shift algorithm is commonly applied for clustering low dimensional data, e.g. in the image domain [29]. But as we will see, being a nonparametric estimator of the gradient, the mean shift is also well capable of dealing with data of high dimensionality. The algorithm is briefly presented in the following.

Several gradient-based optimization approaches calculate the gradient from a difference between neighboring points sampled from the objective function. This is not only a slow procedure as it multiplies the needed number of costly evaluations of the observation function, it is also prone to noise, especially for observation functions defined on image measurements which always comprise some amount of camera noise.

The mean shift algorithm, however, presents an alternative to directly measuring the gradient. It is based on estimating a shift vector $\underline{m}(\underline{x})$ that is easier to compute than explicitly calculating the gradient $\nabla f_K(\underline{x})$ of the observation function. The mean shift vector points in the same direction as the gradient and adapts the magnitude based on the observed function. To calculate the mean shift vector for a given point $\underline{x} \in \mathbb{R}^D$, a number of support points $\underline{x}^n$, $n = 1, \ldots, N$ sampling the observation function are needed, but unlike other deterministic algorithms, the mean shift technique is not very picky about where these points are located. Even random placement is feasible and yet beneficial in some situations. The points $\underline{x}^n$ used to represent the objective function are named *support points* or *sample points*. In general we can say that a higher number of sample points will yield a better approximation of the objective function, but will also be computationally more intensive during the estimation process. This set of points, sampled from the objective function, can also be seen as the outcomes of a random variable. The objective function f can therefore be interpreted as the probability density function (PDF) $f_K(\underline{x}^n)$ of the random process.

To determine the mean shift vector, the procedure performs a density estimation[1] of the PDF in a non-parametric way. For a given point $\underline{x} \in \mathbb{R}^D$, the multivariate *kernel density estimator* (KDE) is calculated by determining the distance of $\underline{x}$ to each of the sample points $\underline{x}^n$, $n = 1, \ldots, N$, rating the distance with a kernel $K(\underline{x})$

$$\widehat{f_K}(\underline{x}) = \frac{1}{N} \sum_{n=1}^{N} K_{\underline{\boldsymbol{H}}}(\underline{x} - \underline{x}^n) \tag{3.4}$$

with the bandwidth matrix $\underline{\boldsymbol{H}} \in \mathbb{R}^{d \times d}$. As a fully parameterized matrix $\underline{\boldsymbol{H}}$ would be difficult to calculate, the simplification $\underline{\boldsymbol{H}} = h^2 \underline{\boldsymbol{I}}$ allows to use the single bandwidth parameter $h > 0$:

$$\widehat{f_K}(\underline{x}) = \frac{1}{Nh^d} \sum_{n=1}^{N} K\left(\frac{\underline{x} - \underline{x}^n}{h}\right) \tag{3.5}$$

The gradient of the PDF is then given by

$$\nabla \widehat{f_K}(\underline{x}) = \frac{1}{Nh^d} \sum_{n=1}^{N} \nabla K\left(\frac{\underline{x} - \underline{x}^n}{h}\right) \tag{3.6}$$

[1] also known as the Parzen window technique [41] named after Emanuel Parzen

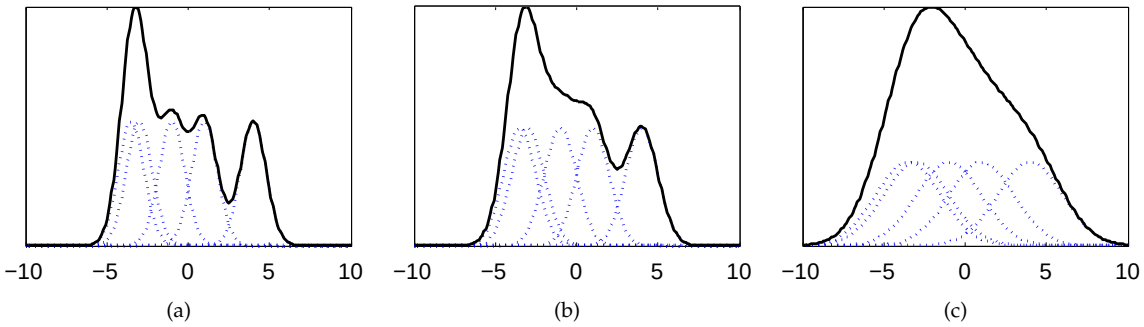


Figure 3.2: *Effect of different kernel bandwidths on the probability density estimation.* For the same positions of the kernel means, the shape of the resulting PDF varies greatly. This example shows gaussian kernels with bandwidths (a) $h_G(\cdot) = 0.5$, (b) $h_G(\cdot) = 1$ and (c) $h_G(\cdot) = 2$.

Either type of multivariate kernel can be used, but radially symmetric kernels are often more suitable. For the application presented in this thesis we use the radially symmetric Epanechnikov kernel:

$$K_E(\underline{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-\|\underline{x}\|^2) & 0 \leq \|\underline{x}\| \leq 1 \\ 0 & otherwise \end{cases} \tag{3.7}$$

where c_d is the volume of the d-dimensional unit sphere. The Epanechnikov kernel has the profile

$$k_E(x) = \begin{cases} 1-x & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases} \tag{3.8}$$

The Gaussian kernel, which is also commonly used, has the form

$$K_G(\underline{x}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\|\underline{x}\|^2\right) \tag{3.9}$$

with its profile

$$k_G(x) = \exp\left(-\frac{1}{2}x\right) \qquad x \geq 0 \tag{3.10}$$

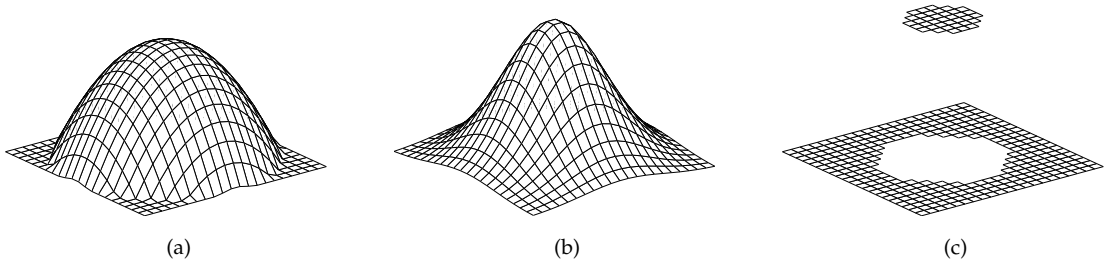


Figure 3.3: *Three kernel functions.* (a) Epanechnikov kernel, (b) Gaussian kernel (c) Unity kernel.

A function k is called profile of a kernel, if it fulfills the following conditions [25, 166]:

(i) k is strictly monotonic decreasing: $a \leq b \Rightarrow k(a) \geq k(b)$.

(ii) k is piecewise continuous.

(iii) $\int_0^\infty k(x)dx < \infty$.

(iv) $\int_0^\infty k(x)dx > 0$.

In the literature, further requirements are often made on the kernel profile, but they are not needed for the scope of this work. As an example, property (iv) is often tightened to $\int_0^\infty k(x)dx = 1$, compare also to [166]. Resigning this property allows us representing the profiles in a simpler way.

Following [29], the profile notation can be used to rewrite Eqn. (3.4) as

$$\widehat{f}_{h,K}(\underline{x}) = \frac{c_{k,d}}{Nh^d} \sum_{n=1}^{N} k\left(\left\| \frac{\underline{x} - \underline{x}^n}{h} \right\|^2\right) \tag{3.11}$$

Given the set of sample points, its mean is determined by

$$m(\underline{x}) = \frac{\sum_{n=1}^{N} \underline{x}^n g\left(\left\| \frac{\underline{x}-\underline{x}^n}{g} \right\|^2\right)}{\sum_{n=1}^{N} g\left(\left\| \frac{\underline{x}-\underline{x}^n}{g} \right\|^2\right)} \tag{3.12}$$

where $g(r) = -\nabla k(r)$ is a profile of kernel G and G is in turn the shadow kernel of K.

According to [25], the kernel G_h is called the shadow kernel of K_h if the associated kernel profiles g and k fulfill the following property:

$$g(r) = l(r) + c \int_r^\infty g(t)dt \tag{3.13}$$

where $c > 0$ is constant and l is a piecewise constant function.

The following statement is valid in particular for kernel profiles with a limited support which are mainly used throughout this work. Let g be a kernel profile of the form

$$g(r) = \begin{cases} l(r) & \text{if} \quad 0 \le r \le 1 \\ 0 & x > 1 \end{cases} \tag{3.14}$$

where $l : \mathbb{R} \to \mathbb{R}$ is a differentiable function. Furthermore let $k : [0, \infty[\to [0, \infty[$ be definined by

$$k(r) = \begin{cases} -\nabla l(r) & \text{if} \quad 0 \le r \le 1 \\ 0 & x > 1 \end{cases} \tag{3.15}$$

If k is a kernel profile, then the kernel G_h with its profile g is the shadow-kernel of the kernel K_h with its profile k.

From these assumptions follows:

$$\int_r^\infty k(t)dt = \int_r^1 k(t)dt = [-g(t)]_r^1 = -g(1) + g(x). \tag{3.16}$$

Which can be rewritten as:

$$g(r) = g(1) + \int_r^\infty k(t)dt \tag{3.17}$$

Togehter with the definition of Eqn. (3.4) this leads to the statement of Eqn. (3.12).

Note that the shadow kernel of the Epanechnikov kernel is the unity kernel, the shadow kernel of a Gaussian kernel is as well a Gaussian kernel and the shadow kernel of a Gaussian kernel with limited support is likewise a Gaussian kernel with limited support. A kernel with a limited support eases calculations based on the PDF estimation, for instance when calculating distances. The Epanechnikov kernel additionally comes with nice properties (cf. [138]) like a simple shadow kernel profile and an easy to calculate gradient. For these reasons, we choose it as kernel for representing the PDF.

As we are applying an optimization technique, we are interested in efficiently finding the maxima of the PDF. The mean shift technique makes use of the fact that the maxima of a function $f(x)$ are at the positions with zero gradient $\nabla f(x) = 0$. It can be shown that the mean shift vector $m(\underline{x}) - \underline{x}$ always points in the direction of steepest ascent of the density function [23]. By repeatedly shifting the point $\underline{x}$ in direction of the mean shift vector, the algorithm finally converges to local maxima of the PDF.

The choice of the kernel bandwidth h from Eqn. (3.5) is of crucial importance in kernel-based density estimation and is usually scaled down at each mean shift iteration in order to concentrate the sample points in the dominant modes. In our implementation, the initial bandwidth h_0 is decreased for each iteration i according to $h = i^\lambda$, with $\lambda =]0, 1[$. A typical value for the annealing parameter is $\lambda = 0.8$. As described by [23] this alters the shape of the kernel for each iteration. At the beginning of the optimization process a wide support is desired to also gather sparsely distributed particles in highdimensional spaces, but this comes at the cost of an inexact representation of the PDF, as such a kernel will flatten out sharp peaks of the function. In consecutive iterations, however, a decreasing bandwith parameter yields a more and more exact representation of the PDF. A smaller support can be tolerated, as from the preceding iteration steps the sample points have already been herded towards the local maxima.

The mean shift procedure is repeated until either a fixed number of iterations is reached or the movement of the sample points falls below a threshold for consecutive iterations. Figure 3.4 shows the trajectories of different points towards local maxima of a PDF.

Carreira [20] shows that mean shift is an expectation maximization (EM) algorithm for Gaussian kernels and a generalized EM algorithm for non-Gaussian kernels. The mean shift technique converges from almost any starting point and, in general, its convergence is of linear order. Although approaching the mode usually takes multiple iterations, the particle moves along the local principal component of the data points within the kernel, yet providing a usable approximation after few steps.

3.3 Probabilistic Optimization Algorithms

The name *probabilistic methods* stands for a category of algorithms that employ a certain degree of randomness as a part of their logic. Such algorithms typically use values from

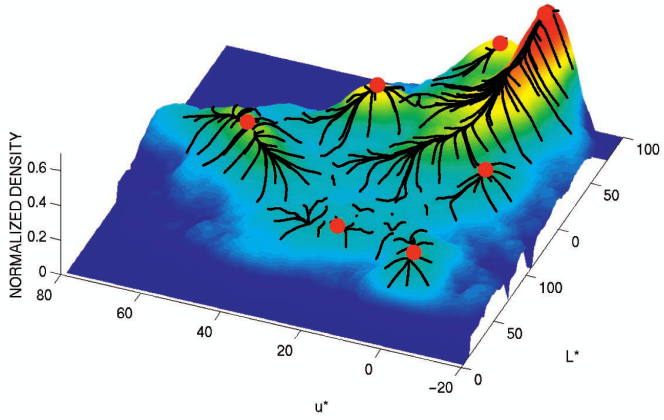


Figure 3.4: *Mean shift optimization in 2D.* An observation function has beed defined in the image domain. A local optimization using mean shift starting arbitrary positions moves the sample points (black lines) towards the modes of the function (red dots). (Image found in [29])

a pseudo-random number generator to alter their behavior with the aim of achieving good performance in the average case. Probabilistic algorithms are particularly useful when faced with multimodal objective functions. The randomness can help to cover the parameter space as a whole without the need to spend too high a number of observations. Employed correctly, this can be a big advantage compared to the discussed deterministic algorithms which are unfavorable for parameter space exploration.

The two types of algorithms that will be discussed in this section are Particle Filters (PF) and Genetic Algorithms (GA). In computer vision, particle filters are often used as the search component within the inference scheme of tracking approaches. Genetic algorithms have proven to be efficient for finding a parameterization to a given problem, e.g. training parameters for cue fusion. Both employ a random sampling scheme and perform well for multimodal search problems. A drawback of the standard particle filter approach is that the accuracy scales poorly with the amount of computational resources spent and that it does not support well tracking of multiple modes of the objective function.To tackle these difficulties, the Kernel Particle Filter (KPF) has been applied during this thesis and will be presented in the following. It combines the advantages of fast parameter space exploration and efficient mode convergence.

3.3.1 Particle Filtering

In general, Particle Filters are categorized as Sequential Monte Carlo Methods. They are based on the idea of approximating an unknown probability density function (PDF) with a set of randomly sampled weighted particles. Particle Filters are often applied to solve dynamic computer vision tasks, like tracking moving objects. In such a context, the PDF models the state of the object, like its position and size. The sequential update process is then used to update the state based on the consecutive images while incorporating the information from former timesteps.

One of the first applications of Particle Filtering to a computer vision task was presented 1998 by Isard and Blake [80]. They propose a Particle Filtering scheme, called CONDENSATION, to track the contour of a walking person. The underlying model has 6 degrees of freedom, but the observed movements are restricted to affine transformations and

a previously learned motion model was used to guide the search process. Their most important contribution was to show that Particle Filters offers advantages over conventional techniques like Kalman Filtering in the context of visual object tracking. Here, the PDF is often non-Gaussian and multimodal, which is hard to estimate with a Kalman Filter. In contrast, Particle Filters do not pose any restrictions on the approximated PDF. This also becomes obvious in the work of Deutscher et al. [38], who use particle filters for tracking human motions, a task which rises the same challenges. In the following, the particle filtering algorithm will be discussed in more detail.

Task Representation

From the perspective of particle filtering, the state of the system is described with a state vector $\underline{x}$ which is equal to the parameter vector in 3.1.1. The state vector contains all variables of the system and can often be interpreted within the given context, e.g., as a pose of a model. The state at a timestep t is expressed as $\underline{x}_t$ and it is expected to change over time. The true state – for instance defined by a ground truth measurement – is denoted as $\underline{x}_t^*$, the estimation of the current system state derived from the particle filter optimization is then called $\widehat{\underline{x}}_t$.

Furthermore, let $\underline{y}_t$ be a measurement – also called observation – that the observation function is based on, and let $\mathcal{Y}_t = \{\underline{y}_1, \ldots, \underline{y}_t\}$ be the history of all measurements. In computer vision, this is typically a sequence of images. For the current timestep, the knowledge about the system state $\underline{x}_t$ under the given history of measurements can then be expressed as the PDF $p(\underline{x}_t|\mathcal{Y}_t)$.

The temporal characteristics of the system can be expressed by the so-called system model

$$p(\underline{x}_t|\underline{x}_{t-1}) \qquad t \geq 1 \tag{3.18}$$

which expresses the probability of the state $\underline{x}_t$ given the last state $\underline{x}_{t-1}$. We assume this to be a first order Markov process, which depends solely on the knowledge of the previous state. When applied to computer vision problems, the system model is often expressed as a motion model which describes the way an object changes its pose over time. Depending on the application it can be favorable to use a history of events for reference rather than a single timestep, which conflicts with the requirement of a first order markov process. For this thesis, however, we simplify the problem formulation to a single timestep only. If at some point more than one timestep is referenced, we furthermore assume that this history can be encoded as a single system state which allows us to handle the system model as a first order process again.

Next, an observation model can be defined as

$$p(\underline{y}_t|\underline{x}_t) \qquad t \geq 1 \tag{3.19}$$

It describes the probability to observe a measurement $\underline{y}_t$ assuming a given state of the system, e.g., how good a recorded image complies with a given model pose. The PDF specified by this observation model is named observation density in the following.

Finally, the task of the particle filter should be to provide an estimate of the best state under the history of measurements, thus estimating $p(\underline{x}_t|\mathcal{Y}_t)$. The particle filter achieves this by implementing a recursive Bayesian filter. It is called recursive, as the current state can be calculated from the last estimation $p(\underline{x}_{t-1}|\mathcal{Y}_{t-1})$, which again depends on the estimation before and so on. Following [80], the current state can be predicted using the system model and given prior observations as

$$p(\underline{x}_t|\mathcal{Y}_{t-1}) = \int p(\underline{x}_t|\underline{x}_{t-1})p(\underline{x}_{t-1}|\mathcal{Y}_{t-1})d\underline{x}_{t-1}. \tag{3.20}$$

After that, the system model 3.18 is included into the prediction 3.20, which yields the posterior PDF $p(\underline{x}_t|\mathcal{Y}_t)$ at time step t. Under the assumption that the new measurements only depend on the current time step, Bayes' rule can be applied which results in

$$p(\underline{x}_t|\mathcal{Y}_t) = \frac{p(\underline{y}_t|\underline{x}_t)p(\underline{x}_t|\mathcal{Y}_{t-1})}{\int p(\underline{y}_t|\underline{x}_t)p(\underline{x}_t|\mathcal{Y}_{t-1})d\underline{x}_t} \tag{3.21}$$

so that the current state can be fully explained given the system model 3.18 and the observation model 3.19. A more detailed description of recursive Bayesian filtering can be found in [4]. Therein, Arulampalam et al. show that under some restrictions, e.g., gaussian PDF's and a linear system model, the exact solution can be found while in the general case, particle filtering still provides a good approximation which is relatively fast to compute.

In particle filtering, the PDF is represented as a set of discrete samples $\underline{s}_t^n, n = 1, \ldots, N$. These samples can be interpreted as points in the state space. Each sample is associated with a weight w_t^n. In the nomenclature of section 3.1.1, a sample $\underline{s}_t^n$ is a candidate parameterization $\underline{x}$ of the objective function and the weight w_t^n is the value y of the objective function $y = f(\underline{x})$ evaluated at position $\underline{x}$. The tuple of a sample and its accompanying weight $(\underline{s}_t^n, w_t^n)$ is called a particle, hence the name particle filter. All particles together form the particle set $\mathcal{S}_t = \{(\underline{s}_t^1, w_t^1), \ldots, (\underline{s}_t^n, w_t^n)\}, n = 1, \ldots, N$.

As we already learned from the Kernel Density Estimator 3.4, the posterior PDF can be approximated by a set of support points. Here, we use a discrete weighted particle set to approximate the PDF as

$$\widehat{p}(\underline{x}_t|\mathcal{Y}_t) = \sum_{n=1}^{N} w_t^n \delta(\underline{x}_t - \underline{s}_t^n) \tag{3.22}$$

with the Dirac-function providing the transition from the continuous to the discrete space. Similar to the KDE in the mean shift, the approximation becomes more accurate, the more particles are used. But, as we will see, a large number of particles also means slow computation. For applications in computer vision, especially for real time processing, a compromise between exactness of the approximation and speed must be found.

Particle Filtering Steps

A well known implementation of the particle filter has been presented 1998 by Isard and Blake [80] with the Condensation algorithm. We will use this as an example imple-

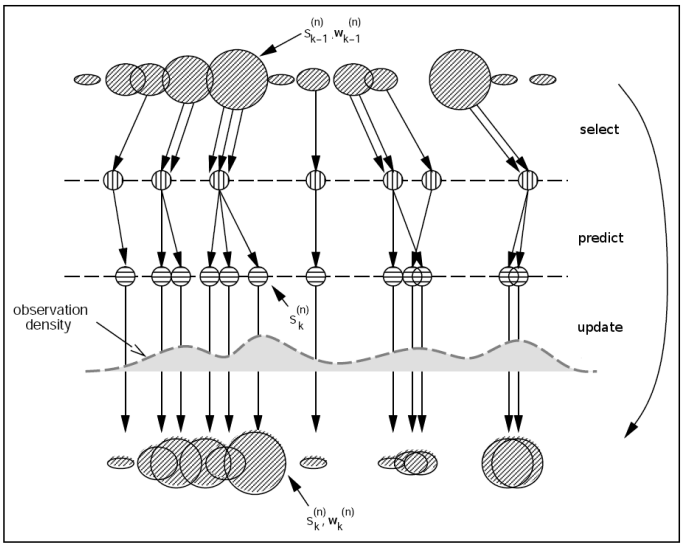


Figure 3.5: *One time step of the* CONDENSATION *algorithm.* From the particle set at the last timestep, a number of exemplars are selected and predicted to new positions. Updating the weights yields a new particle set fro the current timestep. (Image adapted from [80])

mentation to explain the technique of particle filtering. The filter performs a sequence of steps for each new observation, called *select*, *predict* and *update*. The algorithm is outlined in Alg. (3.6), its individual steps are detailed in the the following.

Require: Particle set $\mathcal{S}_{t-1} = \{\underline{s}_{t-1}^n, w_{t-1}^n\}_{n=1}^N$, new image measurement $\underline{y}_t$
For initialization, let $t = 1$ and initialize $\mathcal{S}_{t-1}$ from prior $p(\underline{x}_0)$

for all $n = 1 : N$ **do**
Select $\underline{s}_{t-1}^k$ randomly out of $\mathcal{S}_{t-1}$ with probability w_{t-1}^k and $1 \leq k \leq N$
Predict $\underline{s}_t^n \sim p(\underline{x}_t | \underline{s}_{t-1}^k)$
Update $w_t^n = \widehat{p}(\underline{y}_t | \underline{s}_t^n)$
end for
Normalize weights w_t such that $\sum_n w_t^n = 1$

Ensure: $\underline{S}_t = \{\underline{s}_t^n, w_t^n\}_{n=1}^N$

Figure 3.6: *The* CONDENSATION *algorithm.* An example for the particle filtering technique.

Selection

The first step at each iteration is to generate a new particle distribution to approximate the PDF. As only a limited number of particles is available, it is the task of the selection step to choose a set of particle locations in the parameter space which allows an efficient exploration in order to locate the maximum for the current timestep. The new set of particles is therefore based on that of the previous timestep $\mathcal{S}_{t-1}$ and prefers regions with high weights. In literature, this step is also called *resampling*, as it samples a new set of particles out of the prior set. The positions of the new particles are determined by a technique known as monte carlo random sampling or roulette wheel selection[2] [40], which works as follows.

[2]The name "Monte Carlo" was popularized by physics researchers Stanislaw Ulam, Enrico Fermi, John von Neumann, and Nicholas Metropolis, among others; the name is a reference to a famous casino in

It is assumed that the weights of the particles in $\mathcal{S}_{t-1}$ are normalized, thus they fulfill $\sum_{n=1}^{N} w_{t-1}^{n} = 1$. Monte Carlo sampling uses a cumulative sum function $W(k) = \sum_{n=1}^{k} w_{t-1}^{n}$ that for a given particle $k = 1, \ldots, N$ sums up the weight of all particles up to particle k. A single particle can now be chosen by giving a random number $\rho =]0; 1]$ and searching for the first particle k that makes the cumulative sum bigger or equal to ρ, as depicted in Fig. (3.7).

$$\underset{k \in \{1,\ldots,N\}}{\arg\max}\, W(k) \quad \in \quad \{k \mid \forall l < k : W(l) < \rho\} \tag{3.23}$$

Using this technique, particles with a high weight are more likely to be chosen as they account for a bigger fraction of the sum. The result is a new set of particles at the same positions as they have been in the last timestep, but particles with a low weight are left out and particles with a high weight are multiplied.

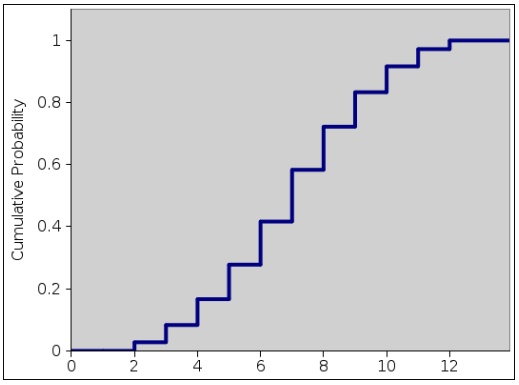


Figure 3.7: *Cumulative sum used for Monte Carlo sampling.* A random number between 0 and 1 is used to select the particle with the corresponding value of the sum. Particles with a high likeihood span a wider region of the cumulative sum and are more likely to be selected.

For the initial iteration $t = 1$, where no information about former system states and observations is available, the particles are sampled at random positions in the parameter space with equal weights $w_1^n = 1/N \quad \forall n = 1, \ldots, N$ for monte carlo sampling.

In the literature, this technique is also called a *darting* method [2], because it can be compared to throwing a dart sideways into the sum function and then look up which particle it hit. It can be generalized [149] to better explore the local mode structure if advance knowledge about the location of these modes can be given.

Prediction

The newly generated particle set is now object to stochastic diffusion, applying the system model as in Eqn. (3.18). The particles are shifted according to the known dynamics by adding the system model to each particle position. If no further information is available, the system model defaults to zero-mean Gaussian noise.

When observing real world objects, e.g. a robot or a human, it is often suitable to choose a model incorporating basic physical rules like inertia. In such a case, the system model is often called *motion model* as the free parameters of the optimization process are the

Monaco where Ulam's uncle would borrow money to gamble [74]. The use of randomness and the repetitive nature of the process are analogous to the activities conducted at a casino.

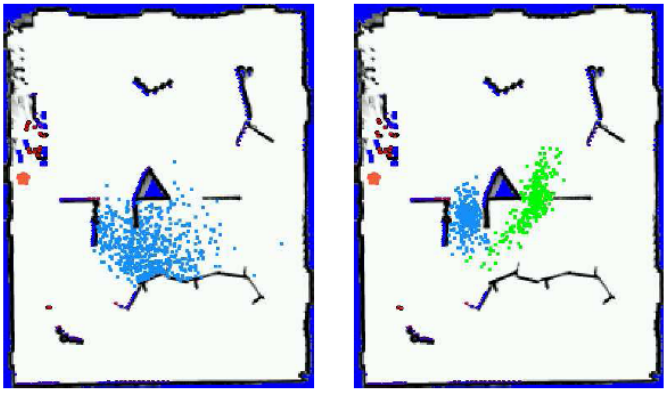


Figure 3.8: *Particle set prediction with a motion model.* Modelling a person walking in a room with obstacles. Left image: Brownian motion model (blue points) incorporating scene constraints. Right image: Additional task knowledge (green points) guides the prediction. (Image found in [17])

position and orientation of the object in the space. Typical implementations are linear prediction, higher-order predictors or a Kalman filter [86] which uses a learned internal model of the observed motion to predict the position for the next timestep.

Knowing about which motions are likely to occur, a detailed model predicting these motions can be derived. An frequently applied technique consists in the use of a motion prior, for instance in the form of a learned motion database.As we are dealing with a distribution of particles, the prediction model can easily be realized as a mixture of different sub-models, which will be applied each for a subset of the distribution. As an example, Bruce et al. show how to incorporate scene and task knowledge [17], which makes the prediction more specific than a general model only. This helps to gather particles in regions of the parameter space that are more likely to match with the new object position, cf. Fig. (3.8).

Note that for all models the diffusion in the parameter space must be wide enough to cover all possible motions – or state changes – of the observed object that can occur within one timestep. For a moving person, for example, it is plausible to define a maximum velocity. If the parameter vector represents variables other than the spatial position, an assumption about the dynamics of the variable is needed. Distributing too widely, however, would waste particles and apart from that it could distract the particle filter to optimize into a wrong local maximum.

Update

The next step in the process is to perform an update of the weights for each sample from the newly generated set. For that, the objective function is evaluated using the new measurement $\underline{y}_t$.

$$w_t^n = \widehat{p}(\underline{y}_t|\underline{s}_t^n) \propto w_{t-1}^n p(\underline{y}_t|\underline{s}_t^n) \tag{3.24}$$

At this point, external information that is for instance provided by an image based distance measure, is introduced into the particle filtering process. Finally, the particle weights are normalized to sum up to 1. The outcome is an approximation of the underlying PDF.

But the question what the result of the optimization process is remains. A set of weighted system states is merely useful. Instead, a single system state $\widehat{\underline{x}}_t$ is often needed

as an estimation. As long as the PDF is unimodal and undistorted, a simple weighted average of all samples estimates well the mean position.

$$\widehat{\underline{x}}_t = \sum_{n=1}^{N} w_t^n \underline{s}_t^n. \tag{3.25}$$

For more complex shaped PDF's, e.g., showing multiple maxima or containing distorted non-Gaussian modes this estimation fails. For two adjacent modes, for instance, it estimates the optimal system state right in-between those two modes, which is a worse result than just choosing the best mode and neglecting the others. Indeed, this is a common workaround, but also bears problems, especially when dealing with visual observations. Herein, the observation process is prone to camera noise and dynamic environments, which easily results in volatile modes. It also frequently happens that the true position is not the global maximum of the objective function but a local maximum only. Reasons for this can be ambiguities in the measurements or the difficulty to find an adequate objective function.

Initialization and Error Recovery

Particle Filters are often used to track an object over time. One of the difficulties is then to choose an appropriate diffusion model that covers all possible motions. If the true position is not within the search radius, it will never be evaluated and the tracking is lost. The same problem arises, if the tracking gets stuck in a false local maximum. This can happen if the diffusion is not wide enough to escape from the false mode. The filter can also be stuck in flat regions of the parameter space, which means that the observation function values to 0 or close to 0 for all samples. With all samples showing equal weights, the selection step samples randomly and cannot provide proper guidance for the optimization.

For all these cases, the principle of probabilistic optimization offers an elegant way to integrate an error recovery component. Mixing a number of *recovery particles* into the distribution, say $5\% - 10\%$, enables the filter to evaluate the objective function at other positions than those defined by selection and prediction. A random coverage of the whole feasible space is the default choice since this resembles random guessing. Such a technique becomes the less efficient the higher the dimensionality of the feature spaces. It is obvious that with any kind of external knowledge, e.g., from other observation processes, the recovery particles can be distributed far more efficiently. Note that the initialization of the particle positions can be seen as a special case of the above. Instead of a just a portion of the distribution, the whole set of particles is replaced by recovery particles. Consequently, the particle weights are all set to an equal value of $\frac{1}{N}$, as the positions have not been based on valid observations.

3.3.2 Kernel Particle Filtering

Many extensions of the standard particle filter approach exist and most of them do for the same reasons, as Arulampalam et al. [4] argue. First, the generation of the weights

is very task specific and so is the resulting objective function in the parameter space. Depending on the shape and number of true and false maxima, simplifications can be made to make the optimization process converge faster, or the search can otherwise be strengthened to avoid as many false maxima as possible. Secondly, a common problem of particle filters is the phenomenon known as particle set degeneration which many algorithms try to compensate. Particle set degeneration describes the fact that after multiple iterations the set of particles might contain a large number of samples which no longer contribute to the approximation of the posterior PDF. This can happen due to particles with almost zero weights but also when too many samples are concentrated in a small region of the parameter space – a state which is often referred to as a collapsed particle set.

Multiple solutions to these problems are known from the literature, some of them are discussed below. After that, an extension of the standard particle filtering approach, the so-called Kernel Particle Filter (KPF), is presented, which has been adapted for this thesis.

An early approach extending the standard technique is the annealed particle filtering concept of Deutscher et al. [37]. Their approach approximates the observation density as $p(\underline{y}_t|\underline{s}_t^n) = w(\underline{y}_t, \underline{s}_t^n)^\beta$ for weights $w(\cdot) \in [0,1]$ and performs up to 10 iterations of CONDENSATION per timestep. Starting with values for β close to zero and increasing it for each iteration narrows down the peaks of the estimated observation density, thus yielding a more and more exact estimation of the modes of the posterior PDF.

As proposed by Chang and Ansari [22] particle filtering can be combined with kernel density estimation. Fitting a kernel around each particle allows the interpolation of the objective function for positions in the state space between individual samples. This kernel representation allows to approximate a PDF with a sparse distribution of particles, which is a particular advantage for high dimensional feature spaces. This idea will also be used in the KPF approach presented in the next section. Furthermore, the particle representation offers a convenient alternative to deal with multi-modal PDFs. But still, the complexity of the standard PF approach dramatically increases with the dimensionality of the sampled PDF.

Both, Maggio et al. [106] and Chang et al. [23] proposed KPF as a tracking algorithm based on a combination of particle filtering and Mean Shift. The proposed tracker generates a smaller number of samples than needed for standard PF and then shifts the samples towards a close local maximum using Mean Shift. They show that the combined tracker outperforms PF with only needing 20% of the number of samples. This idea goes well with the fact that most applications aim at determining the state of a model at a given time, for instance the position and size of an object or the pose of a human. For such applications, one is not interested in approximating the objective function as a whole, instead, finding its modes is sufficient to solve the task which is also true for the applications of KPF in this thesis. In summary, KPF combines the advantages of probabilistic and deterministic optimization techniques. It shows ways to overcome the drawbacks of the standard PF approach like particle set degeneration and deficient scalability for highdimensional feature spaces. The KPF algorithm is detailed in the next section.

Kernel Particle Filtering Steps

The KPF algorithm utilizes a probabilistic particle filtering part to efficiently explore the highdimensional parameter space. Additionally, it employs a meanshift approach as a deterministic optimization procedure to herd particles at local maxima. It thereby allows for a better coverage of the highdimensional parameter space without needing a high number of particles, as they are shifted towards interesting regions of the parameter space. The KPF method thereby significantly enhances the performance of the search process compared to the standard PF approach.

Require: Particle set $\mathcal{S}_{t-1} = \{\underline{s}_{t-1}^n, w_{t-1}^n\}_{n=1}^N$ from last timestep $t-1$,
new image measurement $\underline{y}_t$, bandwith parameter h_0

Update particle set $\mathcal{S}_t$ with a single CONDENSATION step
for all $i = 1 : I$ **do**
  Shift all particles according to mean shift vector $m(s_t^{(n)}, h_0^i)$, $n = 1, \ldots, N$
  Update weights w_t^n for all particles from image $\underline{y}_t$
  Normalize particles $\sum_{n=1}^N \{\underline{s}_t^n, w_t^n\} = 1$
end for
Normalize weights w_t such that $\sum_n w_t^n = 1$

Output: Particle set $\mathcal{S}_t$
next timestep: t=t+1

Figure 3.9: *The* KERNEL PARTICLE FILTER *(KPF) algorithm.* The KPF combines the particle filtering technique with a mean shift optimization approach. This code presents a single timestep of the KPF algorithm.

As for particle filtering, the underlying PDF is approximated by a set of particles. Here, the true density distribution is estimated through placing a kernel function K on each sample:

$$\widehat{p}(\underline{x}_t|\mathcal{Y}_t) = \sum_{n=1}^{N} K_h(\underline{x}_t - \underline{s}_t^{(n)}) w_t^{(n)} \tag{3.26}$$

Extending the PDF approximation in Eqn. (3.22) with weighted kernels, it can now be expressed as

$$K_h(\underline{x}_t - \underline{s}_t^{(n)}) = \frac{1}{Nh^d} \frac{K(\underline{x}_t - \underline{s}_t^{(n)})}{h} \tag{3.27}$$

with the associated weights $w_t^{(n)}$ and the kernel bandwidth h.

For a radially symmetric kernel like the Epanechnikov kernel, we have $K(\underline{x}_t - \underline{s}_t^{(n)}) = ck(\|\underline{x}_t - \underline{s}_t\|)$, where c is a normalization constant which makes the integral $K(\underline{x}_t - \underline{s}_t^{(n)})$ equal one, and $k(r) = k(\|\underline{x}_t - \underline{s}_t\|)$ is the profile of the kernel K.

Next, mean shift is introduced as a local mode finding approach to the kernel particle filter. Given a particle set $\mathcal{S}_t$ and the associated weights $\{w_t^{(n)}\}_{n=1}^N$, the particle mean from Eqn. (3.12) can now be expressed using the kernel representation for the PDF (Eqn. (3.22)) which leads to:

$$m(\underline{s}_t^{(n)}) = \frac{\sum_{i=1}^N H_h(\underline{s}_t^{(n)} - \underline{s}_t^{(i)}) w_t^{(i)} \underline{s}_t^{(i)}}{\sum_{i=1}^N H_h(\underline{s}_t^{(n)} - \underline{s}_t^{(i)}) w_t^{(i)}} \tag{3.28}$$

Using multiple mean shift iterations, each of the particles $\underline{s}^{(n)} = \{\underline{x}^{(n)}, w^{(n)}\}$ is shifted towards a local mode of the estimated posterior.

The choice of the kernel bandwidth H_h is of crucial importance in kernel-based density estimation and is usually scaled down at each mean shift iteration in order to concentrate on the most dominant modes. In our implementation, the initial bandwidth H_0 is scaled at every iteration i according to $H_h = 0.8^i H_0$ where the value 0.8 has been determined empirically, similar to [23].

Following the shifting of particles using the mean shift vector, the particle weights $w_t^{(n)}$ are recomputed. As the shifting of the particles implies that the new particles are not distributed according to the posterior distribution, a reweighting is performed in order to guarantee that each mean shift iteration follows the correct posterior gradient. Using subscript j to denote the particle set after the jth mean shift iteration at time t, the weight is recomputed based on the posterior density evaluated at the new particle positions $\underline{s}_{t,j}^{(i)}$ and a particle density balancing factor [23]:

$$w_{t,j}^{(n)} = \frac{p(\underline{s}_{t,j}^{(n)} \mid \mathcal{Y}_t)}{q_{t,j}(\underline{s}_{t,j}^{(n)})}. \tag{3.29}$$

The balancing factor in the form of the denominator is the new proposal density:

$$q_{t,j}(\underline{x}_t) = \sum_{l=1}^{N} K_h(\underline{x}_t - \underline{s}_{t,j}^{(l)}). \tag{3.30}$$

Without the balancing factor, several particles concentrating after a mean shift iteration at one density mode would result in a high kernel density estimation at this position and would, therefore, heavily influence the particle mean of Eqn. (3.28). This effect is avoided by means of reweighting Eqn. (3.29). So, information on how many particles are located in the kernel window around the new particle position is incorporated in the posterior density evaluation.

The overall posterior density uses a sample-based approximation of the prior density and is given by

$$p(\underline{x}_t \mid \mathcal{Y}_t) \propto p(\underline{y}_t \mid \underline{x}_t) \sum_{l=1}^{N} p(\underline{x}_t \mid \underline{s}_{t-1}^{(l)}) w_{t-1}^{(l)}. \tag{3.31}$$

Equation 3.31 is the Kernel particle filtering equivalent of recursive Bayesian filtering.

In contrast to Deutscher et al., our approach has the advantage that we can control the sensitivity of each cue independently. Another major difference between our KPF and their particle filter is the fact that we need no intermediate condensation iterations on the particle sets in order to migrate particles to local modes in the posterior. Our mean shift approach allows to do this much more efficiently. Typically, 2 or 3 iterations of mean shift are sufficient to locate the posterior modes while, in [37], 10 annealing iterations are proposed.

The outcome of the Mean Shift is a refined probability distribution with the particles being more or less densely gathered around local maxima of the parameter space, depending on the number of Mean Shift iterations and the configuration of the observed space.

3.3.3 Evolutionary Computation

The main idea of Evolutionary Computation is inspired by the theory of evolution according to Charles Darwin (1809–1882), who published[3] it in his work *"On the Origin of Species by Means of Natural Selection"* [35]. Therein he states that in a world with limited resources and a stable population, each individual is in competition with each other. Those individuals exhibiting the best abilities can reproduce and are more likely to survive. Beneficial abilities are bequeathed to their offsprings, thus spreading throughout generations until they dominate the whole population. This is what now name *selection*. Darwin further on states that while passing the abilities on to the offsprings, random events can alter the abilities. This process is nowadays known as *mutation*. If these mutated abilities are beneficial for the new individual, it will have a higher probability to survive.

Genetic Algorithms

To transfer this to optimization, the evolutionary theory is applied in the form of a *Genetic Algorithm* (GA) as follows. Genetic Algorithms are robust computational procedures that try to emulate some of the processes in natural evolution. Evolution is an optimization process aiming at maximizing the survival rate of a population of organisms – or in other words a biological system – while competing in a dynamic environment. Taking up that idea, the technical system to be optimized by a GA is represented by a set of individuals $S(t) = \{\underline{s}_t^{(1)}, \ldots, \underline{s}_t^{(N)}\}$, with a population size of N. Each individual $\underline{s}_t^{(n)}$ of the population stands for a solution of the optimization problem and is rated concerning its fitness $f(\underline{s}_t^{(n)})$, which corresponds to the objective function in the optimization theory. The better an individual is rated with the objective function, the higher its fitness. Each of the possible solutions is a parameter vector composed of variables. In correspondence with their biological meaning, the parameter vector representing the collection of all chromosomes is called the *genetic material* or *genome* of an individual.

The population iteratively develops in discrete timesteps, called *generations*. For each generation, the basic operations *selection*, *reproduction*, and *mutation* are applied to each of its individuals, thus creating a new generation of individuals for the next timestep aiming at a better performance of the total population. The selection operation compares the individuals with each other according to their fitness and selects highly rated

[3] Alfred Wallace (1823–1913) developed the theory of evolution independently from Charles Darwin at the same time. Although Jean-Baptiste Lamarck's (1744–1829) scientific theories were largely ignored during his lifetime, Lamarck began to publish details of his evolutionary theories beginning back in 1801.

individuals. The reproduction operation – also called *recombination* or *cross-over* due to the exchange of the genetic material – creates offsprings from selected individuals. The offspring thereby inherits a mixture of the abilities of its parents while the mutation operation randomly alters the new genetic material.

From generation to generation, the overall fitness of the individuals increases, as only those performing well in the given environment are able to produce offsprings and to pass on their genetic material. The optimization process can be seen in the creation of ever new individuals with increasing performance, their genetic materials contains the parameters to maximize the objective function. This process is called *Genetic Algorithm* (GA), it is briefly outlined in 3.10.

Require: let $t = 0$ be the number of the current generation
let $n = 1, \ldots, N$ be the number of individuals
let $\underline{s}_t^{(n)}$ be the individual n of the generation t

initialize a population $\underline{\boldsymbol{S}}(0) = \{\underline{s}_0^{(1)}, \ldots, \underline{s}_0^{(N)}\}$
repeat
  compute fitness $f(\underline{s}_t^{(n)})$ for each individual $\underline{s}_t^{(n)}$ of the generation $\underline{\boldsymbol{S}}_t$
  select surviving individuals $\underline{\boldsymbol{S}}'_t$ out of $\underline{\boldsymbol{S}}_t$
  recombine pairs of individuals out of $\underline{\boldsymbol{S}}'_t$ to create offsprings $\underline{\boldsymbol{S}}_{t+1}$
  mutate genetic material of offsprings $\underline{\boldsymbol{S}}_{t+1}$
  next generation $t = t + 1$
until break criterion is reached

Output: generate best individual $\widehat{\underline{i}}$ from the current generation $\underline{\boldsymbol{S}}_t$

Figure 3.10: *The* Genetic Algorithm *(GA).* The GA as an example for evolutionary computation.

Throughout the whole process, the different operations are not executed every time but they are subject to probabilities of random processes models as it can be observed in the nature. Mutation, for instance, occurs rarely while selection and recombination are quite common.

In Darwin's Theory of Evolution, there is no need for a break condition, as the theory is meant to describe the natural evolution in an ever-changing environment. It is however the aim of an optimization algorithm to provide an optimal solution for a given problem in a limited amount of time.

The algorithm therefore has to decide if further computation is beneficial, cf. [42]. The optimization can be stopped after a fixed number of generations. Although this helps to restrict the time needed to compute a solution, there is no guarantee that the algorithm has converged until then. Consequently, this criterion is only useful if the convergence behavior can be predicted for the given problem, or it can be used as a fallback if the other criteria fail and convergence is never reached at all. The mean fitness value of a generation can also be used as a cue if observed over time. If there is no increase – or even a decrease – in the mean fitness over some time, it can be assumed that at least a local optimum has been reached and optimization can be stopped. Prior knowledge about the objective of the optimization can also be incorporated, for instance, if an

estimation can be made about what the fitness value of an individual resembling a good solution would be. Optimization can then be stopped as soon as this value is reached.

Finally, an optimal solution – or in other words an optimal individual – has to be determined. Choosing the best individual of the current generation seems to be the obvious choice, but similar to the mode search of the KPF other choices may be more elaborate in some situations. The number of variables, and thus the number of dimensions of the parameter space – can be quite high. By choosing the best individual, one may miss the maximum. A better way is to consider a number of individuals around the best rated individual and to choose their mean position in the parameter space as the final solution.

Problem Coding

In 1975, John Holland developed a theoretical framework for Genetic Algorithms, published in his book *"Adaptation in Natural and Artificial Systems"* [72]. He proposed a building block theory, which encodes the genome in a binary representation and he proved that the optimization process converges for operations on this representation. It is also easier to define the needed operators if all variables in the genome are uniformly coded. Binary coded genomes therefore have been used for a long time with a corresponding transformation scheme for non-binary parts of the genome [110], which can lead to data loss as the binary representation usually implies some kind of discretization. Goldberg explains in [57] that the accuracy of the conversion can be increased by choosing longer binary strings for the genome which can represent numbers in a higher resolution. But the longer the genome, the longer the optimization process takes and the more difficult it gets to accurately determine the maxima. As Antonisse [3] proposed in 1989, a floating point representation for the genome $\underline{x}$ containing J variables $x_{(j)} \in [0,1] \subset \mathbb{R} \quad \forall j = 1,\ldots,J$ offers the same benefits in terms of easily definable operators but it is better applicable to specific problems. Janikow [82] proved this assumption to be true for several test cases with adapted operators. This floating point number representation of the genome will also be utilized for the GA in the course of this thesis.

When defining the transformation functions, there are thee types of variables to account for: floating point numbers, integers, and sets. For each of these variable types, an independent transformation function is needed to map its values to the floating point numbers of the desired domain. Furthermore, an inverse mapping is needed to transfer the variables back into their original domain after optimization.

For the transformation, floating point numbers and integers are assumed to be restricted by boundary constraints ($a_i \leq x_i \leq b_i$), sets are assumed to be finite with the cardinality m for a set $x_i \in \{e_1,\ldots,e_m\}$. If such a restriction does not apply to a specific variable, for instance because it may grow infinitely large, its range must either be restricted to a large number – remember that the range for the floating point representation in standard programming languages is also limited – or a distinctive mapping function has to be found that maps from a finite range to infinite values, e.g. by deliberately exploiting asymptotic behaviour.

As the easiest choice, floating point numbers can be linearly scaled to the desired $[0, 1]$ domain

$$x_i^{[0,1]} = \frac{x_i - a_i}{b_i - a_i} \tag{3.32}$$

The corresponding backprojection is achieved by

$$x_i = x_i^{[0,1]}(b_i - a_i) + a_i \tag{3.33}$$

Similarly, integers are mapped via

$$x_i^{[0,1]} = \frac{x_i - a_i}{b_i - a_i} \tag{3.34}$$

The backprojection is realized by scaling and then rounding the floating point value to the nearest integer.

$$x_i = \lfloor (x_i^{[0,1]}(b_i - a_i) + a_i) + 0.5 \rfloor \tag{3.35}$$

For the transformation of sets, the elements are first mapped to a number of integer values $e_{(i)} \rightarrow i$ corresponding to their indices.

$$\{e_{(1)}, \ldots, e_{(m)}\} \rightarrow \{1, \ldots, m\} \tag{3.36}$$

After that, the mapping can be performed as for integer values.

$$x_i^{[0,1]} = \frac{i - 1}{m - 1} \tag{3.37}$$

The backtransformation is performed accordingly.

$$i = \lfloor (x_i^{[0,1]}(m - 1) + 1) + 0.5 \rfloor \tag{3.38}$$

During optimization, the parameters will typically be assigned with real number values. The lossy backprojection is necessary for integers and sets, as there is normally no such thing as a value between the elements $e_{(n)}$ and $e_{(n+1)}$. But this can pose a problem for the optimization process. Small numerical changes in the genome do not necessarily result in selecting a new element from the set and thus do not necessarily change the behaviour of the objective function if the numerical value lies inbetween two numbers. But for parameters close to an integer, a small parameter change can result in a sudden change between elements which makes the objective function discontinuous and thus hard to optimize for these variables.

Genetic Algorithm Operators

The operators selection, mutation and recombination of genetic algorithms are the equivalents for the steps selection, prediction and refinement of the Kernel Particle Filter as they fulfil similar roles. From the current generation, the selection operator choses a number of members that are likely to best represent the system state in the next timestep. Similar to the prediction process, the mutation operator explores the parameter space by adding random variations to the genome of the existing individuals. Recombination can be compared to the refinement step of the KPF. It generates an offspring that hopefully better approximates the underlying fitness function than its parents. The operators and suitable algorithms are detailed in the following.

Selection

Inspired by natural selection, the selection operator evaluates the fitness values of the individuals and determines their chance to survive. For a best performance of the optimization procedure the selection scheme must be carefully choosen. If the selection is too much oriented at the best individuals – in nature this is called a high seletion pressure – it is more likely that suboptimal members dominate the whole population. These can be seen as local maxima in the objective function, which the population cannot escape if it is not allowed to develop weaker individuals along the way to the global maximum. If the selection pressure is choosen low the abilities of the individuals can freely develop but the direction of the evolution becomes unclear and the chances to miss a local maximum rise. As a consequence, the overall rate of convergence of the optimization process would be very low.

There exist various selection mechanisms with different properties, such as linear and power scaling, σ-truncation, windowing, etc, see [64] for a performance comparison of different selection techniques. As the selection operator is crucial to the performance of the GA, improving it can help to achieve a faster convergence rate [120]. As an intuitive technique, similar to the selection procedure of the standard PF, proportional selection defines the selection probability for an individual $\underline{s}_t^{(n)}$ as a function of its fitness value.

$$p_{ps}(\underline{s}_t^{(n)}) = \frac{f(\underline{s}_t^{(n)})}{\sum_{j=1}^{J} f(\underline{s}_t^{(j)})}. \tag{3.39}$$

The main drawback of this technique is the tendency to create degenerated distributions.

To overcome this disadvantage, linear ranking with stochastic universal sampling is utilized in this work since it handles robustly the posed tasks and prevents from premature convergence and weak search. For linear ranking, the individuals of a population are sorted according to their fitness. The probability to select a certain individual depends on its position in the sorted list rather than its absolute fitness value as for proportional selection. Comparing the two techniques, linear ranking can lower the selection pressure for generations containing a small number of members having outstandigly high fitness values. In the same manner, the selection pressure can be raised for generations mostly comprised of evenly weighted members. The rank selection technique thereby prevents the population from being dominated by few individuals or from being flattened out. The linear ranking method by Baker [5] sorts the N individuals of a population according to their fitness value $f(\underline{s}_t^{(n)})$. The member with the lowest fitness value is sorted as the first element with index 1, the one with the highest value gets the index N. The selection probability p_{lr} for the individual $\underline{s}_t^{(n)}$ at rank n is then assigned by:

$$p_{lr}(\underline{s}_t^{(n)}) = \frac{1}{N}\left(\eta_{max} - (\eta_{max} - \eta_{min})\frac{n-1}{i-1}\right) \tag{3.40}$$

with $\eta_{max} \in [1, 2]$ and $\eta_{min} = 2.0 - \eta_{max}$ being the expected fitness values of the best and worst individual, respectively. Note that the probabilities are normalized to $\sum_n p(\underline{s}_t^{(n)}) = 1$. The selective pressure can be controlled with the free parameter η_{max}. It thereby balances

the exploration and exploitation aspects of the search. Baker found a value of $\eta_{max} = 1.1$ to be feasible for several test cases [110] and it is also used as a default in this work.

The two most common techniques for choosing individuals from a population based on their selection probability $p(\underline{s}_t^{(n)})$ are the roulette wheel algorithm and stochastic universal sampling.

Roulette wheel – also known as stochastic sampling with replacement [6] – is similar to the monte carlo sampling method of the PF. The members of a generation are ordered on a line corresponding to the normalized cumulative sum of their selection probability. A random number chooses the individual at a certain position on the line. A drawback of this approach is that the number of offsprings for a certain individual can not be guaranteed. In theory it is possible that a single individual produces no offsprings at all even with a high selection probability. Although this is unlikely for generations composed of many members as the random selection process tends to average out such anomalies the effect can have negative influence on generations with a small number of individuals as it is common in this work.

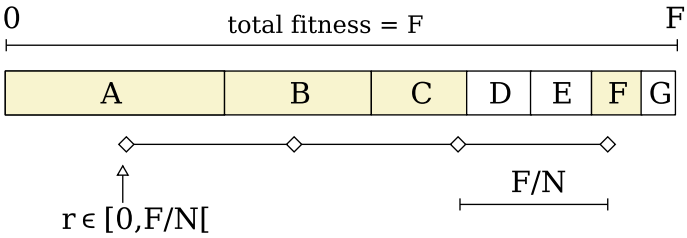


Figure 3.11: *Stochastic universal sampling.* Beginning with a random offset r, samples are drawn in equidistant intervals.

Stochastic universal sampling [6] circumvents this problem as it does not rely on multiple iterations of the random process. Instead, it is based on a single random number only. As before, the individuals are ordered on a line corresponding to their selection probability, see also Fig. (3.11). As the probabilities are normalized, the line has the length 1. For drawing a given number N of samples a second line of equally spaced pointers referencing positions on the line of samples is generated. The distance between two pointers is $\frac{1}{N}$. A random number $r \in [0, \frac{1}{N}]$ is chosen and the line of pointers is offset by r. Thus, the first pointer references the sample at position r, the second pointer the one at $r + \frac{1}{N}$, and so on. This technique results in a new generation of individuals that minimizes the variance of the number of newly selected individuals from a single member. Mitchell [110] states that stochastic universal sampling is in particular suitable for non-deterministic fitness values as it does not over-emphasize single individuals with possibly wrong large fitness values.

Mutation

The process of randomly changing parts of the genetic material an individual is called mutation. In nature, mutation is one of the key elements for evolution, as it helps to develop individuals with a priori unknown capabilities. For a genetic algorithm the role of the mutation operator is the exploration of the parameter space, as a modification of the genetic material means to relocate the individual to a new position in the parameter

space. This also guarantees the diversity of the population. If, for instance, the individuals of a generation are gathered in a local maximum, the mutation operator pushes away some of the members towards unexplored regions.

For each gene x_i, the mutation probability is given by a random number $p_m^{(i)} \in [0,1]$, which is usually chosen to be very small, e.g. $p_m^{(i)} = 0.01$, so that mutation occurs infrequently.

Too high a value for p_m results in frequent modifications of the genotype and spreads the members randomly, thereby counteracting convergence and neglecting already optimized genetic material. Too small a value will instead turn off the mutation operator and the optimization procedure will be stuck in place with only selection and crossover left working. Thus, the mutation probability must be chosen small, but big enough to surmount false local maxima.

Recombination

In a biological sense, recombination – also called *crossover* – is the process to create and to give place to a new generation. The parents pass on their of their genetic material to their children, thereby ensuring the continuity of their characteristics and capabilities. If these are mixed in the right way they will eventually produce offsprings which are even superior in surviving in the environment. The same goes for the recombination operator of the genetic algorithm. The crossover probability $p_c \in [0,1]$ defines how frequent procreation occurs, it is usually chosen to be much higher than the mutation probability, e.g. $p_c = 0.6$. For recombination, two individuals from the current generation are selected and their genetic material is combined. For a general overview of crossover types, see [13].

As a leftover from the times when the genetic material was commonly binary coded comes the so-called uniform crossover. Here, the genetic material from the two parents $\underline{x}$ and $\underline{y}$ is simply exchanged for a randomly chosen gene k:

$$\begin{array}{l} \underline{x} = (x_1, \ldots, x_{k-1}, x_k, x_{k+1}, \ldots, x_J) \\ \underline{y} = (y_1, \ldots, y_{k-1}, y_k, y_{k+1}, \ldots y_J) \end{array} \Longrightarrow \begin{array}{l} \underline{x}' = (x_1, \ldots, x_{k-1}, y_k, x_{k+1}, \ldots, x_J) \\ \underline{y}' = (y_1, \ldots, y_{k-1}, x_k, y_{k+1}, \ldots, y_J) \end{array}$$

For n-point uniform crossover the exchange is performed at n different positions.

Although these techniques work well for a binary coded genotype, neither one- nor n-point crossover are well suited for floating-point represented variables. The coromosomes can only be exchanged but their values are not altered. The resulting position of the offsprings can only be at the corners of a hypercube which is spanned by the parents $\underline{x}$ and $\underline{y}$ lying in opposing corners, also compare to Fig. (3.12). The space within the hypercube or any position on the edges are not reachable with this technique.

For high dimensional feature spaces, this still leaves many possible choices for offsprings, but often the parents are already well positioned and the local maximum is located right between them. For this reason, arithmetic crossover [13] is utilized in this work. It allows the generation of offsprings as a linear interpolation between the

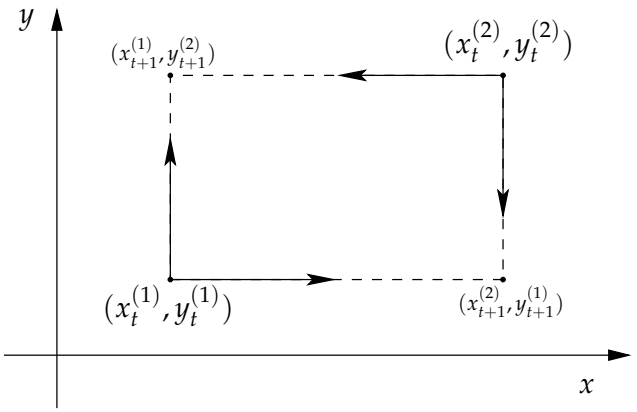


Figure 3.12: *One point crossover.* Illustration of the possible outcomes of a two-dimensional one point crossover. Parents top right and bottom left, possible offsprings top left and bottom right. (Image adapted from [13])

parents which allows to place them anywhere in the hypercube volume. This is achieved by choosing a mixing coefficient $\alpha \in [0, 1]$ which determines the influence rates of the individual parents.

$$\underline{x}_{t+1}^{(k)} = \alpha \underline{x}_t^{(i)} + (1 - \alpha)\underline{x}_t^{(j)} \tag{3.41}$$

After applying the selection and recombination operators to the generation $\underline{S}_t$, a new set of individuals representing a new generation $\underline{S}_{t+1}$ has been generated and can be used for the next iteration of the optimization process.

Elitism

A further improvement of the optimization procedure is introduction of elite individuals. For this work, elitism according to deJong [85] is applied. From each generation, a small number of members with the best fitness values it kept unchanged and passed on into the new generation. This has shown to be a very effective means to stabilize the optimization process and to increase performance for various applications [110].

3.4 Summary

In this chapter, optimization problems are addressed on a theoretical level and a characterization and formal description of such problems is given. That allows us for a given real world problem to abstract from the task and find a formal representation of the optimization problem. Various deterministic and probabilistic optimization techniques were presented and discussed with regard to their competences and restrictions. Usually, there exist many ways to solve a given problem once an abstract formulation has been found. The presented optimization techniques resemble a small portion of the many algorithms existent. They have been presented here in a special chapter, as most of them have shown to perform reliably in many situations. The particle filter, as an example, can be a powerful tool for situations where deterministic approaches fail due to their immense computational complexity. But applying particle filters without considering their random nature easily leads to unpredictable behavior of the whole system they are applied within.

4 Person Localization

It is by no means a coincidence that the human strongly relies on his visual perception. The visual appearance of a scene and the objects within provides rich information about the location of individual objects, their size, their direction of movement and so on. Together with common knowledge about the use of objects in specific situations and the understanding of interactions between objects and their surroundings, humans are able to obtain a detailed representation of a scene just from visual information. In particular, three-dimensional (3D) vision plays an important role in human perception, especially for the recognition of motion and the ability to track objects over time. It is advantageous for vision systems to make use of this very basic information in order to perceive and interprete the environment.

In this thesis, computer vision is used as a technique to perceive and describe the environment. Much like the human, we aim at doing both, deriving a general description of the environment and further localizing and tracking the dynamic parts of the scene. Such automated vision systems can be employed in many situations, but their design is strongly influenced by the constraints of the scenario and the task they are intended to fulfil.

The following chapter presents a method for localising and tracking people and moving objects in a velocity attributed 3D point cloud. We will employ a combination of a bottom-up approach for spatio-temporal scene segmentation and weak models for generation of object hypotheses in order to permit the detection of arbitrary objects. This approach allows task-dependent interpretation without incorporating strong models, which would restrain perception or require prior knowledge about the appearance or structure of the scene.

In the following the scenarios the system has been applied to in the scope of this thesis are introduced. The underlying assumptions and principles are discussed thereafter and the vision-based system for 3D detection and tracking of moving persons and objects in complex scenes is detailed.

4.1 Applicability to Different Scenarios

The system developed in the scope of this thesis has shown to be applicable to various situations, needing only minor modifications to individual system components. In the remainder of this chapter we will detail the system from the viewpoint of two scenarios, as they account for most of the different algorithms used. They share a common goal which is to find and track the moving objects in the scene. A difference lies in the data acquisition process, as different sensors are employed to generate a 3D point cloud. Each sensor and each 3D data generation technique exhibits its own characteristics of

how exact and how dense the point cloud is. Consequently, the data preprocessing techniques and also some later algorithms are adapted to match the requirements posed from the sensor's characteristics.

4.1.1 Industrial Working Cell Safety

The system has shown to robustly perform in an industrial scenario of a working cell with a human worker and a moving robot as presented in [137]. In this scenario, the aim is to localize and track moving objects in the scene for safety purposes. Modern industry robots used within automotive engineering still require strict safety precautions to ensure they do not harm human workers. This often manifests in fences, light curtains, pressure sensitive floor matting and other arrangements which either require complex hardware installations or deter both robots and humans from their normal work flow. The idea is now that if the position and movements of any human or other objects within or close to the working space of the robot is known, the robot can either adjust its working cycle, e.g. lower the speed of its movements if otherwise a collision was likely or it can completely stop in emergency situations. The only sensor needed is a downward-looking stereo camera, which provided the image data to generate a 3D representation of the environment, see also Fig. 4.2. As a part of this industry cooperation, two patents [92, 93] have been published which partly rely on this approach.

4.1.2 Scene Exploration with a Mobile Robot

The second scenario resides in the context of a mobile robot exploring an unknown scene and interacting with people. Acquiring information about the environment is a crucial process. Localizing possible interaction partners and the ability to perceive its surrounding enables a robot to build up and share its representation with the human. In this interactive scene exploration scenario the human guides our mobile robot BIRON around, showing an apartment [27, 62]. In such a highly dynamic scenario the scene is barely static as the human interaction partner will be visible and other people or moving objects may be close to the robot. We can assume that the robot is standing still for only a few seconds (2s – 3s) to localize and track possible interaction partners. Furthermore, the robot keeps looking around, comparable to human's eye saccades, restricting the time to record images without moving the camera to a few frames. Enabling the robot to deal with moving objects is achieved by distinguishing between the moving and the static parts in a scene.

Localizing and tracking those objects is an important feature in itself, as presented in [137], but it furthermore enables the robot to build a model of a small part (subscene) of the complete environment. It has been shown [9, 157] that by neglecting data emerging from moving objects, the reconstruction gets more robust. Here, we will focus on the localization and tracking aspects as presented in [158], as these parts of the presented approach have been developed as part of this thesis; the data acquisition and scene reconstruction parts of the system have been developed by other persons.

4.2 Person Localization System Design

In the following the system design for localizing and tracking persons and moving objects is briefly outlined, after that the individual components developed for the different scenarios are discussed in more detail.

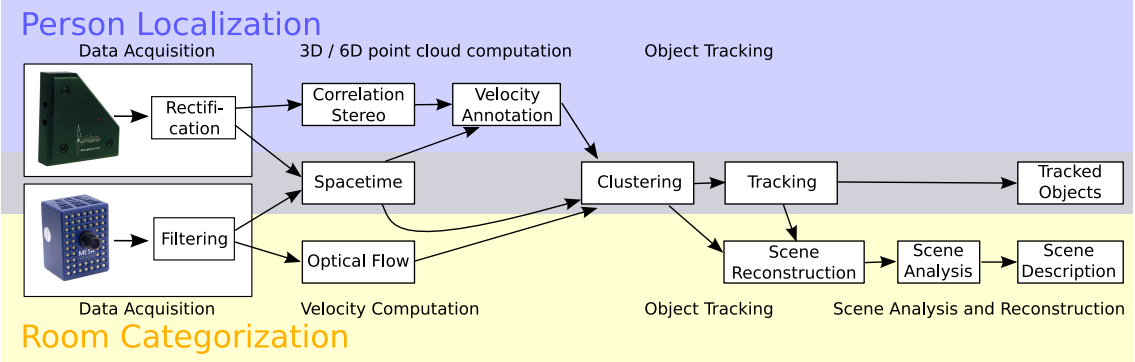


Figure 4.1: *Person localization and tracking system overview.* The two approaches for person localization and room categorization share many modules. After data and image acquisition, velocity components are calculated. Using this 6D information, all objects are clustered and tracked with a particle filtering framework. Finally, the static scene can be reconstructed and analyzed by neglecting data originating from moving objects and persons in the scene.

The presented approach for analyzing a dynamic environment is very similar for the two scenarios it has been applied to, see Figure 4.1 for a comparison. The challenges each of these scenarios pose have inspired us to implement two versions of the localization and tracking approach. The resulting modifications to the system are discussed in the following.

The first realization - the "industry" scenario - is based on a stereo camera setup. Two different techniques are applied to extract 3D information and velocities from the images: A correlation-based block-matching stereo algorithm and a spacetime stereo approach based on local intensity modelling. The data from these two algorithms is merged into a 3D point cloud annotated with velocity components for some of the points.

For the second realization - the "mobile robot" scenario - we use a 3D Time-of-Flight (ToF) sensor to acquire a 3D point cloud representing the environment. Filtering deletes invalid points and reduces noise, which strongly depends on the reflectance characteristics of the observed surfaces. Each of the remaining 3D points is annotated with velocity components using two algorithms: An optical flow method that detects the moving parts in the scene and an adaption of the spacetime approach already used for the "industry" scenario.

For both scenarios, we now have a 3D point cloud that is annotated with velocity components for the moving parts of the scene. The following computation steps are very similar for both scenarios.

For localising persons and objects in the scene the point cloud is segmented into clusters by applying a hierarchical clustering algorithm, using velocity information as an addi-

tional discrimination criterion. Initial object hypotheses are obtained by partitioning the observed scene, also including the tracking results of the previous frame. Using a cylinder as a weak object model, multiple clusters are combined to potential dynamic objects. Multidimensional unconstrained nonlinear minimisation is then applied to refine the initial object hypotheses such that neighboring clusters with similar velocity vectors are grouped to form a compact object. A particle filtering framework is used to select hypotheses which generate consistent trajectories.

For the "industry" scenario, the tracked objects can now be used to predict their positions and check whether a collision with the robot is likely. For the "mobile robot" scenario, the static scene can be reconstructed by neglecting all points belonging to tracked objects. At the same time, the found objects can be used to find possible interaction partners.

In the following sections the individual steps of the approach are detailed.

4.3 6D Point Cloud Generation

In human perception, depth plays an important role for tracking moving objects over time. It is obviously advantageous for vision systems to make use of this very basic information to perceive and interprete the environment. In addition to represent the 3D structure of the scene, we are interested in observing moving objects. Motion is a strong cue to segment objects from the background and can also be used to make predictions about the appearance of the scene in the near future. As an example to recognize motion, we will present methods to compute additional velocity information for the 3D point cloud using two different sensor setups.

4.3.1 Velocity Computation using a Stereo Camera Setup

It is known that humans are able to judge distances based on multiple cues. Depth from triangulation - also called depth from stereo - builds on the fact that the eyes are separated from each other in a known horizontal distance. Although it is only one of many cues it is already sufficient to generate a good estimation on the distance of objects, in particular for small distances. A further advantage of depth from stereo approaches is the fact that they usually work quite well in scenes with enough contrast, e.g. light objects in front of a dark background. Therefore, stereo vision is an efficient approach to obtain 3D information about the scene without the need to place markers or to constrain the appearance of objects.

Correlation-Based Block-Matching Stereo Algorithm

A classical stereo vision algorithm is the block-matching approach [44, 47, 48, 68]. At each pixel of, say, the left image, a rectangular window is centered on the position of that pixel. The algorithm computes the disparity value for that pixel by determining

in the right image the window of identical size on the same epipolar line for which a similarity measure, e.g. the sum of squared differences (SSD), obtains an optimum. Real-time stereo vision systems like those presented in [47] or [68] rely on the principle of establishing correspondence e.g. by computation of SSD. For speed-up, making use of several resolution levels [47] often turns out to be useful, as well as heuristics such as suppression of uniform image regions by employing an interest operator and checking for left-right consistency.

As a first stereo analysis algorithm, we employ the real-time block-matching approach described in [47]. We assume that the cameras are calibrated and the images are rectified to standard stereo geometry with epipolar lines parallel to the image rows [94]. For each interest pixel in the left image for which a sufficiently high intensity gradient is observed, a corresponding point is searched along the epipolar line in the right image. Here, we use the SSD as a similarity measure. A square region around the interest pixel in the left image is compared with regions on the corresponding epipolar line in the right image for all candidate disparities, resulting in an array of correlation coefficients. The disparity corresponding to the minimum SSD value is determined, and a parabola is fitted to the local neighborhood of each maximum, yielding the disparity value at subpixel accuracy. Only well localised correspondences are considered for further processing.

Spacetime Stereo Algorithm based on Local Intensity Modelling

The approach described in [47] is extended in [48] by tracking 3D points individually in a six-dimensional position-velocity space, thus extracting motion information during an additional processing step after correspondence analysis. While virtually all classical stereo vision approaches do not make use of image sequences, [36] describes a block-matching spacetime stereo vision scheme which relies on pairs of image sequences rather than just pairs of images, thus providing more exact results.

The idea of using image sequences for correspondence analysis can be carried on which leads to the spacetime stereo approach of Wöhler [137]. He extends this idea to not only estimate the 3D position of a point but also its velocity component along the epipolar line. This technique is used to generate velocity annotations for the 3D point cloud generated from the stereo camera and has shown to be applicable with minor modifications to data from the Swissranger Time-of-Flight sensor. The algorithm is briefly presented in the following.

Wöhler's spacetime stereo approach exploits the spatio-temporal structure of the acquired sequence of image pairs. To the local spatio-temporal neighborhood of each interest pixel a parameterised function $h(\underline{P}, u, v, t)$ is adapted, where u and v denote the pixel coordinates, t the time coordinate, and $\underline{P}$ the vector of function parameters.

Ideally, an object boundary is described by an abrupt intensity change. In real images, however, one does not observe such discontinuities since they are blurred by the point spread function of the optical system. Therefore, we model the intensity change at an object boundary by a "soft" sigmoid function like the hyperbolic tangent. As we

cannot assume the image regions inside and outside the object to be of uniform intensity, we model the intensity distribution around an interest pixel by a combined sigmoid-polynomial approach:

$$h(\underline{P}, u, v, t) = p_1(v,t) \tanh\left[p_2(v,t)u + p_3(v,t)\right] + p_4(v,t). \tag{4.1}$$

The terms $p_1(v,t)$, $p_2(v,t)$, $p_3(v,t)$, and $p_4(v,t)$ denote polynomials in v and t. The polynomial $p_1(v,t)$ describes the amplitude and $p_2(v,t)$ the steepness of the sigmoid function, which both depend on the image row v, while $p_3(v,t)$ accounts for the row-dependent position of the model boundary. The value of $p_2(v,t)$ is closely related to the sign of the intensity gradient and to how well it is focused, where large values describe sharp edges and small values blurred edges. The polynomial $p_4(v,t)$ is a spatially variable offset which models local intensity variations across the object and in the background, e.g. allowing the model to adapt to cluttered background. All described properties are assumed to be time-dependent. Interest pixels for which no parametric model of adequate quality is obtained are rejected if the residual of the fit exceeds a given threshold.

The parametric model according to Eq. (4.1) in its general form requires that a nonlinear least-mean-squares optimisation procedure is applied to each interest pixel, which may lead to a prohibitively high computational cost of the method. Is is possible, however, to transform the nonlinear optimisation problem into a linear one by assuming that (i) the offset $p_4(v,t)$ is proportional to the average pixel intensity $\bar{I}$ of the spatio-temporal neighborhood of the interest pixel, i.e. $p_4(v,t) = w\bar{I}$, and (ii) the amplitude $p_1(v,t)$ of the sigmoid is proportional to the standard deviation σ_I of the pixel intensities in the spatio-temporal neighborhood with $p_1(v,t) = k\sigma_I$. These simplifications yield the model equation

$$p_2(v,t)u + p_3(v,t) = \operatorname{artanh}\left[\frac{I(u,v,t) - w\bar{I}}{k\sigma_I}\right] \equiv \tilde{I}(u,v,t), \tag{4.2}$$

where the model parameters, i.e. the coefficients of the polynomials $p_2(v,t)$ and $p_3(v,t)$, can be determined by a linear fit to the transformed image data $\tilde{I}(u,v,t)$. Pixels with $|[I(u,v,t) - w\bar{I}] \,/\, [k\sigma_I]| > \theta$ are excluded from the fit, where θ is a user-defined threshold with $\theta < 1$, since arguments of the artanh function close to 1 would lead to a strong amplification of noise in the original pixel intensities $I(u,v,t)$. The factors k and w are further user-defined parameters of the algorithm.

The intensity gradient obtains its maximum value in horizontal direction at the root $u_e(v,t) = -p_3(v,t)/p_2(v,t)$ of the hyperbolic tangent. The horizontal position of the intensity gradient at the current time step for the epipolar line on which the interest pixel is located is given by the value $u_e(v_c, t_c)$, where the index c denotes the centre of the local neighborhood of the interest pixel. The direction δ of the intensity gradient is given by $\delta = \partial u_e / \partial v$. The velocity μ of the intensity gradient along the epipolar line corresponds to the temporal derivative $\mu = \partial u_e / \partial t$ of the location of the epipolar transection. Both derivatives are computed at v_c and t_c.

For correspondence analysis, the SSD similarity measure is adapted to our algorithm by comparing the fitted functions $h(\underline{P}_l, u, v, t)$ and $h(\underline{P}_r, u, v, t)$ rather than the pixel

intensities themselves, where the indices l and r denote the left and the right image, respectively:

$$S = \int \left[h(\underline{P}_l, u - u_e^l(v_c, t_c), v, t) - h(\underline{P}_r, u - u_e^r(v_c, t_c), v, t) \right]^2 du\, dv\, dt, \tag{4.3}$$

where u, v, and t traverse the local spatio-temporal neighborhood of the left and the right interest pixel, respectively. Once a correspondence between two interest pixels on the same epipolar line has been established by searching for the best similarity measure, the disparity d corresponds to $d = \left[u_i^l + u_e^l(v_c, t_c)\right] - \left[u_i^r + u_e^r(v_c, t_c)\right]$ with u_i^l and u_i^r as the integer-valued horizontal pixel coordinates of the left and the right interest pixel, respectively. Given the optical and geometrical parameters of the camera system, the velocity components parallel to the epipolar lines and along the depth axis can be computed directly in metres per second from the values $\bar{\mu} = \left(\mu^l + \mu^r\right)/2$ and $\partial d/\partial t = \mu^l - \mu^r$.

Annotating the 3D Point Cloud with Velocities

Both described stereo techniques generate 3D points based on edges in the image, especially object boundaries. Due to the local approach they are independent of the object appearance. While correlation stereo has the advantage of higher spatial accuracy and the capability to generate more point correspondences, spacetime stereo provides an additional velocity value for each computed stereo point. However, it generates a smaller number of points and is spatially less accurate, since not all edges are necessarily well described by the model defined in Eq. (4.1). Taking into account these properties of the algorithms, the results are merged into a single motion-attributed 3D point cloud.

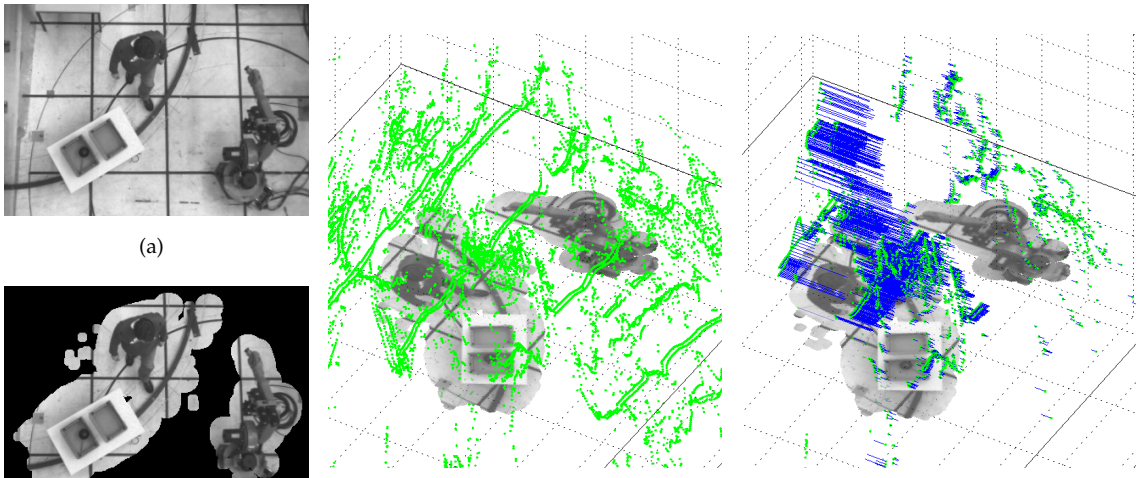


Figure 4.2: *Working cell observed with a stereo camera setup.* (a) Original image (left camera), (b) background subtracted image, (c) full correspondence stereo point cloud, (d) reduced motion-attributed point cloud.

For each extracted 3D point c_k an average velocity $\bar{v}(c_k)$ is calculated, using all spacetime points s_j, $j \in (1, \ldots, J)$ in an ellipsoid neighborhood defined by $\delta_S(s_j, c_k) < 1$ around

c_k. To take into account the spatial uncertainty in depth direction of the spacetime data, $\delta_S(s_j, c_k)$ defines a Mahalanobis distance whose correlation matrix Σ contains an entry $\Sigma_z \neq 1$ for the depth coordinate which can be derived from the recorded data.

$$\bar{v}(c_k) = \frac{\rho}{J} \sum_{j=1}^{J} v(s_j) \; \forall \; s_j : \delta_S(s_j, c_k) < 1 \tag{4.4}$$

The factor ρ denotes the relative scaling of the velocities with respect to the spatial coordinates. It is adapted empirically depending on the speed of the observed objects. This results in a 4D point cloud, where each 3D point is attributed with an additional 1D velocity component parallel to the epipolar lines, see Fig. (4.2(d)).

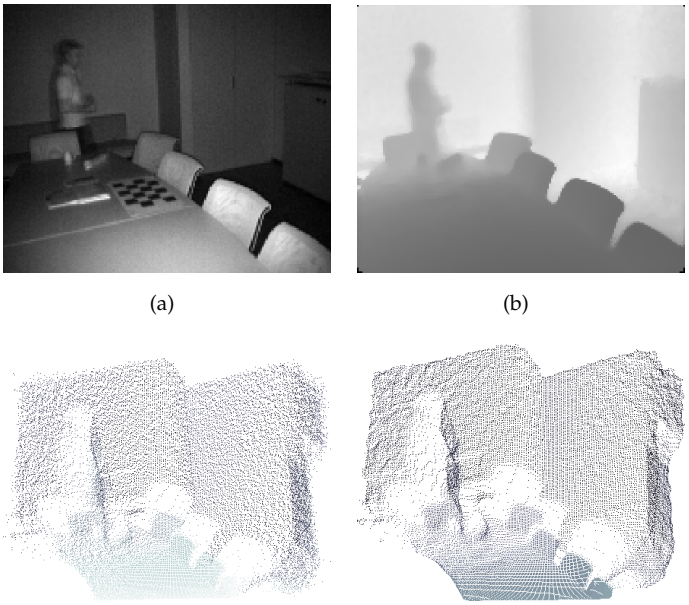
A reference image of the observed scene is used to reduce the amount of data to be processed by masking out 3D points that emerge from static parts of the scene, as shown in Fig. (4.2(b)). Furthermore, only points within a given interval above the ground plane are used, as we intend to localise objects and humans and thus always assume a maximum height for objects above the ground.

4.3.2 Velocity Computation using a Time-of-Flight Sensor

As correspondence-based stereo vision is a passive approach, the quality of the reconstructed point cloud strongly depends on the environmental conditions and on the objects in the scene. Active sensors try to overcome this restriction by generating and sending a signal on their own and measuring the reflected signal. Laser range scanners deliver one scanning line of accurate distance measurements often used for navigation tasks [113, 167]. 3D Time-of-Flight (ToF) Sensors [169, 174] have recently become available at reasonable prices. They combine the advantages of active sensors providing accurate distance measurements and camera-based systems recording a 2D matrix at a high frame rate. Compared to stereo rigs the 3D ToF sensors can deal much better with prominent parts of rooms like walls, floors, and ceilings even if they are not structured. ToF sensors are of special interest to mobile robotics, as they offer dense depth maps from a compact device.

Our system uses the Swissranger SR3000 provided by Swiss Center for Electronics and Microtechnology (CSEM) [169] delivering a matrix of distance measurements independent from texture and lighting conditions. It consists of 176×144 CMOS pixel sensors which are able to determine actively the distance between the optical center of the camera and the real 3D world point by measuring the time-of-flight of a near-infrared signal. Besides the distance value matrix (Fig. 4.3(b)), the camera provides for each frame a matrix of amplitude values (Fig. 4.3(a)). The value indicates the amplitude of the reflected near-infrared signal received by the sensor and therefore the amount of light reflected from the object observed.

Various preprocessing techniques dealing with noise arising from the different reflectance properties and characteristics of the ToF camera have been presented recently. Schiller [134] proposed an automatic calibration of the entire 3D ToF signal using a number of cameras. Color information can be used for outlier detection as shown by Huhle [76]. To



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

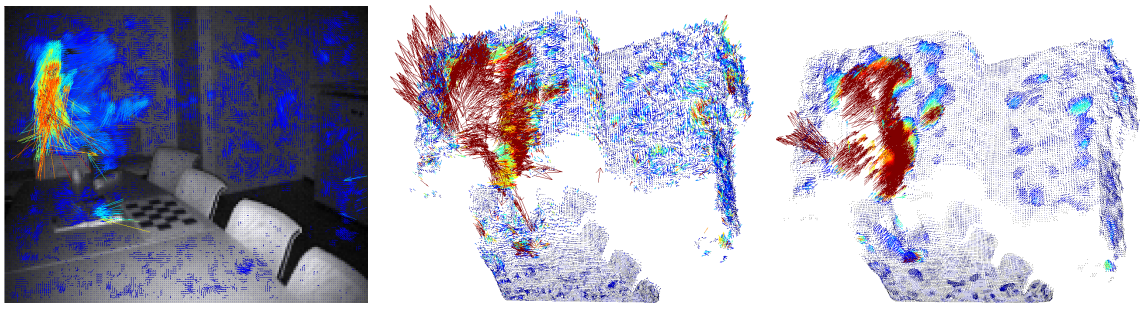
Figure 4.3: *Data acquired from the Time-of-Flight sensor.* (a) amplitude image, (b) distance image, (c) unpreprocessed 3D point cloud, and (d) preprocessed 3D point cloud. (Image found in [158])

remove the so-called "flying pixels" at edges, taking into account the 2D neighborhood can help to iteratively detect geometric outliers [75]. Smoothing techniques that directly rely on the ToF data are amplitude thresholding with a fixed value [108], and correcting the amplitude values using distance values and vice versa [117]. For this thesis we applied the preprocessing techniques proposed by Swadzba in [159], including a distance-adaptive median filter and a rejection of "flying pixels".

In the following two different image based methods for motion computation are presented which can be easily applied here by treating the point cloud as planar depth maps or images respectively. The methods differ in the accuracy of the estimated velocity components and in the number of pixels for which the motion can be calculated. At first glance the optical flow method fits best with the dense depth maps of the Swissranger sensor as it is able to estimate the motion for almost every 3D point. However, the results are prone to be noisy. The presented spacetime method is an adaption from an approach used for stereo computations. It provides exact measurements of the motion present in the image but as it it based on a spatio-temporal edge model, the motion can only be computed at very few points where the model can be fitted.

Optical Flow

In order to distinguish between static parts of the scene and moving objects we determine the motion in the scene as presented by Swadzba [157]. A widely used technique is to compute the optical flow of each 2D image pixel, which is the distribution of apparent velocity of moving brightness patterns in an image and arises both from the relative motion of the object and the viewer [56]. The flow of a constant brightness



(a) (b) (c)

Figure 4.4: *Velocity detection with the optical flow method.* (a) 2D velocity vectors (b) 3D velocity vectors from combining 2D velocities and point correspondences in consecutive images, (c) the latter smoothed componentwise by a median filter. Each 3th velocity vector is displayed and color coded with respect to its length, with red denoting large motion and blue little.(Image found in [158])

profile

$$
\begin{aligned}
I(x,y,t) &= I(x+\mathrm{d}x, y+\mathrm{d}y, t+\mathrm{d}t) \\
&= I(x+v_x\cdot\mathrm{d}t, y+v_y\cdot\mathrm{d}t, t+\mathrm{d}t) \qquad (4.5)\\
\Rightarrow\ & \frac{\partial I}{\partial x}\cdot v_x + \frac{\partial I}{\partial y}\cdot v_y = -\frac{\partial I}{\partial t} \qquad (4.6)
\end{aligned}
$$

is described by the constant velocity vector $\underline{v}_{2\mathrm{D}} = (v_x, v_y)^T$. Usually, the estimation of optical flow is founded on differential methods. They can be classified into global strategies which attempt to minimize a global energy functional [73] and local methods that optimize some local energy-like expression. A prominent algorithm developed by Lucas and Kanade [104] uses the spatial intensity gradient of the images to find a good match using a type of Newton-Raphson iteration. They assume the optical flow to be constant within a certain neighborhood $\mathcal{N}$ which allows to solve the Optical Flow Constraint Equation 4.6 via least square minimization. For the presented approach, we used a hierarchical implementation of Lucas's and Kanade's algorithm written by Sohaib Khan [1] [2].

Spacetime Algorihm Modification

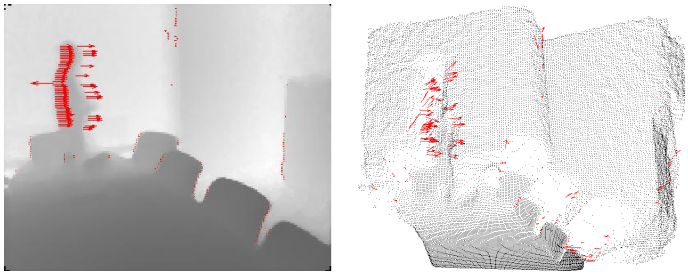
Our second velocity computation approach exploits the spatio-temporal structure of the acquired sequence of images. Although originally designed to detect correspondences with velocities in sequences of image pairs originating from a stereo camera, we found Wöhler's spacetime approach to nicely cope with the depth data of the Swissranger time-of-flight sensor, as presented in [158].

Other than for the original approach as described in section 4.3.1, we are no longer interested in acquireing 3D information for points in the scene, the swissranger already

[1] http://www.cs.ucf.edu/∼khan/
[2] http://server.cs.ucf.edu/∼vision/source.html

provides a dense depth map. We are instead interested in the temporal structure of the scene, namely if there are any moving points. The original spacetime apporach fits a parametrized sigmoid function to the local spatio-temporal neighborhood of each interest pixel in both stereo images to simultaneously calculate correspondences and velocity components. Here, we apply the algorithm to a sequence of depth images from the Swissranger camera and let the spacetime algorithm search for horizontal velocity components only. By using the image representation of the depth data and interpreting distances as intensities in the image, the original approach requires only marginal changes.

We found the assumption that object boundaries are blurred by the point spread function of the optical system to be as well valid for the depth images. Naturally, the velocities can only be calculated for a small number of 3D points for which the local spatio-temporal neighborhood matches well enough with the sigmoid function. Given the optical and geometrical parameters of the camera system, the velocity component parallel to the horizontal line can be computed directly. The result is a number of points annotated with a horizontal velocity component each as shown in Fig. (4.5(a)). These points are mapped to the 3D point cloud from the time-of-flight camera which finally yields a 3D point cloud with an additional horizontal velocity component, see Fig. (4.5(b)).



(a) (b)

Figure 4.5: *Velocity processing with the spacetime method.* (a) 2D velocity vectors along the horizontal lines (b) 3D velocity vectors from mapping the 2D velocities to the 3D point cloud.

One might argue that this is not a full 6D representation, which is obviously true, but for the proposed scenario we are mainly interested in observing persons walking in front of the robot. In such situations, the lateral movement components hold the most important information for us. Consequently, the weak object model that is later applied to these data also does not encode a vertical position or vertical velocities but only a XY-position on the ground. We can therefore use the data generated by this technique in the same way as those from the stereo setup.

The spacetime technique relies on several - we have chosen three - consecutive images to calculate the temporal derivatives. As a consequence, no velocities can be calculated for the very first and very last image of a sequence. For the same reason, the algorithm reacts very sensitively to variations in the time interval between individual images, e.g. due to framedrops. Both arguments are also valid for the optical flow technique, as it similarly relies on a temporal differencing.

When comparing the results of the optical flow and the spacetime technique, see also Figures 4.4 and 4.5, it gets obvious that the velocities computed by the latter method are

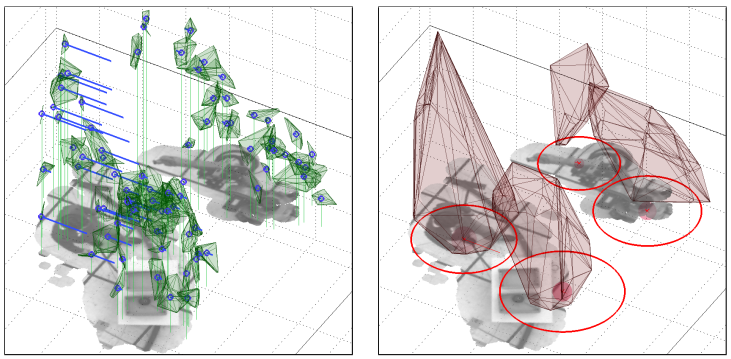
far more accurate but also very sparse. The optical flow method provides velocities for each point, which goes well with the dense depth maps of the time-of-flight sensor, but only at the price of unreliable and noisy results.

4.4 Generation and Tracking of Object Hypotheses

For both scenarios, the preceding processing steps provided velocity annotated point cloud representing the scene. The remaining steps are almost identical for both scenarios. For localising persons and objects in the scene the point cloud is segmented into clusters by applying a hierarchical clustering algorithm, using the velocity information as an additional discrimination criterion. Initial object hypotheses are then obtained by partitioning the observed scene with cylinders, including the tracking results of the previous frame. Multidimensional unconstrained nonlinear minimisation is applied to refine the initial object hypotheses, such that neighboring clusters with similar velocity vectors are grouped to form a compact object. A particle filter is finally used to select hypotheses which generate consistent trajectories.

4.4.1 Over-Segmentation for Motion-Attributed Clusters

To simplify the scene representation and to reduce the computation complexity, the 6D points are clustered. The first step is to span small contiguous regions in the cloud of the 6D points, based on features for spatial proximity and homogeneity of the velocities. By incorporating velocity information for clustering, we expect an improvement in segmentation at these early stages of the algorithm, without needing strong models to ensure separation of neighboring moving objects, e.g. a person walking in front of a wall.



(a) (b)

Figure 4.6: *Hierarchical 6D Clustering.* (a) Over-segmentation and cluster velocities, (b) objects with convex hull.

This procedure deliberately over-segments the scene, generating many small motion-attributed clusters. To build these clusters we apply a hierarchical clustering using the complete linkage algorithm [8], also called furthest neighbor, to describe the distance between two clusters. The resulting hierarchical tree is partitioned by selecting a clustering threshold and addressing each subtree as an individual cluster, see Fig. (4.6(a)).

The criterion for selecting the threshold is the increase in distance between two adjacent nodes in the tree, for which a maximum allowed value is determined empirically.

For each cluster, the following attributes are extracted based on all associated 6D points: The 2D position of the centroid projected on the ground plane, a weight factor based on the number of points, and the mean velocity of all points.

4.4.2 Weak Model for Object Hypotheses

From here on, persons and objects can be represented as a collection of clusters of similar velocity within an upright cylinder of variable radius, see Figure 4.7. The model aims at grouping together as many neighboring clusters with similar velocity values as possible to form a compact object, as shown in Fig. (4.6(b)).

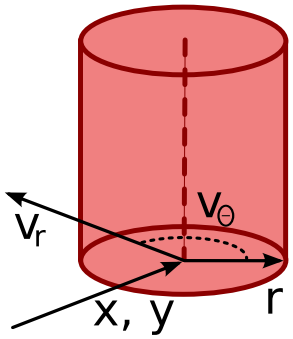


Figure 4.7: *A cylinder as a weak object model.* The appearance of the cylinder is fully describes by the following five parameters: x and y position on the ground, radius r, direction of movement v_θ and magnitude of movement v_r.

This weak model offers an object hypothesis $s(a)$, which is suitable for persons and most encountered objects. It is represented by a five-dimensional parameter vector $a = [x\ y\ v_\theta\ v_r\ r]^T$ with x and y being the center position of the cylinder on the ground plane, v_θ denoting the magnitude and v_r indicating the direction of the velocity of the object, r is the radius of the cylinder.

4.4.3 Kernel Particle Filter for Object Localization

To extract the correct object positions, we utilise a combination of parameter optimisation and particle filter based tracking. We first generate a number of initial hypotheses and optimise their locations in the parameter space. This set of resulting hypotheses is predicted into the next frame and tracked through a kernel based particle filter [137] as follows.

For each new image a set of initial object hypotheses is created for initialization and error recovery. The basis is the set of tracking results from the previous frame. To find any new object that appeared in the scene or that have been lost during tracking, the observed scene is partitioned with cylinders generating additional hypotheses. A multidimensional unconstrained nonlinear minimisation [114] is applied to roughly estimate the positions of the objects in the scene. After optimisation, hypotheses with identical parameterisation are merged and those without any clusters are removed.

The position, size and velocity of each object $s_{t-1}^{(k)}(a)$ that has been localized in the last frame is predicted for the current frame t utilizing a first order motion model $a^* = \Phi(a, \dot{a})$, creating a new set of hypotheses $s_t^{(k)}(a^*) \overset{\Phi}{\leftarrow} s_{t-1}^{(k)}(a)$, $k = 1, \ldots, K$. Each of these K hypotheses representing an object found in the scene can be seen as a specific point in the parameter space, or in the context of the kernel particle filter stands for a particle.

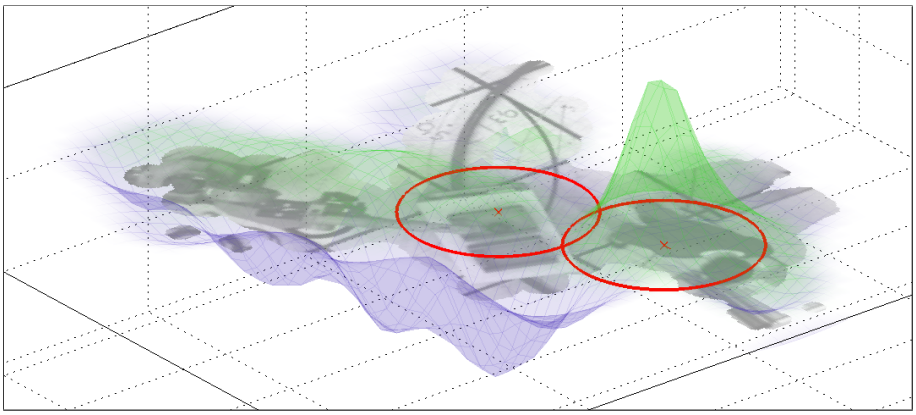


Figure 4.8: *Object hypothesis likelihood function.* Evaluated on a regular grid of X and Y positions for two examplaric hypotheses while keeping the following parameters constant. (blue) object moving upwards: $v_\theta = 0$, $v_r = 0.26$ m s^{-1}, $r = 0.53$ m, (green) object moving downwards: $v_\theta = \pi$, $v = -0.79$ m s^{-1}, $r = 0.53$ m, values mirrored for clearer display.

Each hypothesis is rated based on the relative position, relative velocity, and weight of all clusters l within the cylinder s_k using Gaussian kernels:

$$\rho(s^{(k)}) = K_r(s^{(k)}) \sum_{l \in s^{(k)}} K_d(l, s^{(k)}) K_v(l, s^{(k)}) \tag{4.7}$$

with the weithing functions

$$K_r(s^{(k)}) = \exp\left(-\frac{r(s^{(k)})^2}{2H_{r,\max}^2}\right) - \exp\left(-\frac{r(s^{(k)})^2}{2H_{r,\min}^2}\right) \tag{4.8}$$

$$K_d(l) = \exp\left(-\frac{\|d(l) - d(s^{(k)})\|^2}{2\ H_d^2}\right) \tag{4.9}$$

$$K_v(l, s^{(k)}) = \exp\left(-\frac{\|v(l) - v(s^{(k)})\|^2}{2\ H_v^2}\right) \tag{4.10}$$

The term 4.8 is used to keep the radius in a realistic range, 4.9 reduces the importance of clusters further away from the cylinder centre, and 4.10 masks out clusters having differing velocities. The functions $r(\cdot)$, $d(\cdot)$, and $v(\cdot)$ extract the radius, the 2D position on the ground plane and the velocity of a cluster l or a hypothesis s_k. The kernel widths H are determined empirically.

Eqn. (4.7) denotes the quality of the grouping process for a given hypothesis on the set of clusters. It is used as the observation function $\rho(s^{(k)})$ of the particle filter as shown in Eqn. (3.24). The outcome is a density approximation based on the object hypothesis and the attributes of the appendant clusters, with the maxima corresponding to the actual objects. See Fig. (4.8) for a plot of the likelihood for two moving objects while varying

their position in space. Each found object can be easily represented by its convex hull, see Fig. (4.6(b)), which encloses all 3D points that are assigned to the object.

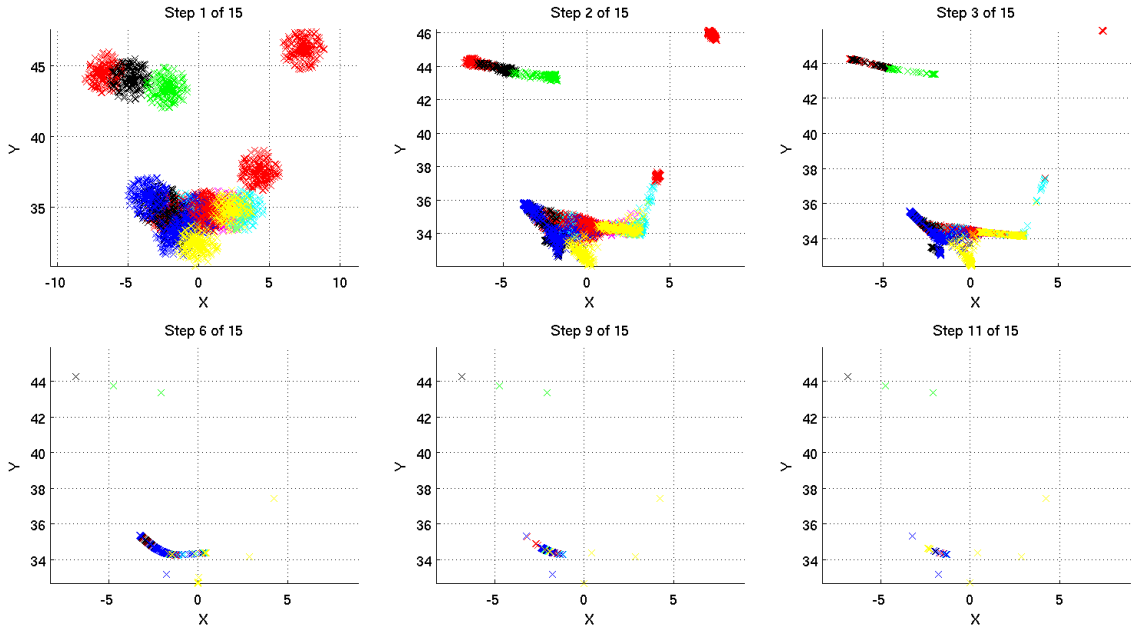


Figure 4.9: *Mean shift iterations for object localization.* Consecutive iterations refine the initial particle distribution depicted in step 1. The colors denote individual objects. Only few iterations are needed to converge towards the local maxima.

The particle filter propagates the found objects from the current timestep to the next image. It thereby also tracks the objects, as not only the position but also the size ans the velocity of the objects are coded in the model and used for matching the model with the clustered point cloud. If, for instance, two similar-sized objects are passing close to each other, they have a small spatial distance and cannot be told apart based on their size but they still have differing velocity vectors which allows the kernel particle filter to distinguish between them. Two hypotheses that always move parallel, as a second example, can very well resemble one and the same object in the world and can be confused unless they are spatially separated far enough.

Fig. (4.9) shows the refinement of the particle distribution using multiple iterations of the mean shift technique. The first step shows the initial distribution as created from the propagation step from the last image. Mean shift converges very fast for well-defined maxima (upper right red blob) and converges along the steepest ascent of the underlying PDF for more complicated situations (upper left). Multiple neighboring objects can only be separated using a higher number of iterations (lower left). Tracking of individual objects is achieved by matching it with particles generated from the same hypothesis for consecutive images.

The motion model used for propagation ensures that only objects forming a consistent trajectory are kept as active hypotheses, see Fig. (4.10). If an object moved faster or changed direction faster than covered by the motion model, its old trajectory ends and a new hypothesis is generated at its current position with its current valocity. Additional

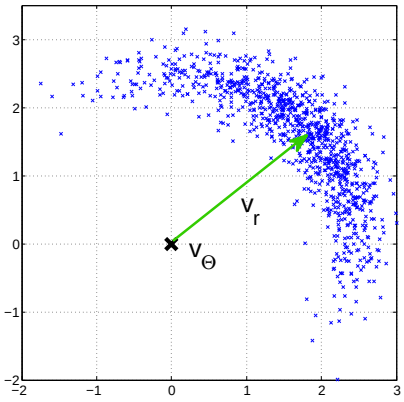


Figure 4.10: *Motion model for person tracking.* Using the velocity information from the tracked clusters, the position of the person is projected to the next timestep. For both velocity v_r and direction v_Θ of movement, gaussian noise is added to create new object hypothesis.

constraints on the maximum and minimum size are ensured by Eqn. (4.7). That leaves us with only those hypothesis forming a consistent trajectory.

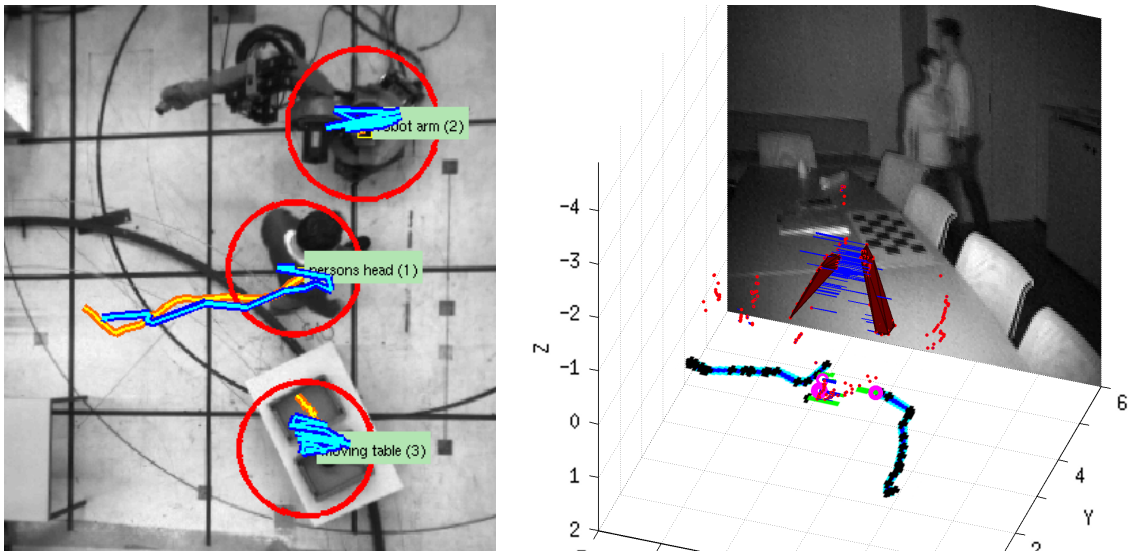


Figure 4.11: *Trajectories of tracked objects.* (a) 'industry' scenario: Person walking in the workspace of a robot. Tracked object positions (blue) and ground truth labels (green text boxes and yellow trajectories), (b) 'mobile robot' scenario: Two persons passing each other. Tracked object positions (blue). The points with velocities, clusters, and the convex hulls are also displayed.

The final outcome is for each frame a list of tracked object hypotheses, see Fig. (4.11) which not only holds their position in space but also their current velocity, their size and the convex hull representing a rough shape in 3D for further computations. An evaluation of the exactness of the tracking is presented in Chapter 6.1, an example for applying the system in a different scenario is presented in Chapter 7.1.

4.5 Summary

In this chapter we presented a system for localizing objects and persons in the context of a mobile robot scenario. A 3d point cloud representing the observed space is acquired

using a stereo camera setup or a time-of-flight camera. Different techniques to extract velocity annotations for the 3D point cloud have been presented. The resulting 6D point clouds can be beneficial to segment the scene into meaningful clusters, the additional velocity information can help to improve the quality of the segmentation process. Subsequently, objects and humans in the scene can be localized by combining multiple moving clusters using a cylinder as a weak object model. The quality of the grouping process is rated based on features like the proximity of the clusters and their similarity of velocity. A kernel particle filtering scheme tracks multiple person hypotheses over time, thus providing robust trajectories and even a motion prediction for each object encountered.

5 Body Pose Tracking

In the last chapter we presented a method to localize and track persons in a scene in the context of a mobile robot scenario. Using the presented techniques we are already able to give answers to some typical questions arising in such a scenario, for instance "Where are objects in the scene" and "where are they moving". Two follow-up questions that have not yet been adressed but that are of special importance for an interactive robot are: "Are these moving objects really persons" and "What is he or she doing"'. These two questions will be the guideline for the following chapter.

In the remainder of this chapter it is explained how the posed questions can be formulated as an optimization problem and which techniques are suitable to provide a solution under the specific constraints of the given scenario. As a possible solution we discuss a technique to track the upper body of a human using an articulated body model and a kernel particle filter framework for the inference process.

The chapter starts with an introduction to the system and gives an overview over the framework.

5.1 Human Robot Interaction Scenario

Let us recall the mobile robot scenario from the previous chapter. It was based on the idea of a typical human robot interaction scenario, i.e. where an individual is communicating with an artificial actor. While the last chapter presented methods to localize possible interaction partners we will now focus on the idea to provide means to understand the gestures of the human.

Figure 5.1: BIRON *hometour scenario.* A person guides the mobile robot BIRON around in a flat and teaches him an object using a pointing gesture.

That leads us to a more challenging scenario. Imagine a person guiding the robot around, showing him different rooms in a flat, teaching him objects to recognize and

introducing him to other persons. It is natural for humans to use gestures in such situations, for instance to give directions and commands, to refer to objects and places, and to support verbal communication. A robot can greatly benefit from being able to recognize and understand these gestures in the right context.

The work presented in this chapter can be seen as the basis for a system to recognize and analyze these gestures. It deals with tracking the movements of a human's arms and hands in 3D as presented in [136] but it stops at the very point where these movements are recorded and can be expressed in the form of a body posture or a trajectory of a single limb. The analysis which gesture is concealed within the recorded motions and what these gestures mean in a specific context is left over to others, it has been exemplary presented in [135] where pointing gestures to objects in a scene are being recognized by the system, see also Sec. (7.2). The system has also shown to be beneficial for other applications, as presented in chapter Sec. (7.4).

For the proposed method, some restrictions which are well suited for human robot interaction can be derived from the given scenario which have been a driving factor for the design of the whole system. The assumptions are as follows: The person is trying to communicate, therefore his or her intention is to cooperate with the system. The person is standing in an upright position, facing the camera. At least the upper body, including the head, the torso and the hands are visible. The person and its appearance, e.g. clothing, and the appearance of the scene are previously unknown to the robot. As a sensor, a single monocular camera is used as it was the intention to build a system that is not dependent on other sensors or special hardware.

5.2 Body Pose Tracking System Overview

There are a number of features that should be considered in order to make a system flexible enough to operate within the described context of a human robot interaction scenario. The person and its body dimensions must not necessarily be known a priori, but the distance of the human to the robot has to be adequate for interaction. Images are acquired using a single monocular color camera as the system is intended to be used on a mobile robot without employing further sensors. During system design, we also avoided specific background models to allow the tracking to be independent from the appearance of the observed scene. To further allow a moving camera, image background substraction based techniques like motion history images are avoided as well.

The presented algorithm uses an articulated body model that is compared to the image through multiple intensity and color cues that rate how well the model pose resembles the actual pose of the human. The inference process is achieved by a kernel particle filtering framework, which efficiently explores the highdimensional pose space. An overview of our algorithm for matching 3D object features of a generic human body model and 2D image features extracted from input images is depicted in Fig. (5.2). One iteration of our algorithm can be described as follows: Input images are acquired with an uncalibrated monocular color camera ① and preprocessed. The results ② are used for matching configurations of the body model with the current image. Several cues

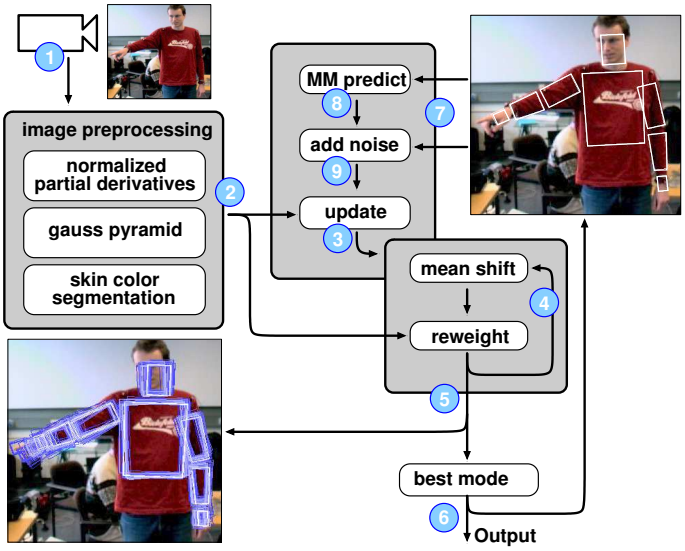


Figure 5.2: *Outline of the algorithm for tracking human upper body motions.* Images are recorded with a monocular camera and preprocessed ①-②. The presented approach employs a combination of particle filter-based pose space exploration ③,⑦-⑨ and mean shift-based pose refinement ④-⑥.

based on intensity and color information each provide a likelihood for the accuracy of the match between the model and the image. The cues are combined ③ to form a probability distribution which is further explored in multiple iterations ④ of the mean shift algorithm. That leads to the identification of different modes ⑤ of the underlying probability distribution from which a single mode ⑥ representing the most likely human body pose is selected as output for subsequent recognition algorithms. This mode is also used as input ⑦ for the next time step of the particle filter. Particles generated from this mode are then - partly after applying a motion model ⑧ - disturbed ⑨ to give an estimation of body configurations for the next time step. Finally, the next picture is acquired ①, preprocessed ②, and the propagated configurations are evaluated ③ on the new image. The outcome is for each image a single pose or posture of the model which is estimated as the best fit. This process is therefore often referred to as pose estimation. Collecting these poses over time leads to a sequence of poses which - depending on the needs of the application - can also be interpreted as trajectories of individual body parts, e.g. the trajectory of the right hand.

5.3 Modeling the Appearence of Humans

The basic idea of model based posture estimation is to use a model of the person to be observed that allows us to inference the person's posture given the current image. The presented algorithm is based on such a model of the human as a reference to find its pose in the image. This model describes the physiology of the human body, its shape and its ability to move, as well as its appearance.

By bringing the model in a position that fits as good as possible with the image of the human, the pose can be determined. Tracking a person or individual body parts in the 2D camera image can be achieved by relatively simple means, but such approaches only provide a position in the image and no depth information.

Using a 3D model, however, enables us to estimate distances even from a 2D image. We

humans do that all day, for instance by knowing that objects further away also appear smaller. Also many optical illusions rely on this principle. But size is only one of many features to judge the distances of a person's individual body parts. As an example, if you see a the hand of a person in front of his chest, you can judge its distance from the torso by looking at the angle how far the ellbow is bended. For a hand far away from the torso the ellbow needs to be almost fully stretched out, for the hand touching the chest the arm needs to be bended. Such information is usually well visibe in the image but it can only be exploited by utilizing a three-dimensional model. The model can also help to constrain the space of allowed postures to those a human is really able to perform. For instance, a pose where the hand is positioned inside the torso can be rejected. In the same way, unnatural or unlikely poses, e.g. the hands touching on the back can be rejected. This reduces the search space and can also help to resolve ambiguities.

An exact pose estimation can only be assumed under the premise that the model exactly reprocudes the properties of the observed human. At the same time, the model must generalize well enough to also bear unpredictable variations to the visual appearance that can, for example, emerge from changing lighting conditions and wrinkling clothes.

The presented algorithm is based on an articulated 3D body model consisting of cylinders with ellipsoid cross sections. With a known camera geometry the 3D body model is back-projected into the image plane using a pinhole-camera model. This yields an approximate representation of the 3D body model consisting of a 2D polygon in the image plane for each limb. The anticipated visual appearance of each limb, in turn, is coded in the form of visual cues - also called filters - that describe independent features of the limb, for instance the edges, the overall color and if it resembles skin. These cues each calculate a likelihood of how well the expected feature is represented in the image. Combining multiple likelihoods per limb and the likelihoods of all the limbs for the whole body yields a final likelihood for the model in the given pose to fit to the visual impression - the higher the likelihood, the better the fit.

The image processing described in this chapter are mainly carried out using the IceWing modular vision framework [102] in combination with routines from the OpenCV vision library [78]. Communication between different instances and modules has been realized using the XCF communication framework [172]

The differnent parts needed to calculate the body model likelihood – the model itself, the image cues, and the rules to combine the filter answers to a model likelihood – are detailed in the following.

5.3.1 Articulated 3D Body Model

Computational models of humans are usually employed to obtain a numerical description of the body and its motions. The list of disciplines interested in aspects of human movements is quite long; it includes athletic coaches, orthopedic surgeons but also sports equipment designers and engineers for automobiles and aircraft, to name just a few. Their interest lies in modeling the biological and physical principles that are common for all humans. Besides the physical appearance that often also includes the

kinematics of the body and the work of muscles and energy transfer throughout the skeletal structure. As Winter [171] declares, a synthesis of human movement incorporating external forces can be achieved using inverse dynamics, for instance to model the performance of an athlete.

The numerical representation of the human body is what we usually call a *body model*. There exist various approaches to implement such a model, the most common are structures of geometric components for the individual body parts that are more or less tightly coupled to represent the kinematic structure of the human body.

An example for such an abstract representation is the work of the Humanoid Animation Working Group [59], who presented the H-Anim specification, an abstract representation for modeling three dimensional human figures. They have been driven by the fact that for a huge number of different software packages for motion capture, 3D modeling and animation exist, but they lack a standardized skeletal system to exchange information. Simplified versions of such a model are most commonly encountered in the field of human body tracking. In 1964, Hanavan presented a segment model [77], cf. Fig. (2.8) that represents the body segments with cylinders and cones. Segment models require only few simple anthropometric measurements, e.g., segment lenths and circumferences. These can be measured individually for each person to be analyzed or they can be chosen from a generic model representing the mean measures of a human. A similar model but with more detail has been employed by Sminchisescu [147], who modeled the body parts by superquadric ellipsoids, see also Fig. (2.6). Other approaches aim at even further simplified representations that incorporate the connections between the different body parts but evade an explicit 3D model as presented by Ramanan [126], cf. Fig. (2.5).

All of these models serve the purpose to enable the application of analytical methods [152] as they offer a unique representation and allow to draw conclusions about the impacting forces. An example would be to analyze gait patterns including the ground reaction forces and the variations for a single human and between multiple subjects.

To represent the appearance of the human body we use an articulated 3D body model consisting of cylinders with ellipsoid cross sections as shown in 5.3(c). The model is organized as a tree structure, beginning with the torso as the root limb. Attached to the torso are the head and the upper arms, then following the lower arms and finally the hands. The limbs are connected among each other by joints, with each having individual joint angle limits which model the physical constraints of the human body. The overall structure of the model is defined by the offsets from one joint to the next. Together, they form a kinematic chain, similar to the skeletal structure of the human, see Fig. (5.3(a)). Motions in the joints are restricted to three degrees of freedom (DOF) in the shoulder $O_{R,L}$ and one DOF for the ellbow $U_{R,L}$. The head and the hand are fixed with respect to the preceding limb. Together with the position $\boldsymbol{t}_T$ and orientation $\boldsymbol{\alpha}_T$ of the torso this results in a upper body model with 14 DOF to be estimated during tracking. A posture of the model at time t is therefore fully described by its 14-dimensional state vector

$$\underline{x}_t = \left[\varphi_T, \vartheta_T, \psi_T, x_T, y_T, z_T, \varphi_{U_R}, \vartheta_{U_R}, \psi_{U_R}, \varphi_{L_R}, \varphi_{U_L}, \vartheta_{U_L}, \psi_{U_L}, \varphi_{L_L}\right]^T \tag{5.1}$$

with the following abbreviations for the limbs: T: torso, H: head, U_R: upper right arm, L_R: lower right arm, U_L: upper left arm, L_R: lower left arm, H_R: right hand, and H_L: left hand. The vector $\underline{x}_t$ spans the space of all possible configurations of the body model. Estimating a pose means estimating the single point in the pose space that resembles the optimum solution to the optimization problem.

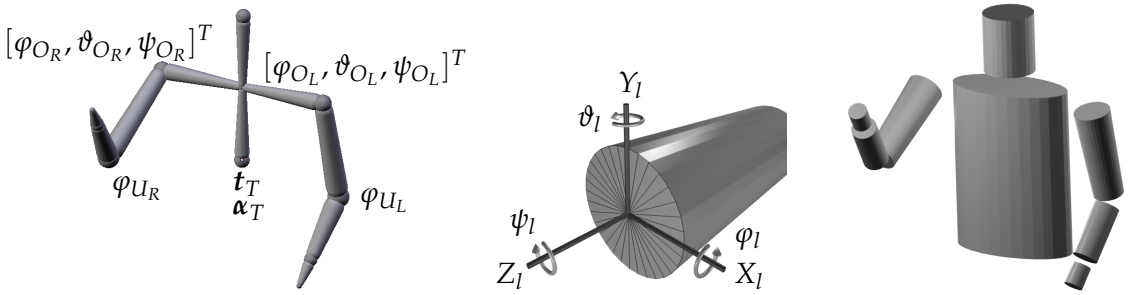


Figure 5.3: *Articulated model of the human's upper body.* (a) The underlying kinematic chain connects the individual limbs, similar to the skeletal structure of the human. (b) Limbs are represented by a truncated cone with a maximum of three rotational degrees of freedom (DOF) in the joint. The shape of the cone defines the appearance of the model, thus resembling the flesh of the model. (c) The resulting articulated model with 14 DOF. (Image found in [66])

The truncated cones are attached to the kinematic structure, they resemle the flesh of the body. For the algorithm, they define the locations where the image cues are to be evaluated. The shape of each truncated cone is defined by an offset from the limb origin, the length of the cone and two values for the semi-major and semi-minor axis of each ellipse forming the cap of the cone, see Fig. (5.3(b)).

Camera Model and Projection

Knowing the camera geometry, each point on the surface of a cylinder can be projected into the image. A projection model transfers from the 3D world coordinate system of the body model to the 2D image coordiante system of the camera image, compare also to [51].

A 2D position in image coordinates can be represented by the vector $\underline{i} = [u, v, 1]^T$, giving the row and the column. Similarly, a 3D point in world coordinates is represented by the vector $\underline{w} = [x, y, z, 1]^T$. We use the homogeneous coordinate notation which is commonly used in robotics and computer vision to express the mapping from pixel to world coordinates as

$$\underline{i} = \underline{\boldsymbol{P}}\underline{w} \tag{5.2}$$

with the 3×4 projection matrix $\underline{\boldsymbol{P}} = [\underline{\boldsymbol{A}}|\underline{\mathbf{0}}_{3\times 1}]$, where

$$\underline{\boldsymbol{A}} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

is the intrinsic camera parameter matrix. Since we assume the images to be rectified we do not need to correct any radial or tangential distortions. Further assuming quadratical pixels, now skew, and u_0 and v_0 being the center point of the image, $\underline{\boldsymbol{A}}$ can be simplified to

$$\underline{\boldsymbol{A}} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

Thus, using Eqn. (5.4), Eqn. (5.2) can be rewritten as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \alpha & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{5.5}$$

giving the direct mapping from world coordinates $\underline{w} = [x, y, z, 1]^T$ to pixel coordinates $\underline{i} = [u, v, 1]^T$.

Furthermore, homogeneous coordinates can be used to easily express affine transformations in the 3D space:

$$\underline{a'} = \underline{\boldsymbol{T}a} \tag{5.6}$$

where

$$\underline{\boldsymbol{T}}_{4\times4} = \begin{bmatrix} \underline{\boldsymbol{R}}_{3\times3} & \underline{\boldsymbol{t}}_{3\times1} \\ \underline{\boldsymbol{0}}_{1\times3} & 1 \end{bmatrix} \tag{5.7}$$

is the homogeneous form of the transformation matrix. It consists of the rotation matrix $\underline{\boldsymbol{R}}$ and the translation vector $\underline{t}$ combined in a single matrix.

The advantage of the homogeneous representation is that affine transformations can be calculated as a matrix operation. As an example, for transforming a point from a limb-local coordinate system of limb l_a into the coordinate system of an adjacent limb l_b we need to know the translation t from one joint to the next and the joint angles ϕ, θ, and ψ, e.g. given in euler ZYX angles. The translation followed by multiple rotations now can easily be calculated.

The rotation matrices are gives as

$$\underline{\boldsymbol{T}}_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

$$\underline{\boldsymbol{T}}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

and

$$\underline{\boldsymbol{T}}_{z,\psi} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

The translation is given by

$$\underline{\boldsymbol{T}}_{\underline{t}} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.11}$$

Matrix multiplication yields the final transformation matrix from limb l_a to limb l_b.

$$\underline{\boldsymbol{T}}_{l_a \to l_b} = \underline{\boldsymbol{T}}_{\underline{t}} \cdot \underline{\boldsymbol{T}}_{z,\psi} \cdot \underline{\boldsymbol{T}}_{y,\theta} \cdot \underline{\boldsymbol{T}}_{x,\phi} \tag{5.12}$$

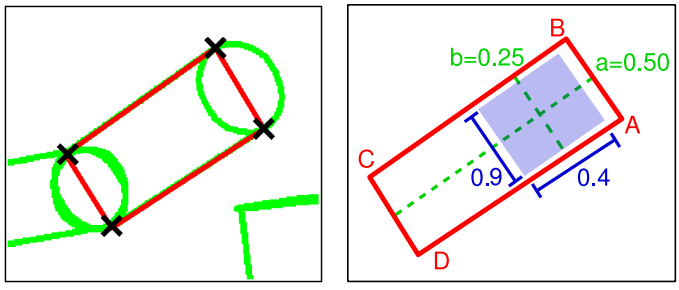
Similarly, a point a_{H_R} given in local coordinates of the right hand limb can be expressed in world coordinates as a_W by multiplying all the transformation matrices that correspond to the joints in the body model hierarchy:

$$a_W = \underline{\boldsymbol{T}}_{\text{world} \to T} \cdot \underline{\boldsymbol{T}}_{T \to U_R} \cdot \underline{\boldsymbol{T}}_{U_R \to L_R} \cdot \underline{\boldsymbol{T}}_{L_R \to H_R} \cdot a_{H_R} \tag{5.13}$$

This representation allows us to transform any point expressed in local coordinates of a limb and given the current pose of the body into the world coordinate system. Consequently, each 3D world point can be projected into the image plane given the camera parameters, see [26] for a more detailed description.

Body Model Projection and Approximation

The image cues described in the following sections are based on evaluating filter responses for defined points on the body model. Although an exact projection of the cylinder surface into the image would be possible, a simplified and much faster to calculate representation is chosen as for each image, several thousands to million feature points are evaluated.



(a) (b)

Figure 5.4: *2D polygon approximation for 3D body model.* (a) A 3D limb cylinder is approximated by a 2D polygon in the image. (b) the position of a samplepoint is defined in a local coordiante system defined on the polygon (green values), regions can be defined similarly (blue).

Using the camera model described above, the 3D body model is back-projected into the image plane. This yields an approximate representation of the 3D body model consisting of a 2D polygon in the image plane for each limb, see Fig. (5.4(a)). A specific point on the limb can then be addressed by giving two coordinates $(a, b) \in [0, 1]$ in the polygon-local 2D coordinate system, see Fig. (5.4(b)). To give an example, the center line of the limb is defined by $(0.5, b)$, the right and left border of the limb are addressed as $(a \in \{0, 1\}, b)$.

The polygon approximation shows the lowest error when the cylinder is observed by the camera from the side. The more the cylinder axis is parallel to the optical axis of the camera, e.g. an arm pointing directly into the camera, the more inaccurate the approximation gets, as the polygon degrades to a single line for frontal views.

5.3.2 The Monocular Challenge

Estimating the pose for a complicated articulated model from monocular observations is a highly ambiguous task. The resulting posterior likelihoods over human pose space are typically multi-modal with a high number of false local maxima. This is due to several reasons, including complicated modeling as well as feature extraction difficulties. The problem of estimating the correct pose of the model for each image can be expressed as finding the one local maximum in the high-dimensional parameter space that fits best the current pose while still obeying all given constraints, e.g. the joint angle limits. It is only feasible if enough data are present, e.g. from using multiple cameras [18] or special sensors like stereo cameras [178] are utilized.

Depending on the number of image cues and their discrimination capabilities, a monocular approach can be successful [141, 147] but the resulting posterior likelihoods over human pose space are typically multi-modal with a high number of false local maxima. Restricting the search space to known motions [103] is a common approach to overcome the highdimensional search problem, but this does not work if the scenario requires arbitrary motions to be allowed. Some motions pose additional problems to monocular approaches, especially motions of body segments in depth, towards or away from the camera. Also axial rotations of limbs are not observable in the image. The ambiguity of body configurations, nonlinearity of observations, and non-observability make the posterior likelihood in the pose space multi-modal and unpredictable.

In our approach, we counter these difficulties with a restricted scenario and by focus on the estimation of the upper body pose only. Motions of body segments in depth, towards or away from the camera, cannot be tracked precisely with a monocular camera, but as long as tracking is not lost, a very rough estimate of the body configuration is still available.

The likelihood for a specific pose is obtained by fusing the filter responses for all cues and all limbs and transferring them into likelihoods with a cue-specific weighting function representing the expected characteristics of the cue. Combining multiple cues makes the estimation more robust against local disturbances but typically also results in a high number of false local maxima in the parameter space. When evolving over time, new modes often emerge from regions with low probability while existing modes degenerate or even vanish. To track the correct pose of the human, the structure of the high-dimensional probability density has to be efficiently exploited, taking into account the constraints and the dynamics of the model. The utilized kernel particle filter [136] propagates such multi-modal distributions and provides a probabilistic search for the best matching body configuration.

The discussed KPF approach has shown to generalize well for the pose reconstruction

and tracking challenges that are present in this thesis. The principle of KPF has been specialized to the problem of assembly pose localization as presented by Stößel [155] with the Extended Kernel Particle Filter. He exploits the availability of exact 3D CAD models of the parts to localize that have to be known before the actual optimization starts but with that he is able to achieve a high spatial accuracy during the localization process. Stößel applies a hierarchical partitioning of the parameter space which eases the search process, but this technique builds on the assumption that each part of the object can be exactly localized without knowing the remaining parts' locations. Not only that this assumption does not hold for human body tracking, even more, the presented monocular approach is designed to exploit for each limb the data that is available and determine the body configuration from the interdependency of the limbs.

5.3.3 Image Cues for Body Pose Tracking

With the 3D model back-projected into the image we have a proper representation at hand to calculate the likelihood for a given pose of the model using multiple image cues. All cues rely on the 2D polygon representation of the model and select the position of their feature points according to the polygon and the type of the cue.

For the indiviudual body parts - in the following they are always referred to as *limbs* - each cue independently calculates a filter response, while not every cue must be used for each limb. The arm, for instance, is found best by combining the edge, ridge and the mean color cue, while the hand is found best using the skin color cue only. For a single limb, all filter responses are first reansfered into a cue likelihood and then combined into a limb likelihood.

The estimation of the likelihood for a given pose is done by combining the likelihoods from each limb $l = 1, \ldots, L$ of the body model using up to four image cues $c \in \{E, R, C, S\}$ or filters, where E stands for the edge cue, R is the ridge cue, C is the mean color cue, and S denotes the skin color cue. Combining multiple cues makes the estimation more robust against local disturbances while still producing good likelihoods for the correct pose.

Some parts of the image processing can - bust must not necessarily - be accomplised before the actual cue calculation starts as a kind of preprocessing. That includes the generation of a gaussian pyramid from the input image and the calculation of partial derivatives for each level of the pyramid. Depending on the given scenario, the system setup can be altered to either benefit from multiple cpu cores or to use more efficient but still serial processing for a single cpu core. In the first case, the calculation of several image features like the gaussian pyramid, the local partial derivatives, a Susan edge filter followed by a chamfer distance transform and the skin color segmentation is performed as a preprocessing step in a separate instance of the image processing framework. As multiple core cpu's are already common in modern cpu architectures the additional instance can be run on a different core and communicates with the main instance via methods provided by the image processing framework. But as it can not be known in before where the model will be positioned in the image and thus which

pixels will be used as feature points, the preprocessing must always be done for the whole image. The situation gets different when the calculation of the derivatives of the gaussian pyramid is done within the cue calculation itself. A dynamic programming approach is then used to only calculate those filter responses actually needed for the current limb. Also multiple requests for the same point do not need to be calculated again. For the discussion of the image cues we simply assume that the filter responses are avaiable the one or the other way.

In the remainder of this section, the image preprocessing and the calculation of the image cues for a given pose of the model are detailed. The combination of the individual filter responses into a pose likelihood is described thereafter.

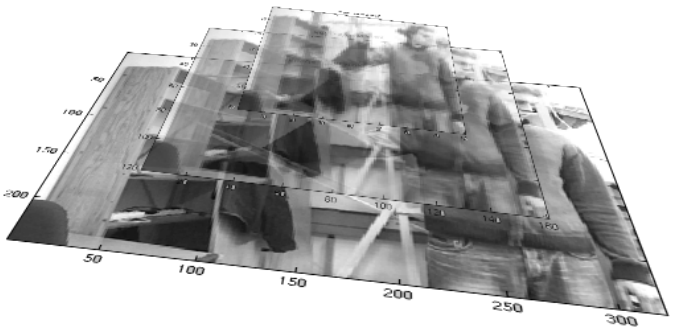


Figure 5.5: *Gaussian image pyramid.* The original image is smoothed and subsampled multiple times. The resulting set of images is a scalespace representation where in each layer resolution is halved.

Gaussian Image Pyramid

A gaussian image pyramid is generated by smoothing the image with a gaussian filter and then subsampling it to half the size of the original image, see Fig. (5.5). This process is repeated multiple times, resulting in a stack of images with each level having a lower resolution as the preceding image. With the smoothing process unwanted aliasing artefacts from subsampling can be avoided.

The gaussian pyramid is a scale-space representation of the image. As we observe people and objects in the real world we can not predict at which size they will appear in the image, but some of the cues are sensitive to the size of the observed features. Using the gaussian pyramid the algorithm is able to choose the layer of the pyramid for which the resolution and the object size match best the needs for the cue.

Image Gradient from Partial Derivatives

An edge in an image can be defined as an abrupt change in the intensity. By interpreting the image as a two-dimensional function $f(x, y)$, such changes can be easily detected by looking at the gradient of the function. Edges will point out as the local extrema of the image gradient.

The gradient of a scalar function $f(x, y)$ is defined by its partial derivatives:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right). \tag{5.14}$$

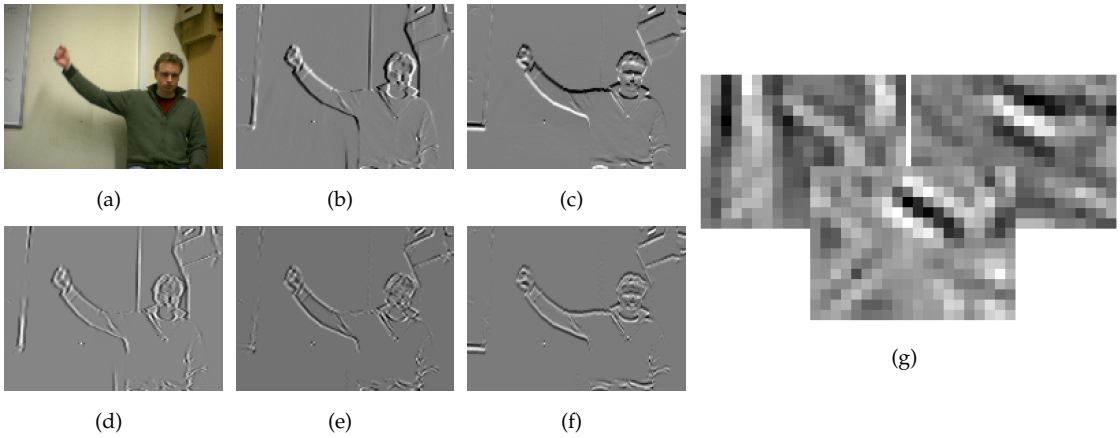


Figure 5.6: *Partial derivatives interpreted as image gradient.* (a) Original image, (b) $\frac{\partial f}{\partial x}$, first partial derivative in x direction, computed at the original image resolution (as are (c)-(f)) (c) $\frac{\partial f}{\partial y}$, first partial derivative in y direction (d) $\frac{\partial^2 f}{\partial^2 xx}$, second partial derivative in xx direction (e) $\frac{\partial^2 f}{\partial x \partial y}$, second partial derivative in xy direction (f) $\frac{\partial^2 f}{\partial^2 y}$, second partial derivative in yy direction (g) second partial derivatives computed at level 4 of the gaussian pyramid.

These derivatives are relatively easy to calculate in the image, as they can be approximated as the forward-backward-difference for a specific pixel.

Skin Color Segmentation

The color of human skin is a very specific feature which on the one hand shows a great variability from person to person and also for different skin regions of the same person, on the other hand it is often well separable from the background. Exceptions are some wood colors as seen in furniture or beige colored walls. But most of the time a skin color segmentation can give a good clue where to look for the hands and the face of a person. Obviously, wearing gloves or looking away from the camera renders this approach useless, but since it is only one of many features we can assume it to be quite robust and live with temporary occlusions.

Here, we applied the skin color segmentation approach proposed by Fritsch [50], which is based on a two-step mechanism as follows. First, the image to be segmented is transformed into the RG color space. The RG color space has the advantage to be illumination invariant. That means that pixels are only classified according to their specific color and not according to their brightness, which is an important feature when dealing with real-world images. Only those pixels with RG color values inside a restricted region, the so-called skinlocus, cf. Fig. (5.7(a)), are processed further. The skinlocus is shaped such that all possible skin colors lie inside the region. This relatively simple pre-segmentation is fast to compute and significantly reduces the amount of data, cf.Fig. (5.7(b)) that needs to be classified with the next step of the approach. In this second step the remaining pixels are classified as being skin or non-skin with a mixture-of-gaussians classifier. It also works within the RG color space and calculates a skin

(a) (b) (c)

Figure 5.7: *Skin color segmentation in the RG color space.* (a) Skinlocus for pre-segmentation, displayed as a region in the RG color space, the gray pixels inside are a histogram of the classified color values, (b) segmentation result after applying the skinlocus, (c) segmentation result after applying the mixture-of-gaussians classifier.

color likelihood by the sum of multiple gaussian weighting functions. See Fig. (5.7(c)) for the final segmentation result.

At this point we would like to point out that the skin color segmentation and the corresponding cue to be described later can serve as an example on how to integrate results from any other segmentation process into the tracking framework, e.g. coming from a texture analysis module. The only requirements are that the feature can be localized to a specific region on the body model, for instance the area of the lower right arm, and that the module can be evaluated in the form of an image filter, which means that it must provide a reliable measure on how well the image region corresponds to the expected appearance.

Edge Cue

The first cue to be presented is the edge cue, which has been inspired by the work of Hedvig Sidenbladh [141]. It uses the first partial derivatives in X and Y direction which are sensitive to strong changes in contrast. The idea is that at the borders of the body model, a strong change in image intensity - which we usually call an edge - can be expected, as there will be the transition from the foreground, the human, to the background.

For recognizing human body parts the presence of edges is most important, not their magnitude. Therefore all images with partial derivatives have been scaled with a nonlinear normalization function. This function has been used to smooth low magnitude edges stemming from textured backgrounds and to emphasize stronger ones, see Fig. (5.6(b)) and Fig. (5.6(c)).

The edge cue provides an accurate match for the position of a limb l by comparing for a point $\underline{z}$ on the border of the limb the estimated limb angle α, which has been obtained from the 3D model, with the angle of the edge gradient $[\partial_x(\underline{z}), \partial_y(\underline{z})]^T$ measured in the image. This is done for $m = 1, \ldots, M_E$ feature points $\underline{z}^{(m)}$ positioned equally spaced on the limb boundaries, see Fig. (5.8) for the positioning of the samplepoints. Let

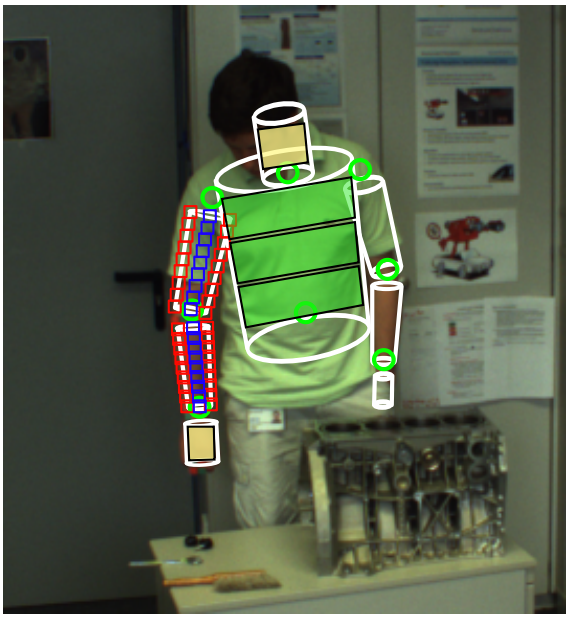


Figure 5.8: *Positioning of the features on the body model.* Regions and sample point locations for the color-based and intensity-based cues, respectively. (green) Regions for the mean color cue, (yellow) regions for the skin color cue, (red) sample points for the edge cue, (blue) sample points for the ridge cue. For a clearer display, the number of features in the figure is reduced.

$\underline{z}^{(m)} = [x, y]^T$ denote the location of one pixel in the image plane. The response of the edge filter is:

$$f_E^{(l)}(\underline{z}^{(m)}, \alpha) = \partial_y(\underline{z}^{(m)})\cos(\alpha) - \partial_x(\underline{z}^{(m)})\sin(\alpha). \tag{5.15}$$

As the filter response is not static but depends on the angle of the border of the limb, it is called a steerable filter. The filter response for the whole limb is calculated by averaging over the M_E feature points, with typical values for $M_E = 20$.

$$\bar{f}_E^{(l)} = \frac{1}{M_E}\sum_{m=1}^{M_E} f_E^{(l)}(\underline{z}^{(m)}). \tag{5.16}$$

All the cues described here generate a separate filter response for each limb they are applied to. To be able to later fuse the cues, the filter responses are converted into likelihoods using the following Gaussian weighting function:

$$p(c, l) = \exp\left(-\frac{(\bar{f}_c^{(l)})^2}{2\,\sigma_c^2}\right). \tag{5.17}$$

This transfer function maps from the range of all possible outcomes of the filter to the uniform likelihood domain $p(c, l) \in [0, 1]$ for any of the different cues $c \in \{E, R, C, S, H\}$. The standard deviation σ_c is derived from the variability of the responses of each utilized cue. It can also be used as a parameter to control the behavior of the cue in the context of fusing it with other cues. A low σ_c makes the cue transfer function very picky, even small changes of the filter response will lead to significantly lower likelihood values. Choosing a large σ_c then again means a wider variance to be allowed which makes the cue more generous. Concrete values for the sigmas always have to be chosen in relation to the values of the expected filter responses. For the following evaluation of the edge cue we have chosen $\sigma_E = 0.1$.

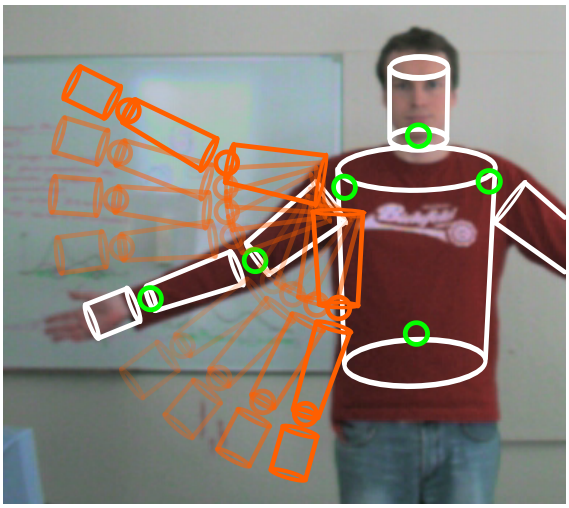
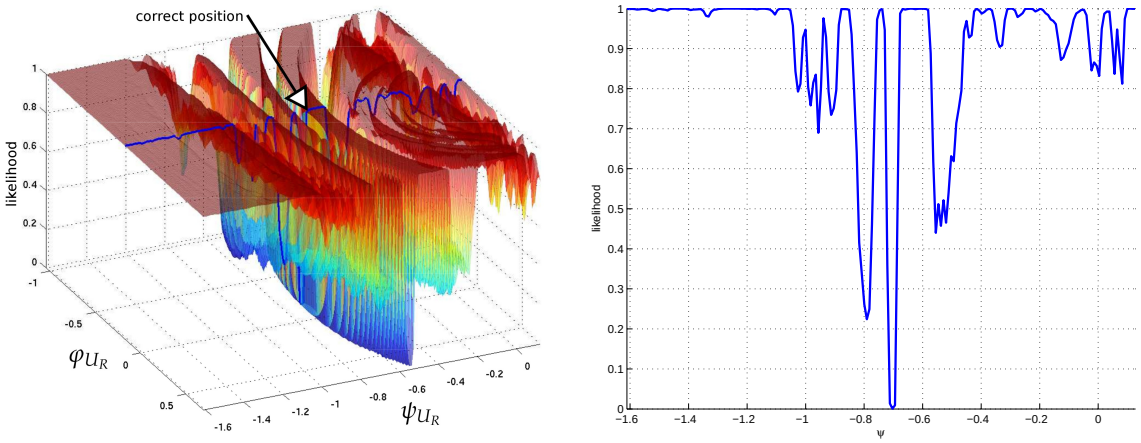


Figure 5.9: *Body model positioning for cue evaluation.* From an initial (correct) position, the right arm is moved up and down using the shoulder joint ψ_{U_R}, exemplaric body model poses are shown in red. Additionally, the arm is moved to the front and back using the φ_{U_R} joint (not shown here).

In order to understand the principle of this cue and to compare the results to the other cues, it has been applied to a range different postures for a sample image, see Fig. (5.9). For the evaluation, the only cue activated is the edge cue on the lower right arm of the body model. For this evaluation, the body model has been placed in the correct position at approximately and then the arm has been moved using two joints of the shoulder. The first angle, ψ_{U_R}, is responsible for moving the arm up and down, the second one, φ_{U_R}, moves the arm back and forth. Together, they span a 2D subspace of the full 14D parameter space used during optimization. Fig. (5.10) shows a plot of the resulting limb likelihood for different poses of the model, the optimal position is at $\varphi_{U_R} = -0.16$, $\psi_{U_R} = -0.75$ approximately. Fig. (5.10(a)) displays the likelihood surface for iterating both parameters, the correct position is denoted by the arrow. Fig. (5.10(b)) is generated by slicing Fig. (5.10(a)) at the blue line which runs directly through the correct position for φ_{U_R}. This scheme is repeated for the remaining cues for a better comparability.



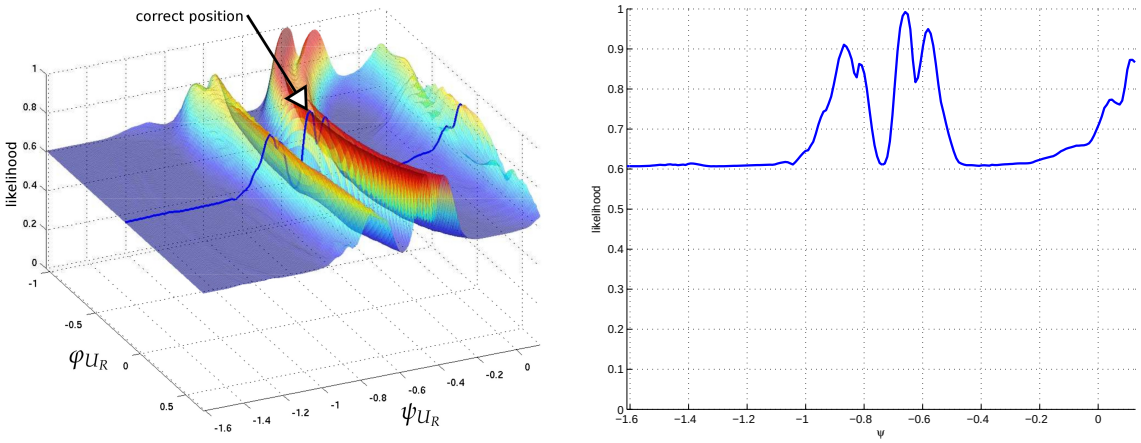
(a) (b)

Figure 5.10: *Edge cue likelihood.* Calculated by averaging over 10 feature points, $\sigma_E = 0.1$, and without importance reweighting $\lambda_E = 0$: (a) varying angles φ_{U_R} and ψ_{U_R} of the upper arm, (b) varying angle ψ_{U_R} only.

A high likelihood at the correct position and low values for any other configuration are the desired characteristics for an ideal cue. In the plot, such a behaviour would show as a single sharp peak at the correct position. It is obvious that the cue provides a high likelihood for the correct position but unfortunately it also does for many other configurations. The likelihood values sharply decline for moving the arm vertically but much slower when moved backwards or forwards. The reason is that vertical movements of model easily create a misalignment with the person's arm, whereas for movements in the depth direction, the projection simply gets distorted but still in place overall. As a result, the edge cue seperates well the arm's correct position from slight variations, but it does not cope well for motions in depth direction. Even worse, it exhibits extensive false maxima for other regions in the image as it gets confused easily when edges are present.

Ridge Cue

The design of the ridge cue has also been inspired by [141]. It is utilized to find elongated structures of a specified thickness - called ridges - which is excellent for finding a person's arms in the image. Here, the second partial derivatives are used, see Fig. (5.6(d)), Fig. (5.6(e)) and Fig. (5.6(f)). As the cue depends on the size of the limbs in the image, it only provides appropriate results if the observed limb is in a particular distance to the camera. To overcome this restriction, we select a resolution level μ in the Gaussian image pyramid based on the current distance of the limb to the camera. In the image from this pyramid level, the limb has the correct size to be recognized by the ridge cue, cf. Fig. (5.6(g)). Similar to the edge cue, a normalized representation for the derivatives is utilized.



(a) (b)

Figure 5.11: *Ridge cue likelihood.* Calculated by averaging over 10 feature points, $\sigma_R = 1.0$, and without importance reweighting: (a) varying angles φ_{U_R} and ψ_{U_R} of the upper arm, (b) varying angle ψ_{U_R} only.

For finding ridges, the cue suppresses point-like edge features by searching for elongated edge structures parallel to the expected limb angle α and missing edges in perpendicular direction. This is achieved by evaluating the normalized second partial deriva-

tives $[\partial_{xx}^{(\mu)}(\underline{z}), \partial_{xy}^{(\mu)}(\underline{z}), \partial_{yy}^{(\mu)}(\underline{z})]^T$ at $m = 1, \ldots, M_R$ feature points $\underline{z}^{(m)}$ equally distributed on the main limb axis, see Fig. (5.8).

$$\begin{aligned} f_R^{(l)}(\underline{z}, \alpha) &= \left| \sin(\alpha)^2 \, \partial_{xx}^{(\mu)}(\underline{z}) + \cos(\alpha)^2 \, \partial_{yy}^{(\mu)}(\underline{z}) - 2\sin(\alpha)\,\cos(\alpha)\, \partial_{xy}^{(\mu)}(\underline{z}) \right| - \\ &\quad \left| \cos(\alpha)^2 \, \partial_{xx}^{(\mu)}(\underline{z}) + \sin(\alpha)^2 \, \partial_{yy}^{(\mu)}(\underline{z}) + 2\sin(\alpha)\,\cos(\alpha)\, \partial_{xy}^{(\mu)}(\underline{z}) \right| . \end{aligned} \tag{5.18}$$

Similar to the edge cue, the ridge filter response is dependant from the asked limb angle, it is therefore also a steerable filter. The filter response for the whole limb l is computed by averaging over all M_R feature points, with typical values for $M_R = 20$.

$$\bar{f}_R^{(l)} = \frac{1}{M_R} \sum_{m=1}^{M_R} f_R^{(l)}(\underline{z}^{(m)}, \alpha) \tag{5.19}$$

The filter responses are transfered into likelihoods using Eqn. (5.17), here using $\sigma_R = 1.0$.

For the evaluation, the only cue activated is the ridge cue on the lower right arm of the body model. Looking at Fig. (5.11) shows us that the ridge cue gives a coarser estimate of the limb position than the edge cue, but produces less false maxima. With its smooth slopes it is better suited to guide the search process towards the correct position than the edge cue with the sharp peaks. Both the smoothness and the slight inaccuracy can – at least partly – be explained by the use of the higher levels of the gaussian pyramid.

Mean Color Cue

Color can be a very meaningful feature, in particular if the color of the clothing significantly differs from the background. The mean color cue models the appearance of a limb using an learned color model. Instead of looking at the whole limb at a time, the algorithm uses B_l regions on each limb l, see Fig. (5.8). Each region is defined by a square in the local coordinate system of the back-projected limb, the number of regions B_l and their positions are chosen on the basis of the limb type. Using an initial image and a corresponding body pose a mean color value $\bar{C}_t(\mathcal{Z}^{(b,l)})$ is learned for each region from all pixels within the region $\mathcal{Z}^{(b,l)} = \{\underline{z}_1^{(b,l)}, \ldots, \underline{z}_m^{(b,l)}\}$.

To calculate the filter response, the mean color $C_t(\underline{z}^{(b,l)})$ of each polygon b at position $\underline{z}^{(b,l)}$ is compared to the learned mean color $\bar{C}_t(\mathcal{Z}^{(b,l)})$ of this polygon on limb l using the L2 norm in the utilized RGB color space:

$$f_C^{(b,l)} = \begin{cases} \left\| C_t(\underline{z}^{(b,l)}) \; - \; \bar{C}_{t-1}^{(b,l)} \right\| - \rho_M & \text{if } \left\| C_t(\underline{z}^{(b,l)}) \; - \; \bar{C}_{t-1}^{(b,l)} \right\| > \rho_M \\ 0 & \textit{otherwise.} \end{cases} \tag{5.20}$$

where ρ_M defines a minimum error threshold that first has to be exceeded before an error value is generated. We call this threshold the *mean plateau*, as it appears like a flat region in the surface of the error function. It has been introduced to neglect the small unpredictable errors originating from the camera noise that could distract the optimization.

The filter response for the limb l is calculated by taking all regions into account:

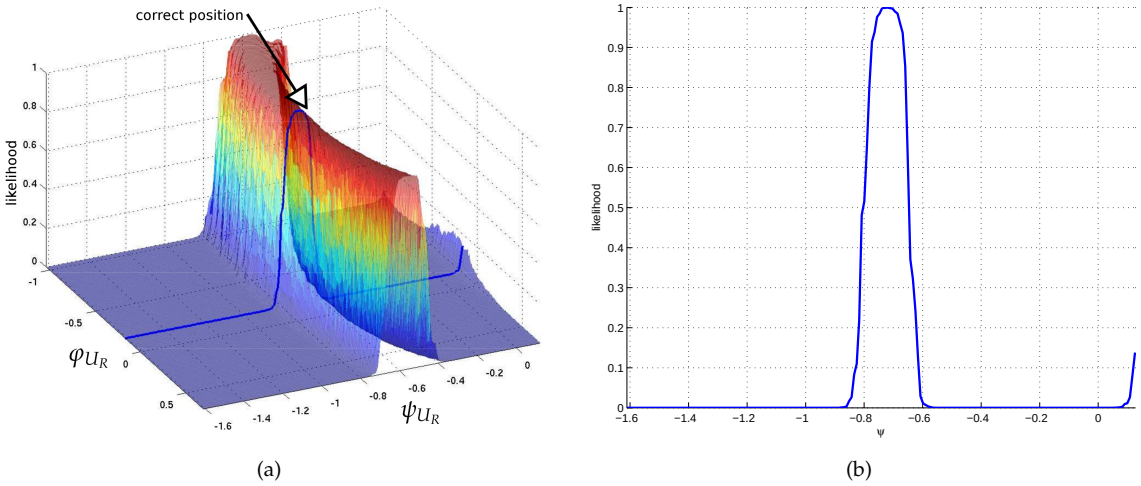
$$\bar{f}_C^{(l)} = \frac{1}{B_l} \sum_{b=1}^{B_l} f_C^{(b,l)}. \tag{5.21}$$

The cue likelihood is again determined using the transfer function Eqn. (5.17). For the following evaluation we used $\sigma_M = 35.0$ and $\rho_M = 0.0$ Note that this value for the variance is significantly higher than those of the other cues. The reason is that the filter generates much larger responses as these resemble distances in the RGB color cube. Here, the maximum filter response is $d = \sqrt{3} \cdot 255 \approx 441,7$.

To deal with varying illumination conditions we adapt the current mean color model according to those values $\hat{C}_{t-1}(\underline{z}^{(b,l)})$ that have been extracted on the basis of the best pose in the last time-step $t-1$:

$$\bar{C}_{t-1}^{(b,l)} = \beta \cdot \hat{C}_{t-1}(\underline{z}^{(b,l)}) + (1-\beta) \cdot \bar{C}_{t-2}^{(b,l)} \tag{5.22}$$

where β is an adaptation factor, typically chosen to be small, e.g. β=[0.01,0.05], to make only a small adaption for each image. This adaption is not sensitive to the accuracy of the tracking results, therefore a too fast adaption would mean to degrade the color model. In case of a false tracking, the limb could for instance be slightly off the body and the background color would be learned.



(a) (b)

Figure 5.12: *Mean color cue likelihood.* Calculated utilizing three patches in the lowest level of the gaussian pyramid, $\sigma_M = 35.0$, mean plateau $\rho_M = 0.0$, and without importance reweighting $\lambda_M = 0$: (a) varying angles φ_{U_R} and ψ_{U_R} of the upper arm, (b) varying angle ψ_{U_R} only.

For the evaluation, the only cue activated is the mean cue on the lower right arm of the body model, using three neighboring regions. The mean cue reliably finds the coarse limb position as color is a very discriminative cue, see Fig. (5.12). It is robust to false maxima as long as the learned colors do not appear in the background. Additionally, it is generous concerning minor disturbances and therefore in general very robust. Unfortunately, it very much depends on the appearance of the scene, e.g. the clothing of the person to be tracked. A uniformly colored shirt means, for example that the arms

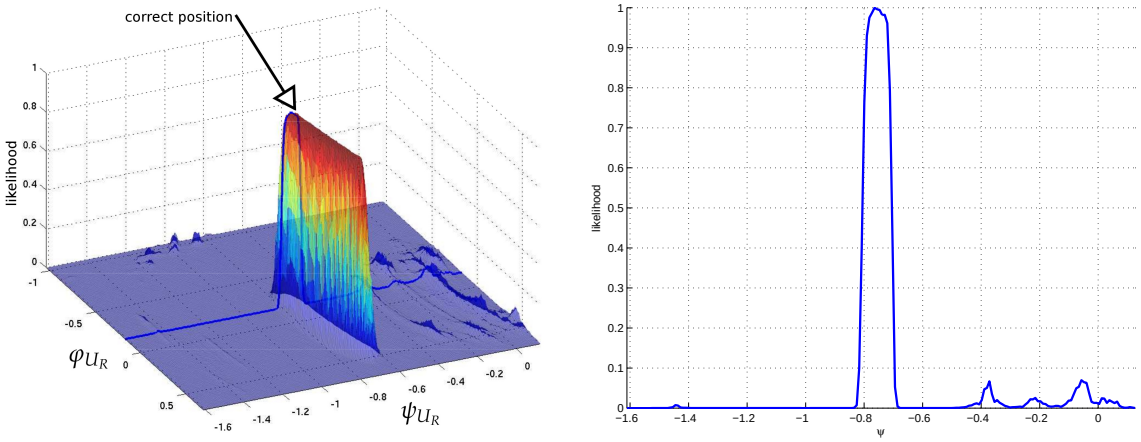
and the torso can not be told apart. For textured clothing, a mean value is not very informative and the cue is a bad choice.

Skin Color Cue

The skin color cue is based on the segmentation image presented in Sec. (5.3.3). Assuming that the only skin colored regions originate from the person to be tracked, this cue allows to find the positions of the hands and the head. Similar to the mean color cue, it uses one or more regions defined on a limb. All M_S pixels $\mathcal{Z}^{(b,l)} = \{\underline{z}_1^{(b,l)}, \ldots, \underline{z}_m^{(b,l)}\}$ originating from a region b are analyzed. The filter response for the limb l is calculated using the ratio of pixels being classified as skin or non-skin:

$$\bar{f}_S^{(l)} = \frac{1}{M_S} \sum_{m=1}^{M_S} \rho(\underline{z}_m^{(b,l)}) \tag{5.23}$$

where $\rho(\underline{z}_m^{(b,l)}) = 1$ means that the pixel has been classified as being skin colored and $\rho(\underline{z}_m^{(b,l)}) = 0$ if not.



(a) (b)

Figure 5.13: *Skin color cue likelihood.* Calculated utilizing a single patch on the hand limb, $\sigma_S = 0.1$, and without importance reweighting $\lambda_S = 0$: (a) varying angles φ_{U_R} and ψ_{U_R} of the upper arm, (b) varying angle ψ_{U_R} only.

The cue likelihood is again determined using the transfer function Eqn. (5.17), we used $\sigma_M = 35.0$, $\rho_M = 0.0$ and all pixels from the hand limb region for the cue evaluation.

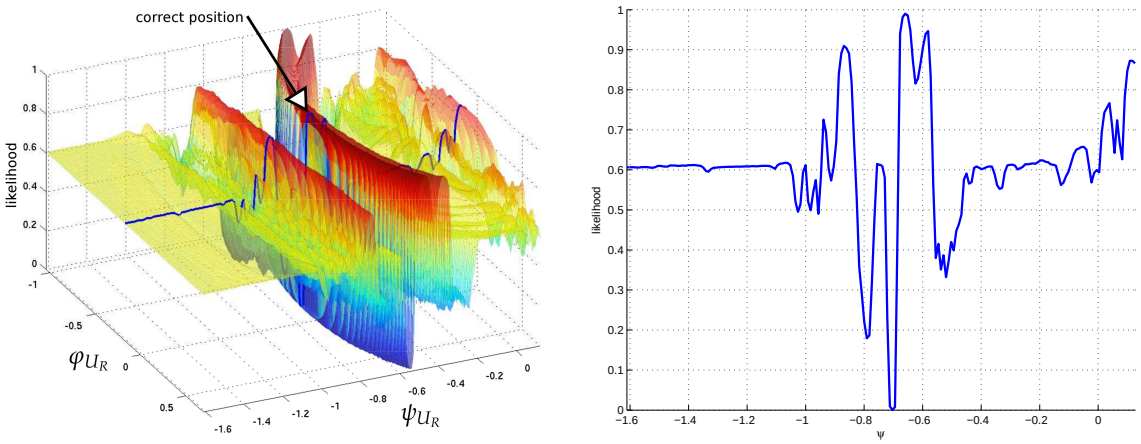
Note that for mapping the filter responses to the likelihood domain using the transfer function is only an option, not a must. The filter responses of the skin cue are already in the correct domain that is $\bar{f}_S^{(l)} \in [0,1]$, as it resembles the ratio between skin and non-skin pixels. We nevertheless apply Eqn. (5.17) here with a $\sigma_S = 0.1$, as it allows us to better control the sensitivity of the cue. This is – at least up to a certain degree – also true for the other cues, as using a gaussian transfer function is only a proposal. If it was only for the transfer to the correct domain, a simple linear or second order scaling

with an offset would also have been sufficient, but during our experiments, the use of the transfer function to customize the behavior of the cues has proven to be an effective and intuitive means.

The cue likelihood displayed in Fig. (5.13) shows the best localized and compact maximum. It even allows some reasoning about the distance of the hand limb which is mainly an effect of the forward kinematics of the body model. Unlike the arm used for the mean color cue, the hand used to find skin color is a very small region. Moving the arm back and forth means a circular movement in euclidian space which very soon pushes the small hand limb outside the likewise small hand in the image. This example shows that an accurate skin segmentation can be very beneficial for tracking a person's movements. The advantage of the proposed system is, however that it does not fail completely if the skin color segmentation is temporarily inadequate or the hand is occluded from time to time, the system will rather benefit from accurate data if available and otherwise be content with the other cues still providing enough data to keep up a robust tracking.

5.3.4 Body Pose Observation Model

The presented results from the cue evaluations are promising and give proof of the fact that a pose of the body model can be rated based on different image features. But at the same time, the discussion made us question the reliability of each individual cue. It is easy to imagine situations in which some cues may not show satisfactory performance. That calls for a combination of multiple cues to enhance their accuracy and their reliability.



(a) (b)

Figure 5.14: *Combined edge and ridge cue likelihood.* Calculated by multiplying the two individual cue likelihoods. (a) varying angles φ_{U_R} and ψ_{U_R} of the upper arm, (b) varying angle ψ_{U_R} only.

As an example, let us have a closer look again at the edge and the ridge cue. To briefly recall the qualities, the edge cue is very exact but produces many false maxima, the ridge cue is less exact but produces less false maxima. Multiplying the likelihoods of

both cues yields a new combined likelihood which much better resembles the desired characteristics of a cue to be used for an optimization process. Fig. (5.14) shows a clear maximum at the correct position and much less disturbances from false detections.

For rating a pose of the body model given the current image the responses from all cues are combined into a compound likelihood:

$$p(\underline{y}_t|\underline{x}_t) = \prod_{c\in\{E,R,M,S,H\}} \prod_{l=1}^{L} p(c,l)^{\rho_c^{(l)}} \tag{5.24}$$

where

$$\rho_c^{(l)} = \delta_c^{(l)} \frac{1}{\lambda_c N_c^{(l)}} \tag{5.25}$$

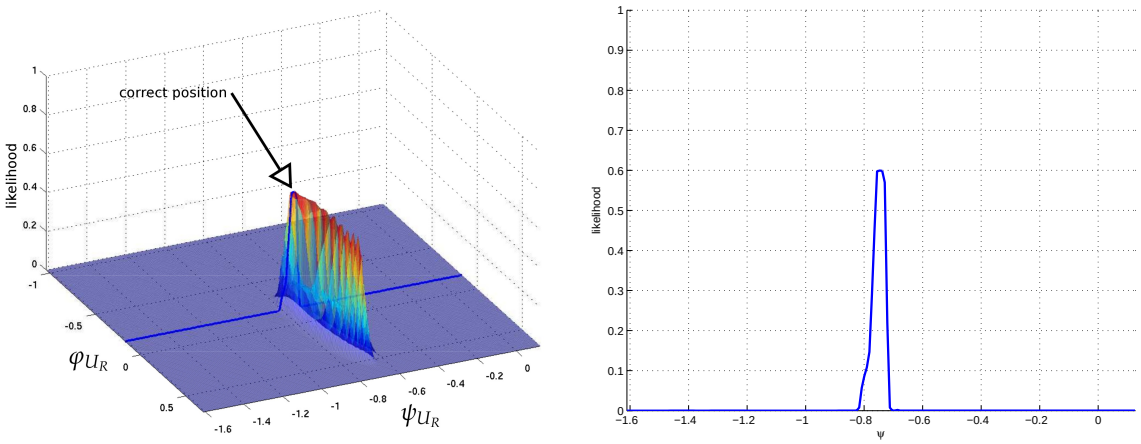
is a limb- and cue-specific balancing factor. Eqn. (5.24) resembles the probability for a given pose of the model $\underline{x}_t$ the observation $\underline{y}_t$ can be made, where the latter is defined by the image cues at time t. The fact that a simple multiplication can effectively be used to combine results from very different image processing techniques is made possible by the transfer into the likelihood domain and under the assumption that the cues are independent from each other.

The number of cues used for a single limb $N_c^{(l)}$ differs. The skin cue, for instance, is the only cue used for the hand limb, whereas the position of the lower arm is estimated using three or more cues. To account for the variations in the number of cues per limb, the cue likelihoods of an individual limb l are scaled according to the total number of cues N_l for this limb. This way, the likelihood of each limb contributes equally to the likelihood of the overall body pose. The function $\delta_c^{(l)}$ as part of the balancing term Eqn. (5.25) can be used to express whether a cue c is activated for a limb l:

$$\delta_c^{(l)} = \begin{cases} 1 & \text{if cue is active for limb } l \\ 0 & \text{else} \end{cases} \tag{5.26}$$

A fusion of such dissimilar features as presented here always requires to reason about the importance of each cue in relation to the other cues and its importance for the rating of the whole body pose. The factor λ_c is a term to re-weight the importance of a cue c regardless for which limbs it is applied. Such a modification of the cue importance can be made if the scenario is known in before to raise difficulties for a single cue, e.g. if the skin color can not be segmented robustly. If no assumptions can be made, λ_c defaults to one for all cues, rendering it inactive.

The compound likelihood as in Eqn. (5.24) is the final result of the observation process. For a given image and a given pose it provides a likelihood of how well the model pose fits to the observed image. As an example, Fig. (5.15) shows a plot of the combined likelihoods of all of the cues above. The resulting density is unimodal and well shaped. Only the exact position in the depth direction can not be determined that well. It must also be noted that the example image used here, cf. Fig. (5.9), shows an almost perfect situation. The person is facing the camera which makes the body model projection error small. The skin color can be segmented easily and the background color is neither



(a) (b)

Figure 5.15: *Final Pose Likelihood.* Calculated by combining all of the likelihoods above. (a) varying angles φ_{U_R} and ψ_{U_R} of the upper arm, (b) varying angle ψ_{U_R} only.

similar to skin nor to the color of the shirt. Furthermore, the background is more or less uniform, which produces less false detections for the edge cues.

For this example only the cues for the lower right arm and the hand have been used and they have been sampled for two degrees of freedom. To get an idea of how complex the posed problem can get, imagine that the displayed likelihood density is multimodal and ill-shaped and has to be explored for a 14-dimensional parameter space. At this point it gets obvious that an exhaustive search of the parameter space is not suitable and other means of exploration have to be found. The next section focuses on the kernel prticle filtering mechanism to solve this problem.

5.4 Kernel Particle Filtering for Body Pose Tracking

The problem of estimating the correct pose of the model for each image can be expressed as finding the one local maximum in the high-dimensional parameter space that fits best the current pose while still obeying all given constraints, e.g. the joint angle limits. To track the correct pose of the human, the structure of the high-dimensional probability density has to be efficiently exploited, taking into account the constraints and the dynamics of the model. The utilized kernel particle filter propagates such multimodal distributions and provides a probabilistic search for the best matching body configuration. The theory of the KPF method has been presented in section 3.3.2, we will now focus on its adaptation for model based tracking of the human body.

In the literature, many different techniques based on particle filtering can be found for tracking persons or a person's body parts. The idea of using the mean shift mode seeking within a particle filter has mainly been utilized in experiments consisting of tracking objects or isolated body parts in the 2D image space (e.g., [23, 63]). The last chapter gave an example for a localization and tracking approach in 3D. It already employed a kernel particle filter based optimization of a model with multiple degrees of

freedom. But the data provided was more more detailed as depth information and even velocities were present instead of monocular images only. The task of localizing a peron in space is less ambigious than recognizing his or her pose. This also becomes apparent in the number of parameters that need to be estimated. The person localization approach employed a model with six degrees of freedom, here we already need to estimate 14 degrees of freedom.

The kernel particle filtering technique proposed in section 3.3.2 is well capable of performing tracking in the high-dimensional space associated with 3D human modeling. Let us briefly review the filtering steps. The distribution of particles is rated based on the image cues, a mean shift procedure herds the particles in regions with a high probability. The best configuration is extracted and used to propagate the distribution to the next timestep. The following adaptions allow us to use the kernel particle filter technique for pose estimation.

The most individual part of each optimizsation approach, the problem representation itself, has been presented in the previous sections of this chapter. For each pose, a request can be forulated to rate the agreement of the model pose with the recorded image. The body model and the image cues provide the basis for the definition of the objective function. We can now focus on the remaining steps: refining the distribution, estimating the best pose and from that generating a new particle distribution for the next time step.

5.4.1 Refinement of the Particle Distribution

As already addressed before, degenration of the particle distribution is a common problem for particle filter approaches and many solutions to this problem have been presented. To give a well known example, the specific variant of sequential importance sampling also known as "Condensation" [79] applies a re-sampling step to avoid degeneration of the particle based representation.

We tackle this problem in a different way by designing the system for the specific task of human pose tracking in an interaction scenario. In order to perform tracking with a small number of particles, an iterative mode-seeking in the form of the mean-shift algorithm is applied to shift the particles to high weight areas. The advantage of mean shift is its principle of being a local approach. The kernel used for calculating the mean shift vector only takes into account a local region around the particle to be shifted. Besides, it employs an annealing technique which narrows the local support with each iteration. As a result, the distribution rapidly converges towards the next local maximum as depicted in Fig. (5.16), regardless of the global structure of the underlying PDF.

The choice of the kernel bandwidth h is of crucial importance in kernel based density estimation and it is usually scaled down at each mean shift iteration in order to concentrate on the most dominant modes. A small value can generate a very ragged density approximation with many peaks, while a large value can produce an over-smoothed density estimate. In particular, if the bandwidth of the kernel is too large, significant

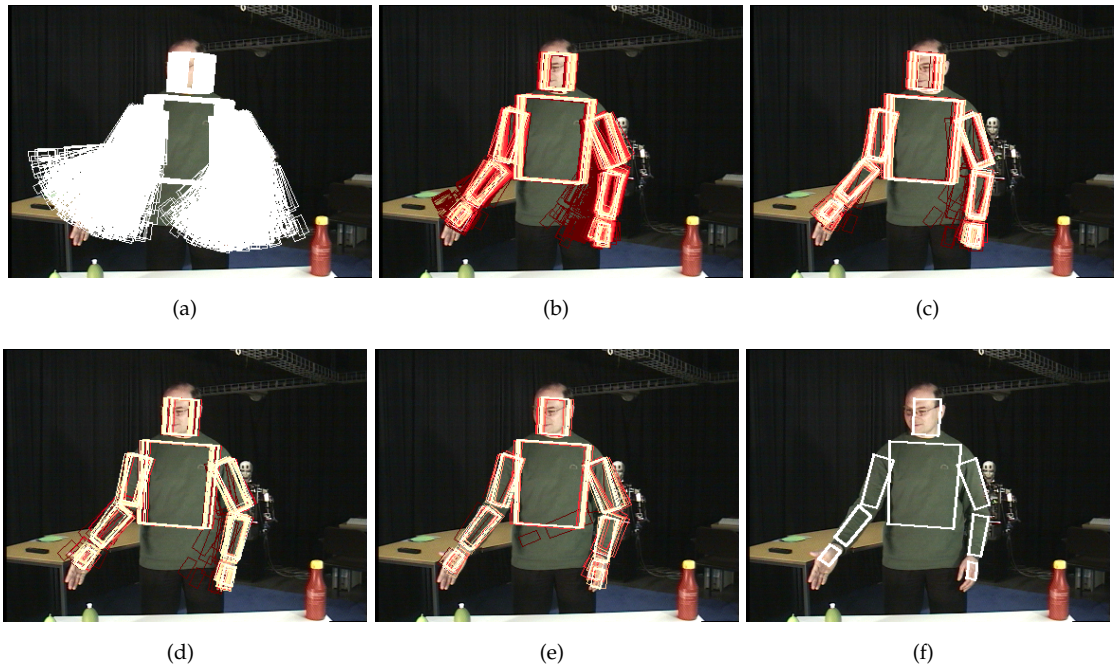


Figure 5.16: *Mean shift iterations for human body tracking.* Consecutive iterations of the mean shift procedure. (a) Initial distribution generated from the propagation model of the last time step. The particle distribution after (b) one, (c) two, (d) three, and (e) four mean shift iterations. The colors denote the likelihood of each pose after being evaluated with the image cues with white denoting a high likelihood and yellow towards red decreasing likelihood values. (f) The estimated body pose extracted from the most dominant mode of the distribution.

features of the distribution, like multi-modality can be missed. In our implementation, the initial bandwidth h_0 is scaled at every iteration i according to $h = 0.8^i h_0$ where the value 0.8 has been determined empirically, similar to [23]. To cope with the different scalings of the values for each dimension, the bandwith is adapted separately for each parameter using the joint deviation vector, resulting in a bandwith vector $\underline{H} = h\underline{B}$. Other kernel particle filtering approaches, like the EKPF presented by Dirk Stößel [156] employ both whitening and automatic bandwith selection techniques, which also aim at unifying the dynamics of the different parameters. These techniques are profitable to use for unknown configurations but as we are well aware of the meaning of the different parameters, e.g. joint angles and spatial positions, and we furthermore know about the applied propagation model, we can directly use this knowledge to perform a manual whitening to decorrelate the parameter space.

Iterating the mean shift procedure continues until a maximum number of iterations has been reached or until the Euclidean distance between the corresponding modes in the last two iterations is below an empirically determined threshold. Although a fixed number of iterations, e.g. 3, is more likely to produce distributions that are still spread out in the parameter space, this has the advantage that the runtime of the refinment step is more assessable. Additionally, the next step in the algorithm, the extraction of the best body pose, can handle such non-collapsed distributions to a certain degree.

5.4.2 Extracting the Best Body Pose

Following the particle distribution refinement, the most dominant mode is obtained by a weighted averaging over all particles in a window centered at the peak of the posterior. This mode serves as estimate of the current body pose, see Fig. (5.16(f)), and is output to other algorithms, like a trajectory based gesture recognition system. The back-projection of this pose into the image plane is also utilized as reference model for updating the mean color using Eqn. (5.22).

Over time, the best poses form a trajectory in the highdimensional pose space, which can be interpreted as the movements of the person. Motions of single limbs, for instance of the right hand, can be dervied similarly from the trajectory. In principle, the trajectory of any point on the limb surface or any point which is defined in relation to a limb can be calculated given the pose history and applying forward kinematics for the mody model.

Additional to the output to external modules the current pose also serves as the reference for estimating the particle distribution for the next timestep, as presented in the following.

5.4.3 Motion Models for Body Pose Tracking

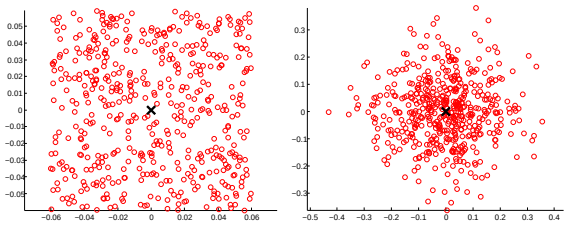
A key element for the efficient exploration of the parameter space is the propagation step of the particle filtering technique. It generates a new distribution of particles for the next timestep at those positions in the parameter space that the model is likely to take. Only the correct choice of the propagation model Φ ensures the particle filter to track any motion of the body that is possible. The generated distribution therefore must be widespread to capture the whole range of different parameterizations and also dense enough to enable the mean shift procedure to work properly. At the same time, the computational complexity of the particle filter roughly scales linearly with the number of particles generated. The most costly part is the evaluation of the objective function based on the image cues. Typically, a fixed number of particles is chosen to keep up a predictable runtime behaviour which implies that the avaiable particles have to be distributed wisely.

For propagating the particles from the dominant mode to the next time step we combine three different strategies: For some joints, velocity templates are used to model the dynamics of the human musculoskeletal system, a linear motion model covers the remaining degrees of freedom, and a minimum percentage of the particles are subject to random propagation. The reason for not relying on a specific motion model for the particle filter is that we aim at tracking general human motion and at the same time we need the ability to recover the tracking in case of failure. For each of the strategies the cumulated probabilities of the particles in the posterior are calculated to decide on the ratio of particles for the next time step. The method which generated the particles with the highest probabilities is favored. A minimum percentage for each method is always enforced to guarantee the ability to recover from tracking failures. If the propagation

results in an invalid body configuration, the propagation step is repeated until a valid body pose is obtained.

5.4.4 Random Noise Propagation

Propagating the particles with a random noise model Φ_N is often used as a fallback for error recovery and if the limb movements can not be predicted accurately. The particle distribution for the next timestep $t+1$ is defined by a normal distribution centered at the current position of the best mode $\underline{\mu}_{t+1} = \underline{x}_t$ in the parameter space, see also Fig. (5.17).



(a) (b)

Figure 5.17: *Random Noise Models.* Distributions of 500 particles generated for a given position (x) of the best mode. (a) uniform noise, also called white noise, (b) normal noise, also called gaussian noise.

The normal distribution is often used within classic particle filtering approaches. It offers a high density at the center but still spreads wide enough to catch strong motions.

$$\Phi_N = \mathcal{N}(\underline{\mu}_{t+1}, \underline{\sigma}_{t+1}) \tag{5.27}$$

We chose the standard deviations $\underline{\sigma}_{t+1} = \underline{B}$ to equal the initial bandwiths given in the body model definition. The values of the joint angle bandwith parameters B have been determined experimentally based on the model definition, the camera view and the application domain.

In the context of a kernel particle filter, however, a uniform noise model Φ_U has shown to perform better compared to the normal distribution model.

$$\Phi_U = \mathcal{U}(\underline{\mu}_{t+1}, \underline{\sigma}_{t+1}) \tag{5.28}$$

Kernel particle filtering employs the mean shift procedure to concentrate the particle distribution in successive iterations. A prior concentration of the particles is not needed, instead they are better off with being used for exploring the parameter space. Fig. (5.17) nicely shows that the same amount of particles covers a larger area of the parameter space.

Linear Velocity Model

More elaborate approaches for propagating the particle distribution often involve predicting the pose of the body model to the next time step and then forming a distribution appropriately. A reliable prediction can be achieved by taking into account the history of body poses for the last timesteps. For each joint, a new position can be estimated by extrapolating from the recorded positions.

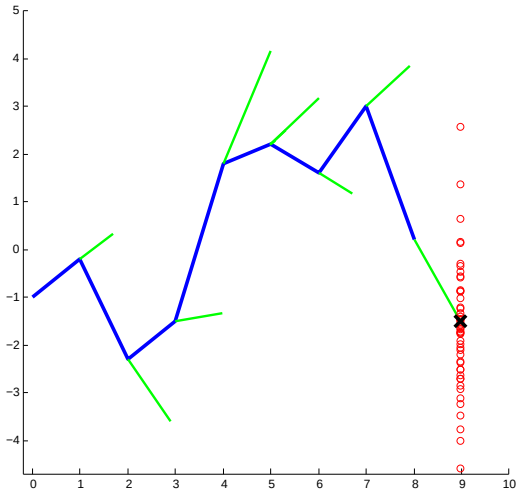


Figure 5.18: *Linear motion model.* The joint velocity (green) is estimated from the current trajectory (blue). At the predicted position of the joint, new particles are formed by applying a random noise model.

As shown in Fig. (5.18), a new position for a joint can be predicted by estimating a velocity vector $\underline{v}_t$ recursively:

$$\underline{v}_t = \alpha \frac{\underline{x}_t - \underline{x}_{t-1}}{\delta t} + (1 - \alpha)\underline{v}_{t-1} \tag{5.29}$$

with α being a smoothing factor that determines the influence of the latest measurements. A recorsive definition is in particular useful for noisy trajectries, as single outliers can be smoothed out. For smaller values of α, the velocity estimation is more robust but also reacts slower to changes in the trajectory. For recognizing pointing gestures with the proposed method, a value of $\alpha = 0.7$ is typical.

At the estimated position $\underline{x}_t + \underline{v}_t$, the noise model from Eqn. (5.27) is used to create a particle distribution. We can assume that the velocity estimation is already somewhat close to the correct position, so the derivation can be chosen to be smaller as when using a noise model only. We chose the derivation to be $\underline{\sigma}_{t+1} = 0.25\underline{B}$.

Velocity Template Models

Human kinetics shown typical patterns in joint velocities. Admiraal et al. [1] discuss the possibility to model such movements by an analysis of the kinetics and dynamics of a human arm. One example of such a modelling is Fitt's Law[1], which describes the motion during a pointing gesture, in particular the variation in velocity depending on the size and the distance of the target of the pointing gesture. This law can not only be used as a description, it also plays an important rule in the perception of motions [58] and is often applied in human machine interaction.

Fig. (5.19) depicts the typical progression of the joint angle velocities during a pointing gesture. Such sequences of velocities can now be reduced to a model representation. In contrast to other approaches that are based upon a database of poses and motions, the presented approach uses a more abstract representation as it tries to model each joint

[1] published by Paul Fitts in 1954

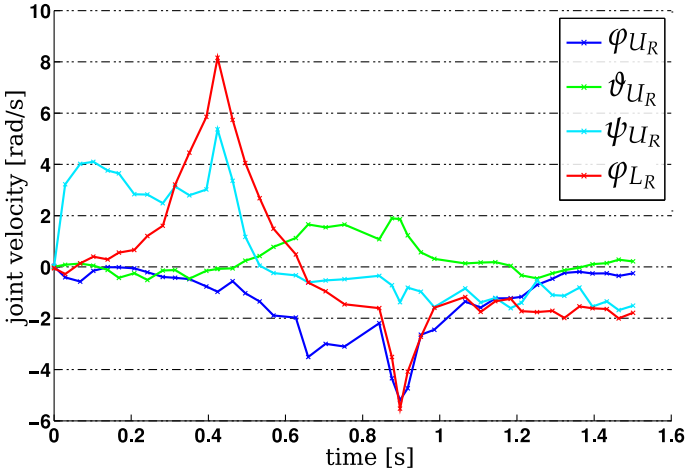


Figure 5.19: *Arm angle velocities during a pointing movement.* These velocities have been extracted from the tracking results of a pointing gesture recorded at 30Hz. The figure shows the velocities of the shoulder joints $\varphi_{U_R}, \vartheta_{U_R}, \psi_{U_R}$ and the elbow joint φ_{L_R}.

independently. It could be compared to approaches using a model of the dynamics of the human body, although our model is much more coarse and does not incorporate important features like masses of the limbs, inertia and gravity. It must be seen on a phenomenological level, as it tries to describe the typical velocities present in pointing gestures.

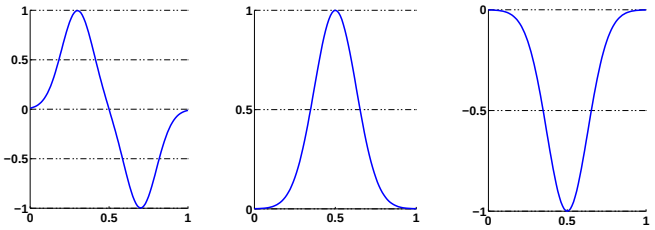


Figure 5.20: *A choice of trajectory template models.* The leftmost model is intended to represent the elbow's typical velocities during a pointing gesture, which is first bended an then stretched again. In combination with the scaling during the inference process, the two more generic models resemble the prospects of Fitt's law.

We use three models to cover most of the occurring motions. They are built from scaled gaussian function and are depicted in Fig. (5.20). The basic idea behind the shape of the models is that for each motion, the arm first needs to be accelerated to start moving and when getting close to the desired target position, it is decelerated again. Such motions of the arm result in according patterns in the joint angles. We deliberately chose not to use the acceleration defined by the second derivation as a feature, as the tracking results are typically quite noisy and often result in corrupted velocities given by the first derivation.

The basic idea of template based motion modeling is to record a history of joint angles, calculate the according velocities and match this velocity history with the templates using a multitude of different scalings and offsets. The parameterized template that best fits the original data is then used to predict the joint velocity – and thereby also the joint position – for the next timestep. This is done separately for each joint to reduce the complexity of the inference process. This approach has been inspired by the gesture detection presented by Hofemann [70] and uses a similar technique for the template representation and the search process. It has been implemented by Luemkemann and is described in more detail in [105].

Scaling the templates is necessary as the absolute intensity and the duration of a gesture is not known in before. We use three parameters to describe a scaled template, as depicted in Fig. (5.21). Amplitude scaling adapts the absolute intensity of the gesture to match with the maximal velocity during motions. Time scaling stretches or compresses the template to adapt to differnt length of motions. The time offset o_τ determines a starting point inside the template. This parameter is important in the context of the search process that is described in the following.

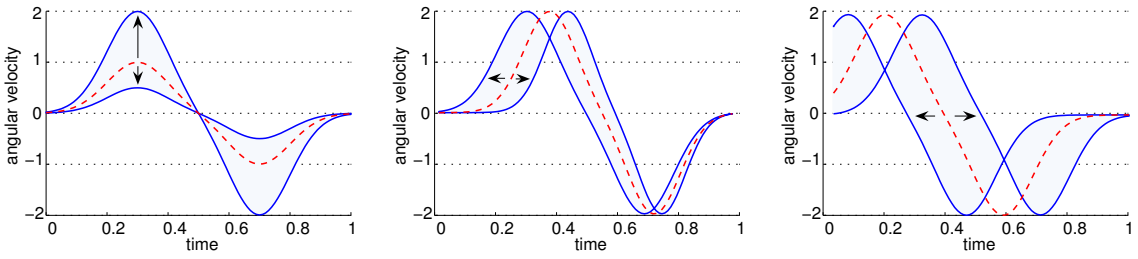


Figure 5.21: *Template scaling.* (a) Amplitude scaling s_a, (b) time scaling s_τ, (c) time offset o_τ.

Again, we use a particle filter approach to select the best fitting template parameterization $\underline{x}_t$ for a given history of joint velocities $\underline{z}_t$ at time t. The parameter vector $\underline{x}_t = \{m, s_a, s_\tau, o_\tau\}$ fully describes a specific parameterized template with m selecting one of the three template models. The likelihood of a template model $\underline{x}_t$ to resemble the observed joint velocities $\underline{z}_t$ is expressed by

$$p_{mot} = (\underline{z}_t | \underline{x}_t)\,. \tag{5.30}$$

The error measure comparing the history of joint velocities with the template model uses a window function that restricts the time frame to a number of time steps. Here, the time offset o_τ becomes important as it defines the starting point of the comparison window. For a more detailed description of the underlying algorithm, please see [105].

When observing a gesture, the result of the condensation process is a history of template parameters, giving the model chosen, the current offset and the scaling factors. For a completed gesture, the offset runs through the complete model from beginning to end. The scaling must not be the same for the whole gesture, as the condensation process allows gradually changes to the parameters. Thus, a gesture may start slowly and then become faster, as it also may exhibit different motion intensities. Fig. (5.22) shows an exemplary motion that has been fitted to the first gesture model. The different scaling for the first and the second part of the motion can be observed well.

The prediction accuracy of the joint position can be significantly improved if the velocities follow one of the templates. The effect on the tracking framework is that the new particles are better positioned for the next timestep and eventually less particles are sufficient for a good prediction. However, motion template modeling is a far more complex technique than the previously proposed motion models and the benefits for the tracking process should well surmount the additional effort if using this approach.

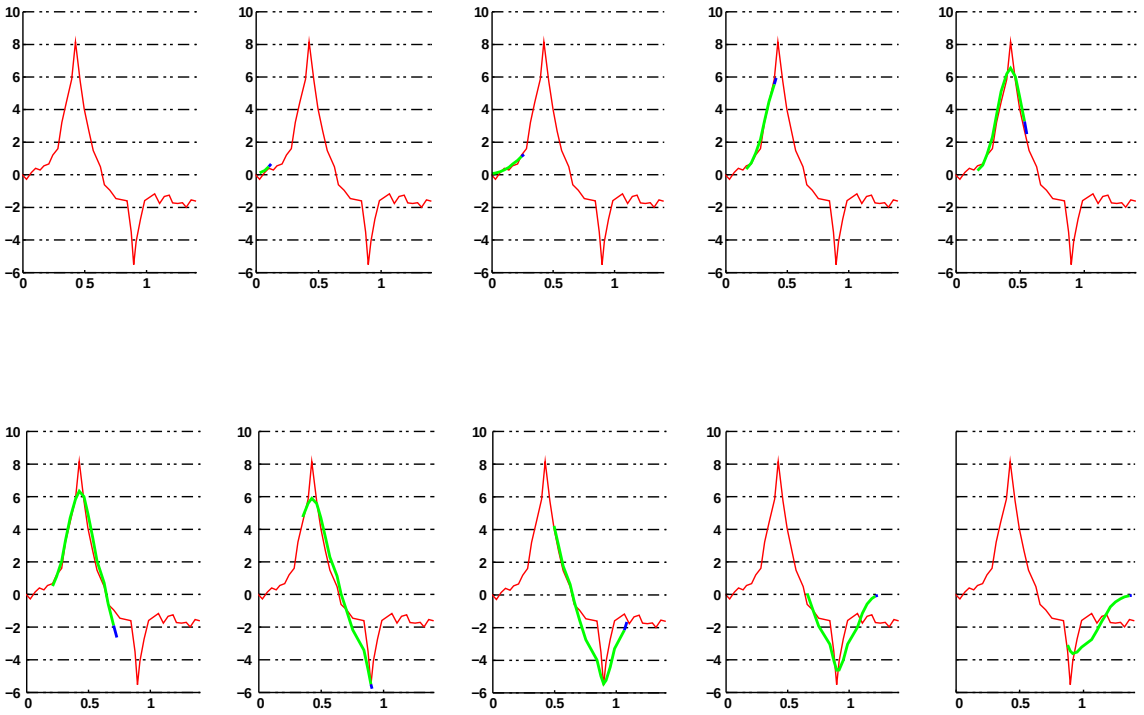


Figure 5.22: *Matching the joint velocity with a template.* The trajectory template follows the true velocity over time and also provides a good prediction for the next timestep.

The biggest chances for this model can be seen if it was combined with a full dynamic model or a gesture detection algorithm as the extracted information is far more valuable than just to be used for a prediction of joint positions.

5.5 Body Model Initialization

The body model tracking system that has been presented so far assumes that a manual initialization of the body pose is given for the first image. As long as the system is used for analyzing recorded image sequences such as presented in Sec. (7.4), a manual initialization is fasible. For many related body posture tracking approaches that can be found in the literature, the question of initialization also stays open or is subject to manual or semi-automatic procedures.

Related tracking systems incorporating automatic or semi-automatic initialization often make use of learned appearance models [125], rely on stereotyped poses [163] or on combining a repertoire of learned pose estimates and visual appearance [11, 145, 161]. Initialization can also be formulated as the problem of pose estimation or object reconstruction from a single image using strong models [98]. Summarizing, recent literature has described different approaches focused on tracking people. However, there is still a gap between tracking algorithms and systems working in the real world, mainly due to the fact that for most tracking approaches the challenges of automatic initialization and error recovery are not addressed thoroughly.

As a possible solution, we define in the following an initialization procedure that incorporates knowledge about the appearance of a human in an image and knowledge about the human robot interaction scenario to automatically derive an initial pose and initial parameters for the image cues of the pose tracking algorithm.

5.5.1 Automatic Initialization Procedure Overview

The presented system is situated within a typical human robot interaction scenario, where an individual is communicating with an artificial actor. The basic idea guiding the design of the initialization procedure is to integrate robust face and hand detection results into a model representation that can be used for automatic initialization and failure recovery of a pose tracking algorithm.

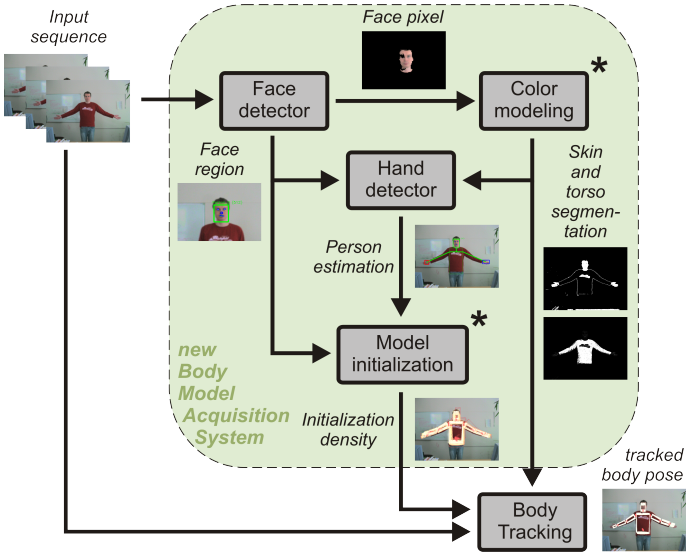


Figure 5.23: *Automatic initialization system overview.* Faces are detected, shirt and skin color are learned, hand and face positions are used to estimate a density for initializing the tracking and for error recovery. Modules denoted by * have been used in the existing body model tracking framework, but had to be manually initialized.

Using body tracking in human robot interaction often comes with strong restrictions, e.g., the number and type of cameras available or the gesture repertoire to be observed. There are a number of features that should be considered in order to make a system flexible enough to operate within this context: The person and its body dimensions must not necessarily be known a priori, but the distance of the human to the robot has to be adequate for interaction. Images are acquired using a single monocular color camera as the system is intended to be used on a mobile robot without employing further sensors. During system design, we also avoided specific background models to allow the tracking to be independent from the appearance of the observed scene. To further allow a moving camera, image background substraction based techniques like motion history images are avoided as well.

To ease the initialization process, we can pose additional requirements to the initialization stage compared to the tracking stage, where they are no longer neccessary. We assume the following requirements, which are well suited for a human robot interaction scenario, to hold for the initialization stage: 1) The person is trying to communicate, therefore his or her intention is to cooperate with the system. If, for instance, the

```
estimate initial pose from single image:
if face is detected then
    extract skin and shirt samples
    update color models for skin and shirt
    create segmented images
    detect hands using skin and shirt color segmentation
    generate initial density estimating the correct pose:
    if hands are detected then
        find probable model poses based on the distances of the hands
        and the face, use 5 DOF for model
    else
        body model facing the camera, arms hanging down, use 3 DOF
        for model
```

```
track human body:
do
    wait
until initial density is provided
if tracking for the first time then
    start tracking:
    use density as prior
else
    keep tracking:
    if initial density is provided then
        use density as recovery component
        use updated skin and shirt color models for tracking
    else
        track relying on current model
```

Figure 5.24: *The automatic initialization algorithm.*

tracking fails at some point, the person can help recover the tracking by entering poses that are easier to recognize for the system. 2) The upper body, including the head, the torso and both hands, is visible in the camera image and no large self occlusion occurs. 3) The person is standing in an upright position, facing the camera and having the arms outstretched.

The initialization procedure, as depicted in Fig. (5.23) and briefly outlined in Alg. (5.24), is presented in the following. As a first step, a face has to be detected for estimating an initial pose density and starting the tracking. Accounting for the current lighting conditions, a personalized skin color model is learned and based on this, both hands are located. Face and hands information, if detected, is utilized to estimate the likelihood for an initial body model location and pose. Once initialized, the body tracker performs continuously considering face and hands information whenever available. Future face and hand features allow the tracking system to have an additional validation control useful to recover the tracking from failures.

5.5.2 Face and Hands Detection

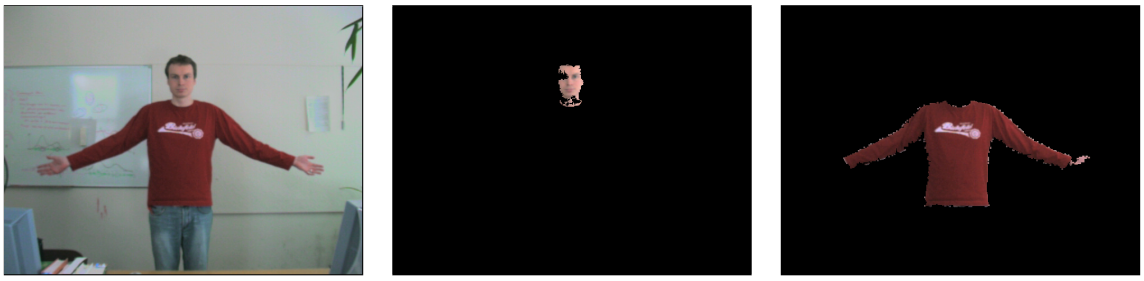
The ENCARA2 face detection system employed has been developed by Modesto Castrillón [21]. It integrates, among other cues, several classifiers based on the general object detection framework by Viola and Jones [165], skin color, multilevel tracking, and more. The chosen detection system provides not only face detection but also facial feature location information in many situations.

Figure 5.25: *Internal face feature detection.* Normalized face sample and likely locations for nose and mouth positions after normalization.

The facial element detection procedure is only applied in those areas which bear evidence of containing a face. This is true for regions in the current frame, where a face has been detected, or in areas with detected faces in the previous frame.

Color Modeling

To add further details to the representation of the individual detected, our system learns models of the skin and the shirt color taking into account the previously detected face region. This is done only for a robustly detected face, for which at least three facial features – the face itself, its eyes and the nose or the mouth – have been detected as a trusted result. This consideration is used to reduce the possibility of false detections, i.e. false positives. A color model is then learned, or updated if already created for that individual, only from these trusted faces, reducing the probability of using erroneous face detections.



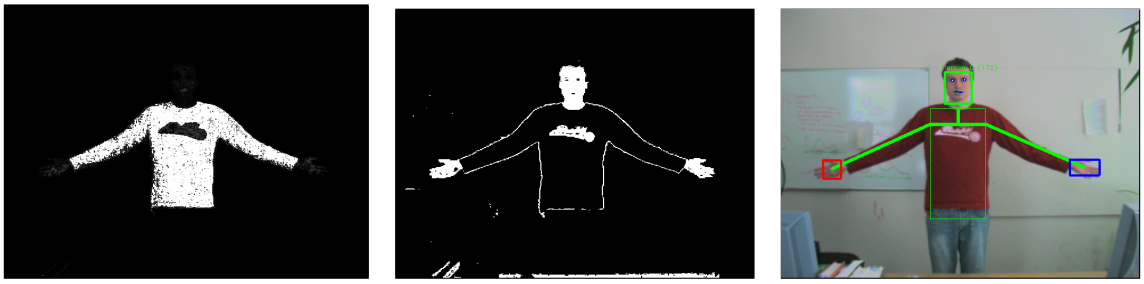
(a) (b) (c)

Figure 5.26: *Color training.* (a) Input image, (b) Skin color training pixel, (c) Torso color histogram segmentation.

From the internal features' position, a face container is estimated to provide training data for the skin color model. In a lower position a second container is used to select an area of the user's shirt. Both containers are utilized to model respectively the skin and shirt colors of the user by means of a histogram based color model [160]. Fig. (5.26) presents an examplary segmentation of the input image based of these histograms, see Fig. (5.26(b)) for skin and Fig. (5.26(c)) for shirt color-like areas. It can be observed that

the shirt is segmented easily but hands and skin-like areas, are not necessarily clear and compact.

Due to the sensitivity of the histogram based skin color model, we chose another color model representation. The mask of the skin blob extracted from the face container determines the skin pixels to be employed as training samples, see Fig. (5.27(a)). The skin color of each individual is then learned and further adapted throughout tracking using the mixture of gaussians color model as described in Sec. (5.3.3), see Fig. (5.27(b)) for the segmentation results using the learned MoG color model. The histogram based color model, learned from training pixel drawn from a region below the face, is similarly used to provide training data for the mean color cue described in Sec. (5.3.3).



(a) (b) (c)

Figure 5.27: *Head and hands detection.* (a) Skin color training pixels produced from the face detection, (b) resulting segmentation using the skin locus and learned mixture of gaussians (MoG) in RG-color space, (c) detected head and hands location.

Hand Detection

Searching the full 14-dimensional pose space is not feasible for an initialization procedure. But robustly finding an initial pose of the body model can be achived with less effort by assuming to have an exact position of the head and the two hands of a person. The corresponding limbs of the body model can then be attached to the found body parts which drastically reduced the complexity of the search.

Multiple difficulties are present regarding robust and efficient hand detection in video, mainly due to the inherent variability of the articulated hand structure, the large domain of gestures, the restriction of real-time performance, varying illumination conditions and complex background clutter. Therefore, different restrictions are commonly considered or even manual initialization is performed for this task.

Literature is rich in hand detection approaches that have traditionally been based on skin color segmentation [154], due to their reduced processing cost. Recent approaches [90, 151], however, have utilized the Viola-Jones' object detection framework [165] even when hands are not that easy to describe as faces. They are highly deformable objects, so training a single cascade classifier for detecting hands is a complex and arduous task. For that reason, a different classifier for each recognizable gesture can been trained [151], but also a single classifier can be sufficient for a limited set of hands [90].

Considering the unrestricted context, where the use of multiple detectors would produce an approach not suitable for real time processing, we have chosen the skin color approach for faster processing. However, instead of using a predefined color space definition, the information obtained from the face blob is used, as described above, to estimate the skin color model for that individual, see Fig. (5.27). The skin color model is then employed to locate other skin-like blobs in the image.

As we mentioned above, the approach considers that both hands are visible, no gloves are used, their distance to the face is similar, and that a vertical line falling from the face center would leave each hand on one side. If all those conditions fit, and two well proportionated and coherent skin blobs are located close to the shirt-colored region, then they are suggested as hands candidates, and provided to the 3D tracker initialization module, see Fig. (5.27(c)) for the detected positions.

5.5.3 Integration into the Body Pose Tracking System

After determining the position of the head and hand limbs in the image, we now must transfer this into a valid pose of the body model. Estimating the correct pose is even more difficult on a single frame than on a sequence where also temporal dependencies could be used. Restrictions on the scenario and the expected poses help to simplify the task. For initialization, we assume the person to be facing the robot with the arms outstretched. As a consequence, the model can be constrained to be oriented towards the camera and the arm limbs to move only in a plane parallel to the image plane, having the elbow relaxed. This allows reducing the number of DOF to be determined during initialization to only 5 parameters: Three for the model position and one for the elevation of each arm. In situations in which no hands can be found, the arms are assumed to be close to the torso, reducing the dimensionality d of the parameter space further to 3. An additional scaling factor for the body size is needed, as due to the orthographic projection, variations in the absolute size and in the distance to the camera can result in the same appearance. For the presented interaction scenario, however, we can assume that the distance of the person is close to the robot.

The 2D distances between the detected face, d_H, and left and right hand features, d_{H_L} and d_{H_R}, and the corresponding model limbs are converted into likelihoods using the following Gaussian weighting function

$$p(c) = \exp\left(-\frac{(d_c)^2}{2\,\sigma_c^2}\right), \tag{5.31}$$

where the standard deviations σ_c are choosen to cover the maximal observable distance depending on the image size for each utilized feature $c \in \{H, H_L, H_R\}$. For a number of different poses, these distances are almost the same, as the positions of the face and the hands in the image will not change drastically when translating the model in the depth direction. Similar to Eqn. (5.24), the likelihood that a model pose $\underline{x}_t$ at the current time t causes the observation $\underline{y}_t$ can be formulated as

$$p(\underline{y}_t \mid \underline{x}_t) = \prod_{c \in \{H, H_L, H_R\}} p(c) \tag{5.32}$$

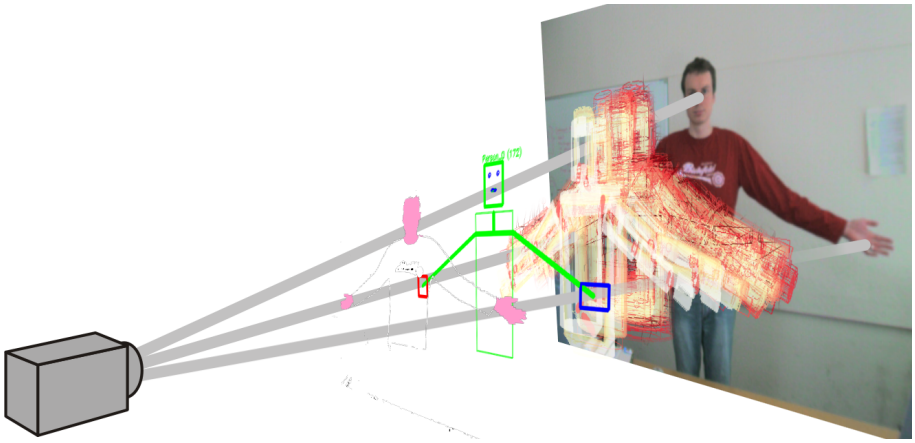


Figure 5.28: *Generating a distribution for automatic initialization and error recovery.* The particle distribution approximates the likelihood density defined by the face and hands detection.

with $p(c) = 1$ if the feature is not present. In the d-dimensional space $\mathcal{R}^d$ of all possible poses, an initialization particle set $\mathcal{SI}_t = \{\underline{s}_t^{(n)}\}_{n=1}^N$ is used to represent the observation density with the associated weights $\{w_t^{(n)}\}_{n=1}^N$ distributed according to $p(\underline{x}_t \mid \underline{y}_t)$ and w normalized to $\sum_{n=1}^N w_t^{(n)} = 1$.

Using this particle set, we employ a kernel particle filter for searching the pose space for initialization postures agreeing with the results from the face and hand detection. The result is a particle distribution estimating the likelihood density in the reduced parameter space. Figure 5.28 illustrates the result when both face and hand position are given. These image positions each form a line when projected into the 3D space, on which the corresponding limbs of the model have to be placed. The distribution indeed shows a good estimation seen from the image plane perspective, but it covers a wider range in the depth direction, as the distance to the camera respective the size of the model can only be determined indirectly.

Up to this point, the problem of fitting the model to three given points in the image can be solved much more easily, e.g., making use of inverse kinematics which could provide a deterministic set of possible solutions. Using a multiple-hypothesis approach for both tracking and initialization instead gives us the advantage that the generated results can easily be integrated into the tracking framework. This approach also leaves room for future extensions, which will possibly result in higher dimensional statespaces, e.g. including occlusion and texture information.

A major problem of all tracking approaches is the tendency to get stuck in false local maxima. To overcome this drawback, the presented approach adds a recovery component to the existing tracking framework. Recovering from tracking errors is achieved by inserting a fixed percentage α (e.g., 5% - 20%) of particles from the initialization distribution $\mathcal{SI}_t$ into the tracking distribution $\mathcal{ST}_t$

$$\{\mathcal{ST}_t\}_{n=1}^{\lfloor \alpha \cdot N \rfloor} = \Phi(\mathcal{SI}_t, n) \tag{5.33}$$

with $\Phi(\mathcal{SI}_t)$ sampling from the distribution according to the weight of the particles. These particles do not necessarily represent the correct pose and in such cases will

be neglected during the tracking process. If the tracking gets stuck in a wrong pose, however, the recovery particles enable the algorithm to explore the parameter space in a region outside of the current search radius of the tracking process while they are still more directed and therefore more likely to resemble the correct pose than randomly distributed particles.

5.6 Summary

In this chapter, we presented a system for tracking human upper body motions in a human robot interaction scenario that is based on a monocular approach and uses no specialized hardware. The system does not depend on a database of learned motions or on a mapping from 2D shilouettes to 3D poses. Instead, a generic 3D body model is employed for matching the estimated pose with the appearance of the human in the image. The inference process is based on multiple image cues that are fused to obtain a likelihood for a given pose of the model. A kernel particle filtering scheme efficiently explores the highdimensional search space to track the body pose over time. Using a face recognition module, the system is self starting and learns the appearance of the observed human from a generic model.

6 System Evaluation and Optimization

"In theory, there is no difference between theory and practice. But, in practice, there is."

Jan L. A. van de Snepscheut

The last chapters have presented various techniques for vision-based localization, posture detection and tracking of humans. Employing these techniques in the context of interactive scenarios means that they are used to extract position and pose information from the images and provide these data to subsequent processing steps, e.g. for engineering safety measures or to recognize gestures in order to understand the actions of a human. Integrating the proposed recognition techniques into bigger systems calls for an evaluation of the exactness of the reconstruction as subsequent processing steps need to cope with the expected inaccuracies.

The following sections present evaluations that examine the reconstruction accuracy for both the person localization and the posture tracking algorithm. To determine the error during localization and tracking the image sequences have been annotated with ground truth either manually or based on active and passive markers. From this error a quality measure determining the performance of the algorithm under the current parameterization can be derived. Having this at hand, we can even go a step further and develop a method to automatically determine an optimal set of parameters for the algorithms. The probabilistic optimization technique and the results obtained are presented in the last section of this chapter.

6.1 Evaluating the Person Localization

For evaluation of the person localization, multiple real-world image sequences have been recorded with a PointGrey Digiclops multiple CCD camera system with an image size of 640×480 pixels, a pixel size of 7.4 μm, and a focal length of 4 mm. The stereo baseline corresponds to 100 mm. The sequences display an industrial working cell with a human worker, a robot, and a moving platform. The distance of the camera to the scene is 5.65 m.

We empirically found for the correlation matrix element Σ_z in Eqn. (4.4) the value $\Sigma_z = 0.292$, regarding a set of 3D points obtained with the spacetime stereo algorithm and belonging to a plane scene part, while $\Sigma_x = \Sigma_y = 1$ are equally scaled. The velocity scaling factor is set to $\rho = 380$ s, where the velocity is expressed in meters per second and the spatial coordinates in meters. The kernel widths for Eqn. (4.7) are chosen as $H_{r,\max} = 1.88$ m, $H_{r,\min} = 0.135$ m, $H_d = 0.113$ m, and $H_v = 0.188$ m.

			with velocity		without velocity	
seq.	# pic.	object	RMSE	% tracked	RMSE	% tracked
industry1	69	person	26.5	100.0	38.3	84.8
		table	60.3	100.0	21.8	69.7
		robot	87.8	95.5	111.8	98.5
industry2	79	person	42.7	100.0	31.8	94.8
		table	43.5	100.0	27.5	100.0
		robot	12.1	98.7	17.7	96.1
industry3	24	person	19.6	100.0	14.7	100.0
		table	24.9	100.0	22.5	90.9
		robot	17.1	100.0	29.3	100.0
industry4	39	person	24.7	75.7	35.2	89.2
		table	27.0	100.0	24.5	97.3
		robot	9.1	100.0	20.0	97.3
industry5	24	person	20.8	90.9	25.4	81.8
		table	21.9	100.0	32.9	100.0
		robot	8.6	77.3	33.1	100.0

Table 6.1: *Person localization tracking results.* Tracking accuracy by comparing to manually labeled ground truth. The RMSE is given in centimeters.

For each sequence, ground truth was generated manually by marking the center of the objects of interest in each frame, e.g. the head of the person or the center of the car, and transforming them into 3D coordinates using the known geometry of the scene and the objects, e.g. the tallness of the person and the position of the ground plane. The trajectories of the tracked objects are compared to the ground truth based on the corresponding value of the root mean square error (RMSE). The results in Table 6.1 show that objects can be tracked in a stable manner at reasonable accuracy. Using velocity as an additional feature yields a more accurate localisation result for 10 of 16 detected objects, and detection is usually possible in a larger fraction of the frames. For four other objects the RMSE but at the same time also the detection rate is lower when velocity information is neglected. The system is designed to segment the point cloud into clusters of differing velocity. As a consequence, the proposed system works best for objects with homogeneous velocity. For example, we observed that for a walking person moving the arms backwards the object hypothesis does not contain the arms. As it is illustrated by the trajectories in Fig. (4.11), the system is able to track objects and persons in a top-view surveillance setup as well as in a side-view setup.

6.2 Evaluating the Body Pose Tracking

Being able to make statements about the exactness of the tracking is a prerequisite for a qualitative evaluation and in consequence for measuring improvements of the algorithm. This can be achieved by a comprison of the estimated body pose and the true pose of the human, the so called ground-truth. The challenge is therefore to record images of the human and his body pose simultaneously and to define an appropriate measure

for comparison. In the following a ground truth corpus based on acitve markers is presented and the utilized quality measure is motivated. The corpus has been recorded with the help of Andre Zielinski. For technical details, a motivation on the ground truth measure and first evaluations of the body tracking system you may also see [179].

6.2.1 Marker-Based Ground Truth

To determine the "true pose" of a human seems to be an easy task given the right technical equipment. This is only true up to a certain point. Motion capuring systems are well known nowadays from special effects in movies. Our approach to record a ground truth dataset is based on active markers that are attached to the limbs of the subject, quite similar to the passive marker systems commonly used for motion capturing. But in fact, the motion capturing system only records the position of the markers, not the pose of the human. You might ask what the difference between those two is. The markers can be seen as a projection of the human posture. As the markers are attached to the subjects' skin or clothing, they easily shift their position during movements. This can be due to wrinkles or loose fitting clothes but also due to unavoidable disturbances of the body surface from the muscles. Therefore, the recorded data must be seen as an approximation of the body pose and calls for a closer examination of the utilized distance metric for comparing the tracked pose with the ground truth.

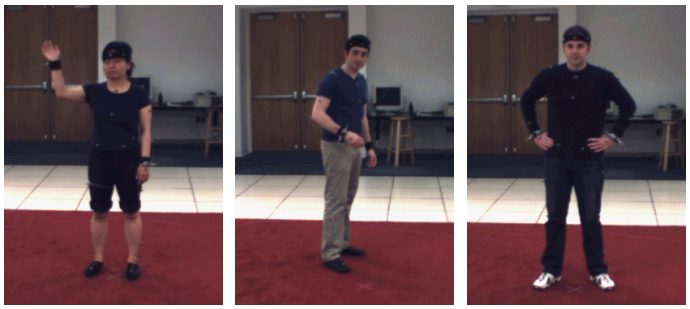


Figure 6.1: *Sample images from the HumanEva-I corpus.* The subjects are wearing black clothes with retro-reflective infrared markers and perform a variety of actions.

Before detailing our own ground truth data set, let us have a look at the HumanEva-I corpus [144], which Leonid Sigal and Michael Black presented in 2006. It is freely available in the Internet[1] and shares some technical and methodological features with our corpus, so we will discuss it briefly. The idea to make this corpus public is to provide a possibility to quantitatively compare different body tracking approaches.

The HumanEva-I corpus contains recordings of four persons performing a number of different actions, like walking, jogging, boxing and gesturing, see Fig. (6.1). Black and Sigal used a ViconPeak Motion Capturing System which uses a number of infrared cameras with active lighting to localize retro-reflective markers attached to the subjects' body. Image data has been recorded at 60Hz with four Pulnix TM6710 gray scale cameras and three UniQ UC685CL color cameras from different points of view. Ground truth is provided in the form of a true full body pose, which has been optimized using all tracked markers, cf. Fig. (6.2). A comparison to tracking results can be done either

[1] http://vision.cs.brown.edu/humaneva/index.html

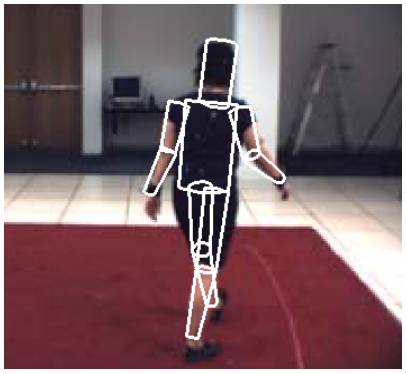


Figure 6.2: *HumanEva-I ground truth.* An optimal position for the body model has been determined using all tracked markers. This can be used as a direct reference or by placing virtual markers on the model.

by comparing the body postures or by placing virtual markers on the ground truth model. A more detailed description of the dataset and further information about the data processing can be found in [144].

For evaluating the body tracking framework, however, the HumanEva-I dataset has been only partly usable. We aim at a monocular setup in an interaction scenario, where the focus lies on frontal views of the upper body of the human. The HumanEva-I dataset is optimized for a multi-camera setups employed in full body tracking approaches. For many actions, the person is too small as it is further away from the camera and it is moving around which reduces the reasonably usable observation time for a single camera to only a few frames. That prohibits the evaluation of tracking failures and the recovery abilities of our system for longer image sequences. Additionally, the person's appearance often provides too little detail for our image cues, as the subjects are mostly wearing dark clothes, probably to ease the calculation of silhouettes.

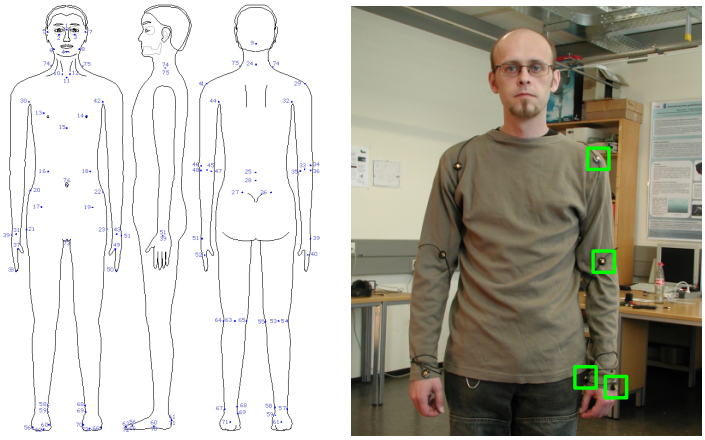
The following paragraph explains the setup that has been used for our own recordings of ground truth data that much better fits the purpose the system has originally been designed for.

Figure 6.3: *Lukotronic AS200 motion capture system.* Active infrared marker chain with transmitter and trinocular infrared camera system for 3D marker tracking.

Recording Setup and Configuration

For our experiments, a Lukotronic AS200 Motion Capturing System has been used to record the marker information, cf. Fig. (6.3). This system is comprised of an active infrared marker chain with a transmitter that is attached to the subject and a trinocular infrared camera system that tracks the markers and calculates the according 3D information. Each marker sends out infrared pulses with a unique code sequence and therefore can be identified unambiguously. The design of the markers restricts the visibility to a cone angle of approximately 170°. The maximum allowed distance from the camera is 12m, in the range of 1–5m, the manufacturer assures a maximum error of 0,1mm. The system is able to record marker data at a framerate of 60Hz. It is self-calibrating, which means that the marker coordinates are output in a metric world coordinate system. Simultaneously, images have been recorded using a Sony DFW-V500 firewire camera at 15Hz on a different laptop. Both systems recorded timestamps for each frame.

All recordings have been made in a lab environment under natural lighting with the aid of two photo lamps. Prior to the actual recording sessions we captured multiple calibration images.



(a) (b)

Figure 6.4: *Marker positioning for ground truth recording.* (a) Marker positions as proposed by the H-Anim specification. (Image found in [59]) (b) The marker positioning as used for our experiments. For each arm, one marker is placed on the shoulder, one at the elbow and two at the wrist to better handle occlusions. The markers on the left arm are highlighted.

We used a total of eight markers to record the pose of the human that have been positioned as follows: For each arm, one marker is placed on the shoulder, one at the elbow and two at the wrist at opposite sides of the arm. This allows a better coverage in terms of visibility and more robustness against occlusions which are quite common while gesturing.

Contents of the Corpus

The corpus is comprised of the data of two subjects. For each of them, multiple streams have been recorded with a total of approximately 11.000 images and the according marker trajectories. The subjects repeatedly perform various actions and different types of gestures, including pointing at objects in the scene, raising both arms outstretched

(for calibration purposes) and deliberately occluding body parts. For some streams, the person has been standing still in front of a black curtain, cf. Fig. (6.5(a)), for others the person has been moving around in a cluttered lab environment, cf. Fig. (6.5(b)). For some streams, the subjects changed their clothes to provide a wider variety of appearances throughout the corpus. These recordings have been cut into multiple sequences with 30–150 frames each. A single sequence holds a single gesture, like pointing at one object or raising the arms.

(a) (b)

Figure 6.5: *Sample images from the ground truth data set.* (a) The author of the thesis pointing at an object in the scene. (b) The second subject presenting his dancing skills.

Fig. (6.6) displays the recorded marker data for one stream containing multiple gestures. Note that the wrist markers on the left arm (green and pink trajectories) are not visible all the time due to occlusion and visibility constraints.

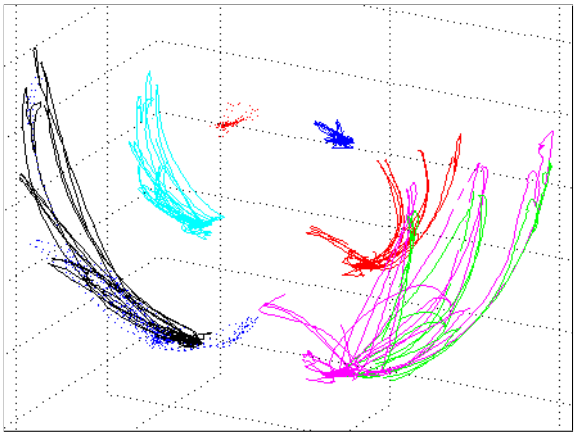


Figure 6.6: *Marker trajectories.* 3D trajectories recorded from the eight active infrared markers for a person performing pointing gestures to multiple targets.

Synchronization and Calibration

Images and marker data have been recorded independently on two separate laptops at different framerates, as depicted in Fig. (6.7). Although each image and each set of marker positions is annotated with a timestamp, the internal clocks never run absolutely synchronously and a later synchronization is required for a unique mapping of images and marker data. Furthermore, the two data streams need to be aligned in order to find a common starting point. Thus, two parameters for the temporal scaling $a_k \in \mathbb{R}$ and the temporal offset $b_k \in \mathbb{R}$ have to be determined. As the marker data is (almost) never recorded at the very same moment the image is captured, the position information

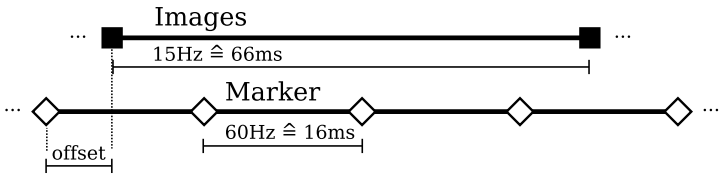
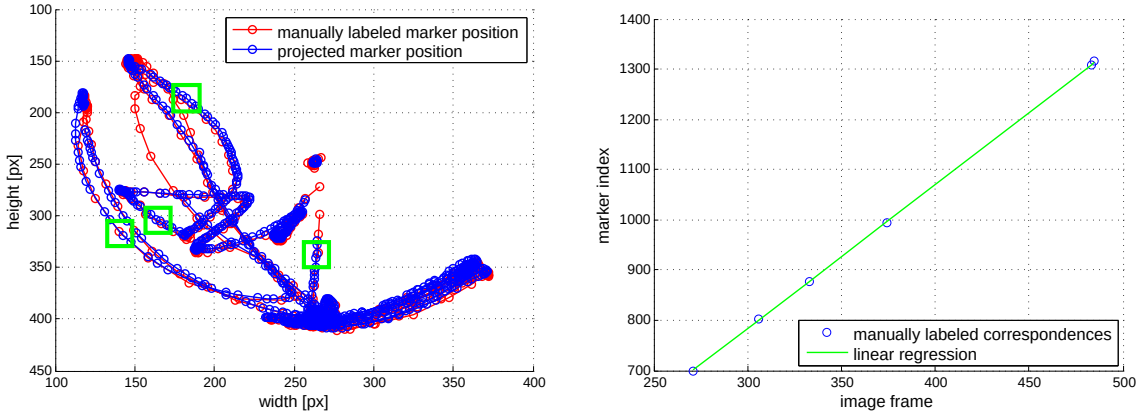


Figure 6.7: *Image and marker framerate.* The camera provides images at a framerate of 15Hz, the marker data is recorded at a four times higher rate of 60Hz.

needs to be interpolated to reconstruct the correct point in space at the time of the image recording. Although making a small error here, we choose a linear interpolation to be sufficient for our needs. A polynomial regression or spline interpolation would provide more exact results, especially when observing accelerated motions.

We used a semi-automatic technique for synchronization, similar to Black's method applied on the HumanEva-I dataset, see Sec. (6.2.1) for reference. Camera calibration using the Matlab Camera Calibration Toolbox of Jean-Yves Bouguet[2] with a standard checkerboard pattern yields an estimation for the intrinsic parameters of the camera k: The focal length $\underline{f}_k \in \mathbb{R}^2$, the principal point $\underline{c}_k \in \mathbb{R}^2$ and the radial distortion coefficients $\underline{k}_k \in \mathbb{R}^5$. To calibrate the camera with the marker tracking system, we further need to know the relative orientation $\underline{r}_k \in \mathbb{R}^3$ and the translation vector $\underline{t}_k \in \mathbb{R}^3$ representing the extrinsic parameters.



(a) (b)

Figure 6.8: *Determining spatial and temporal parameters for synchronization.* (a) Manually labeled correspondences (green boxes) for the 3D markers. (b) linear regression yields the temporal parameters a_k and b_k.

Our synchronization technique uses the Nelder Mead Simplex algorithm, see Sec. (3.2.1), to determine a usable combination for the free temporal and spatial parameters. For the 3D markers $\underline{\Gamma}_t^{(3)}$, $t \in \{1 \dots T^{(3D)}\}$, their position in the 2D image $\underline{\Gamma}_t^{(2D)}$, $t \in \{1 \dots T^{(2D)}\}$ has been manually labeled for a number of reference images, preferably such with motion present as this makes the following synchronization less ambiguous. Synchronization and calibration can be achieved simultaneously by minimizing the Euclidean

[2] `http://www.vision.caltech.edu/bouguetj/calib_doc/`

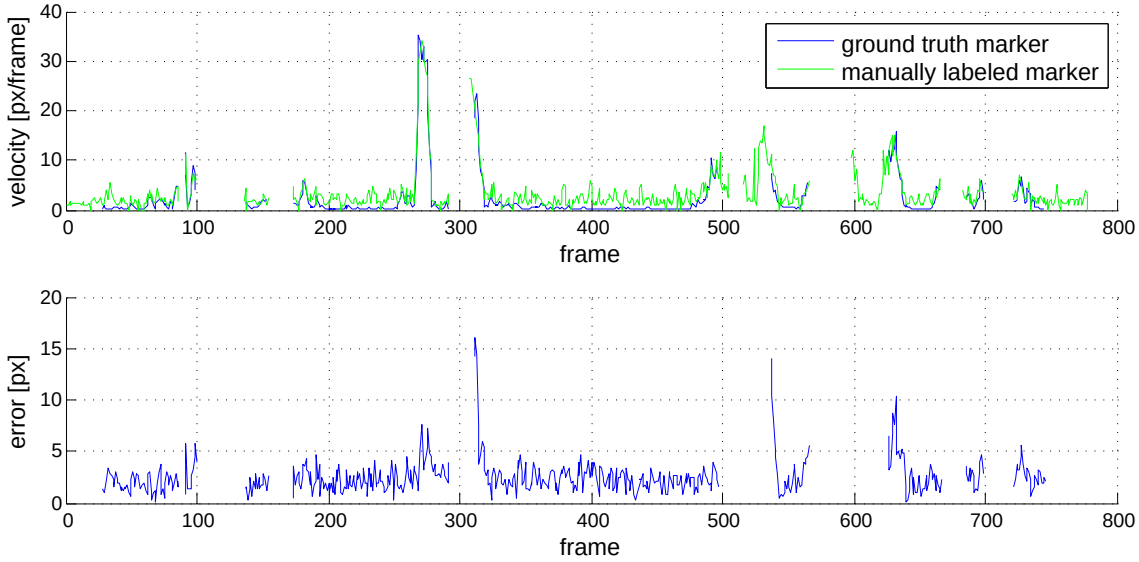


Figure 6.9: *Error of the synchronization over time.* The plot at the top displays the magnitude of the markers' velocities calculated by differencing adjacent position values. The bottom plot displays the synchronization error for all values where both markers and tracking results are available.

distance $\underline{f}(\underline{\Gamma}_t^{(3D)}; \underline{r}_k, \underline{t}_k) \in \mathbb{R}^2$ between the markers and their labeled image position:

$$\underset{\underline{r}_k, \underline{t}_k, a_k, b_k}{\operatorname{argmin}} \sum_{t=1}^{T^{(2D)}} \left[\delta(t; a_k, b_k) \cdot ||\underline{\Gamma}_t^{(2D)} - \underline{f}(\underline{\Gamma}_{t*a_k+b_k}^{(3D)}; \underline{r}_k, \underline{t}_k)||^2 \right]. \tag{6.1}$$

The result of the minimization yields a unique mapping of the marker positions recorded in 3D space to the images. Note that the 3D marker position $\underline{\Gamma}_{t*a_k+b_k}^{(3D)}$ is linearly interpolated, to incorporating the previously mentioned temporal interpolation, which means that a_k and b_k are being real-valued. The $\delta(\cdot)$ function serves as a domain test, to ensure that only indices are used for which both, marker data and images, are present.

$$\delta(t; a_k, b_k) = \begin{cases} 0 & \text{if } \; t \cdot a_k + b_k > T^{(3D)} \\ 0 & \text{if } \; t \cdot a_k + b_k < 1 \\ 1 & \text{otherwise} \end{cases} \tag{6.2}$$

Initial values for $\underline{r}_k$, $\underline{t}_k$, a_k and b_k are being chosen by hand to roughly resemble the configuration of the experimental setup. The results of the optimization are shown in Fig. (6.8). The captured 3D marker positions nicely fit to the manually annotated positions. A linear regression yields the offset to find a common starting point and the temporal scaling factor. Using these information, a new list of interpolated marker positions for each image frame can be calculated and used for later evaluation.

When viewed over time, the synchronization error displayed in Fig. (6.9) mostly stays below 5 pixels. Higher errors of up to 17 pixels occur only in situations where the marker is more or less rapidly moved. At first sight this could be taken as a result of our inexact interpolation. But a closer examination of the raw marker data reveals that the pose always lags behind the position as observed in the image. So either the camera changes

its framerate when observing motion or the motion capturing device sometimes has problems with the exact timing during motions. But for calling this behavior an error, the effect was too weak too momentary and thus has been disregarded for the further experiments.

6.2.2 Error Measure Definition

To judge the exactness of the tracking results, a measure is needed that expressed the error between the estimated body pose from the tracking framework and the recorded ground truth.

The error measure that is used for this evaluation follows the proposal of Bălan et al. [18]. It is based on the Euclidian distance between virtual markers on the body model and the real markers from the motion capturing system. In the following, we will briefly discuss the definition of virtual markers with respect to the body model as defined in Sec. (5.3.1). Thereafter, the definition of the error measure is presented.

For each real marker of the motion capture system, a corresponding virtual marker id defined at the same position as the real marker was placed. The measure is then based on the mean distance for all markers, see also [144] for a more in-depth examination of different measures for body tracking evaluation. Let M the number of virtual markers on the estimated body model $\underline{\hat{X}} = \{\underline{\hat{x}}_1, \underline{\hat{x}}_2, \ldots, \underline{\hat{x}}_M\}$ of the tracking system, and accordingly the markers $\underline{X} = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_M\}$ of the ground truth. For an estimated posture $\underline{\hat{X}}$, the distance to the ground truth posture $\underline{X}$ can be given as the mean absolute error of the corresponding markers by

$$D(\underline{X}, \underline{\hat{X}}, \Delta) = \sum_{m=1}^{M} \frac{\delta_m ||\underline{x}_m - \underline{\hat{x}}_m||}{\sum_{i=1}^{M} \delta_i}. \tag{6.3}$$

As already seen in Fig. (6.6), the presence of all markers can not be ensured for every timestep due to occlusions and visibility constraints. We define a set $\Delta_t = \{\delta_1, \delta_2, \ldots, \delta_M\}$ of selection variables that defines for each marker $\underline{x}_i$ whether the data is valid $\delta_i = 1$ or not $\delta_i = 0$ at the current timestep t.

For a sequence of images with the length T, the mean error over all tracked body postures can be determined as follows:

$$\mu_{seq} = \frac{1}{T} \sum_{t=1}^{T} D(\underline{X}_t, \underline{\hat{X}}_t, \Delta_t). \tag{6.4}$$

6.2.3 Evaluating the Accuracy of the Body Pose Tracking

As a first step, we are interested in the overall exactness of the body tracking framework. It is commonly known that the employed kernel particle filtering technique shows better results for a large number of particles, as with more particles, the parameter space can be covered more densely. But still, using double the number of particles we still do not

anticipate double the exactness for two reasons: First, in the highdimensional parameter space, twice as many particles do not double the exactness of the search process. Secondly, we are employing a twofold optimization process which uses alternating particle filtering and means shift steps. It is the effect of these two statements what we are trying to document with the following evaluation.

Effect of the Number of Particles on the Accuracy

This evaluation shows the effect of changing the number of particles available for the kernel particle filtering process. We used numbers of 100, 250, 500, 750 and 1000 particles. As the KPF is a probabilistic approach, the results vary for each iteration. Although we expect this effect to abate the more particles are used, it may be well noticeable for a small number of particles. Therefore we used 10 separate runs for each parameterization and evaluated the mean error and the variance. For each run, the same model pose is used for initialization. It has been manually determined to resemble best the ground truth marker configuration, much like Black and Sigal did to generate ground truth body poses. The mean shift is set to use a fixed number of three iterations. The parameters for scaling the cue likelihood transfer function σ_c and the importance reweighting factors λ_c of the image cues c have been set to a configuration known to track robustly as follows. Edge cue: $\sigma_E = 1,0$, $\lambda_E = 1,5$; Ridge cue: $\sigma_R = 1,5$, $\lambda_R = 1,0$; Mean cue: $\sigma_M = 35,0$, $\lambda_M = 0,5$; Skin cue: $\sigma_S = 0,3$, $\lambda_S = 1,0$.

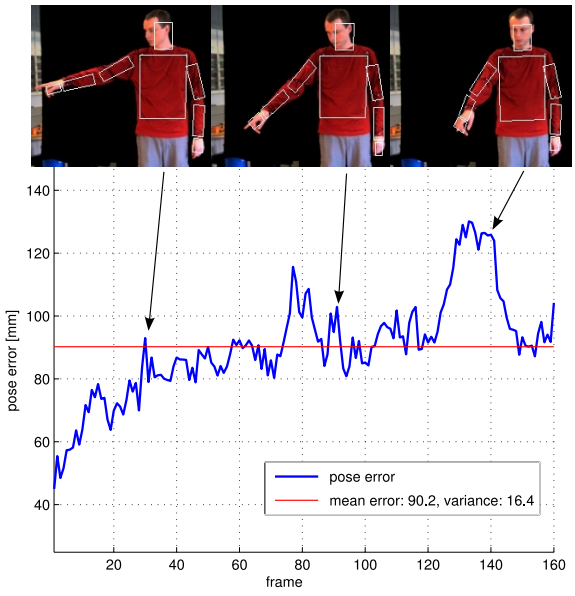


Figure 6.10: *Pose error over time.* Results from sequence A containing three pointing gestures with increasing difficulty. The last gesture produced the highest error.

For the experiment, two sequences from the corpus have been used. The first, sequence A, has a length of 161 frames. The subject performs three pointing gestures with the right arm towards objects positioned in the scene. This sequence is expected to be easier to track as no large self-occlusions occur. Sequence B, comprised of 118 images, is more difficult, as the subject performs two pointing gestures with the left arm towards objects right in front of the person. Thus the arm points directly towards the camera which is more ambiguous and temporarily occludes other body parts.

# particles	sequence A $\mu \pm \sigma$	sequence B $\mu \pm \sigma$
100	111,44±31,44	140,37±53,94
250	90,20±16,40	99,57±38,05
500	83,75±19,58	94,57±35,71
750	81,53±19,21	89,23±38,97
1000	78,19±17,17	88,76±32,9

Table 6.2: *Effect of the number of particles on the tracking accuracy.* Mean error and variance are given for two sequences while varying the number of particles.

To better understand the behavior of the tracking algorithm let us have a look at the results for the tracking error of sequence A plotted over time, as displayed in Fig. (6.10). Here, the system uses 250 particles. Shortly after initialization the error is smallest. That was expected, as the initial body pose was chosen to minimize the error. After a few frames the error stabilizes at a value of approximately 90mm, which represents the exactness that the algorithm is able to achieve in average. The three gestures pose an increasing difficulty to the tracking framework. For the first two gestures, the tracking error shortly raises during the translational movement of the arm. During the holding phases, however, it falls back to normal values. In the third gesture, the arm points more towards the camera. Here, the tracking algorithm is not able to recover the pose exactly, which results in a high position error of up to 128mm. Even though the tracking is not that exact, it is able to recover as soon as the disadvantageous pose is abandoned.

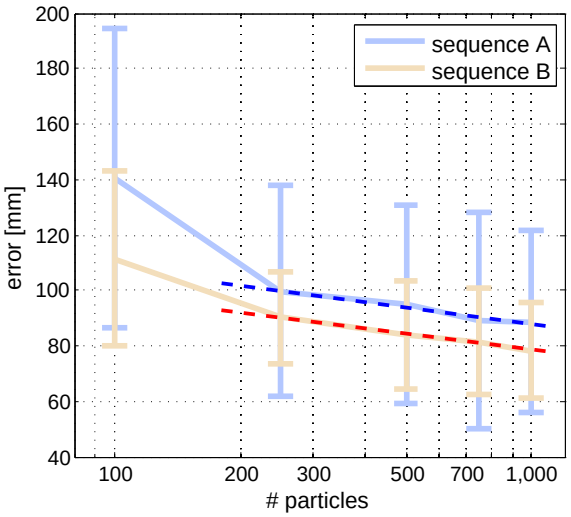


Figure 6.11: *Marker error on a logarithmic scale.* When plotted on a logarithmic scale, the mean error shows a linear behavior for 250 and more particles.

Now let us inspect the change of the error for different numbers of particles used. Tab. (6.2) lists the results for the two sequences with five parameterizations each. The error obviously decreases with more particles available to cover the search space. Fig. (6.11) reveals an even more interesting detail. For 250 particles and more, the error is linear on a logarithmic scale. This means for doubling the number of particles – and by that also doubling the calculation time of the algorithm – the error decreases only by a fraction. Here, the error for 500 particles is a mere 8% lower than for using 250 particles. Even using 1000 particles instead of 250 only lowers the error by 13% using up four times the computational resources. Note that these numbers are valid only for the current config-

uration of the system and the current image sequence. The general behavior, however, stays the same and has been observed throughout the whole evaluation and also became obvious during the calculations performed for other projects. The even bigger errors for 100 particles can be explained by the very sparse coverage of the parameter space that results from such a low number of points distributed in a 14-dimensional space. Assuming the particles are used to independently sample the individual parameter dimensions, this number would just suffice to put 7 sample points for each parameter. Although this example is constructed and the truth is that the particles can not be seen as being that independent from each other, the unexpectedly high error suggests that the algorithm often fails to find the correct pose. The similarly higher variances further support this claim.

Our findings coincide with the results that Bălan, Sigal and Black [18] published in 2005. They also claimed a logarithmic relation between the tracking exactness and the number of particles, although their optimization process uses a different particle filtering technique. The achieved tracking accuracy, however, is hard to compare. Bălan et al. used a multi-camera setup and they were only able to reliably track with at least three camera views for a longer period of time without losing tracking. For three cameras, the error can be constantly kept at 50mm or less. For the same reason it was not reasonably for us to use the HumanEva-I dataset to evaluate the body tracking framework. Stable observation periods would have been too short to provide meaningful results.

Summarizing, the results demonstrate that the kernel particle filter allows tracking of human motions in cluttered environments as long as no large self-occlusions occur. It works best for scenes with a simple background and for motions that do not produce ambiguous results. Consequently, with a standard camera and a laptop computer mounted onboard a mobile robot, our approach can be used for gesture recognition to improve human-robot interaction.

6.3 Automatic Parameter Optimization for Body Pose Tracking

The huge number of variables in the model and in the algorithm and their mutual dependencies complicate applying the body tracking system to new scenarios. In principle, the effect of all parameters are well known and can be adjusted precisely. In practice, it has shown that the same set of parameters does not always perform equally well in different scenarios and for different system configurations. The algorithms to be presented in this section have been implemented by Hermes and are described in more detail in [66]. The following explains how the principle of evolutionary computation can help to formulate a method to optimize the existing body tracking frameworks given an optimization criterion. Let us make this clear with two examples.

First, imagine a master student developing a new cool feature to be integrated into the tracking framework, for instance a texture descriptor. He develops and tests the feature on its own, it performs well and provides a robust segmentation between foreground and background. It is integrated into the tracking framework and at the next live demonstration a robust tracking is almost impossible. What has happened? The new

feature has been tested extensively on the video the student had used all the time and he also found a working parameterization for this one videos. But he never realized the fact that the outcome depends on the appearance of the scene and that the individual features might combine quite differently when applied to a new scenario. The reliability of the cues strongly depends on the appearance of the scene. The skin segmentation, for instance, sometimes produces false positives for wooden furniture, which then distracts the tracking process. You could blame this to be an error of the skin segmentation module, but the fact is that a perfect functioning of all cues is more an exception than the rule. Here, the system could use a learning phase to try out the newly integrated feature on a large pool of collected videos, which resemble a widespread variety of different scenarios, lighting conditions, cameras used, people observed and so on. Instead of finding one optimal parameterization for the tracking framework, the optimization method could help to find a set of parameters that generates robust results for many different scenarios.

Second, imagine that the body tracking system has been developed as a stand-alone component and is now planned to be integrated on a mobile robot with limited computational resources but which offers a clear scenario. Here, the optimization method can take the functioning algorithm and optimize its parameters to find a suitable trade-off between accuracy and computational complexity.

These examples make clear that we have to deal with a multivariate problem. Even if we assume that a single parameter has a big impact on the accuracy, for example the number of particles as presented in Sec. (6.2.3), some parameters exhibit a much more unpredictable behavior and are strongly interdependent, for instance the cue likelihood transfer function scaling parameters σ_c and the importance reweighting factors λ_c of the image cues.

6.3.1 Genetic Algorithms for Parameter Optimization

Genetic algorithms have shown to perform well on multivariate problems with strong nonlinearities in the parameters. We will now investigate how the body tracking as a system can be introduced as the subject of optimization to a genetic algorithm. An instance of the tracking system will therefore function as an individual of the genetic algorithm, the genetic material is determined by the parameters of the tracking system. A generation of individuals will therefore be composed of a number of instances of the body tracking system.

Defining a Fitness Function

First, we need to define a fitness function that represents how well the body pose tracking system – our individual – performs at the given task. The genetic algorithm expects the fitness value to increase with better performance and to lower otherwise. We intend to use the ground truth measure μ_{seq} presented in Eqn. (6.4) as the basis for the fitness value. From the examples we learned that a second figure to be taken into

account is the runtime of the tracking process. In is based on measuring the time τ_t needed to estimate a body pose at the time t. For a sequence with the length T, this leads to a mean time per frame

$$\omega_{seq} = \frac{1}{T} \sum_{t=1}^{T} \tau_t. \tag{6.5}$$

Unfortunately, both μ_{seq} and ω_{seq} and thus also $F_{\mu,\omega}(\underline{x})$ behave exactly the opposite than required, as a lower error means a better tracking result and a lower processing time means better performance. Therefore, we first need to transfer these measures into a fitness value. A simple negation of μ_{seq} and ω_{seq} would correct the gradient direction, but we also must obey the constraints selection operators, which demands the selection probability to be greater than zero. We chose a nonlinear scaling using a sigmoidal transfer function to best fit the demands of the genetic algorithm concerning the fitness function. The logistic function $\sigma_a(x)$ – also known as the basic form of the neuron activation function used in artificial neural networks – offers an almost linear projection for values close to zero ($\sigma'(0) = 1$) and an asymptotic behavior for large input values.

$$\begin{aligned} \sigma_a(x) &= \frac{1}{(1 + e^{-a \cdot x})} \\ \sigma'_a(x) &= a \cdot \sigma_a(x) \cdot (1 - \sigma_a(x)) \end{aligned} \tag{6.6}$$

We now search for a function $\mathrm{sc}_R(x)$ following the above conditions and with the characteristics

$$\mathrm{sc}_R(0) = R \quad \wedge \quad \mathrm{sc}'_R(0) = -1 \quad \wedge \quad \mathrm{sc}_R(\infty) = 0. \tag{6.7}$$

Here, R defines the maximum value of $\mathrm{sc}_R(x)$. The following function fulfills the requirements of a transfer function:

$$\mathrm{sc}_R(x) = 2R \cdot (1 - \sigma_{\frac{2}{R}}(x)), \tag{6.8}$$

where

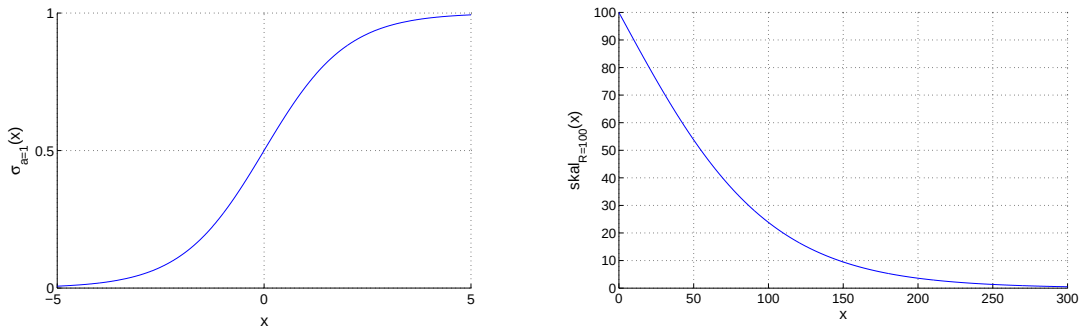
$$\tilde{\mu}_{seq} = \mathrm{sc}_R(\mu_{seq}) \tag{6.9}$$

represents the fitness value of the measure μ_{seq}, the fitness value $\tilde{\omega}_{seq}$ is calculated accordingly.

Using Eqn. (6.8), low values of the measures μ_{seq} and ω_{seq} are bring transfered into high fitness values, while large error values and a long processing time leads to low fitness values.

The two fitness criteria $\tilde{\mu}_{seq}$ and $\tilde{\omega}_{seq}$ can be used separately, e.g.for only optimizing the accuracy and ignoring the runtime, which then would be a uni-objective optimization problem. If both criteria are to be optimized, we talk about a multi-objective problem, which can be handled by defining a new criterion $\tilde{F}_{\tilde{\mu},\tilde{\omega}}(\underline{x})$ that combines the fitness values of $\tilde{\mu}_{seq}$ and $\tilde{\omega}_{seq}$ for a given set of parameters $\underline{x}$ as

$$\tilde{F}_{\tilde{\mu},\tilde{\omega}}(\underline{x}) = \alpha_F \cdot \tilde{\mu}_{seq}(\underline{x}) + (1 - \alpha_F) \cdot \tilde{\omega}_{seq}(\underline{x}), \tag{6.10}$$



(a) (b)

Figure 6.12: *Fitness scaling function.* (a) The logistic function $\sigma_a(x)$ offers the characteristics needed by the operators of the genetic algorithm. (b) The final scaling function $\text{sc}_R(x)$ can be used to transfer the results of the ground truth measure into fitness values.

where the weighting coefficient $\alpha_F \in [0,1]$ determines the importance of the two measures. A low value of α_F emphasizes the time criterion and forces the optimization to find a parameterization with a low computation time, while big values of α_F put the focus more on the exactness of the tracking. Most interesting are mid-range values, as they represent a trade-off between efficiency and accuracy. We found values of $\alpha_F \in [0.5, 0.8]$ to provide the reasonable results.

Computation Framework for a Genetic Algorithm

Now having a fitness function at hand that rates the performance of the body pose tracking, it can be used as the objective function for the optimization problem of finding the set of parameters that maximizes the fitness values. For the search process, we employ a genetic algorithm that is based on a series of generations consisting of evolving individuals. Here, each individual represents an instance of the body tracking framework. To determine its fitness for a given parameter vector, the tracking framework has to be evaluated on one or multiple sequences if images, comparing the tracking results with the marker-based ground truth. The genetic algorithm then uses the fitness values to breed a new generation using the mutation and selection operators while aiming at a higher mean fitness of the population.

In this optimization process, the body pose estimation with the tracking framework is the most time consuming part. For a single individual, a whole sequence of images has to be tracked, which can take several minutes to hours, depending on the length of the sequence and on several parameters that can be changed during optimization, as, for instance, the number of particles. Each generation again consists of a number of individuals each with a unique set of parameters. The advantage is now that these individuals are independent from each other and that the calculation can therefore easily be parallelized to speed up the total runtime of an experiment. We chose a client server architecture as the basis for the implementation of the computation framework that allows the distribution of the time-consuming tracking runs to multiple machines in a

local network of Linux machines.

The Server

The server provides a user interface to set up new experiments and is responsible for the administration and distribution of the upcoming calculation tasks. It holds a list of the machines currently available as clients and monitors them. Communication is based on XML-messages over the TCP/IP-network protocol. It allows to send an receive textual information as well as the exchange of serialized objects to ease the integration process of the different software modules. The server collects and integrates the results from the clients into the optimization procedure. As soon as the fitness values for all individuals of a generation are completely calculated, the genetic algorithm calculates a new generation of individuals whose parameter sets are again sent out to the clients. The server schedules the calculation tasks such that the results can be obtained as fast as possible. If one client does not complete the calculation of the tracking process for a long time, for instance due to a malfunction, the server submits the according parameter set to a different machine. If for a specific set of parameters, the clients never report back any results, the server assumes that this specific combination of parameters is prone to enforce an error – e.g. a deadlock – on the clients. The individual is then neglected in the genetic algorithm as a single individual only account for a small piece of information in the optimization process.

The Client

Each client communicates with the server, receives new parameter and image data and stores them locally. It then runs the tracking procedure while monitoring the computation time. Finally , it calculates the ground truth error, and stores and reports back the results to the server. Calculations on each client machine are run with a low priority in order not to disturb the local user. At wish of the user, calculation can also be totally stopped and the client can be excluded from the list of available machines. The client also offers the possibility to remotely monitor the progress of the tracking process. Each client sets up a new virtual desktop on which the IceWing tracking instance is started and that can be exported via a virtual network client to be viewed remotely, e.g. from the server machine.

Normalization of the Time Measure

For a faster completion of the genetic algorithm optimization process, using a big number of machines is desirable. Remembering the second optimization criterion, the time constraint that puts us in a difficult situation. Measuring the total processing time to complete a run of the tracking algorithm may not be the best choice to calculate this measure, as the result would then mainly depend on the performance of the machine. Using a slow machine, the tracking inherently takes longer than on a fast machine.

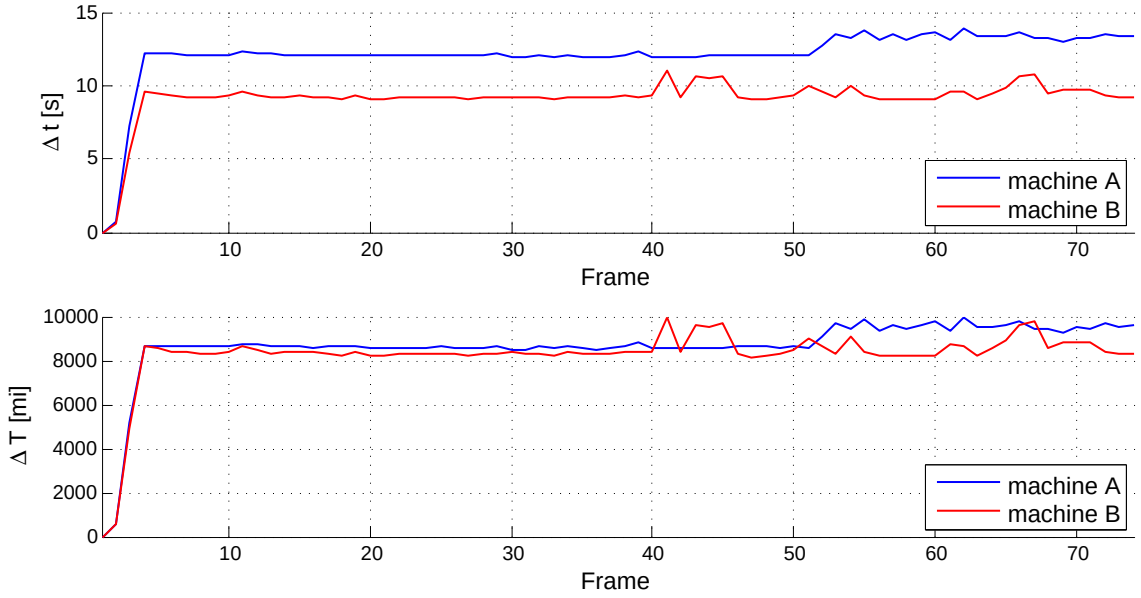


Figure 6.13: *Normalization of the time measure.* (top) Time measurements in seconds per frame for two machines using the same parameter set. Obviously, machine B is faster. (bottom) Normalizing the measurement reveals that both machines needed approximately the same number of instructions. The values are given in million instructions per frame.

We use the Whetstone-benchmark after Curnow [32], that repeatedly calculates a number of different arithmetic operations and averages the needed time to a MIPS-value (*millon instructions per second*) that represents the speed of the machine. This benchmark is run on each client and the time measurement is then normalized using the benchmark result. Fig. (6.13) shows the success of the normalization. Even though the two machines have different speeds, in absolute they need approximately the same number of operations to complete the tracking. Note that the probabilistic tracking technique is based on random processes and thus the computation time may vary for repeated runs of even for the same set of parameters. Also, fluctuations due to network traffic, hardware interrupts and processes of other user can not be regarded reliably. The only way to avoid mistakes due to outliers is to repeat the tracking a number of times and use the average or median measurement. 3-5 runs have already shown to be sufficient to mask out the worst effects, but that still means a threefold computational effort.

6.3.2 Parameter Optimization Results

To better understand the functioning of the optimization framework we will first present it as a tool to calculate a sequence of parameterizations without applying any optimization procedure. As a response to the two examples from the beginning of this section, we will begin with optimizing the parameters such to produce accurate results, secondly we will try to find a parameterization that represents a good trade-off between speed and accuracy.

Accuracy for Different Numbers of Particles

Similarly to the evaluation in Sec. (6.2.3), we will use the computation framework to perform a series of tracking runs while changing the number of particles employed. As the calculation is not deterministic, the measurement for each number of parameters is repeated 30 times, resulting in a total number of 750 tracking runs for the evaluation displayed in Fig. (6.14). As expected, the mean error decreases for a bigger number of particles. The increase of the standard deviation can be explained by the strong influence of the random effects in the tracking module. For a low number of particles, the tracking is never able to achieve good results, while using more particles performs better in average, but few runs using 1000 particles still perform worst than the best results for 100 particles. However, this could be an effect of the specific image sequence that has been used for this evaluation. Substantial statements could only be made if multiple image sequences posing different challenges to the tracking module were used, which was not possible due to the long processing time of the tracking framework.

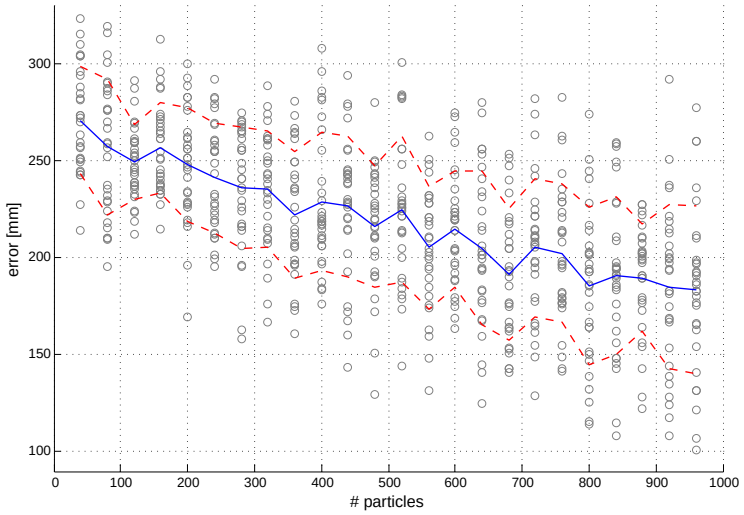


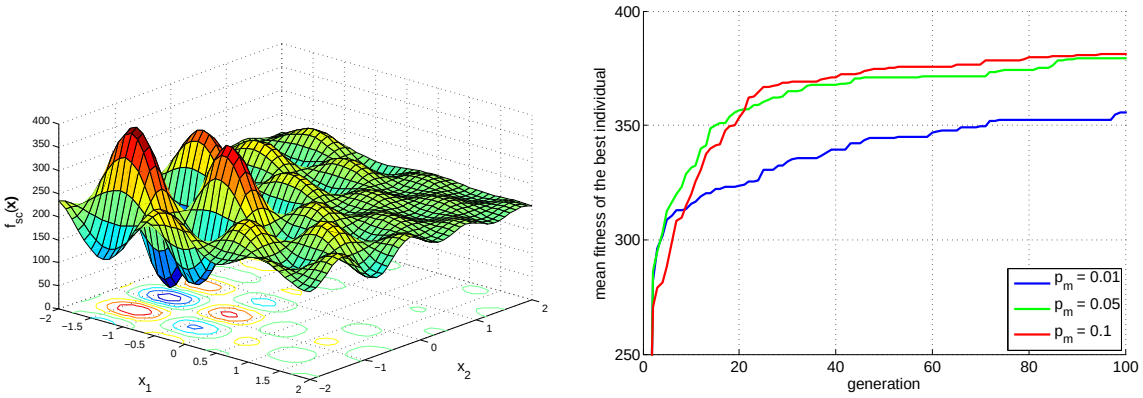
Figure 6.14: *Effect of changing the number of particles.* For each parameterization, tracking is carried out 30 times. The mean value (blue) clearly shows a higher accuracy when using many particles. The standard deviation (red) is also increasing.

Note that Fig. (3.1) has been created using the same technique. It displays the ground truth error values for varying the two parameters λ_R and σ_R of the ridge cue. The blue valley-like area represents good parameter combinations. Interestingly, a combination of $\lambda_R = 1.0$ and $\sigma_R = 1.5$ is almost equal to $\lambda_R = 2.0$ and $\sigma_R = 3.0$ in terms of a low error, but the latter combination can be expected to generalize better for other scenarios, as it offers a bigger margin for the two parameters and thus should perform more robustly.

Determining Operator Parameters with a Benchmark Function

After having verified the functionality of the computation framework we can now examine the optimization process for correct functioning. A benchmark function providing artificial and reproducible fitness values is used for testing. In contrast to the tracking framework, the benchmark function is very fast to calculate which eases trying out different setups for the parameters of the optimization framework. Exemplary, we will

here discuss the parameter for the mutation probability in more detail which has a default value of $p_m = 0.01$.



(a) (b)

Figure 6.15: *Benchmark optimization of the genetic algorithm.* (a) Fitness values of the scaled *Levy Nr. 5* test function $f_{sc}(\underline{x})$. (b) Optimization by varying the mutation rate p_m.

The benchmark function we employed for testing is the two-dimensional function *Levy Nr. 5* [121], which is scaled to a fitness function f_{sc} as follows:

$$\begin{aligned} f(\underline{x}) &= \sum_{i=1}^{5} [i \cdot \cos((i-1)x_1 + i)] \cdot \sum_{j=1}^{5} [j \cdot \cos((j+1)x_2 + j)] \\ &\quad + (x_1 + 1{,}42513)^2 + (x_2 + 0{,}80032)^2 \qquad (6.11) \\ f_{sc}(\underline{x}) &= -f(\underline{x}) + 220 \qquad (6.12) \end{aligned}$$

The scaled Levy-function expects a two-dimensional parameter vector $\underline{x} = (x_1, x_2)^T$, its response is depicted in Fig. (6.15(a)) for the range of input values $-2 \leq x_{\{1,2\}} \leq 2$. It features a number of local maxima that easily distract the optimization from the correct global maximum $\underline{x}^*$ at $(-1.31, -1.42)^T$ with a value of $f_{sc}(\underline{x}^*) = 396.14$.

A population size of 10 individuals with randomly chosen positions for initialization is used for the genetic algorithm. The mentioned range $-2 \leq x_{\{1,2\}} \leq 2$ has also been used as boundary constraints. For the two-dimensional parameter space, we deliberately chose a small number of individuals to simulate the sparse distributions of particles in the later highdimensional parameter space. Three different values for the mutation probability $p_m = \{0.01, 0.05, 0.10\}$ are evaluated. Each optimization has been repeated 30 times for a duration of 100 generations. For each generation, the fitness value of the best individual has been chosen and is averaged over the number of repetitions. The results are depicted in Fig. (6.15(b)).

It shows that for a mutation rate of $p_m = 0.01$ the optimization almost gets stuck and converges only slowly. Both of the other values perform much better. It is remarkable that doubling the mutation rate from $p_m = 0.05$ to $p_m = 0.10$ hardly shows any effect on the convergence behavior and produces similar results. Although $p_m = 0.10$ leads to a slightly higher final fitness, we chose $p_m = 0.05$ for further evaluations as it produced slightly better results in early stages of the optimization process.

(a) (b) (c)

Figure 6.16: *Streams used for parameter optimization.* Some of the occurring challenges are (a) partial occlusions, (b) pointing gestures, (c) wide movements.

Optimizing for Accuracy

The previous evaluations have shown the functionality of the genetic algorithm. The following evaluation aims at simultaneously optimizing multiple parameters of the body pose tracking framework. The parameters to be optimized and the boundary constraints are displayed in Tab. (6.3). The experiment is divided in two parts. In a first step, a general parameter combination is found, for which in a second step selected parameters are further refined.

parameter	min	max
# particles	1	2000
# meanshift iter.	1	10
H_h	0,1	10
σ_E	0,01	3,0
σ_R	0,01	3,0
σ_M	0,1	500,0
ρ_M	0,0	200,0
σ_S	0,01	5,0
λ_E	0,01	10,0
λ_R	0,01	10,0
λ_M	0,01	10,0
λ_S	0,01	10,0

Table 6.3: *Boundary constraints for the body pose tracking parameters.* The parameters to be optimized by the genetic algorithm have direct influence on the image cues calculating the pose likelihoods.

This experiment is reduced to use only a single image sequence, Fig. (6.16(a)) depicts an example image. The decision to base the optimization on a single sequence has been made to compensate for the long processing time. Furthermore, we are interested in a coarse estimation of suitable parameters, a more detailed examination will be carried out in the next experiment.. The genetic algorithm is set up to use a population size of 20 individuals that are allowed to develop for 50 generations. For a single individual the calculation of the fitness value is not repeated to reduce the workload. Still, the two presented experiments presented in this paragraph took several days to complete using all available machines (20 or more) in our working group. As we deliberately failed

to define a break condition, the optimization always is carried out for the maximum number of generations unless the researches who conducts the experiments stops it premature. The best individual that has ever existed in any generation will then be used as output providing a new optimized set of parameters for the tracking framework.

parameter	mean value
# particles	1170,24 ± 64,40
# meanshift iter.	4,17 ± 0,65
σ_E	1,31 ± 0,39
σ_R	0,95 ± 0,07
σ_M	143,14 ± 67,02
ρ_M	82,32 ± 11,14
σ_S	2,80 ± 0,52
λ_E	3,81 ± 0,62
λ_R	7,66 ± 0,36
λ_M	7,42 ± 0,45
λ_S	6,82 ± 0,86

Table 6.4: *Best parameter configuration.* Averaged from all individuals of generation 27.

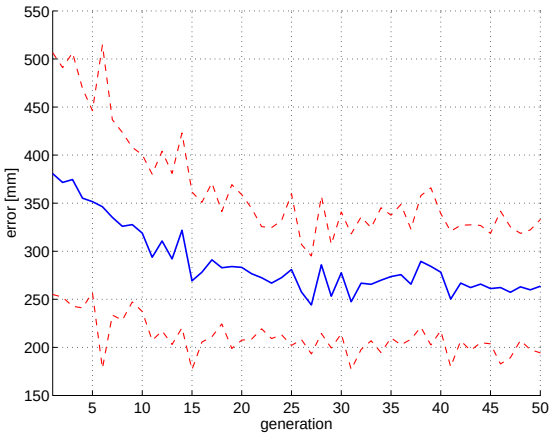


Figure 6.17: *Development of the parameter optimization.* Mean error for generation 27: 244.13mm.

The development of the average error μ_{seq} for all individuals is depicted in Fig. (6.17). After 30 generations, some kind of saturation effect seems to begin, possibly due to a local minimum. The best average result is produced by generation 27 with a mean error of 244.13mm, the averaged parameters for this generation are shown in Tab. (6.4).

As we learned from the evaluation in Sec. (6.3.2), using only a single image sequence bears the risk of losing generalizability. Therefore a second experiment is set up to use three sequences for the parameter optimization. They comprise postures and gestures of varying difficulty, while the overall appearance has been kept equal. Example images from the sequences are shown in Fig. (6.16). This second parameter optimization experiment uses 50 individuals per generation and the calculation of the fitness values is repeated 9 times for each individual. As for one fitness calculation, the three streams are not all evaluated in sequence but one is randomly chosen by the optimization framework, repeating the calculation 9 times means that each sequence is in average repeated three times for one individual, although the actual number may vary. In contrast to the former experiment, we chose only the cue fusion parameters σ_c and λ_c to be optimized, which slightly reduces the parameter space and also takes out the number of particles as a free parameter, which otherwise surmounts the other parameters concerning its influence for the final result. Therefore, a fixed number of 1000 particles and three mean shift iterations has been used. The results from the last experiment are used as initial values for the parameters.

Fig. (6.18) depicts the development of the error during optimization. As we already chose an optimized set of initial parameters, convergence is much slower than for the former experiment. The lowest error of 177.10mm can be found for generation 46. However, this error value only partially reflects the performance of the new param-

parameter	mean value
# particles	1000 fixed
# meanshift iter.	3 fixed
σ_E	1,56 ± 0,44
σ_R	1,78 ± 0,28
σ_M	362,43 ± 64,62
ρ_M	101,71 ± 18,54
σ_S	2,77 ± 0,61
λ_E	6,16 ± 1,38
λ_R	5,57 ± 1,57
λ_M	5,33 ± 0,91
λ_S	5,04 ± 1,24

Table 6.5: *Best parameter configuration using three sequences.* Averaged over all individuals of generation 46.

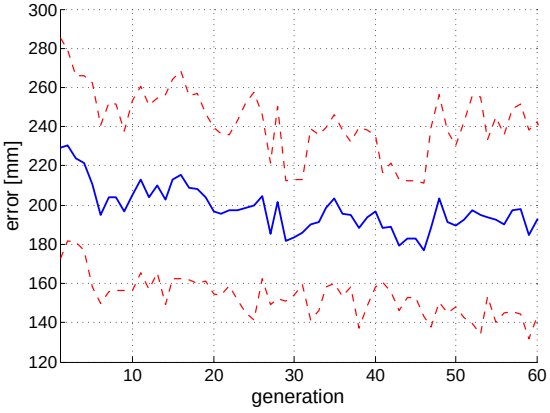


Figure 6.18: *Error of the parameter optimization using three sequences.* Mean error for the best generation # 46: 177.10mm.

eter combination, as it has been determined from three sequences each with its own challenges. Tab. (6.6) details the achieved accuracy for each sequence. The results have been obtained by averaging over 30 calculations of the tracking error for each sequence using the resulting parameter set as shown in Tab. (6.5).

sequence	error [mm]
A	94,43 ± 14,22
B	193,83 ± 32,41
C	200,48 ± 123,18

Table 6.6: *Error for the three streams in detail.* While the optimization has been successful for sequence A, the other two sequences still produce a higher tracking error with the same set of parameters, which shows the varying difficulty.

Optimizing for Speed and Accuracy

For the last experiment concerning the automatic parameter optimization, we will explore the effect of integrating an additional time criterion in the fitness function $\tilde{F}_{\tilde{\mu},\tilde{\omega}}(\underline{x})$, as described in Eqn. (6.10). The scaling factor for the influence of the temporal criterion has been set to $\alpha = 0.7$. The parameters to be optimized here are those with the greatest impact on the total calculation time of the tracking algorithm, namely the number of particles, the number of means shift iterations and the kernel bandwidth of the mean shift. The last two parameters are tightly coupled and are therefore both chosen for optimization. As before, the optimization has been performed using the three sequences, the population size is 20, and the calculation of the fitness values is also repeated 9 times for each individual. The remaining parameters are set to the optimal values as determined by the last experiment.

The results for the optimization combining the spatial and temporal criteria are quite surprising. Even though we initialized this optimization run with already optimized parameters, a slight improvement is still visible, especially for the first few generations, see Fig. (6.19). After that, the search process seems to drift away from the local maximum

parameter	mean value
# particles	749,09 ± 47,56
# meanshift iter.	1,27 ± 0,55
H_h	5,06 ± 1,34
σ_E	1,56 ± fixed
σ_R	1,78 ± fixed
σ_M	362,43 ± fixed
ρ_M	101,71 ± fixed
σ_S	2,77 ± fixed
λ_E	6,16 ± fixed
λ_R	5,57 ± fixed
λ_M	5,33 ± fixed
λ_S	5,04 ± fixed

Table 6.7: *Best parameter configuration combining spatial and temporal criteria.* Averaged over all individuals of generation 59.

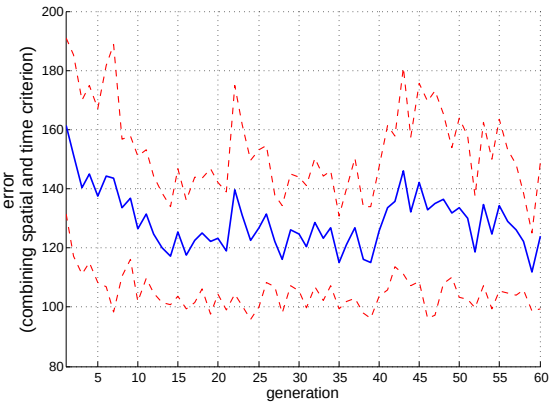


Figure 6.19: *Error of the parameter optimization combining spatial and temporal criteria.* Mean error for the best generation # 59: 111.86. Note that the fitness values have been back-transformed into the error domain for better readability.

as the increase in both error and variance suggest. What happens here, is that the number of particles is steadily decreased which on the one hand results in a higher fitness of the time criterion, but at the same time makes tracking much harder. Interestingly, the optimization finds an better configuration in the later steps which also offers a low variance. The mean error is even significantly lower compared to the previous results.

	spatial criterion only		spatial + temporal criterion	
	mean error [mm]	time per frame [s]	mean error [mm]	time per frame [s]
sequence A	94,43 ± 14,22	4,46 ± 0,01	83,62 ± 12,57	1,97 ± 0,01
sequence B	193,83 ± 32,41	4,50 ± 0,02	196,55 ± 21,21	1,98 ± 0,01
sequence C	200,48 ± 123,18	4,46 ± 0,02	204,97 ± 141,64	1,98 ± 0,01

Table 6.8: *Comparison of optimization results combining spatial and temporal criteria.* Adding the time criterion yields a comparable accuracy but twice the speed of the original parameter set. Values are averaged over 30 runs using the best parameter set. Results from Tab. (6.6) are repeated for a better comparability.

The biggest surprise, however, holds the evaluation of the runtime of this configuration. The runtime is more than cut in half while the error stays almost the same. As before, Tab. (6.7) is generated by averaging over 30 runs independently for each sequence. The runtime measurement reflects the mean time the tracking framework needed for each frame. The shorter runtime corresponds to the lower number of observations that had to be carried out for each frame. The previous experiment used 1000 particles and three mean shift iterations, totaling to 3000 observations per frame, the new setup proposes 749 particles $\times 1.27$ mean shift iterations which makes a total of 951.23 observations. An explanation for the good performance needing less particles than before can be found in the kernel bandwidth H_h, which determines the scaling of the Epanechnikov kernel that defines the neighborhood for the calculation of the mean shift vector. Its initial value of 1.0 has been scaled up to 5.04. This means that each particle has a bigger support which means that the mean shift algorithm is enabled to function even for a sparse coverage of

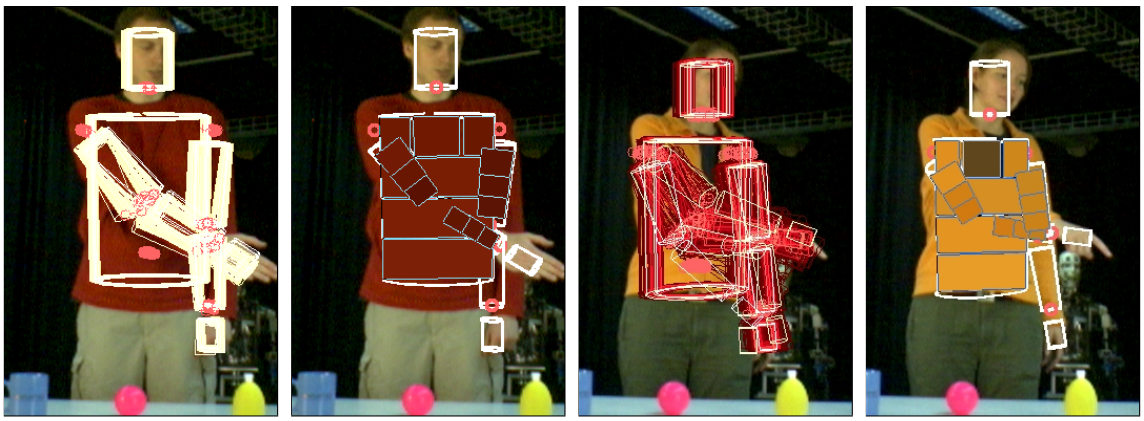
the parameter space.

A question that may arise at this point is why this combination of a low number of particles combined with a high accuracy has not been found before, as it obviously even outperforms the result of the optimization without the time constraint which we already thought to be optimal. In principle, all the parameters except the kernel bandwidth were free to be optimized in the first experiment. But exactly this single parameter seems to play an important role in combination with the functioning of the mean shift in sparsely covered parameter spaces. It must be said at this point that all the parameters in question were thoroughly tested even before the automatic optimization had been created. A finding like this, or remember also the discovery of the more robust parameter combination for the ridge cue as depicted in Fig. (3.1), only shows that some of the interrelations between the parameters may be hard to predict or may be just a case of "Never thought of that".

Although a numerical comparison to related body tracking approaches stays difficult, as the applied techniques differ in too many points, a calculation time of two seconds per frame given the achieved exactness can be seen as a good result. As presented in other parts of this thesis, the speed may be increased by using even less particles and constraining the search space but that also lowers the accuracy. Chen et al. [24] presented an investigation on parallelizing the body tracking using multiple cores. They applied the annealed particle filtering method of Deutscher et al. [37] and optimized the code for both, the vision processing and the particle filtering steps. The fastest version needed 3.2s per frame with 8 cores. For a single core, the computation time increased to 21s per frame, but it must be mentioned that they used images from four cameras, which naturally is more complex than using a single image. Bălan et al. [18] support this claim. They achieved low tracking errors of 41mm in average on the HumanEva dataset, but only for using three or more cameras. The error of 283mm they give for a single camera must not be taken seriously, as it is just the result of the tracking drifting further and further away from the correct position. Thus, a mean error between 80mm and 200mm is still comparable taking into account the extreme challenges of a monocular tracking that these approaches are not able to comply with. Many different approaches to achieve a fast monocular tracking have been proposed recently [19, 103, 145], but they usually rely on strong constraints concerning the appearance and the scenario or predefined motions. The presented system, in contrast, has been designed to initially pose as few constraints as possible. If a fast calculation is required, the system can be adapted by introducing similar constraints and then using the automatic optimization procedure to find an optimal parameter configuration under these constraints.

6.4 Evaluating the Automatic Initialization Procedure

The following section evaluates the automatic initialization procedure that has been proposed for the body pose tracking algorithm as presented in Sec. (5.5). Given the position of the face in an image, either the height of the person or the distance to the camera can be determined due to the monocular setup. For the experiments we assumed a fixed height for the person, thus only varying the distance during initialization.



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

Figure 6.20: *Body tracking results using automatic initialization.* Results for subjects B (a-b) and C (c-d) from tracking with learned body model: (a) and (c) show the most likely poses, (b) and (d) the final tracking results.

For evaluation, the system has been applied to image sequences of three persons pointing at objects on a table with 836 frames in total. Ground truth has been generated by manually annotating the position of the hands and the head. To show the effectiveness of the presented approach both for initialization and recovery from tracking failures, the automatic initialization setup is compared to the manually initialized setup as described in [136]. For the latter, we still need a relatively high number of particles to ensure robust tracking over a longer image sequence, furthermore it is necessary to adjust the measures of the body model for each person accurately.

The system is configured in two ways: 1) 1500 particles and 6 mean shift iterations, 2) 500 particles and three meanshift iterations. The automatically initialized system is also set up to use 500 particles, three mean shift iterations but uses a generic body model for all subjects. For each iteration, $\alpha = 5\%$ recovery particles are inserted into the tracking distribution. All setups employ the same cues with identical parameterization.

		manual initialization				automatic init	
		1500 particles		500 particles		500 particles	
sequence	# pict.	RMSE	σ	RMSE	σ	RMSE	σ
subj A	318	18.73	13.87	52.65	41.31	30.11	26.05
subj B	242	14.52	8.58	32.86	23.26	21.96	22.13
subj C	276	12.04	10.14	72.64	38.30	54.82	57.76

Table 6.9: *Position error using automatic initialization and error recovery.* RMSE (root mean squared error) position error and standard deviation σ given in pixel. Comparison of three setups: manual initialization with 1500 particles and with 500 particles, automatic initialization and error recovery with 500 particles. The colored mean error values for subject B can also be seen in Figure 6.21.

Figure 6.20 shows typical tracking results using the automatically acquired body model. For Fig. (6.20(a)) and Fig. (6.20(c)) the most likely poses are colored white, less likely poses red. Note the multi-modal distribution in Fig. (6.20(c)) due to ambiguous measurements of the pose. Fig. (6.20(b)) and Fig. (6.20(d)) show the final tracking result and the learned shirt color.

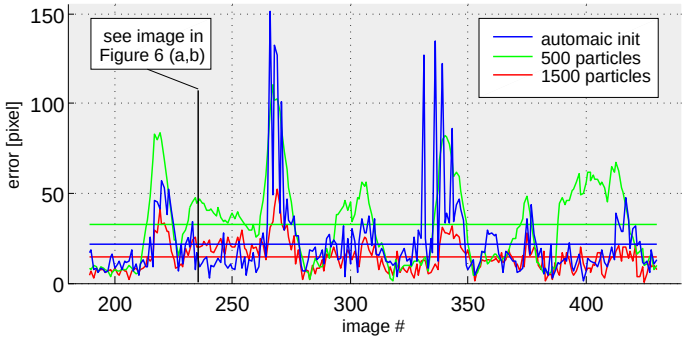


Figure 6.21: *Tracking quality for three setups..* Comparing tracking errors of subject B for manual and automatic initialization setups. Also compare to Table 6.9. Note the tracking loss around frames 265 and 340 and the recovery afterwards.

Table 6.9 shows the tracking quality as the differences between the annotated position of the right hand and the model position projected into the image plane as RMSE and standard deviation in pixel for each sequence. For the presented approach, the RMSE stays between 20 – 55 pixel, which is accurate enough for detecting gestures in a human robot interaction scenario [61]. In contrast to the existing tracking approach, the standard deviation is much higher for the presented approach, which suggests that the tracking suffers from losses but is able to recover again as depicted in Figure 6.21 taking subject B as an example (blue line). Tracking gets lost around frames 265 and 340, but as soon as the situation gets less complicated, the inserted recovery particles are able to guide the tracking towards the correct pose again. Thus, the loss is only temporary and results in a comparably low RMSE as for 1500 particles (red line), but using only a third of the number of particles. Employing an identical parameterization using 500 particles but without the automatic initialization and recovery behaviour leads to an even less accurate tracking (green line) with a 30% higher error in average. For subject C, the system tends to lose tracking over and over again for sophisticated postures, e.g., the hand pointing directly towards the camera, a situation where the recovery component also does not work resulting in high errors for both approaches. The scaled standard body model does not suit this subject well enough. This is also reflected by the high RMSE of more than 70 pixel. Actually, the tracking was stuck in one position while the person moved, producing varying error values. Robust tracking for this subject is nevertheless possible. The former approach applied a personalized body model and an increased computational effort, yielding accurate results.

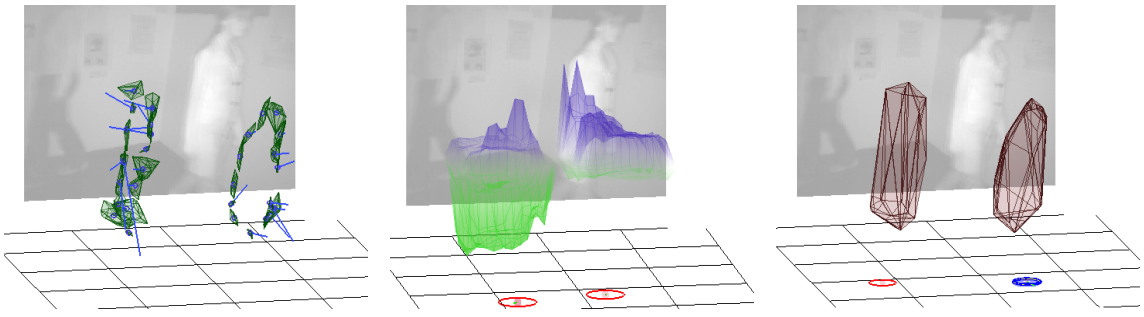
This clearly shows the limits of the presented approach adapting a generic model and calls for automatically adapted model kinematics and limb sizes. The usability of the presented system for the human robot interaction is still much higher compared to the former tracking approach. Persons interacting with such a system now have the possibility to repeat unrecognized gestures if tracking has been lost unintentionally.

7 Applications

In the scope of this chapter, the results for applying the presented techniques in different applications are presented. While the last chapter was meant to provide quantitative analyses of the accuracy and speed of the developed approaches, here, an overview will be given over applications, scenarios, and systems, where the person localization and the body pose tracking played an important role. The works presented in this chapter have been realized using a variety of tools and frameworks, e.g. Matlab[1] for the localization modules, IceWing[102] and OpevCV [78] for the image processing of the body pose tracking, and XCF [173] for communication between the individual modules.

7.1 Person Localization for Scene Reconstruction

For a mobile robot destined to interact with people acquiring information about its environment is a crucial process. Localizing possible interaction partners and the ability to perceive its surrounding enables a robot to build up and share its representation with the human. Recalling the scene exploration scenario described in Sec. (4.1.2) we will now briefly present the results of applying the person localization and tracking system in this scenario. The following evaluation have mainly been carried out by Swadzba and Beuter, see [158] for a more in-depth description. They are presented here anyway to underline the usefulness and the ease of integration of the person localization for the scene reconstruction system.



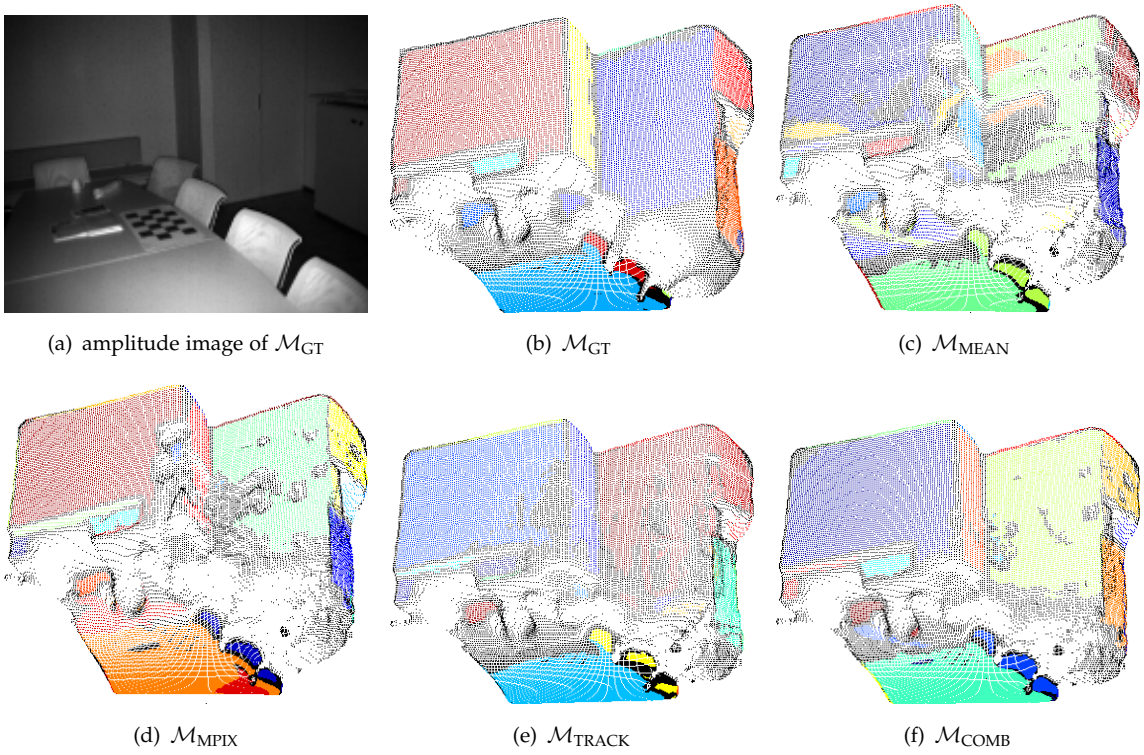
(a) (b) (c)

Figure 7.1: *Detecting two persons walking through a corridor.* (a) Clusters with velocities. (b) The blue and green error surfaces represent the observation function $\rho(s^{(k)})$, see Eqn. (4.7), evaluated for hypotheses with opposing velocities. Using multiple mean shift iterations, the hypotheses are shifted towards the local maxima of the density. (c) Objects found by searching for modes in the density and their convex hulls.

Reconstructing an environment is trivial for static scenes as a simple integration over time is often sufficient both for image and 3D data. The situation gets more complex

[1] `http://www.mathworks.com/`

if neither the scene is static nor the duration of the recording lasts longer than a few frames. Simple integration produces in severe reconstruction errors, as moving objects will misleadingly be added to the background model. Localizing and tracking those objects is an important feature in itself, as presented in [137], furthermore it allows the reconstruction module to neglect data emerging from moving objects, yielding more exact results. In this paper we are concentrating on reconstructing a static scene model whereas trajectories of moving objects are also an important information for a robot in order to keep track of events occurring in the robot's environment.



(a) amplitude image of $\mathcal{M}_{\text{GT}}$ (b) $\mathcal{M}_{\text{GT}}$ (c) $\mathcal{M}_{\text{MEAN}}$

(d) $\mathcal{M}_{\text{MPIX}}$ (e) $\mathcal{M}_{\text{TRACK}}$ (f) $\mathcal{M}_{\text{COMB}}$

Figure 7.2: *Room reconstruction results under different motion filtering techniques.* The different colors encode planar surfaces extracted on the generated 3D background models. (a) Amplitude image of the ground truth. (b) 3D point cloud of the ground truth $\mathcal{M}_{\text{GT}}$. (c) Background model $\mathcal{M}_{\text{MEAN}}$ produced by averaging over the entire sequence without excluding any points. (d) Background model $\mathcal{M}_{\text{MPIX}}$ excluding points with large motion. (e) Background model $\mathcal{M}_{\text{TRACK}}$ excluding points from tracked objects. (f) Background model M_{COMB} resulting from our combined approach.

As presented in Sec. (4.2), our proposal for reconstructing the static background of a dynamic environment is a combination of tracking dynamic objects and reconstructing the static parts to a complete scene representation. The person localization and tracking is able to robustly segment the 3D point cloud, see Fig. (7.1(a)), and to apply the weak object model to generate hypotheses of the possible location of persons in the scene, see Fig. (7.1(b)). Finally, the objects are tracked using the particle filter, see Fig. (7.1(c)), and the information about all tracked objects is forwarded to the scene reconstruction and analysis module, see also Sec. (4.2).

Figure 7.2 shows the reconstruction results for a sequence where a person passes the camera while picking up a chair. The empty scene is depicted in Fig. (7.2(a)), generated

$\bar{e} \pm \sigma$	scene 1	scene 2a	scene 2b	scene 2c
$\mathcal{M}_{\text{MEAN}}$	259 ± 238	93 ± 120	125 ± 201	94 ± 139
$\mathcal{M}_{\text{MPIX}}$	112 ± 227	27 ± 74	90 ± 221	37 ± 91
$\mathcal{M}_{\text{TRACK}}$	44 ± 65	46 ± 88	46 ± 98	58 ± 103
$\mathcal{M}_{\text{COMB}}$	33 ± 26	22 ± 71	41 ± 105	21 ± 175

Table 7.1: *Reconstruction accuracy for the different approaches.* Mean error $\bar{e}$, see Eqn. (7.1), and standard deviation σ of the four models $\mathcal{M}_{\text{MEAN}}$, $\mathcal{M}_{\text{MPIX}}$, $\mathcal{M}_{\text{TRACK}}$, and $\mathcal{M}_{\text{COMB}}$ compared to the ground truth $\mathcal{M}_{\text{GT}}$. All values are given in millimeter (mm). The combined method $\mathcal{M}_{\text{COMB}}$ performs best for all scenes.

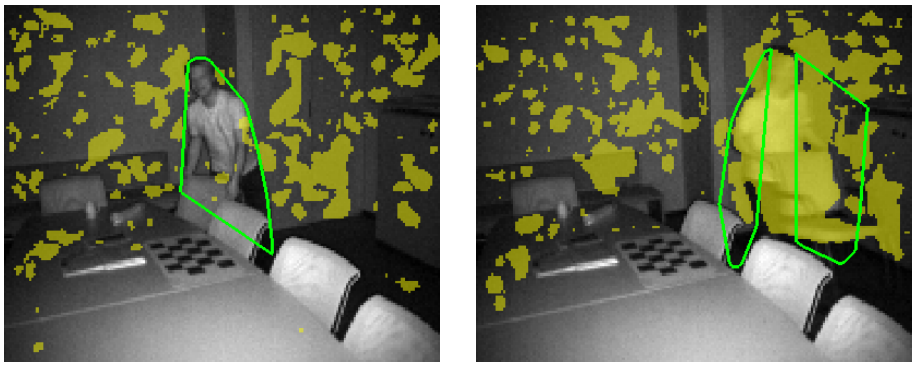
by acquiring data from the scene without any moving objects. Four different approaches for reconstructing the static scene are compared to each other. We start with the simplest approach where the frames of the entire sequence are averaged without removing any 3D data per frame, which is referred to as $\mathcal{M}_{\text{MEAN}}$, see Fig. (7.2(c)). The next approach is to remove from each frame those 3D points annotated with a velocity vector larger than a certain threshold – in the following named as *moving pixels*. Here, for each pixel the motion is computed by determining the optical flow at each pixel taking into account the current and the proceeding frame. The resulting model will be referred to as $\mathcal{M}_{\text{MPIX}}$, see Fig. (7.2(d)). The second approach is neither able to detect non-moving persons within few frames nor able to distinguish between motion produced by moving objects and motion introduced from noise. Tracking of moving objects using spacetime in order to gain reliable motion information as presented in Section 4.3.2 enables us to build a more sophisticated model $\mathcal{M}_{\text{TRACK}}$, see Fig. (7.2(e)). Combining tracking of moving objects and rejection of moving pixels leads to an approach which is not sensitive to failures during tracking producing an even better model $\mathcal{M}_{\text{COMB}}$, see Fig. (7.2(f)).

The models are compared to the ground truth $\mathcal{M}_{\text{GT}}$ by computing a mean error. As each 3D point is associated with a pixel of a 2D image, both in the model and the ground truth, the Euclidean distance between the model 3D point and the ground truth 3D point for each pixel is computed. The quality is indicated by the mean error $\bar{e}$

$$\begin{aligned} \bar{e} &= \frac{1}{|i|} \sum_i |\vec{p}_i^{\,\text{GT}} - \vec{p}_i^{\,\text{M}}| \qquad (7.1) \\ \text{M} &= \{\text{MEAN}, \text{MPIX}, \text{TRACK}, \text{COMB}\} \\ i &: \text{index of valid pixels (3D points, respectively)} \end{aligned}$$

and the standard deviation σ of all valid pixels $\vec{p}_i^{\,\text{M}}$.

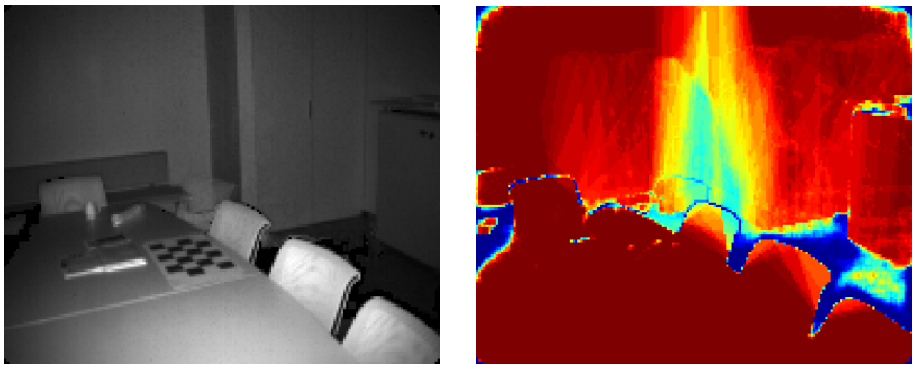
As shown in Tab. (7.1) and Figure 7.2 the combined model $\mathcal{M}_{\text{COMB}}$ estimates the ground truth model $\mathcal{M}_{\text{GT}}$ best with a small mean error $\bar{e}$ of 21mm compared to the three other models – $\mathcal{M}_{\text{TRACK}}$ with 58mm, $\mathcal{M}_{\text{MPIX}}$ with 37mm and $\mathcal{M}_{\text{MEAN}}$ with 94mm mean error. $\mathcal{M}_{\text{COMB}}$ uses the advantages of both methods which is tracking of moving objects and removing 3D points annotated with large velocity vectors. As $\mathcal{M}_{\text{MPIX}}$ and $\mathcal{M}_{\text{MEAN}}$ are generated from methods not based on tracking they suffer from the problem that situations like standing persons cannot be handled well and therefore these persons are integrated into the scene model since less optical flow/3D motion can be observed between two consecutive frames. This problem can be resolved by tracking moving



(a) frame 14 (b) frame 43

Figure 7.3: *Functioning of the combined approach.* Green polygons indicate the convex hulls from the object tracking approach, the yellow highlighted areas contain points annotated with large motion vectors. (a) The person is standing still, it can still be tracked robustly and thus be neglected from reconstruction. (b) The person is tracked but separated into two objects. The motion information from the optical flow corrects the missing object information for the central part of the body.

objects which also leads to the detection of temporarily standing persons as depicted in Fig. (7.3(a)). Also, excluding 3D points with large motion vectors is beneficial for the background extraction in the case of failed tracking or splitting of a person into two objects, as shown in Fig. (7.3(b)). $\mathcal{M}_{\text{COMB}}$ determines the points on the object between the two polygons by considering additionally optical flow on the entire image. The standard deviation of our reconstruction is quite small which indicates that the ground truth is estimated quite well even considering the extensive motions involved.



(a) virtual amplitude image (b) reliability map

Figure 7.4: *Reliability of the reconstruction results.* (a) The reconstructed amplitude image for $\mathcal{M}_{\text{COMB}}$, (b) the corresponding reliability map, with the color encoding the number of measurements fused. Red color depicts regions with many data present and therefore a reliable reconstruction, and green towards blue becoming less and less reliable. Note that the floor has a low reliability as the carpet absorbs the infrared light from the Swissranger sensor and provides only poor measurements. Furthermore, the person stood still in the central part of the image and has been neglected during reconstruction, leaving only some - but still enough - points for a robust reconstruction.

Fig. (7.4(b)) shows a reliability map of our model $\mathcal{M}_{\text{COMB}}$. The color encodes the amount of measurements which contributed to a certain model point (red: a lot of measure-

ments, blue: only few measurements). It can be seen that only few measurements from the ground floor and the edges could be collected due to the poor reflectance properties and big amount of motion arising from the persons' feet. The walls and the table, however, provide a lot of stable measurements. Additionally, it can be seen that between the corner and the table less measurements are collected as this is the place where the person has been standing for a short period of time. Still, gaining only few measurements is not a problem since only reliable measurements are chosen and therefore a stable 3D background model can still be computed.

7.2 Body Pose Tracking for Object Attention

Intended to be a personal robot companion, the mobile robot BIRON (**Bi**elefeld **Ro**bot Companio**n** [62]) is a research and evaluation platform used in human-robot studies focused on social interaction. The basic functionalities of our robotic interaction system are a model for human awareness, the ability to navigate in unknown environments and a dialog system for naturally spoken language, see also [150]. The following section presents the vision system that is used for resolving object references, it has been presented at the IROS 2008 conference [135]. The vision components presented here are designed to be integrated into a multimodal system that can bear the requirements of human-robot interaction in everyday life.

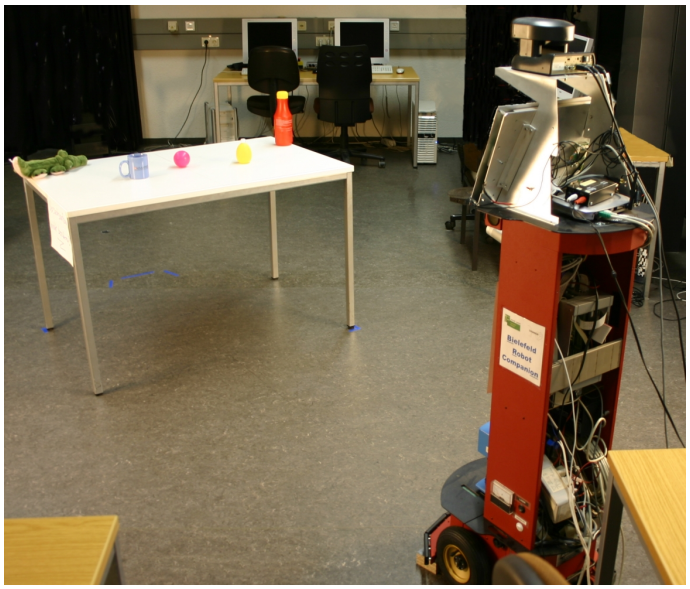


Figure 7.5: *Object attention scenario.* The mobile robot BIRON is facing a table with different objects. The person interacting with the robot will stand behind the desk and point at the objects.

The contribution of this work can be seen as part of a bigger learning scenario: Imagine the robot to be placed in an unknown environment, e.g. the first time to see the flat of the human owner. The task here is to let the robot learn new objects by pointing at them, therefore having the human and the robot observing each other and interacting on objects in the real world. The vision system provides information about the environment, especially about the gestures of the human interaction partner.

7.2.1 Object Attention System Overview

Regarding the technical basis, we use two cameras, one for recognizing the human and his gestures and one active camera for the acquisition of closeup images of objects. For body tracking an Apple iSight (body tracking camera) is used, whose field of view is wide enough to completely observe the upper body of the interaction partner at a distance of 2.0 - 2.5 m. The second camera (object camera) is a Sony Evi pan-tilt camera, utilized to get a closeup of the referenced object. Calibrating both cameras with respect to the robot's coordinate system allows the object camera to aim at the target position of a pointing gesture seen from the body tracking camera. The software basis for the integrated system presented in this paper is provided by the XML-based communication framework XCF [173] and the image processing framework IceWing[102].

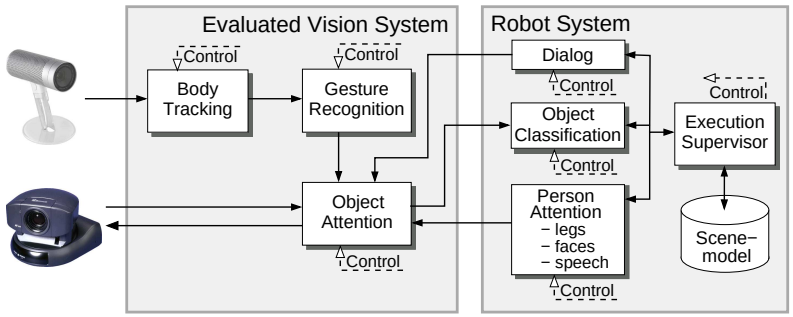


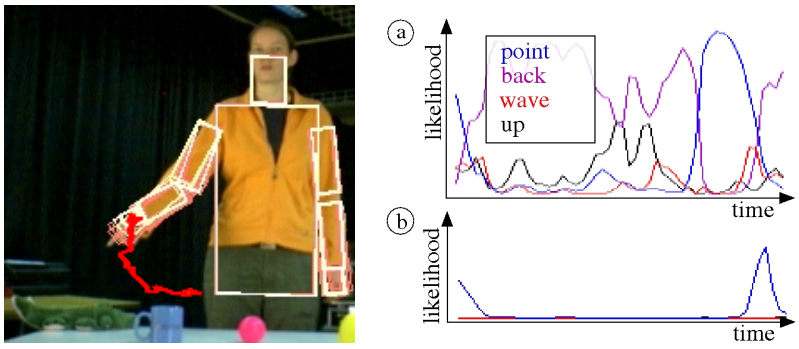
Figure 7.6: *Object attention system overview.* This sketch outlines the vision system for recognizing pointing gestures on a mobile robot. The robot system is only shown as a reference here to illustrate the integration into the framework. The evaluation presented here has been done on the vision system only.

The vision system evaluated in this paper consists of three modules that are presented in the following, see also Fig. (7.6). We start with tracking the human's upper body, especially his acting hand (Body Tracking). These motions are interpreted in order to detect actions like pointing (Gesture Recognition). Based on the recognized action, a region of interest (RoI) is determined (Object Attention) to finally enable detecting and learning the referenced object.

7.2.2 Trajectory-Based Gesture Recognition

The trajectory-based gesture recognition module developed by Hofemann [69] uses the tracked motions are used as input in order to detect meaningful gestures. Due to the constraints of the presented scenario, the trajectory of the right hand as depicted in Fig. (7.7(a)) is sufficient for recognizing the human's gestures. The hand's motions are represented in a cylindrical coordinate system with the basis in the human's shoulder. The features used for the recognition process are the relative radial and the vertical velocities with respect to the torso, therefore making the recognition process independent from the viewpoint of the camera. In a preprocessing step the sequential data is smoothed by a causal Savitzky-Golay filter [123].

Trajectory models are generated from annotated training examples. For the presented approach, we trained the following gestures: pointing at an object (point), retracting the



(a) (b)

Figure 7.7: *Gesture recognition results.* (a) Hand trajectory provided from the tracking module. (b) Activation potentials for different gesture models (top) and gesture completion likelihood (top).

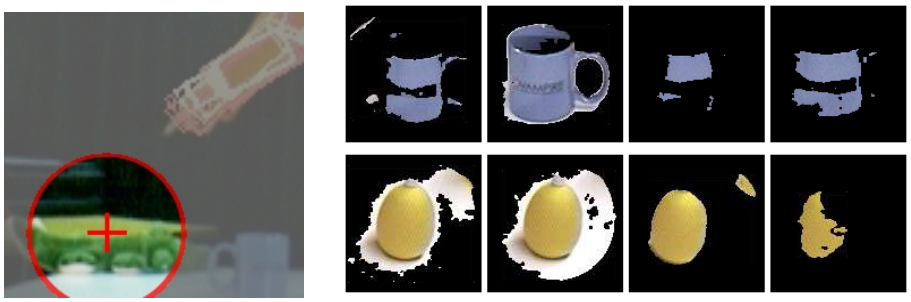
arm after pointing (back), raising the arm (up), waving (wave) and lowering the arm finally (down).

Briefly, the actual recognition is done by comparing the current motion simultaneously with the different possible trajectory models. As a new gesture can start any time, we employ a particle filtering scheme [70] tracking a multitude of hypotheses each with different parameters. Optimized parameters are the time when the gesture started and scaling factors for duration and amplitude of the motion, respectively. The parameters of one hypothesis define the optimal trajectory for this hypothesis that is compared with the trajectory actually observed based on a similarity measure. From this comparison the likelihood for each hypothesis can be calculated and used for the propagation step in the particle filter. For each trajectory model the likelihoods of all hypotheses can be added over the complete particle set resulting in the likelihoods for each trajectory model and those for completing the gesture successfully, see Fig. (7.7(b)). The gesture recognition thereby segments the continuous hand trajectory into meaningful gestures. External triggering, e.g., by speech is not needed.

7.2.3 Object Attention

The presented interaction scenario eventually aims at recognizing objects the human points at. The tracked and recognized gestures provide the basis for calculating a region of interest (RoI) to resolve the referenced object. The object attention module has been developed by Haasch [60].

As soon as a gesture is recognized as pointing the position of the human is extracted from the tracking results and affirmed by comparing it to the person attention information. To determine the RoI, the pointing direction is assumed as a line extending from the users head over his hand, cf. [153]. The region of interest is represented as a sphere in the world coordinate system, centered at the estimated object position. The distance between the tracked hand position and the RoI center can be scaled comprising verbal information, e.g., describing the size (“large”, “tiny”) of the object. The zoom of the object camera is set accordingly. A fixed distance is used for this evaluation, as speech input has not been considered. The RoI position is then transformed into the coordinate system of the robot’s object camera and an image is captured for analysis, see Fig. (7.8).



(a) (b)

Figure 7.8: *Region of Interest (RoI) for Object Attention.* (a) The RoI position is defined by the pointing direction. (b) At the expected position of the object, sample images are recorded for further processing, e.g. an object detection module.

The object attention verifies if the verbal and visual information correspond during interaction. Predefined color words, e.g. "blue", are used to apply a simple color based object segmentation using a direct mapping to preassigned RGB-values. This enables the user to resolve ambiguities from pointing, e.g., for two nearby objects by naming their color. Thus a coupling of verbal and gestural modalities is supported by the object attention. The color segmentation can be used to acquire a closeup view of the referenced object for later use, e.g. training an object recognition system.

7.2.4 Evaluating the System Performance

For evaluation the following setup has been used: The human is standing in front of a table with five objects, facing the robot BIRON. The person is asked to show the objects to the robot by pointing at them. The human's actions are recorded by the body tracking camera while we ensured that the upper body is in its field of view. Likewise, the object positions are well within range of the pan-tilt camera. The exact position of each objects has been manually measured. The objects are the targets for pointing gestures and therefore serves as a reference for the following experiments.

The experiments were performed with 4 persons, two female and two male. Half of the participants were inexperienced and performed the gestures for the gesture recognition for the very first time. The other half were experienced users. After an initialization phase for the body tracking the person pointed (point) at each object in sequence (crocodile, cup, ball, lemon, bottle), withdrawing their hand (back) after each gesture. The next gestures are raising the hand (up) and waving (wave) into the camera, finally lowering the arm (down). All participants had to perform the same sequence of gestures three times changing the order of targets each time.

We recorded each subject performing the experiments four times, resulting in 16 sequences with a total number of 18572 images, equivalent to more than 20 minutes of video, counting 496 performed gestures in total.

Compared to our previous work [61], appropriate depth information about the current body pose can significantly improve the robustness of action recognition. Probabilistic

fusion of data from gesture and speech recognition, however, could help to validate assumptions, as speech and gestures naturally co-occur in everyday communication [170].

Both, body tracking trajectories and recognized gestures are essential for determining the region of interest. For evaluating the object attention module only the 167 correctly recognized pointing gestures are used.

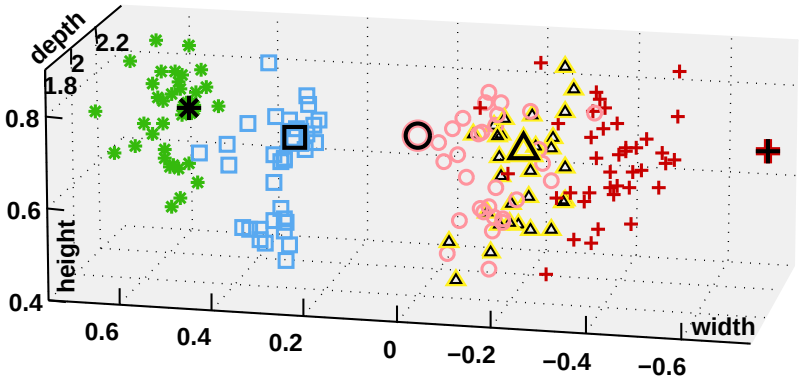


Figure 7.9: *Recognized object positions.* RoI center positions for recognized pointing gestures and ground truth (bold markers) for each object. Objects (1-5) from left to right.

The final object position error is calculated as the euclidian distance in $[m]$ in the world coordinate system between the measured object position and the RoI center. Table 7.2 presents the RMSE error μ and the variance σ for the RoIs separately for each subject. The objects (1-5) are ordered in the same manner as in Figures 7.9 and 7.6.

The RoI errors are for most objects not bigger than 25cm. Even without further validation from the robot's person attention, the object RoI is determined robustly. The error in the RoI positions is mainly due to the coarse gesture information from the preceding modules. Also, estimating the pointing direction as a line from the head through the hand does not always suit well. For objects (4) and (5) to the left of the person (at the right in the image), the RoI is estimated to be too close to the human for two reasons: 1) The body tracking cannot handle the self occlusion that good and produces more noisy results; 2) for the described posture it is uncommon to have the hand in the line of sight towards the object, the gesture is more like "shooting from the hip". This calls for learning task-specific dependencies. The error could be overcome by altering the RoI offset for this specific gesture, as the recognized RoI positions are still compact, which explains the low σ in Table 7.2, but their mean position is slightly offset as Fig. (7.9) clearly shows.

	subj A			subj B			subj C			subj D			all		
Object	# g	μ	σ	# g	μ	σ	# g	μ	σ	# g	μ	σ	# g	μ	σ
(1) green crocodile	9	0.22	0.04	7	0.10	0.03	7	0.15	0.04	9	0.13	0.06	32	0.15	0.06
(2) blue cup	8	0.28	0.02	10	0.21	0.07	7	0.23	0.02	7	0.23	0.03	32	0.24	0.05
(3) pink ball	8	0.29	0.06	7	0.14	0.05	8	0.25	0.04	7	0.22	0.07	30	0.23	0.08
(4) yellow lemon	7	0.38	0.05	7	0.25	0.03	7	0.25	0.05	8	0.34	0.03	29	0.31	0.07
(5) red bottle	8	0.58	0.04	10	0.47	0.07	11	0.51	0.05	10	0.50	0.04	39	0.51	0.06
all	40	0.35	0.14	41	0.25	0.15	40	0.30	0.14	41	0.29	0.14	*162*	*0.30*	*0.14*

Table 7.2: *Position error for the calculated region of interest (RoI).* Objects (1-5) and subjects (A-D). # g: number of recognized gestures, μ: mean error, σ: standard deviation, both in $[m]$.

Objects (1-3) show that better results are achievable. Comparing the different subjects,

the errors vary only slightly although the gestures were performed very differently across subjects. Individual gesture models therefore seem to be a good way to deal with such variations.

The exactness could be further increased if verbal descriptions for size and color were available and information from the person attention (e.g., position of the gesturer, acoustic input present) were utilized. Integrating context knowledge about the environment and the current task could also help to reduce the search space for the body tracking as well as the gesture recognition.

7.3 Hand Gesture Detection using the Body Pose Tracking

As we already learned, gesture are an important means for everyday communication between humans. The advantages of a gesture recognition process for mobile robotics has been presented in the last chapter. On of the disadvantages of the presented approach is that the type of the gesture performed is solely determined by the trajectory of the hand, which obviously works for dynamic gestures. But our life is also full of static gestures, like the "stop" sign shown in Fig. (7.10). These gestures are meaningful on their own, but also during dynamic gestures, the hand pose can determine the exact purpose of the action. The difference between pointing and grasping, as an example, is hard to tell observing the trajectory only, but can be made possible when including appearance information. A well known approach using hand gestures to control a user interface is the HandVU system presented by Kölsch and Turk [89]. The following technique for appearance-based hand gesture detection has been implemented by Gärtner and is presented in more detail in [53].

Figure 7.10: *Stop command gesture.* Basic commands can be given to the robot using static gestures.

The body model tracking framework does not model the hand in detail, it only provides a rough estimate of the hand's position and the orientation in space given by the lower arm. The idea of the hand gesture detection using the body pose tracking is to let the tracking determine a likely region for the hand and then use an appearance based

detector to determine the exact hand gesture. A big problem of appearance-based detectors is the dependency on the orientation and scaling of the feature. This is where the body tracking really comes into play. The orientation of the lower arm is known in 3D from the 3D body model. Backprojecting it into the image, it can be used to straighten up and scale the hand region to be free from rotation and perspective transformation, as depicted an Fig. (7.11).

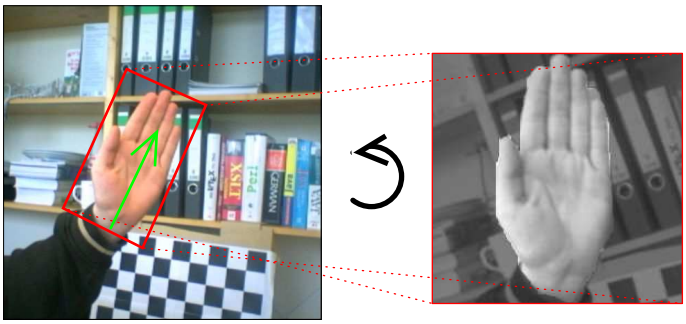
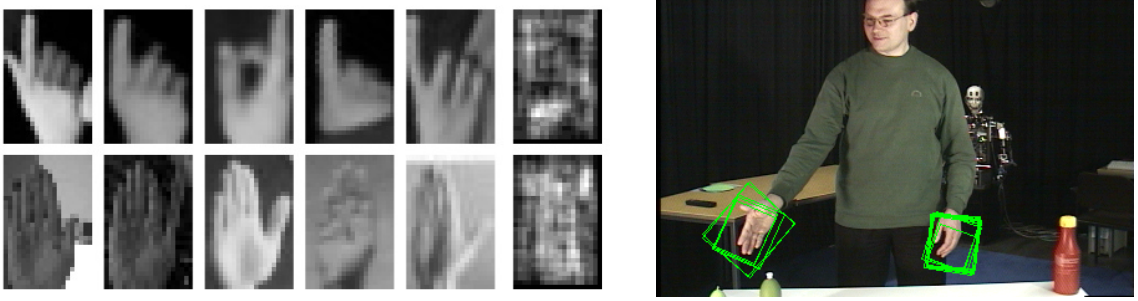


Figure 7.11: *Hand region normalization.* Using information from the body tracking, the hand region is straightened up.

The appearance based hand pose detector utilizes the AdaBoost learning technique that constructs a cascade of multiple efficiently to calculate features, similar to the detector proposed by Viola et al. [164] whose approach is well known as a face and object detector. Kölsch and Turk [89] extended the features of the Viola approach and used it to detect hand gestures. Training this kind of detectors usually requires a huge training data set, comprised of both positive and negative samples, see Fig. (7.12(a)) for some images from our positive training set. The AdaBoost strategy picks for each learning



(a) (b)

Figure 7.12: *Hand gesture training data and detection results.* (a) Training examples for the "pointing" (top row) and "stop" (bottom row) gestures. The rightmost column shows the according feature distribution of the chain boost cascade classifier. (b) Detection results. Similar to other appearance-based detector approaches, a multitude of detections for the same object are created.

step those features, that best separate the remaining positive and negative examples. The resulting cascade of classifiers follows the principle "from coarse to fine", which means that the uppermost classifiers react on those regions in the input image that give a strong indication whether the image contains a hand or not. Using a cascade allows a fast execution of the final classifier, as early stages in the cascade neglect most of the non-hand images and therefore do not need to be inspected further. The detector employed

for this approach is based on the chain boost detector developed by Peters [122], and has been extended to use four-block features in addition to the usual two- and three-block features known from the Viola approach. This extension proved to perform efficiently for the task of recognizing hand gesture which, in contrast to face classical detection, is much more depending on the recognition of fine and complex details, as the human hand is a highly articulated structure.

"stop" gesture		actual value positive	actual value negative
detection	positive	250	92
	negative	71	4133605039
		0.779 sensitivity	0.999 specificity

"pointing" gesture		actual value positive	actual value negative
detection	positive	100	30
	negative	241	383035805
		0.293 sensitivity	0.999 specificity

Table 7.3: *Classification results for the two classifiers "stop" and "pointing".* Both classifiers have been trained until a detection rate of at least 0.999 had been achieved, without posing further restrictions on the sensitivity or specificity.

Training of the classifier is continued until a given detection rate, e.g. 99,9% is achieved. An allowed false-positive rate of up to $30\% - 50\%$ for each cascade level have shown to be reasonable [100]. The classification results for the two classifiers "stop" and "pointing" are displayed in Tab. (7.3).

enforcing less false negatives		actual value positive	actual value negative
detection	positive	659	131802
	negative	3	177971733
		0.995 sensitivity	0.999 specificity

Table 7.4: *Classification results for the combined cascade classifier.* Training both classifiers in a cascade and using more positive examples drastically improves the performance.

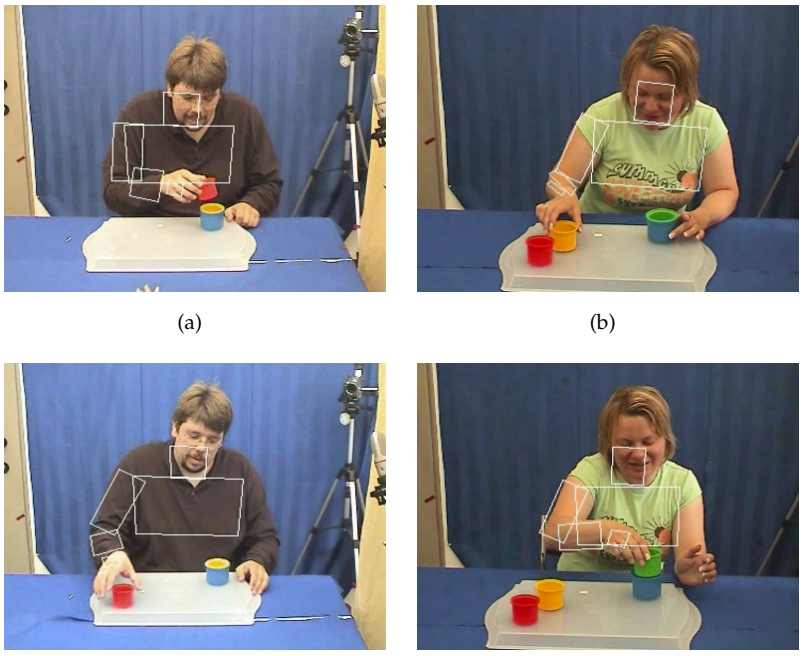
The classification results of these two classifiers are moderate. A high false negative rate prevents them from being used in a real application as too many hand gestures would not be correctly determined. This mostly results from the poor training data. The tables already suggest that only few positive examples were used, whereas negative examples were generated automatically and therefore are virtually available in an unlimited number. Further manual labeling of hand gestures provided new training material and has been used to train a combined cascade classifier for both gestures, see Tab. (7.4) for the results. A single cascade for both classifiers has the advantage that they can benefit from sharing the uppermost cascade levels, which can be interpreted as being a "hand presence" detector. This also helps to further reduce processing time, as incoming samples only need to be classified with a single cascade instead of one per detector.

Efforts to train a general hand classifier or to introduce more than two gestures were not carried on due to the labor-intensive manual labeling of the hand gestures. Also, a study using a "fist" and an "idle" gesture (hand is relaxed) supposed that not all gestures are equally well detectable with the presented approach. Although the hand gesture detection had not been integrated on the robot, the prospects of combining the body pose tracking with the hand gesture detection approach are promising.

7.4 Motionese Developmental Studies

An interesting approach related to human robot interaction has been contributed by a field that – at the first sight – seems to be unrelated to robotics. Developmental studies investigated the importance of motion learning in early childhood development in the so-called "motionese" studies. The term motionese is a neologism combining the words "motion" and "motherese". The latter is a term known from linguistics, where it describes the changes in speech that parents often feature when communicating with their children in the early childhood speech acquisition phase, for instance that the prosody becomes emphasized. Motionese describes the emphasis of specific features in motion that can be observed during gestural interaction of parents with their children, as described by Brand et al. [14].

Studies by Rohlfing et al. [129, 131] aimed at finding quantitative evidence for the described effect by observing parents during gestural interaction with a child and by comparing the observed motions for an action to the motions of the same action when performed towards an adult. One of the biggest challenges was to extract the parents' motion without posing too much constraints on the experimental setup, as letting the parents wear special clothing or attach marker to them could have their affected their unbiasdness.



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

Figure 7.13: *Tracking motions for developmental studies.* (a,b) Two subjects explain a task to their particular adult partner, (c,d) the same task is explained to their child. Multiple features of the hand trajectory reveal differences between child-directed and adult-directed motions.

The body tracking system was seen as a good option to analyze the parents' motions, as it combines all the features needed for the experiment: It does nor require markers, it does not require specific clothing or a specific environment, it is not specific to a single person, it does not make any assumptions on the type of gestures observed, it is able to provide a 3-dimensional reconstruction of the gestures performed and, probably most

important, it eases the experimental setup as it does not require specific hardware like stereo cameras but functions with a single monocular color video camera. 16 families took part in the recordings, which makes a total of 32 persons to be analyzed. From these 32, 22 persons had to be excluded from the final evaluation, partly due to variations in the manner the gestures were performed, partly due to technical issues like errors during recording, but some also due to failed tracking. The most common reasons for lost tracking were fast motions and occlusions and ambiguities due to the camera viewpoint and the fact that the persons were sitting at a table, sometimes bending forward towards their child which the tracking framework could not handle. In general it can be said that the tracking quality strongly depended on a good visibility in the image, on clothing that was distinguishable from the background but also showed enough features to allow the separation of individual body parts and las but not least on a good fitting model, which puts the suitability of the body tracking approach for a general unconstrained experiment back into perspective. Some results of the tracking can be seen in Fig. (7.13), the full results of the studies are presented in [130, 131]. Quantitative analysis of the motions indeed revealed a significant interrelation between some features of the trajectories, like pace and roundness, and the presence of motionese.

8 Outlook

In my thesis, I introduced a system that enables the automatic localization of persons and the tracking of upper body gestures in a human robot interaction scenario. The importance of such systems has been shown with the variety of scenarios they have been applied in. Further improvements, however, could open up new fields of activities.

First signs for newly emerging applications are already in sight. Lowering production costs for sensors and increasing computational power in all kinds of electronic equipment that penetrates our daily life makes methods for observing human motions easily available. As an example, the Swissranger Time-of-Flight sensor that has been used to record depth information costs several thousand euros. The 3DV company[1] recently presented the ZCam, a ToF sensor that provides depth information synchronized with RGB images and thereby outperforms the Swissranger to be sold at a reasonable price of approximately 100 euros. It will probably soon be found as an interface for game consoles, similar to the well-known EyeToy camera[2]. Depth information could also be used within the body pose tracking framework to achieve more reliable pose estimation, similar to [133].

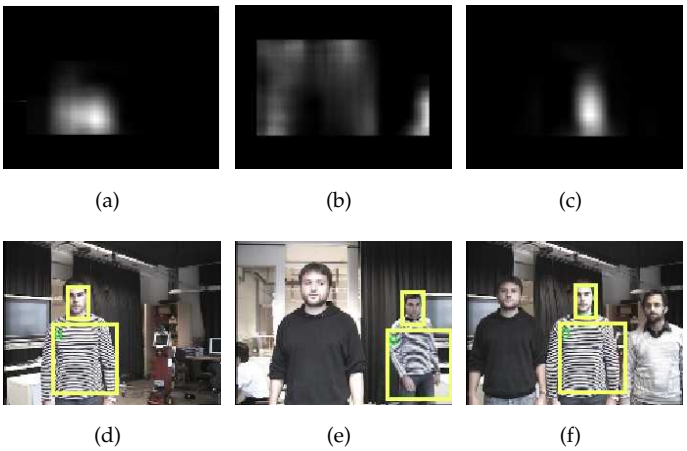
More professionally oriented, an automated observation of industrial workplaces for safety purposes has already been achieved with the vision based SafetyEYE[3] system which is commercially available. Adding motion information has already shown to improve the performance of the segmentation and tracking. But still, the SafetyEYE and also the presented localization system are only reactive systems. The next step towards interactive systems would be to understand the actions of the human [99] through observing his motions in context of the objects in the scene and the performed task. Knowledge about typical trajectories in a given scenario can be used to classify the observed motions and to predict future states, as Hermes et al. [67] presented for a driver assistance system, but those methods could be transferred to other scenarios as well. The prospect of enabling humans and robots to work together are aspects that also inspired the formulation of the patent [92].

As mentioned above, cheaper sensors may revolutionize the whole field of human motion analysis. The soon to come broad availability of standard computer hardware offering multiple cores in a single machine calls for algorithms that can benefit from these developments. The presented particle filtering technique clearly is such an algorithm. The inference of possible model configurations can easily be parallelized, the overall system performance scales well with the number of cores available, as it was shown in [24]. In the last years, the processing units of standard graphic cards have undergone

[1] http://www.3dvsystems.com/

[2] http://www.eyetoy.com/

[3] http://www.pilz.de/products/sensors/camera/f/safetyeye/index.jsp



(a) (b) (c)

(d) (e) (f)

Figure 8.1: *Directed attention for fast person localization.* (a)-(c) Top-down object attention maps representing the most salient regions using a learned directed attentional model. (d)-(f) The found object position given by the most salient point.

a similar trend. GPUs already offer 800 and more parallel processing units, big effort is taken to enable the mapping of computational concepts to this standard hardware [65].

In contrast to the recent hardware developments, understanding the humans' visual perception has been of interest to cognitive scientists for a long time. An approach to detect humans in images is presented by Beuter [10], summarizing the work of Lohmann [101]. In contrast to applying strong models which necessitate a certain appearance, the directed attention approach uses the idea of bottom-up feature analysis to detect salient regions in the image without making any assumptions about their origination. Finding humans and other objects is achieved by selecting only those features that separate best fore- and background. A weak top-down model that describes humans as a union of multiple regions is not introduced until this late point of the algorithm, see Fig. (8.1) for exemplary results. Such a bottom-up feature analysis could be integrated into the body pose tracking framework to restrain the search process. It also seems promising to employ it as a basic vision component of future frameworks for robotic applications, where it could function as a module that can learn the typical appearance from higher level vision modules, say a face detector or an object classifier, and then provide a fast to calculate directed attention towards specific image regions.

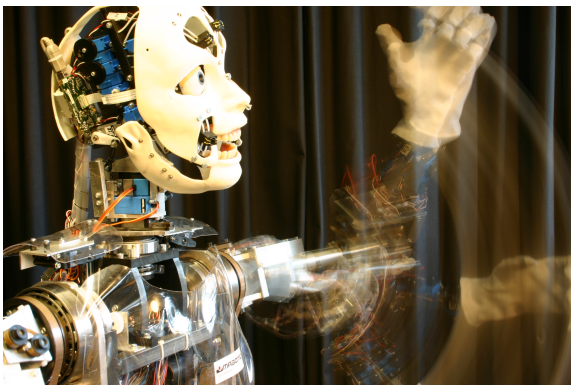


Figure 8.2: BARTHOC *humanoid robot.* **B**ielefeld **a**nthropomorphic **R**obo**t** for **h**uman **o**riented **c**ommunication.

Understanding and interpreting human motions is undoubtedly of interest to any robotic system trying to interact with humans. The growing discipline of humanoid robotics has a particular interest in gesture perception and understanding, as such robotic system of-

fer a much more intuitive human robot interaction [150], cf. Fig. (8.2). If the robot is able to track and interpret human gestures, interaction is no longer limited to perception, as imitation and synthesis of gesture becomes possible. Concurrently, developmental sciences become of interest for robotics, as technical systems can likewise benefit from the understanding of learning in humans. Perception and understanding are fundamental parts of cognition, the work presented in this thesis hopefully can provide inspirations for further realizations of technical systems that percept and interpret human motions to enable the development of interactive cognitive systems.

Appendix: Publication List

Patents

- U. Kreßel, L. Krüger, W. Progscha, C. Wöhler, F. Kummert, G. Sagerer, J. Schmidt, R. Ott: *System zum Training von Personen und zur Überprüfung deren Lernfortschritts*, German Patent No. DE 10 2006 048 165 A1, 10.10.2006.
- U. Kreßel, L. Krüger, W. Progscha, C. Wöhler, F. Kummert, G. Sagerer, J. Schmidt, R. Ott: *Verfahren zur Beobachtung einer Person in einem industriellen Umfeld*, German Patent No. DE 10 2006 048 166 A1, 10.10.2006.

Journal Papers

- A. Swadzba, N. Beuter, J. Schmidt, and G. Sagerer: *6D Scene Analysis: From a Dynamic Environment to a Static Scene*, Computer Vision and Image Understanding (CVIU): Special Issue on Time-of-Flight based Computer Vision. (submitted for review).

Conference Papers

- N. Beuter, O. Lohmann, J. Schmidt, F. Kummert: *Directed Attention – A cognitive vision system for a mobile robot*, ROMAN 2009, 18th IEEE International Symposium on Robot and Human Interactive Communication, Japan, September 2009. (submitted for review).
- J. Schmidt: *Monokulare Modellbasierte Posturschätzung des Menschlichen Oberkörpers*, 8. Oldenburger 3D-Tage, Oldenburg, January 2009.
- N. Beuter, A. Swadzba, J. Schmidt and G. Sagerer: *3D-Szenenrekonstruktion in Dynamischen Umgebungen*, 8. Oldenburger 3D-Tage, Oldenburg, January 2009.
- J. Schmidt, N. Hofemann, A. Haasch, J. Fritsch, and G. Sagerer: *Interacting with a Mobile Robot: Evaluating Gestural Object References*, IROS 2008, Nice, France, September 2008.
- N. Beuter, A. Swadzba, and J. Schmidt: *Simultaneous Tracking And Scene Reconstruction For Robot Perception*, Workshop for Cognitive Humanoid Vision: IEEE-RAS, Daejeon, Korea, 2008.
- A. Swadzba, N. Beuter, J. Schmidt, and G. Sagerer: *Tracking Objects in 6D for Reconstructing Static Scenes*, CVPR 2008 Workshop On Time of Flight Camera based Computer Vision (TOF-CV), Anchorage, Alaska, June 2008.

- J. Schmidt, and M. Castrillón Santana: *Automatic Initialization for Body Tracking - Using Appearance to Learn a Model for Tracking Human Upper Body Motions*, 3rd International Conference on Computer Vision Theory and Applications (VISAPP), Funchal, Madeira - Portugal, January 2008.
- J. Schmidt, C. Wöhler, L. Krüger, T. Gövert, and C. Hermes: *3D Scene Segmentation and Object Tracking in Multiocular Image Sequences*, In Proc. of 5th International Conference on Computer Vision Systems (ICVS), Bielefeld, Germany, March 2007.
- J. Schmidt, B. Kwolek and J. Fritsch: *Kernel Particle Filter for Real-Time 3D Body Tracking in Monocular Color Images*, In Proc. of Automatic Face and Gesture Recognition (FG), pages 567-572, Southampton, UK, April 2006.

List of Figures

List of Tables

Bibliography

[1] M. A. Admiraal, M. J.M.A.M. Kusters, and S. C.A.M. Gielen. Modeling kinematics and dynamics of human arm movements. In *Motor Control*, pages 312–338. Human Kinetics Publishers, Inc., 2004.

[2] I. Andricioaei, A. F. Voter, and J. E. Straub. Smart Darting Monte Carlo. In *J. Chem. Phys.*, volume 114, pages 6994–7000, 2001.

[3] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding. In David J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89)*, pages 86–91, San Mateo, California, 1989. Morgan Kaufmann Publishers, Inc.

[4] S. Arulampalam, S. Maskell, and N. Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

[5] J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 101–111, Mahwah, NJ, USA, 1985. Lawrence Erlbaum Associates, Inc.

[6] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc.

[7] C. Bauckhage, M. Hanheide, S. Wrede, T. Käster, M. Pfeiffer, and G. Sagerer. Vision Systems with the Human in the Loop. *EURASIP J. on Applied Signal Processing*, 2005(14):2375–2390, 2005.

[8] M. Berthold and D. J. Hand. *Intelligent Data Analysis*. Springer, 2nd edition, 2003.

[9] N. Beuter, A. Swadzba, and J. Schmidt. Simultaneous tracking and scene reconstruction for robot perception. *Workshop for Cognitive Humanoid Vision at the International Conference on Humanoid Robots*, 2008.

[10] Niklas Beuter, Okko Lohmann, Joachim Schmidt, and Franz Kummert. Directed attention - a cognitive vision system for a mobile robot. In *18th IEEE International Symposium on Robot and Human Interactive Communication*, Toyama, Japan, September 2009. IEEE, IEEE.

[11] A. Bissacco, M.-H. Yang, and S. Soatto. Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[12] J. Bjørnstrup. Estimation of human body segment parameters — historical background. Technical report, Laboratory of Image Analysis, Institute of Electronic Systems, Aalborg University, 1995.

[13] L. B. Booker, D. B. Fogel, D. Whitley, P. J. Angeline, and A. E. Eiben. Recombination. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Evolutionary Computation 1 – Basic Algorithms and Operators*, chapter 33, pages 256–307. Institute of Physics Publishing, 2000.

[14] R. J. Brand, D. A. Baldwin, and L. A. Ashburn. Evidence for 'motionese': modifications in mothers' infant-directed action. *Developmental Science*, 5(1):72–83, 2002.

[15] A. G. Brooks and C. Breazeal. Working with robots and objects: revisiting deictic reference for achieving spatial common ground. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 297–304, New York, NY, USA, 2006. ACM.

[16] T. Brox, B. Rosenhahn, U. Kersting, and D. Cremers. Nonparametric density estimation for human pose tracking. In K. Franke, R. Mueller, B. Nickolay, and R. Schaefer, editors, *Pattern Recognition 2006, DAGM*, volume 4174, pages 546–555, Berlin, 2006. LNCS, Springer-Verlag, Berlin Heidelberg.

[17] A. Bruce. Better motion prediction for people-tracking. In *In Proceedings of ICRA 2004*, 2004.

[18] A. O. Bălan, L. Sigal, and M. J. Black. A quantitative evaluation of video-based 3d person tracking. In *ICCCN '05: Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 349–356, Washington, DC, USA, 2005. IEEE Computer Society.

[19] D. Bullock and J. Zelek. Towards real-time 3-d monocular visual tracking of human limbs in unconstrained environments. *Real-Time Imaging*, 11(4):323–353, 2005.

[20] M. A. Carreira-Perpinan. Gaussian mean-shift is an em algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(5):767–776, 2007.

[21] M. Castrillón Santana, O. Déniz Suárez, M. Hernández Tejera, and C. Guerra Artal. ENCARA2: Real-time detection of multiple faces at different resolutions in video streams. *Journal of Visual Communication and Image Representation*, pages 130–140, April 2007.

[22] C. Chang and R. Ansari. Kernel particle filter: iterative sampling for efficient visual tracking. In *IEEE Int. Conference on Image Processing*, volume 2, pages 977–980, 2003.

[23] C. Chang and R. Ansari. Kernel particle filter for visual tracking. *Signal Processing Letters*, 12(3):242–245, 2005.

[24] T. P. Chen, D. Budnikov, C. J. Hughes, and Y. Chen. Computer vision on multi-core processors: Articulated body tracking. In *International Conference on Multimedia and Expo (ICME)*, Beijing, China, July 2007. Springer-Verlag.

[25] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):790–799, 1995.

[26] R. Cipolla and P. Giblin. *The Visual Motion of Curves and Surfaces*. Cambridge University Press, 2000.

[27] COGNIRON. The cognitive robot companion, 2004. (FP6-IST-002020), http://www.cogniron.org.

[28] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *ICCV (2)*, pages 1197–1203, 1999.

[29] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[31] J. L. Crowley and J. Coutaz. Vision for man machine interaction. In *Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction*, pages 28–45, London, UK, 1996. Chapman & Hall, Ltd.

[32] H. J. Curnow and Brian A. Wichmann. A synthetic benchmark. *Comput. J.*, 19(1):43–49, 2003.

[33] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893, 2005.

[34] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, New Jersey, 1963.

[35] C. Darwin. *On the origin of species by means of natural selection, or, The preservation of favoured races in the struggle for life*. Grant Richards, London, 1902.

[36] J. Davis, D. Nehab, R. Ramamoorthi, and S. Rusinkiewicz. Spacetime stereo: A unifying framework for depth from triangulation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.27(2), 2005.

[37] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Int. Conf. on Pattern Recognition*, pages 126–133, 2000.

[38] J. Deutscher, B. North, B. Bascle, and A. Blake. Tracking through singularities and discontinuities by random sampling. In *In IEEE International Conference on Computer Vision*, pages 1144–1149, 1999.

[39] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *Int. J. Comput. Vision*, 61(2):185–205, 2005.

[40] A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo methods in practice*. Springer, New York, 2001.

[41] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc, 1973.

[42] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.

[43] A. P. Engelbrecht. *Computational Intelligence – An Introduction*. John Wiley & Sons, 2007.

[44] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[45] R. Fay, U. Kaufmann, A. Knoblauch, H. Markert, and G. Palm. Integrating object recognition, visual attention, language and action processing on a robot in a neurobiologically plausible associative architecture. In G. Palm and S. Wermter, editors, *NeuroBotics Workshop Proceedings*. 27th German Conf. of Artificial Intelligence, University of Ulm, 2004.

[46] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *Int. J. Comput. Vision*, 61(1):55–79, 2005.

[47] U. Franke and A. Joos. Real-time stereo vision for urban traffic scene understanding. In *Conf. on Intelligent Vehicles*, Detroit, 2000. IEEE.

[48] U. Franke, C. Rabe, H. Badino, and S. K. Gehrig. *6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception*, pages 176–183. Lecture Notes in Computer Science 3663. Springer-Verlag Berlin Heidelberg, pattern recognition. proc. 27th dagm symposium, vienna, austria edition, 2005.

[49] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[50] J. Fritsch, S. Lang, M. Kleinehagenbrock, G. A. Fink, and G. Sagerer. Improving adaptive skin color segmentation by incorporating results from face detection. In *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication (ROMAN)*, pages 337–343, Berlin, Germany, September 2002. IEEE.

[51] P. Fua, A. Gruen, R. Plaenkers, N. D'Apuzzo, and D. Thalmann. Human Body Modeling and Motion Analysis From Video Sequences. In *International Symposium on Real Time Imaging and Dynamic Analysis*, 1998.

[52] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, 1975.

[53] M. Gaertner. Integration eines Handgestenerkenners zur intuitiven Systemsteuerung. Diplomarbeit, *Faculty of Technology:* Bielefeld University, 2005.

[54] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding: CVIU*, 73(1):82–98, 1999.

[55] T. Germa, F. Lerasle, P. Danes, and L. Brethes. Human / Robot Visual Interaction for a Tour-Guide Robot. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[56] J. J. Gibson. The perception of the visual world. *Riverside Press*, 1950.

[57] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[58] M. Grosjean, M. Shiffrar, and G. Knoblich. Fitt's law holds in action perception. *Psychological Science*, 18:95–99, 2007.

[59] Humanoid Animation Working Group. Information technology – computer graphics and image processing – humanoid animation (h-anim).

[60] A. Haasch. *Attention-controlled acquisition of a qualitative scene model for mobile robots*. PhD thesis, *Faculty of Technology:* Bielefeld University, 2007.

[61] A. Haasch, N. Hofemann, J. Fritsch, and G. Sagerer. A multi-modal object attention system for a mobile robot. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1499–1504, Edmonton, Alberta, Canada, August 2005. IEEE.

[62] A. Haasch, S. Hohenner, S. Hüwel, M. Kleinehagenbrock, S. Lang, I. Toptsis, G. A. Fink, J. Fritsch, B. Wrede, and G. Sagerer. BIRON – The Bielefeld Robot Companion. In E. Prassler, G. Lawitzky, P. Fiorini, and M. Hägele, editors, *Proc. Int. Workshop on Advances in Service Robotics*, pages 27–32, Stuttgart, Germany, May 2004. Fraunhofer IRB Verlag.

[63] B. Han, Y. Zhu, D. Comaniciu, and L. Davis. Kernel-based bayesian filtering for object tracking. In *Int. Conf. on Computer Vision and Patt. Recognition*, pages 227–234, 2005.

[64] P. J. B. Hancock. An empirical comparison of selection methods in evolutionary algorithms. In *Selected Papers from AISB Workshop on Evolutionary Computing*, pages 80–94, London, UK, 1994. Springer-Verlag.

[65] M. Harris. Mapping computational concepts to gpus. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 50, New York, NY, USA, 2005. ACM.

[66] C. Hermes. Evolutionäre Algorithmen für die zielgerichtete Optimierung eines Systems zur Verfolgung von 3D-Körperposturen. Diplomarbeit, *Faculty of Technology:* Bielefeld University, 2008.

[67] C. Hermes, C. Wöhler, K. Schenk, and F. Kummert. Long-term vehicle motion prediction. In *IEEE Intelligent Vehicles Symposium*, June 2009.

[68] H. Hirschmueller. Improvements in real-time correlation-based stereo vision. In *Int. Conf. on Computer Vision and Pattern Recognition, Stereo Workshop*, Hawaii, 2001.

[69] N. Hofemann. *Videobasierte Handlungserkennung für die natürliche Mensch-Maschine-Interaktion*. Phd thesis, *Faculty of Technology:* Bielefeld University, Bielefeld, 2007.

[70] N. Hofemann, J. Fritsch, and G. Sagerer. Recognition of deictic gestures with context. In C. E. Rasmussen, H. H. Bülthoff, M. A. Giese, and B. Schölkopf, editors, *Pattern Recognition, 26th DAGM Symposium, Tübingen, Germany. Proceedings*, volume 3175 of *Lecture Notes in Computer Science*, pages 334–341, Heidelberg, Germany, 2004. Springer-Verlag.

[71] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2137–2144, Washington, DC, USA, 2006. IEEE Computer Society.

[72] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[73] B. K. P. Horn and B. G. Schunck. Determining optical flow. In *Artificial Intelligence*, volume 17, pages 185–204, 1981.

[74] D. Hubbard. *How to Measure Anything: Finding the Value of Intangibles in Business*. John Wiley & Sons, 2007.

[75] B. Huhle, P. Jenke, and W. Strasser. On-the-fly scene acquisition with a handy multisensor-system. In *Workshop on Dynamic 3D Imaging (Dyn3D)*, 2007.

[76] B. Huhle, T. Schairer, P. Jenke, and W. Strasser. Robust non-local denoising of colored depth data. In *Intl. Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Time of Flight Camera based Computer Vision (TOF-CV)*, 2008.

[77] R. L. Huston, C. E. Passerello, and M. W. Harlow. On human body dynamics. *Annals of Biomedical Engineering*, 4:25–43, 1976.

[78] Intel. Intel Open Source Computer Vision Library, v1.0. www.intel.com/research/mrl/research/opencv, October 2006.

[79] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Europ. Conf. on Computer Vision*, pages 343–356, 1996.

[80] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[81] T. Jaeggli, E. Koller-Meier, and L. J. Van Gool. Monocular tracking with a mixture of view-dependent learned models. In *AMDO*, pages 494–503, 2006.

[82] C. Z. Janikow and Z. Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In *ICGA*, pages 31–36, 1991.

[83] O. C. Jenkins, G. González, and M. M. Loper. Tracking human motion and actions for interactive robots. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 365–372, New York, NY, USA, 2007. ACM.

[84] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201–211, 1973.

[85] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, *University of Michigan*, Ann Arbor, MI, USA, 1975.

[86] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME-Journal of Basic Engineering*, 82:35–45, 1960.

[87] M. Keck, J. Davis, and A. Tyagi. Tracking mean shift clustered point clouds for 3d surveillance. In *VSSN*, pages 187–194, 2006.

[88] R. Kehl, M. Bray, and L. Van Gool. Full body tracking from multiple views using stochastic sampling. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 129–136, Washington, DC, USA, 2005. IEEE Computer Society.

[89] M. Kölsch and M. Turk. Fast 2d hand tracking with flocks of features and multi-cue integration. In *In IEEE Workshop on Real-Time Vision for Human-Computer Interaction (at CVPR*, page 158, 2004.

[90] M. Kölsch and M. Turk. Robust hand detection. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, May 2004.

[91] D. Kortenkamp, E. Huber, and R. P. Bonasso. Recognizing and interpreting gestures on a mobile robot. In *Nat. Conf. on Artificial Intelligence*, pages 915–921, 1996.

[92] U. Kreßel, L. Krüger, W. Progscha, C. Wöhler, F. Kummert, G. Sagerer, J. Schmidt, and R. Ott. System zum Training von Personen und zur Überprüfung deren Lernfortschritts, October 2006. DE 10 2006 048 165 A1.

[93] U. Kreßel, L. Krüger, W. Progscha, C. Wöhler, F. Kummert, G. Sagerer, J. Schmidt, and R. Ott. Verfahren zur Beobachtung einer Person in einem industriellen Umfeld, October 2006. DE 10 2006 048 166 A1.

[94] L. Krüger, C. Wöhler, A. Würz-Wessel, and F. Stein. In-factory calibration of multiocular camera systems. In *SPIE Photonics Europe (Optical Metrology in Production Engineering)*, pages 126–137, Strasbourg, 2004.

[95] G.-J. M. Kruijff, J. D. Kelleher, and N. Hawes. Information fusion for visual reference resolution in dynamic situated dialogue. In *Perception and Interactive Technologies, PIT 2006*, volume 4021 of *LNCS*, pages 117–128. Springer Berlin / Heidelberg, 2006.

[96] X. Lan and D. P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 470–477, Washington, DC, USA, 2005. IEEE Computer Society.

[97] H. J. Lee and Z. Chen. Determination of 3d human body postures from a single view. *Computer Vision Graphics and Image Processing*, 30(2):148–168, May 1985.

[98] M. Lee and I. Cohen. Human upper body pose estimation in static images. In *Proc. of European Conference on Computer Vision ECCV)*, pages 126–138, 2004.

[99] Z. Li, N. Hofemann, J. Fritsch, and G. Sagerer. Hierarchical modeling and recognition of manipulative gesture. In *Proc. IEEE ICCV, Workshop on Modeling People and Human Interaction*, Beijing, China, October 2005.

[100] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, Microprocessor Research Lab, Intel Labs, December 2002.

[101] O. Lohmann. Objektrepräsentation durch visuelle Salienz zur Erzeugung gerichteter Aufmerksamkeit. Diplomarbeit, *Faculty of Technology:* Bielefeld University, 2009.

[102] F. Lömker, S. Wrede, M. Hanheide, and J. Fritsch. Building modular vision systems with a graphical plugin environment. In *Proc. of International Conference on Vision Systems*, St. Johns University, Manhattan, New York City, USA, January 2006. IEEE.

[103] Z. Lu, M. Carreira-Perpinan, and C. Sminchisescu. People tracking with the laplacian eigenmaps latent variable model. In J.C. Platt, D. Koller, Y. Singer,

and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1705–1712. MIT Press, Cambridge, MA, 2008.

[104] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 121–130, 1981.

[105] J. Luemkemann. Kinematik und Dynamik eines 3D-Körpermodells für bildbasierte Gestenerkennung. Master's thesis, *Faculty of Technology:* Bielefeld University, 2005.

[106] E. Maggio and A. Cavallaro. Hybrid particle filter and mean shift tracker with adaptive transition model. In *Proc. of IEEE Signal Processing Society International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Philadelphia, PA, USA, 19–23 March 2005.

[107] E.-J. Marey. *Chronophotograph*. Deutsches Filmmuseum, Frankfurt am Main, Germany, reprint of: berlin, mayer u. müller, 1893 edition, 1985.

[108] S. May, B. Werner, H. Surmann, and K. Pervolz. 3D Time-of-Flight cameras for mobile robotics. In *Intl. Conference on Intelligent Robots and Systems (IROS)*, pages 790–795, 2006.

[109] D. McNeill. *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press, 1992.

[110] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[111] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding: CVIU*, 81(3):231–268, 2001.

[112] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding: CVIU*, 104(2):90–126, 2006.

[113] M. Montemerlo, W. Whittaker, and S. Thrun. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *ICRA*, 2002.

[114] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, vol.7:308–313, 1965.

[115] K. Nickel, E. Seemann, and R. Stiefelhagen. 3D-Tracking of Heads and Hands for Pointing Gesture Recognition in a Human-Robot Interaction Scenario. In *Int. Conf. on Face and Gesture Recognition*, 2004.

[116] O. Oechsle and A. F. Clark. Feature extraction and classification by genetic programming. In Antonios Gasteratos, Markus Vincze, and John K. Tsotsos, editors, *ICVS*, volume 5008 of *Lecture Notes in Computer Science*, pages 131–140. Springer, 2008.

[117] S. Oprisescu, D. Falie, M. Ciuc, and V. Buzuloiu. Measurements with tof cameras and their necessary corrections. In *Intl. Symposium on Signals, Circuits & Systems (ISSCS)*, 2007.

[118] G. Orwell. *1984 Nineteen Eighty-Four*. Secker and Warburg (London), June 1949.

[119] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system, 1998.

[120] J. Park, M. Jeon, and W.S. Pedrycz. Score-based resampling method for evolutionary algorithms. In *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, volume 38, pages 1347 – 1355. IEEE, October 2008.

[121] K. Parsopoulos and M. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2–3):235–306, 2002.

[122] V. Peters. Effizientes Training ansichtsbasierter Gesichtsdetektoren. Diplomarbeit, *Faculty of Technology:* Bielefeld University, 2006.

[123] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[124] W. Qu, D. Schonfeld, and M. Mohamed. Distributed bayesian multiple-target tracking in crowded environments using multiple collaborative cameras. *EURASIP J. Appl. Signal Process.*, 2007(1):21–21, 2007.

[125] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 467–474, 2003.

[126] D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 271–278, Washington, DC, USA, 2005. IEEE Computer Society.

[127] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):65–81, 2007.

[128] D. Ramanan and C. Sminchisescu. Training deformable models for localization. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 206–213, Washington, DC, USA, 2006. IEEE Computer Society.

[129] K. Rohlfing, J. Fritsch, and B. Wrede. Learning to manipulate objects: A quantitative evaluation of motionese. In *Third International Conference on Development and Learning (ICDL 2004)*, page 27, La Jolla, CA, October 2004. ISBN 0-615-12704-5.

[130] K. Rohlfing and T. Jungmann. Referenz durch bewegung: Eine studie zu motionese, 2005. Poster presented at 17th EPSY, Bochum, 2005.

[131] K. J. Rohlfing, J. Fritsch, B. Wrede, and T. Jungmann. How can multimodal cues from child-directed interaction reduce learning complexity in robotos? *Advanced Robotics*, 20(10):1183–1199, 2006.

[132] R. Rosales, M. Siddiqui, J. Alon, and S. Sclaroff. Estimating 3D body pose using uncalibrated cameras. In *Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 821–827, 2001.

[133] R. Dillmann S. Knoop, S. Vacek. Sensor fusion for 3d human body tracking with an articulated 3d body model. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Walt Disney Resort, Orlando, Florida, May 2006.

[134] I. Schiller, C. Beder, and R. Koch. Calibration of a pmd camera using a planar calibration object together with a multi-camera setup. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume 37 Part B3a, pages 297–302, 2008.

[135] J. Schmidt, N. Hofemann, A. Haasch, J. Fritsch, and G. Sagerer. Interacting with a mobile robot: Evaluating gestural object references. In *Intl. Conference on Intelligent Robots and Systems (IROS)*, Nice, France, November 2008.

[136] J. Schmidt, B. Kwolek, and J. Fritsch. Kernel Particle Filter for Real-Time 3D Body Tracking in Monocular Color Images. In *Proc. of Automatic Face and Gesture Recognition*, pages 567–572, Southampton, UK, April 2006. IEEE.

[137] J. Schmidt, C. Wöhler, L. Krüger, T. Gövert, and C. Hermes. 3D Scene Segmentation and Object Tracking in Multiocular Image Sequence. In *Proc. of 5th International Conference on Computer Vision Systems (ICVS'07)*, Bielefeld, Germany, March 2007.

[138] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, September 1992.

[139] V. Sharma and J. Davis. Integrating appearance and motion cues for simultaneous detection and segmentation of pedestrians. In *ICCV*, 2007.

[140] V. D. Shet, J. Neumann, V. Ramesh, and L. S. Davis. Bilattice-based Logical Reasoning for Human Detection. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[141] H. Sidenbladh. *Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences*. PhD thesis, KTH Sweden, 2001.

[142] H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3D human figures using 2D image motion. In *Europ. Conf. on Computer Vision*, pages 702–718, 2000.

[143] L. Sigal, S. Bhatia, S. Roth, M. J. Black, and M. Isard. Tracking loose-limbed people. In *Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 421–428, 2004.

[144] L. Sigal and M. J. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Technical Report CS-06-08, 2006. http://vision.cs.brown.edu/humaneva/index.html.

[145] L. Sigal and M. J. Black. Predicting 3d people from 2d pictures. In *IV Conference on Articulated Motion and Deformable Objects - AMDO 2006*, volume 4069, pages 185–195, Mallorca, Spain, July 2006. IEEE Computer Society, LNCS.

[146] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Learning joint top-down and bottom-up processes for 3d visual inference. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1743–1752, Washington, DC, USA, 2006. IEEE Computer Society.

[147] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3d body tracking. In *CVPR*, pages 447–454, 2001.

[148] C. Sminchisescu and B. Triggs. Mapping minima and transitions of visual models. *Int. J. of Computer Vision*, 61(1), 2005.

[149] C. Sminchisescu and M. Welling. Generalized Darting Monte-Carlo. In *Artificial Intelligence and Statistics*, volume 1, 2007.

[150] T. Spexard, M. Hanheide, and G. Sagerer. Human-oriented interaction with an anthropomorphic robot. *IEEE Transactions on Robotics*, 23, Special Issue on Human-Robot Interaction(5):852–862, October 2007.

[151] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla:. Hand pose estimation using hierarchical detection. In *ECCV Workshop on HCI*, pages 102–112, 2004.

[152] N. Stergiou. *Innovative Analysis of Human Movement*. Human Kinetics, 2004.

[153] R. Stiefelhagen, H. Ekenel, C. Fügen, P. Gieselmann, H. Holzapfel, F. Kraft, K. Nickel, M. Voit, and A. Waibel. Enabling multimodal human-robot interaction fort the karlsruhe humanoid robot. *IEEE Transactions on Robotics*, 23, Special Issue on Human-Robot Interaction(5):840–851, October 2007.

[154] M. Storring, T. B. Moeslund, Y. Liu, and E. Granum. Computer vision-based gesture recognition for an augmented reality interface. In *4th IASTED International Conference on VISUALIZATION, IMAGING, AND IMAGE PROCESSING*, pages 766–771, September 2004.

[155] D. Stößel. *Automated Visual Inspection of Assemblies from Monocular Images*. PhD thesis, *Faculty of Technology:* Bielefeld University, 2007.

[156] D. Stößel and G. Sagerer. Kernel Particle Filter for Visual Quality Inspection from Monocular Intensity Images. In Katrin Franke, Klaus-Robert Müller, Bertram Nickolay, and Ralf Schäfer, editors, *DAGM06*, volume 4174 of *Lecture Notes in Computer Science*, pages 597–606, Heidelberg, Germany, 2006. Springer-Verlag.

[157] A. Swadzba, N. Beuter, J. Schmidt, and G. Sagerer. Tracking objects in 6d for reconstructing static scenes. *Computer Vision and Pattern Recognition Workshop: Time of Flight Camera based Computer Vision*, 2008.

[158] A. Swadzba, N. Beuter, J. Schmidt, and G. Sagerer. 6d scene analysis: From a dynamic environment to a static scene. *Computer Vision and Image Understanding: Special Issue on Time-of-Flight based Computer Vision*, 2009. submitted for review.

[159] A. Swadzba, B. Liu, J. Penne, O. Jesorsky, and R. Kompe. A comprehensive system for 3d modeling from range images acquired from a 3d tof sensor. In *Proc. of International Conference on Computer Vision Systems*, Bielefeld University, Bielefeld, Germany, 2007. University Library of Bielefeld.

[160] M. J. Swain and D. H. Ballard. Color indexing. *International Journal on Computer Vision*, 7(1):11–32, 1991.

[161] L. Taycher, G. Shakhnarovich, D. Demirdjian, and T. Darrell. Conditional random people: Tracking humans with crfs and grid filters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 222–229, 2006.

[162] A. Tyagi, M. Keck, J. W. Davis, and G. Potamianos. Kernel-based 3d tracking. In *CVPR*, 2007.

[163] R. Urtasun, D.J. Fleet, and P. Fua. Monocular 3d tracking of the golf swing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, 2005.

[164] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance, 2003.

[165] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):151–173, May 2004.

[166] M. P. Wand and M. C. Jones. *Kernel Smoothing (Monographs on Statistics and Applied Probability)*. Chapman & Hall/CRC, December 1994.

[167] C. Wang, C. Thorpe, M. Hebert, S. Thrun, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *IJRR*, 26(6), 2007.

[168] V. Wank. *Modellierung und Simulation von Muskelkontraktionen für die Diagnose von Kraftfähigkeiten*. Berichte und Materialien des Bundesinstituts für Sportwissenschaft, 1st edition, 1996.

[169] J. Weingarten, G. Gruener, and R. Siegwart. A state-of-the-art 3D sensor for robot navigation. In *IROS*, 2004.

[170] R. M. Willems, A. Özyürek, and P. Hagoort. When language meets action: The neural integration of gesture and speech. *Cerebral Cortex*, 2006.

[171] D. A. Winter. *Biomechanics and Motor Control of Human Movement*. Wiley, 1990.

[172] S. Wrede, J. Fritsch, C. Bauckhage, and G. Sagerer. An XML Based Framework for Cognitive Vision Architectures. In *Proc. Int. Conf. on Pattern Recognition*, pages 757–760, 2004.

[173] S. Wrede, M. Hanheide, S. Wachsmuth, and G. Sagerer. Integration and coordination in a cognitive vision system. In *Proc. of International Conference on Computer Vision Systems*, St. Johns University, Manhattan, New York City, USA, 2006. IEEE.

[174] Z. Xu, R. Schwarte, H. Heinol, B. Buxbaum, and T. Ringbeck. Smart pixel – Photometric Mixer Device (PMD) / new system concept of a 3D-imaging-on-a-chip. In *M2VIP*, pages 259–264, 1998.

[175] X. Zhao and Y. Liu. Generative tracking of 3d human motion by hierarchical annealed genetic algorithm. *Pattern Recognition*, 41(8):2470–2483, 2008.

[176] Q. Zhu, S. Avidan, M. C. Yeh, and K. T. Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, pages 1491–1498. IEEE Computer Society, 2006.

[177] Y. Zhu, B. Dariush, and K. Fujimura. Controlled human pose estimation from depth image streams. In *CVPR 2008 Workshop On Time of Flight Camera based Computer Vision (TOF-CV)*, 2008.

[178] J. Ziegler, K. Nickel, and R. Stiefelhagen. Tracking of the articulated upper body on multi-view stereo image sequences. In *Conference on Computer Vision and Pattern Recognition - CVPR*, New York, USA, June 2006. IEEE Computer Society.

[179] A. Zielinski. Quantitative Evaluierung eines Systems zur Bildbasierten Verfolgung von 3D-Körperposturen. Diplomarbeit, *Faculty of Technology:* Bielefeld University, 2006.