Project Acronym:	GRASP
Project Type:	IP
Project Title:	Emergence of Cognitive Grasping through Introspection, Emulation and Surprise
Contract Number:	215821
Starting Date:	01-03-2008
Ending Date:	28-02-2012

Deliverable Number:	D3
Deliverable Title :	Human Body Detection and Tracking
Type:	PU
Authors	A. Argyros, T. Asfour, D. Baldauf, H. Baltzakis, H. Deubel, M. Do, G. Floros, F. Hecht, G. Kastellakis, D. Michel
Contributing Partners	FORTH, LMU, UniKarl

Contractual Date of Delivery to the EC:	28-02-2009
Actual Date of Delivery to the EC:	28-02-2009

Contents

Chapter 1

Executive summary

Deliverable D3 presents part of the developments within workpackage WP1 - "Learning to Observe Human Grasping and Consequences of Grasping" of GRASP, for the first twelve months of the project.

WP1 is responsible for translating raw visual data into higher level descriptions of where humans and their hands are in a certain scene and how they move. These descriptions should have multi-scale representations, since different levels of detail are required at different levels of reasoning about human actions (e.g. global hand position might be relevant during hand pre-shaping before grasping, but finger positions might be interesting just before grasping occurs). The output of WP1 is provided to WP2 that is responsible for understanding human activities. This output is at the level required to be able to interpret these activities, map them to robotic embodiments and finally execute them. At the same time, WP2 provides valuable feedback to WP1 since the interpretation results of WP2 can feed lower level detection and tracking modules with valuable constraints. Information on the location and motion of humans and human hands is also very useful to the work carried out in WP4 with respect to the perception of the environment and the exploitation of contextual knowledge. The perception of consequences of grasping can directly assist the maintenance of the background/foreground model of the environment. Additionally, "interesting" things (i.e. in terms of what constitutes foreground/background) are expected to happen at the vicinity of hands or at a direction compatible to their intended motion. Therefore, knowledge on the location, posture and motion of the hands may constitute an important cue in driving attention. Inversely, contextual knowledge (e.g. knowledge of the geometry of environmental structures) may be used to constraint the perception of humans and hands.

In particular, and, according to the Technical Annex of the project, D3 presents activities in the context of tasks 1.3 and 1.4. The objectives of these tasks are the following:

- **[Task 1.3] - Observing humans:** Definition and development of a system that detects and tracks humans and their movements in particular. Activities in this task will focus on the important problem of acquiring real 3D motion of the arms while the human is interacting with objects. The tracking should be successful also in cases when the robot does not have a frontal view of the human.
- **[Task 1.4] - Observing human grasping:** Definition and development of a computational method that detects, tracks and represents human hands in action. The derived representation includes aspects and features in the full 4D spatiotemporal space (3D space and time dimensions). The aim is to extract from a sequence of stereoscopic hand observations, the information that is necessary and sufficient for subsequent (WP2) parsing and interpretation of observed hand activities that, in turn, support future repeats by a robotic hand. Activities within this task will address important subproblems such as figure-ground segmentation (environmental modelling, motion/colour based segmentation, coarse object categorisation) tracking humans/hands in 2D/3D (feature selection, hand models, representation of prior knowledge of motion models, prediction and search strategies), etc.

The related WP1 activities follow the spiral development model. According to this model, the development proceeds in cycles, each of which improves incrementally the sophistication, functionality and quality of the results of the previous cycles. The development of the tracking modules in GRASP proceed in several such cycles, each consisting of three phases: a) specifications phase, b) research and integration

phase and, c) trials and validation phase. Each cycle addresses the tracking problem at different levels of complexity regarding the information extracted from the sensory data, the assumptions regarding the human actor and the environmental conditions. The definition of these phases makes it possible to use the evaluation of the results of one cycle as important feedback for the next cycle, the early detection of potential problems and the reorientation of method design and implementation. It should also be noted that the adoption of the spiral development model in WP1 of GRASP capitalizes heavily on know-how, research results and technologies that are already available to the involved members of the consortium. Along these directions, several activities have been carried out in the course of the twelve first months of the GRASP project. The rest of this document is organized in three chapters.

Chapter 2 reports on the activities related to the task 1.3, "Observing humans". The basic approach to the problem is presented in section 2.1 where UniKarl presents a method for 3D human body detection and tracking. This work was accepted for publication in the ICRA'09 conference. The manuscript of this publication is provided as an attachment at the end of this deliverable. Section 2.2 presents a method for partial, elastic 2D shape matching technique that has been developed by FORTH, which can be used to improve the observation model of a human body tracking method. Additionally, in section 2.3 FORTH presents a novel technique that, based on skin color, can robustly track naked human body parts such as faces and hands.

Chapter 3 reports on the activities related to the task 1.4, "Observing human grasping". This chapter presents in detail the work carried out by FORTH along this direction. The description includes the rationale behind developing a hand tracker specifically for grasping hands, the construction of a 3D hand model and of the tools to manipulate it, the observation model as well as the techniques considered for tracking the 3D hand model in image sequences based on the adopted observation model. In section 3.2, FORTH describes preliminary work on a parallel and, hopefully, very useful approach towards a compact representation of hand trajectories that is based on Restricted Boltzmann Machines. The rationale behind this work is that this approach might well enhance the robustness and the quality of hand tracking results when appropriately integrated with a probabilistic hand tracking framework. Section 3.3 reports on joint activities among LMU, UniKarl and FORTH in acquiring an annotated data set of hand grasping sequences. Several subjects perform different type of grasps on several types of objects. Each and all of these experiments is observed by a calibrated stereo vision system and image sequences are recorded. Simultaneously, a Polhemus tracker, records (synchronously to the image sequences) the trajectories of eight carefully selected points on the subject's hand (five fingers, wrist, thumb, and center of palm). The resulting data set is expected to prove very useful as a benchmark for the qualitative and quantitative evaluation of hand tracking in the context of grasping activities.

The deliverable concludes in chapter 4, with a brief summary of current and future activities towards the goals of WP1.

Chapter 2

Human body detection and tracking

This chapter reports on the activities related to the task 1.3, "Observing humans" of WP1. The basic approach to the problem is presented in section 2.1 where UniKarl presents a method for 3D human body detection and tracking. This work was accepted for publication in the ICRA'09 conference. The manuscript of this publication is provided as an attachment at the end of this deliverable. Section 2.2 presents a method for partial, elastic 2D shape matching technique that has been developed by FORTH, which can be used to improve the observation model of a human body tracking method. Additionally, in section 2.3 FORTH presents a novel technique that, based on skin color, can robustly track naked human body parts such as faces and hands.

2.1 Human motion tracking

We have investigated new methods to robustly track the motions of persons in real-time without special marker setups. Our focus was on methods that can be implemented with the sensors available on a humanoid robot (e.g. stereo-cameras) and that can track human motions in every-day environments. In order to perceive and understand human grasping, it is not sufficient to track the hands only, but whole body tracking is required. This task becomes particularly hard when the human is perceived through the robot's onboard cameras only. By using the cameras built-in the robot head only, the angle of view is limited to essentially a single perspective and depth resolution is limited due to the small baseline. In the attached paper [FHD09], we present a novel approach, which is able to deal with these issues and still run in real-time on regular hardware.

2.1.1 Method description

In our previous work, we presented a stereo-based approach that combines 3D position tracking of key body parts (hands and head) with articulated upper body tracking in real-time [AUAD07]. The goal was on accurate acquisition of real 3D upper body motion, intended for reproduction and imitation with humanoid robot systems. The system runs with a processing rate of approx. 15 Hz on regular hardware.

Our primary goal was now to remove the dependency on the skin-color based hand tracker, in order to achieve more robust application in the presence of occlusions and heavily cluttered backgrounds. Secondly, our previous method required a uniformly colored shirt for the purpose of figure ground segmentation, in order to distinguish body contour edges from background edges. Furthermore, tracking of the legs was to be incorporated. Note that the increase in robustness by removing the dependency on a hand tracker is bought at the expense of less accurate estimation of the hand positions.

In the attached paper [FHD09], a novel approach to the 3D human motion tracking problem is proposed, which combines several particle filters with a physical simulation of a flexible body model. The flexible body model allows the partitioning of the state space of the human model into much smaller subsets, while finding a solution considering all the partial results of the particle filters. The flexible model creates the necessary interaction between the different particle filters and allows effective semi-hierarchical tracking of human body motion. The physical simulation does not require inverse kinematics calculations and is

hence fast and easy to implement. Furthermore an appearance model is built on-the-fly, which allows successful application without figure ground segmentation. The system offers a convenient initialization procedure, which allows to start tracking automatically. The processing rate amounts to 10 Hz on a regular PC using a stereo camera and is hence suitable for Human-Robot Interaction applications.

The core of the system is a physical simulation of a flexible body model. This model is a mass-spring system that models important joints as mass-points and the bones as springs between the joints. The structure of the model is illustrated in Fig. 2.1. The model is enhanced with further constrains for the range of angular motions and self collisions.

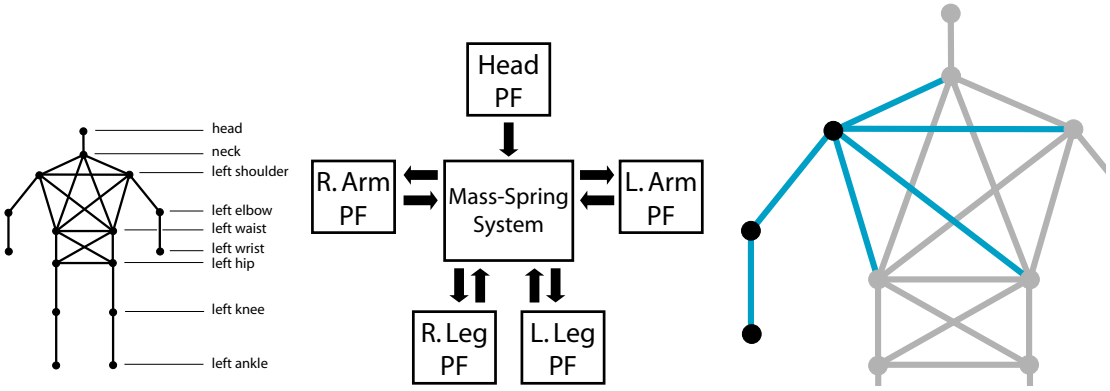


Figure 2.1: Left: Mass-Spring model of a human. Center: Interaction between the particle filters (PF) and the simulation. Right: Subset of the model used for filtering.

The actual tracking consists of two steps: First the head is tracked with a regular skin-color based head tracker – this is the only part that makes use of the stereo camera – and the position of the model is moved according to the head position. Then the arms and legs are tracked by first estimating the state of the arms and legs individually with the particle filters. In the next step, the four estimates are integrated into a final estimate for the body state by inserting the results for the extremities into the body model and then running the physical simulation under gravity. Fig. 2.1 (center) shows the interaction between the particle filters and the model. The particle filters and the simulation interact with each other and can mutually correct mistrackings.

The particle filters for the extremities make use of a motion model that is tailored to tracking with a single perspective, in particular by increasing noise in the depth direction. The constraints of the flexible model are enforced within the motion model. This allows to deal with non-linear arm and leg motion (see Fig. 2.1, right). The measurement model for evaluating a given hypothesis combines and edge cue and a region-based color cue. The arms and legs are modeled as cylinders, which allows efficient projection into the image. The measurement model incorporates an occlusion model, which is important when using only a single perspective. The color cue compares the color in the image with a color model for the arms and legs, which is learned during the automatic initialization procedure.

The single-threaded tracking system runs on a Pentium 4 3.2 GHz CPU with a processing rate of 10 Hz. Image processing is performed on 640×480 color images. The particle filters use 200 particles per extremity and 100 particles for the head. Processing time amounts to ≈35 ms for the image processing and ≈50 ms for particle filtering, where the motion model is consuming about half the processing power.

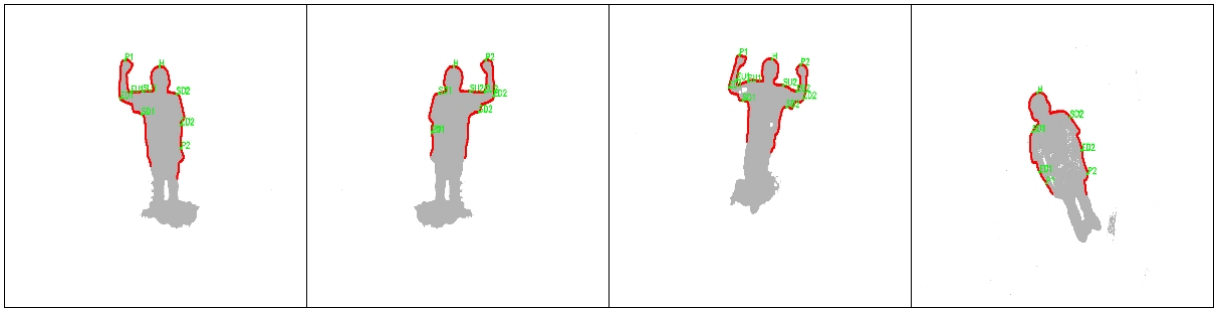
The tracking of the arms is able to follow the motions of a single arm robustly, even through complicated and ambiguous situations. It is even able to detect if parts of a limb are hidden behind the body or the head. The interactions of the two arms are tracked as long as both arms are visible to a certain extent, cross-over situations are successfully tracked due to the occlusion model. Legs are tracked individually, including the detection of knee bends and folded up calves. When a person turns sideways, one leg is completely occluding the other, which can cause significant confusion for the filters. As the tracking depends strongly on the localization of the head, more precisely the face, the person cannot turn away more than 90 degrees from the camera. The suspension of the head together with the simulation of gravity implies that the person must be standing. The constraint framework allows easy implementation of external constraints such as collisions with objects or furniture.

2.2 Recognition of human postures based on elastic, partial, scale invariant contour matching

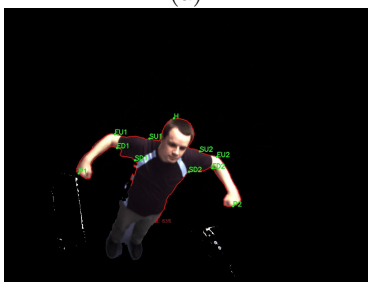
2D shape matching is a fundamental problem in computer vision which amounts to describing a 2D shape and calculating its similarity to others. The rationale behind this work in the context of GRASP is that given such a method, the 2D outline/contour of a human figure in an image can be matched with one of a few example, prototype contours, representing a human in a particular posture. This might be helpful in both (a) identifying a segmented object as a human and (b) recognizing their posture. Despite the fact that this approach has been originally developed for human detection and posture recognition, the methodology is broad enough and can be applied also to perform hand posture recognition or even object recognition. As such, the interest in it is not isolated to the task 1.3 of GRASP (human detection and tracking) but extends to the task 1.4 (hand detection and tracking) and also to the WP4 activities related to environment perception. The methodology followed to the problem of 2D shape matching is the theme of a publication under preparation.

2.2.1 Brief overview of the proposed method

Most of the existing shape matching methods [LJ07, BMP02, EAACC08, FS07, BCB09, WTYY09, dSTF07, BET09] work by defining shape descriptors which are then compared by the selection of an appropriate distance function. The smaller the distance of the representations, the closer the shape similarity between source and reference shapes. The quality of the shape matching process depends on whether the shape descriptor really captures the essential, intrinsic aspects of shape and on whether the similarity measure between such descriptors is a function that agrees with the human perception of shape similarity. Despite their simple definition, both these dimensions of shape matching prove



(a)



(b)

Figure 2.2: Matching open with closed contours. A set of open contour prototypes (a) have to be matched with a closed contour (b). Figure (b) also shows the results of matching.

to be challenging problems. Shapes belonging to the same class may differ substantially in scale and orientation. For the particular case of articulated and/or deformable objects such as humans and hands, transformations between instances of the same object may result in quite different shapes. Contours to be matched are the result of some kind of segmentation process which, being imperfect, may result in considerable amounts of noise that needs to be tolerated. Objects may appear occluded by other objects, so the identification of the correct class should also be based on partial evidence. Being viewpoint dependent, 2D shapes of the same objects may differ considerably. On the other hand, shapes that

are globally quite similar may correspond to different classes because of the existence of some local discriminating property. Last but not least, in realistic settings, all of the above complexities do not appear isolated, but combined, making the solution of the 2D shape matching problem a very challenging one. The problem is further complicated when one needs to match *part* of a closed contour with an open one [CFH+09, LMWY07].

Consider for example the situation shown in Fig.2.2. Figure 2.2(a) shows four prototype open contours (shown in red) corresponding to parts of a human figure. These prototypes are annotated with points corresponding to critical aspects of human shape (joints). Given another, possibly scaled, rotated and deformed closed contour of a human figure (see figure 2.2(b)), we are interested in determining which of the open prototype contours of figure 2.2(a) matches best to the closed contour of the human figure in figure 2.2(b). It can easily be verified that all the aforementioned difficulties contribute in complicating the 2D shape matching problem. None of current shape matching techniques provides solutions to all problems listed above [LMWY07]. The proposed approach solves efficiently this problem and gives, as a byproduct, the correspondence of the selected points between the open and the closed contours (also shown in figure 2.2(b)).

In the context of this work we developed a new technique for matching 2D shapes. First, we proposed a new descriptor as a means of local, 2D shape representation. The proposed descriptor is, by construction, totally scale and rotation invariant. Moreover, it tolerates substantial articulations and non-rigid shape deformations. Given this descriptor, we introduce a scheme for sampling a given 2D contour for deciding where 2D shape descriptors should be computed. The rationale behind this spatially non-uniform sampling method is to provide sparser shape representations in areas of the shape that are less discriminant and denser shape representations in areas of the shape that are more discriminant. This makes it possible to provide less ambiguous input to the global shape matching method performed in subsequent steps. Another key property of this sampling process is that it produces approximately the same number of shape descriptors in scaled and rotated versions of the same shape. Once the shape descriptors have been computed, a variant of an existing dynamic-programming based matching technique [SFC07] is proposed to take care of the global 2D shape matching process. The key novelty in this variant is its ability to handle partial matching. Thus, matching of a source, open contour to the best matching part of another closed, target contour can be established. Figure 2.2(c) shows an example of matching the open contour of Fig.2.2(a) to the contour of the foreground region in Fig.2.2(b). The details of the proposed approach is part of a publication which is currently under preparation.

2.2.2 Experimental results

The proposed approach for 2D shape matching has been validated by several experiments. The experiments can be classified into two classes, one related to the performance of the variant of the method for closed contours, and another one that assesses the capabilities of the proposed method to match closed and open contours.

For the problem of full shapes matching, and for the sake of quantitative evaluation, we performed experiments with MPEG-7 Core Experiment CE-Shape-1 data set [Lat]. This data set contains 70 different classes of objects, each containing 20 class representatives, resulting in 1400 different images. We did an exhaustive classification test that employed each of the 1400 images as a query object. The goal of the proposed method was to rank the similarity of the query object against the full data set of 1400 images. Representative results from this experiment are shown in Figure 2.3.

Figure 2.4 presents quantitative results on the performance of the proposed method on the Bullseye test over the MPEG7 data set. The performance of another state of the art method [LJ07] is also provided for comparison.

Similar results have been also obtained in “GESTURES” and “MARINE” data sets [Pet].

On the MPEG7 data set, we also tested the performance of the proposed method in matching open contours. Characteristic examples are shown in figure 2.5.

Additionally, the proposed method has been tested in more GRASP-oriented, home-made data sets. More specifically, open contours corresponding to hand postures have been manually defined (figure 2.6, top) and then matched in long sequences of performing hands. Snapshots of the results obtained are shown in figure 2.6, bottom. Analogously, open contours corresponding to human body postures have been manually defined (figure 2.7, top) and then matched in long sequences of performing humans. Snapshots

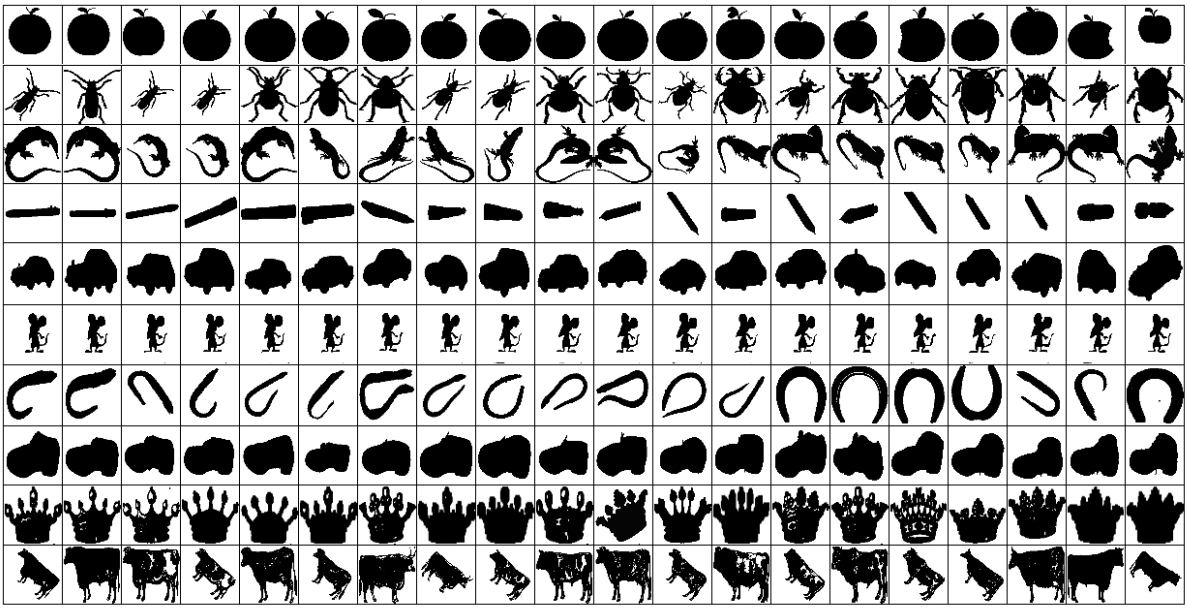


Figure 2.3: Characteristic results for the MPEG7 sequence. The first column corresponds to the query image, the rest columns are retrieved shapes in the order of decreasing similarity.

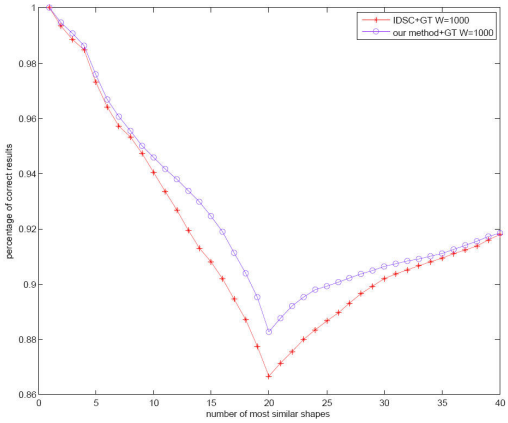


Figure 2.4: Performance of the proposed method in the Bull's Eye test on the MPEG7 data set.

of the results obtained are shown in figure 2.7, bottom. As a byproduct, the results show the identified human body joints in the test snapshots.

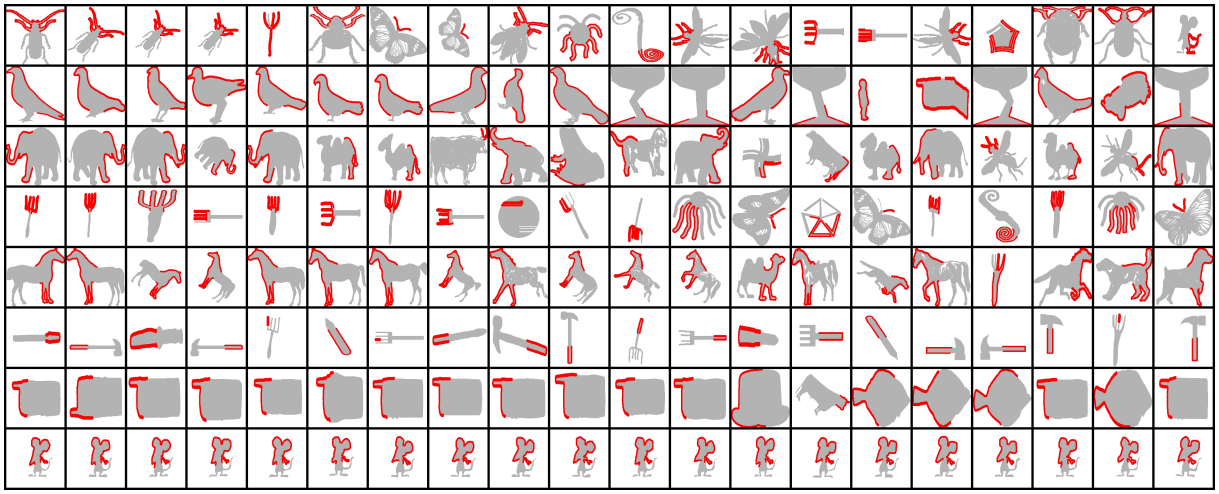


Figure 2.5: Characteristic examples of open (partial) contour matching in the MPEG7 data set.

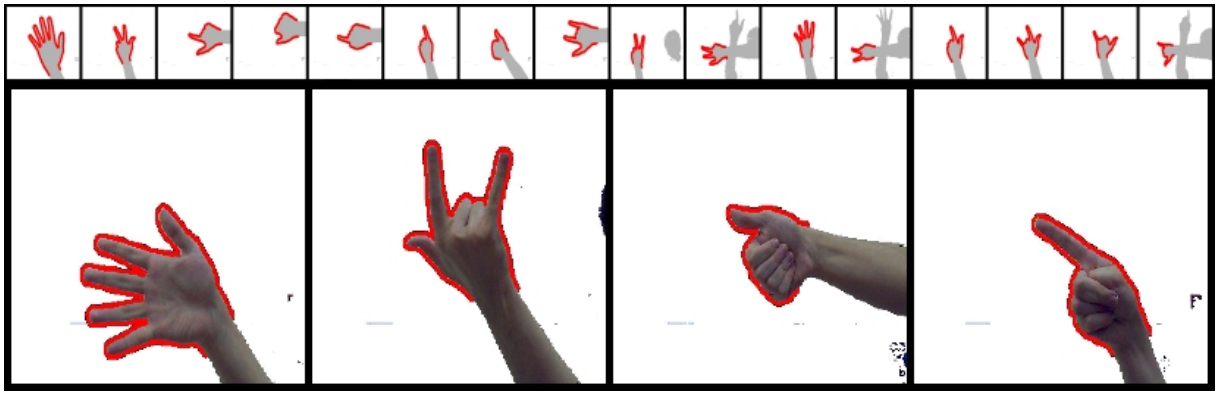


Figure 2.6: Matching open with closed hand contours. A set of open contour prototypes top) have to be matched with closed contours (bottom).

Figure 2.7: Matching open with closed body contours. A set of open contour prototypes (top) have to be matched with closed contours (bottom).

2.3 Skin-color based tracking of human body parts

In this section, we present a novel technique that, based on skin color, can robustly track naked human body parts such as faces and hands. Such body parts may move in complex trajectories, occlude each other in the field of view of the camera and vary in number over time [AL04, BALT08, ZBA08]. A block diagram of the proposed method is presented in Fig. 2.8. As shown in this figure, the proposed approach consists of three layers.

2.3.1 Layer 1: Assignment of probabilities to pixels

Within the first layer, the input image is processed to identify pixels that potentially depict human hands and faces. For this purpose, we utilize color information as well as information from a foreground/background extraction algorithm. The results of this first step is the probability $P(H \mid C = c, F = f)$ for each pixel that this pixels belongs to a hand or a face (H), given its color c and the information of whether it belongs to foreground or not ($F = f$).

2.3.2 Layer 2: From pixels to blobs

This layer applies hysteresis thresholding and connected components labeling on the probabilities determined at layer 1. These probabilities are initially thresholded by a "strong" threshold T_{max} to select all pixels with $P(H \mid C = c, F = f) > T_{max}$. This yields high-confidence skin pixels that constitute the seeds of potential hand/face blobs. Then, a second thresholding step is performed with a "weak" threshold T_{min} that also takes into account connectivity with already identified skin-colored pixels. During this step, pixels with probability $P(H \mid C = c, F = f) > T_{min}$ where $T_{min} < T_{max}$, that are immediate neighbors of skin-colored pixels, are recursively added to each blob. A connected components labeling algorithm is then used to assign different labels to pixels that belong to different blobs. Size filtering on the derived connected components is also performed to eliminate small, isolated blobs that are attributed to noise and do not correspond to meaningful skin-colored regions.

2.3.3 Layer 3: From blobs to hypotheses

Within the third processing layer, blobs are assigned to hand and/or face hypotheses which are tracked over time. Tracking over time is realized through a scheme which can handle multiple objects that may move in complex trajectories, occlude each other in the field of view of a possibly moving camera and whose number may vary over time.

According to this approach, a new hypothesis is created each time a new blob appears. A Kalman tracker is assigned to each hypothesis in order to track its location and its speed. In order to solve the

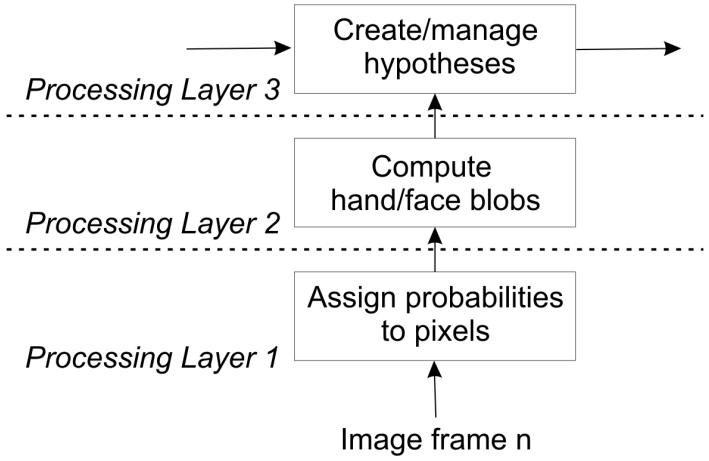
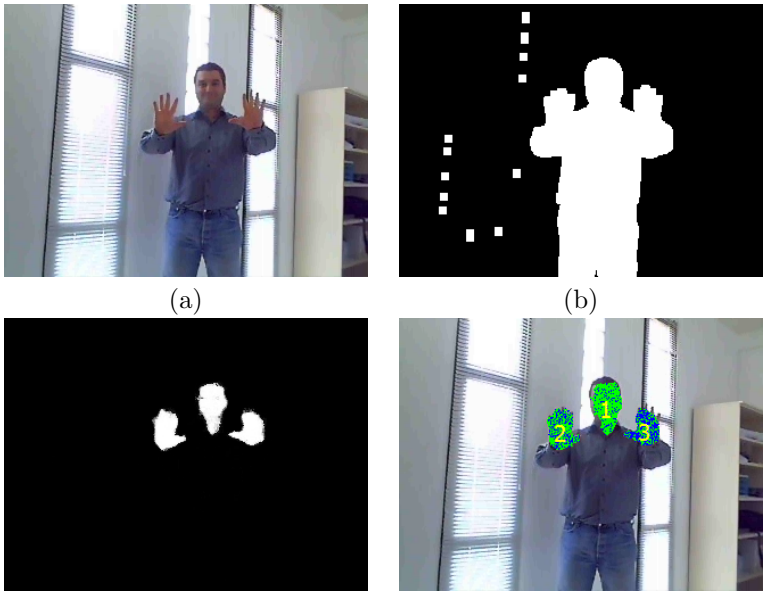


Figure 2.8: Block diagram of the proposed approach. Processing is organized into three layers.

data association problem, i.e., to assign image pixels to hypotheses in the presence of occlusions, a novel approach has been developed and used. Image pixels are propagated in time according to a motion model that corresponds to the current Kalman filter state and the uncertainty in the position and the speed of the whole blob. Pixels belonging to observed blobs are assigned to old hypotheses according to the density of the propagated pixels.



(a) (b)

(c) (d)

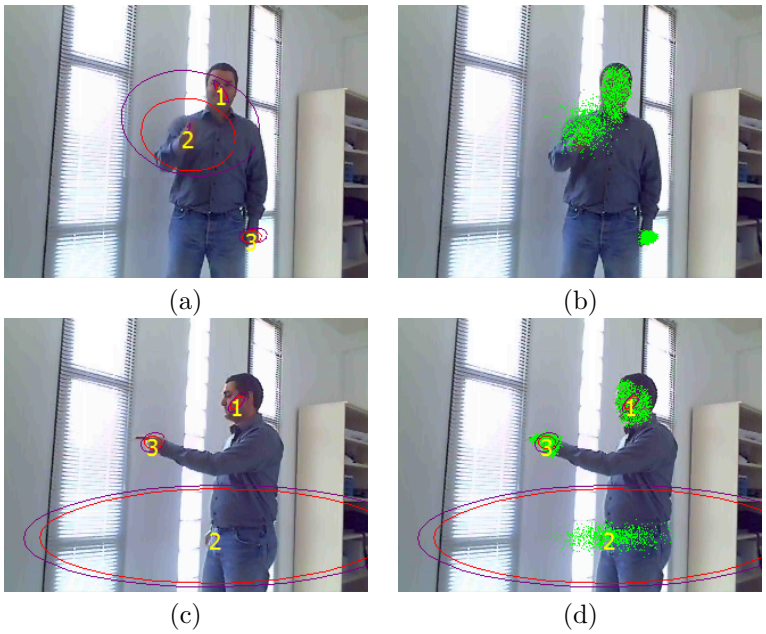
Figure 2.9: The proposed approach in operation. (a) Initial image, (b) background subtraction result, (c) pixel probabilities, (d) hand and face hypotheses.

Fig. 2.9 depicts various intermediate results obtained at different stages of the proposed approach. A frame of the test sequence is shown in Fig. 2.9(a) Fig. 2.9(b) depicts the result of the background subtraction algorithm, i.e., $P(H \mid C = c, F = f)$. Fig. 2.9(c) depicts the pixel probabilities and, finally, Fig. 2.9(d) depicts the hand and face hypotheses as tracked by the proposed approach

The operation of the tracker developed for layer 3 of the proposed approach is illustrated in Fig. 2.10 and Fig. 2.11. In Fig. 2.10(a) and Fig. 2.10(c) we can see the magnified 95% uncertainty ellipses for the predicted location and speed of each hypothesis. As can be easily observed, hypotheses that move rapidly (e.g., hypothesis 2 in Fig. 2.10(a)) or hypotheses that are not visible (e.g., hypothesis 2 in Fig. 2.10(c)) have larger uncertainty ellipses. On the other hand, hypotheses that move very slowly (e.g., face hypotheses) can be predicted with more certainty. This is also reflected on the location of the predicted pixels (Fig. 2.10(b) and Fig. 2.10(d)) which tend to be more concentrated (higher density) for hypotheses that are more predictable.

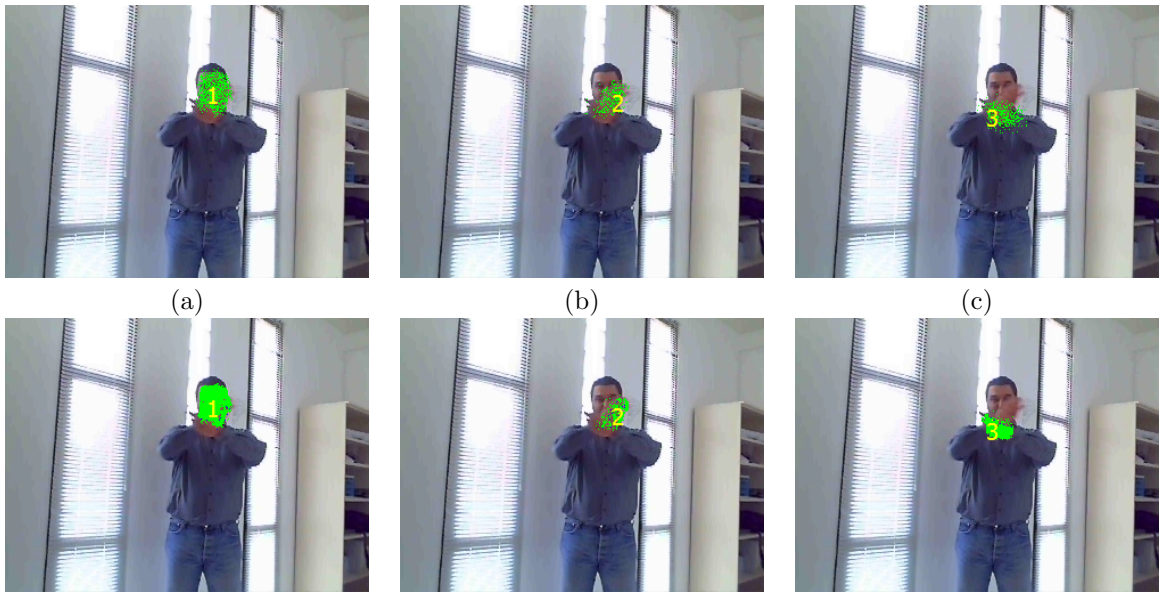
In Fig. 2.11 we can see how the algorithm behaves in a case where three hypotheses simultaneously occlude each other, leading to a difficult data association problem. The top row depicts the predicted pixel locations for each of the three valid hypotheses. The bottom row depicts the final assignment of blob pixels to hypotheses, according to the density of the predicted pixel locations.

The performance of the proposed tracker is also demonstrated in Fig. 2.12 which shows the state of the tracker corresponding to a number of frames out of the same image sequence. As can be easily observed, the tracker succeeds in keeping track of all the three hypotheses despite the occlusions introduced at various fragments of the sequence.



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

Figure 2.10: Tracking hypotheses over time (a,c) magnified uncertainty ellipses corresponding to predicted hypotheses locations and predicted hypothesis speed, (b,d) predicted pixel locations.



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

Figure 2.11: A difficult case with three hypotheses that appear as a single blob. (a,b,c) predicted pixel locations for each of the three hypotheses, (d,e,f) pixels finally assigned to each hypothesis.

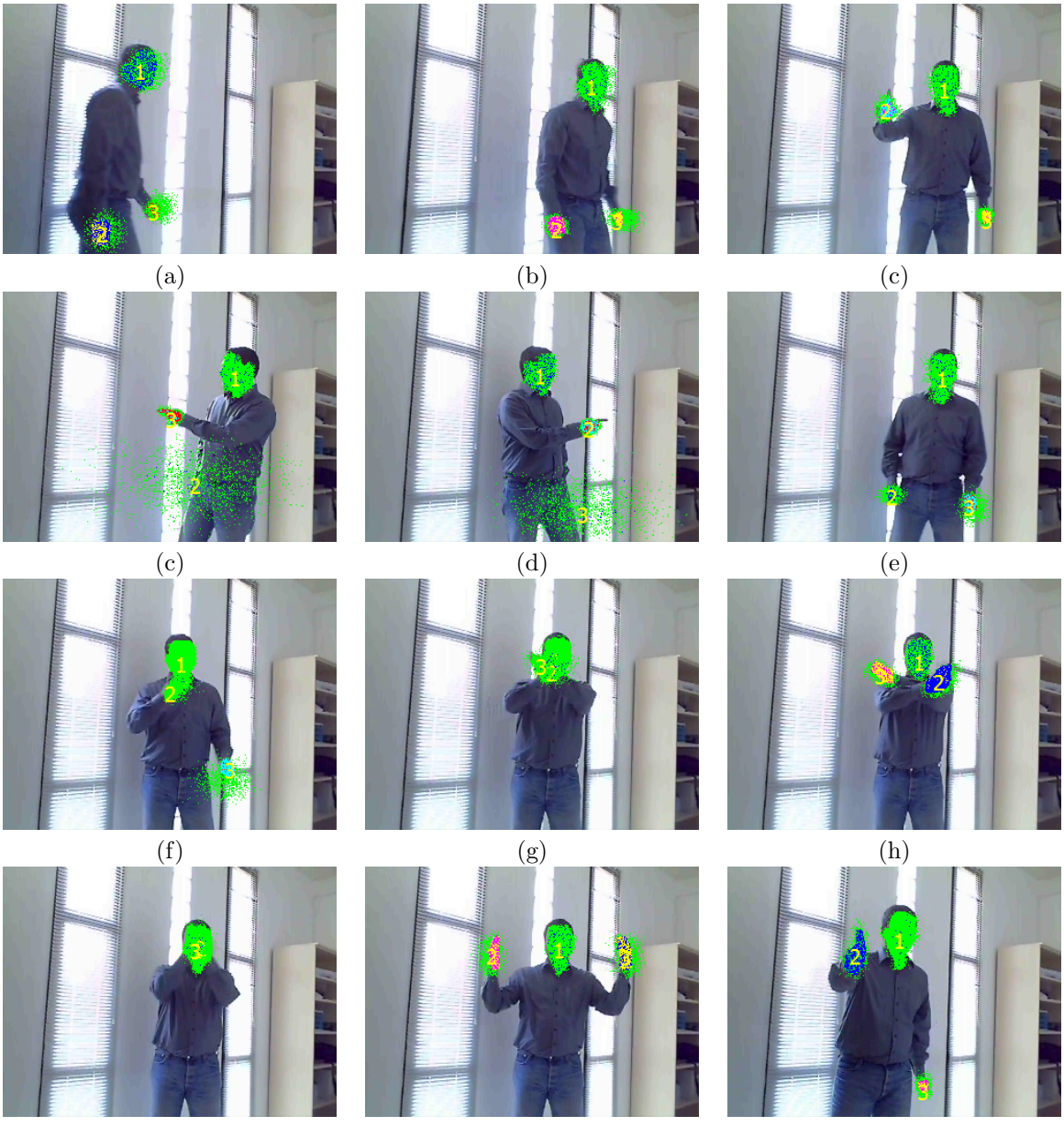


Figure 2.12: Indicative tracking results for twelve segments of the office image sequence used in the previous examples. In all cases the algorithm succeeds in correctly tracking the three hypotheses.

Chapter 3

Human hand detection and tracking

This chapter reports on the activities related to the task 1.4, "Observing human grasping". This chapter presents in detail the work carried out by FORTH along this direction. The description includes the rationale behind developing a hand tracker specifically for grasping hands, the construction of a 3D hand model and of the tools to manipulate it, the observation model as well as the techniques considered for tracking the 3D hand model in image sequences based on the adopted observation model. In section 3.2, FORTH describes preliminary work on a parallel and, hopefully, very useful approach towards a compact representation of hand trajectories that is based on Restricted Boltzmann Machines. The rationale behind this work is that approach might well enhance the robustness and the quality of hand tracking results when appropriately integrated with the probabilistic hand tracking framework. Section 3.3 reports on joint activities among LMU, UniKarl and FORTH in acquiring an annotated data set of hand grasping sequences. Several subjects perform different type of grasps on several types of objects. Each and all of these experiments is observed by a calibrated stereo vision system and image sequences are recorded. Simultaneously, a Polhemus tracker records (synchronously to the image sequences) the trajectories of eight carefully selected points on the subject's hand (five fingers, wrist, thumb, and center of palm). The resulting data set is expected to prove very useful as a benchmark for the qualitative and quantitative evaluation of hand tracking in the context of grasping activities.

3.1 Detection and tracking of a grasping hand

Hand tracking has great potential as an attractive method for providing natural human-computer interaction (HCI). Other areas of hand tracking application include navigation in virtual environments, gesture recognition, human-robot interaction and motion capture for animation use.

Capturing hand articulation is a challenging problem, because the hand is the most effective, general-purpose interaction tool among the other body parts, presenting dexterous functionality in communication and manipulation and exhibiting many degrees of freedom. Representing the hand pose by the angles at each joint, the configuration space spans a 27-dimensional space. In addition, self-occlusions of fingers introduce uncertainty for the occluded parts. Another reason that makes the problem really difficult is that a huge amount of data needs to be processed for each sequence of frames and a real-time implementation of any algorithm becomes quite demanding in terms of computational power. Finally, the fact that the hand moves often in front of cluttered background and that it has very fast motion capabilities make the real-time tracking procedure a very demanding task.

Two general approaches have been pursued to capture hand articulation. The first one is the *appearence-based* approach, which estimates hand configurations from images directly after having learned the mapping from the image feature space to the hand configuration space [AS03, RASS01, Shi01, SSKM98, WH00]. The mapping is highly nonlinear due to the variation of the hand appearences under different views. Further difficulties are posed by the collection of large training data sets and the accuracy of pose estimation. On the other hand, appearence based methods are usually fast, require only a single camera and have been successfully employed for gesture recognition tasks.

The second approach is the *model-based* approach, which uses a 2D or 3D hand model [RK94, SMC01, SMFW04]. In case of a 3D model the hand pose is estimated by matching the projection of the model

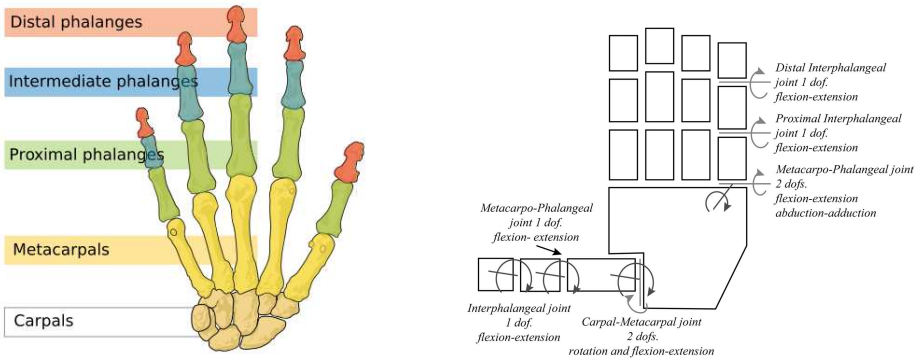


Figure 3.1: Skeletal hand model: hand anatomy (left) and hand kinematics (right)

and the observed image features. The task is then formulated as a search problem in a high dimensional configuration space. The research areas of this method include the efficient construction of realistic 3D hand models, the dimensionality reduction of configuration space and the development of fast and reliable tracking algorithms to estimate the hand posture.

3.1.0.1 Literature review

Different methods have been proposed to capture human hand motion. A recent overview of hand tracking methods in the context of Human Computer Interaction (HCI) can be found in [EBN+07]. The following review will be divided into four parts, corresponding to hand modeling methods, observation modeling methods, filtering methods used for tracking and dimensionality reduction methods used to handle the high dimensional nature of the problem.

Hand modeling

Several approaches have been presented for the modeling of hand's kinematics and shape, ranging from simple and rough to complete and analytical models.

Regarding the kinematics modeling, most of the works adopt a 27 DOF model, which complies with recent biomechanical studies [DPD+07] (see Figure 3.1). The nine Intermediate phalanges (IP) joints can be accurately described as having only one DOF, flexion-extension. All five Metacarpo-Phalangeal (MCP) joints, however, are described with two DOF, namely abduction/adduction in the plane defined by the palm and flexion/extension. The kinematic hand model described above is the most natural choice for parameterizing the 3D hand configuration but there exist a few exceptions using other types of representations. Sudderth et al. [SMFW04] proposed independent rigid bodies for each component of the hand, leading to a highly redundant model. The kinematic constraints between these rigid bodies were enforced using a prior model in a nonparametric belief propagation scheme. Heap et al. [HH96] modeled the entire surface of the hand using PCA applied on MRI data. Such a representation requires further processing to extract useful higher-level information, such as pointing direction. However, it was shown to be very effective in reliably locating and tracking the hand in images.

As far as the shape modeling of the hand is concerned, both articulated and elastic components have to be taken in consideration. However, computational efficiency reasons make the use of very complex shape models inevitable, because the hand model needs to be projected many times on the input image(s) to obtain matching features. The above fact has motivated the use of geometric primitives such as cylinders, spheres and ellipsoids attached to each link of the hand skeleton. Stenger et al. [SMC01] used quadrics as shape primitives. Using projective geometry, fast algorithms were provided for projecting the hand model on the image plane and accounting for the self-occlusions during the projection phase. An even more economical, view-dependent model called *cardboard model* was used by Wu et al. [WH01]. Although these rough hand models can be processed efficiently, one may also anticipate systems with better performance using more complex models. In [KH94] a B-spline surface was used to model the hand's surface and in [BKMVG04, DDH04, dlGPF08] a deformable skin model was implemented using a skinning technique. However, some of these studies make use of 3D features obtained from depth sensors or stereo vision,

eliminating complex projection operations.
Finally, the problem of kinematic fitting is tackled in various ways. In most cases manual calibration is performed, using markers or landmark points. Recenty, Lien et al. [Lie05] introduced a marker-based calibration method using scalable inverse kinematic solutions.

Observation modeling

Feature extraction and matching is a very crucial module in a hand tracking system. The design and implementation of this module has a considerable effect on the robustness of the system to self-occlusions and background clutter. The hand motion creates images that are very difficult to analyze in general. High level features (such as fingertips, fingers etc) provide a compact representation of the input supporting high speeds. However, the extraction of features can not be done robustly without any severe pose restrictions. Therefore the majority of studies rely on low-level features.

Edges or contours are features that can be used in any model-based technique. In [SMC01] for example a volumetric model of the hand is projected to the images and the occluded contours are calculated. The point correspondences between 3D model contour points and image edges are computed based on a certain proximity criterion. The distance between corresponding points give the matching error. Edge orientation is combined with chamfer matching to a more robust application in [TSTC03]. In [STTC04, SMFW04] skin color models were employed to increase robustness. The likelihood of the segmentation, asserted by the projection of the model, was calculated using background and skin color models as a measure of similarity. A combination of edges, optical flow and shading was proposed in [SLSO03] to successful track hand motion, under severe occlusions.

There are also a few studies that use 3D features. 3D data contain valuable information that can help eliminate problems due to self-occlusions which are inherent in image-based approaches. On the other hand, an exact, real-time and robust reconstruction of the hand from multiple cameras is very difficult and computationally inefficient.

Filtering methods

The tracking algorithms that have been applied to hand tracking problem can be divided to two categories. The first one contains single hypothesis algorithms while the second one constains algorithms that can track multiple hypotheses.

Single hypothesis tracking corresponds to a best fit search where the matching error is minimized. For this purpose, Gauss-Newton method augmented with a stabilization step was used in [RK94]. A plethora of other minimization algorithms were used, including Nelder Mead Simplex (NMS) [OH99a], Genetic Algorithms (GA) and Simulated Annealing (SA) [NSMO96]. Bray et al. [BKMVG04] used Stochastic Gradient Descent along with depth features. A small number of points were selected randomly at each iteration to reduce computational cost and avoid spurious local minima. The resulting algorithm was called Stochastic Meta Descent (SMD). Finally, the Unscented Kalman Filter has been used for single hypothesis tracking in [SMC01]. UKF applies a deterministic weighted sampling of the Gaussian posterior, in order to be able to handle a nonlinear observation model.

There are many approaches for multiple hypotheses tracking. The majority of them use a Bayesian approach or some similar formulation. Particle filtering is a well-known technique for implementing recursive Bayesian estimation using Monte Carlo simulations. It is used in [SMFW04] in combination with a Bayesian network, in which inference was made using Nonparametric Belief Propagation algorithm to produce robust tracking results in the presence of self-occlusions. Thayananthan et al. [TSTC03] have implemented Bayesian filtering in a grid-based manner, resulting in the algorithm called tree-based filtering. Bayesian filtering was performed over the tree by assuming piecewise constant distribution over its leaves. During tracking, the tree was traversed to update the probabilities.

Dimensionality reduction methods

Although active motion of the hand is highly constrained, this fact is not reflected in the kinematic model. An attempt to capture natural hand motion constraints is by complementing the kinematic model with direct constraints. However, the very intricate structure of the hand does not allow expressing all the

constraints in a closed form. Furthermore, in some specific applications (e.g. grasping) the motion of the hand is further restricted by the scope of the spcecific movement and therefore other constraints which have nothing to do with structural limitations are also posed. These problems have motivated the use of learning-based approaches, which exploit some ground truth data (collected using data gloves or other types of sensors). Thus, the feasible configurations of the hand are expected to lie on a lower dimensional manifold due to the constraints that exist. Lin et al. [LH04] applied PCA on a large amount of joint angle data to construct a 7-dimensional space. Another way to use data collected from a glove is to generate synthetic hand images to build a template database of all possible hand configurations as in [STTC04, AS03]. In another approach, the dynamics of hand motion were learned in an attempt to help the tracking algorithm. Precisely, [ZH03] presented an eigen-dynamic analysis method for modeling the non-linear hand dynamics and applied PCA to reduce the dimensionality of the problem. Finally, Thayananthan et al. [TSTC03] represented the configuration space as a tree, which was constructed using hierarchical clustering techniques. This tree structure enables fast hierarchical search through Bayesian Filtering.

3.1.0.2 Design decisions

Being motivated by the aforementioned literature review we have taken some decisions regarding the components that will be used to design our hand tracking system. The fact that the hand tracker will be used during hand grasping motions played a principal role to the design of our system.

Therefore we decided to adopt the hand model proposed by Stenger et. al in [SMC01]. This model, despite the fact that it is not complete it captures the kinematics and the shape of the hand in a quite precise way, such that the residual error between model projection and hand image is assumed to be negligible. Additionally, the projection of the 3d model into the image plane is done in a computationally efficient way, allowing the tracking system to run in real-time or close to real-time.

Another design decision that we have taken is the use of edge and skin color cues in the observation model. These low-level features can be efficiently computed and matched with the model projection. This characteristic turns to be very important as the feature extraction and matching procedure is repeated many times during the minimization of the residual error.

Finally, regarding the tracking algorithm and the dimensionality reduction approach that will be followed, research is still in progress. Although some initial tracking system has been implemented (UKF), it does not fulfill our requirements. Thus, another tracking system in combination with some dimensionality reduction algorithm will be developed. Some more details can be found in the future work section of this report.

3.1.1 3D model of a human hand

3.1.1.1 Types of models and model selection

The choice of a particular model for the human hand raises important questions and is a crucial task that affects its robust and effective tracking in a sequence of images. Most of the common models used for modeling the human hand or other human limbs could be described as being based on a skeleton that contains a set of articulations connected by links that define their relative positions. Different types of primitives representing either parts of the hand or some features that can be observed or measured, are attached to these skeletons.

Some authors use models that approximate parts of the human hand, in terms of shape, while others prefer simpler models that are rough representations of the shape but precise in terms of relative position of some particular features. Most models that are created for animation purposes are based on splines or meshes, as they can produce a more realistic visual representation of the modeled part (i.e. human hand, human body), whereas those created for computer vision applications are mostly based on quadrics or other simpler geometric shapes.

More specifically, the works of Stenger [SMC01], Ouhaddi [OH99b], Sidenbladh [SBF00] and Delemarre [QO01] employed rough geometric models based on cylinders, spheres and even parallelepipeds. More precise models were used by Terzopoulos [TM91] who used superquadrics, Sminchisesku [ST03] who used meshes derived from sampling the surfaces of superquadrics, Deutscher et al. [DBR00] who used quadrics

whose surface was sampled to create a mesh of 3D points and Lerasle [FGD99] who used a mesh of points to model a human leg.

The advantage of the use of quadrics comes from their simple manipulation and the possibility of applying all kinds of affine transformations and still obtain their definition in analytical form, as well as their projection in the image plane of some camera. On the contrary, analytical expressions for mesh based models are, in general, not possible to derive and their projection on an image must be done on a point-by-point basis while also taking care of their neighborhood relationships. Although mesh-based models are adequate for applications that require a great modeling precision, they may be completely inappropriate for other applications that require computational efficiency.

Deformable models have the advantage that they can be adjusted precisely to each tracked target, but these deformations represent additional degrees of freedom that increase the dimensionality of the problem. Another reason for rejecting this kind of models is that the articulated structures, as said before, are based on a skeleton whose configuration defines the appearance of the object. The deformations that appear on the object's surface due to some elasticity of the flesh and skin are normally small when compared with changes in the relative position of the limbs due to rotation around some articulations. Thus, rough models (e.g. based on quadrics) are better suited for vision-based tracking applications where the ease of model manipulation is of greater importance relative to the visual realism.

Taking into account the aforementioned considerations, the human hand is modeled as a 3D articulated structure composed of truncated quadrics. These geometric primitives can be easily manipulated and projective geometry provides the tools to obtain their projections in an elegant way. The adopted hand model does not model local deformations. This implies that when the model and the hand are in the same pose, a residual error between model projection and hand image still remains. However, it may be assumed that this error is not significant for the purpose of pose recovery, and may be tolerated given the efficient contour projection. In the following sections we provide a detailed description of the construction of the proposed hand model.

3.1.1.2 Projective geometry of quadrics and conics

A quadric in projective 3-space P_3 is the locus of all points $\boldsymbol{X}$ satisfying a homogeneous quadratic equation

$$\boldsymbol{X^T Q T} = 0 \tag{3.1}$$

where $\boldsymbol{X} = [x, y, z, 1]^T$ is a homogeneous vector representing a 3D point and $\boldsymbol{Q}$ is a 4x4 symmetric matrix only defined up to a scale. It can be easily verified that the equation above corresponds to the quadratic equation:

$$q_{11}x^2 + q_{22}y^2 + q_{33}z^2 + 2q_{12}xy + 2q_{13}xz + 2q_{23}yz + 2q_{14}x + 2q_{24}y + 2q_{34}z + q_{44} = 0 \tag{3.2}$$

Thus, a quadric depends only on nine independent parameters.

Quadrics of interest to modeling a hand

There are 17 standard types of quadratic surfaces, but only four of these types are of particular interest: ellipsoids, cones, elliptic cylinders and parallel planes. These types of quadrics are shown in Fig.3.3 and their properties are presented below in detail.

Ellipsoids are represented by

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{3.3}$$

and the corresponding matrix is given by

$$\boldsymbol{Q} = \begin{bmatrix} \frac{1}{a^2} & 0 & 0 & 0 \\ 0 & \frac{1}{b^2} & 0 & 0 \\ 0 & 0 & \frac{1}{c^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{3.4}$$

and it is full rank.

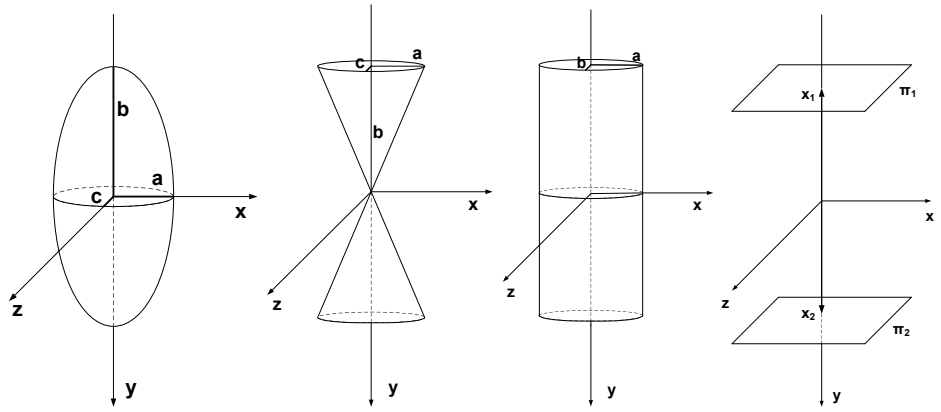


Figure 3.2: Quadrics of particular interest to modeling a human hand: ellipsoid, cone, elliptic cylinder, pair of planes

Cones are represented by matrices $\boldsymbol{Q}$ with rank($\boldsymbol{Q}$)=3. The equation of a cone aligned with the y-axis is

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} + \frac{z^2}{c^2} = 0 \tag{3.5}$$

and the corresponding matrix is

$$\boldsymbol{Q} = \begin{bmatrix} \frac{1}{a^2} & 0 & 0 & 0 \\ 0 & -\frac{1}{b^2} & 0 & 0 \\ 0 & 0 & \frac{1}{c^2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.6}$$

Elliptic Cylinders are represented by matrices $\boldsymbol{Q}$ with rank($\boldsymbol{Q}$)=3. The equation of an elliptic cylinder aligned with the y-axis is

$$\frac{x^2}{a^2} + \frac{z^2}{b^2} = 1 \tag{3.7}$$

and the corresponding matrix is

$$\boldsymbol{Q} = \begin{bmatrix} \frac{1}{a^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{b^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{3.8}$$

Pairs of planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ are represented by matrices $\boldsymbol{Q} = \boldsymbol{\pi}_1\boldsymbol{\pi}_2^T + \boldsymbol{\pi}_2\boldsymbol{\pi}_1^T$ with rank($\boldsymbol{Q}$)=2. The equation of a pair of planes, which are parallel to the xz-plane is

$$(y - y_1)(y - y_2) = 0 \tag{3.9}$$

where $\boldsymbol{\pi}_1 = [0, 1, 0, -y_1]^T$ and $\boldsymbol{\pi}_2 = [0, 1, 0, -y_2]^T$ and the corresponding matrix is

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{(y_1+y_2)}{2} \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{(y_1+y_2)}{2} & 0 & y_1 y_2 \end{bmatrix}. \tag{3.10}$$

Coordinate transformations of quadrics

Any point $\boldsymbol{X}$ that verifies equation (3.1), i.e. is situated on quadrics' surface can be transformed by pre-multiplying it by some homogeneous matrix $\boldsymbol{H}$. This matrix can represent transformations like rotation, translation, scaling and shear, and combinations of those. As long as this transformation can be inversed, we may rewrite equation (3.1) as

$$(\boldsymbol{H}^{-1}\boldsymbol{H}\boldsymbol{X})^T\boldsymbol{Q}\boldsymbol{H}^{-1}\boldsymbol{H}\boldsymbol{X} = 0 \tag{3.11}$$

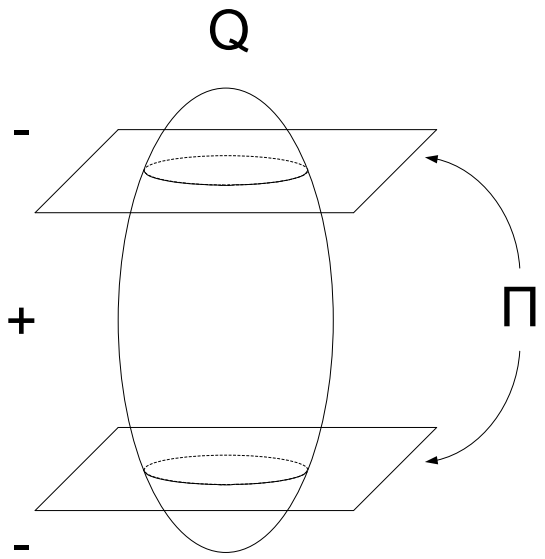


Figure 3.3: A truncated ellipsoid $\boldsymbol{Q}_\Pi$can be obtained by finding the points of the quadric $\boldsymbol{Q}$ that satisfy $\boldsymbol{X}^T\boldsymbol{\Pi}\boldsymbol{X} \geq 0$

Thus, replacing $\boldsymbol{X'} = \boldsymbol{H}\boldsymbol{X}$ we get

$$\boldsymbol{X'}^T\boldsymbol{H}^{-T}\boldsymbol{Q}\boldsymbol{H}^{-1}\boldsymbol{X'} = 0 \tag{3.12}$$

Then, it holds that

$$\boldsymbol{X'}^T\boldsymbol{Q'}\boldsymbol{X'} = 0 \tag{3.13}$$

where $\boldsymbol{Q'} = \boldsymbol{H}^{-T}\boldsymbol{Q}\boldsymbol{H}^{-1}$.

Equation 3.13 implies that any quadric matrix $\boldsymbol{Q'}$ can be factored into components representing its shape, given in normal implicit form and motion in the form a Euclidean transformation. Furthermore, given this result, it is possible to arrange a set of quadrics in any relative configuration by applying the required transformation to each of them. This is also required to perform the animation of a model composed of quadrics linked by articulations. For any model based on a kinematic chain (i.e. the human hand model), a transformation matrix needs to be computed for each parameter of the model, be combined with matrices resulting from the precedent parameters and then be applied to each subsequent part. The combined transformation matrix $\boldsymbol{H}_j$ to be applied to part j of the chain is given by

$$\boldsymbol{H}_j = \boldsymbol{H}_{j-1}\boldsymbol{T}_j \tag{3.14}$$

where $\boldsymbol{T}_j$ represents the transformation to be applied to part j relatively to the previous one. $\boldsymbol{H}_0 = \boldsymbol{T}_0$ represents the transformation encoding the rotation and translation of the base part of the kinematic chain relative to the world coordinate system.

Truncated quadrics

Truncated quadrics can be used as building blocks for modeling more complex shapes [RK94, SMC01, TM91]. For any quadric $\boldsymbol{Q}$ the truncated quadric $\boldsymbol{Q}_\pi$ can be obtained by finding the points $\boldsymbol{X}$ for which:

$$\boldsymbol{X}^T\boldsymbol{Q}\boldsymbol{X} = 0 \tag{3.15}$$

and

$$\boldsymbol{X}^T\boldsymbol{\Pi}\boldsymbol{X} \geq 0 \tag{3.16}$$

where $\boldsymbol{\Pi}$ is a matrix representing a pair of planes that delimit the quadric (see Figure 3.3).

Geometry of imaged quadrics

Consider an image of a quadric surface. Any homogeneous point on the surface, $\boldsymbol{X}$, projects into the image as $\boldsymbol{x} = \boldsymbol{P}\boldsymbol{X}$ where $\boldsymbol{P}$ represents the 3×4 projection matrix for the camera. This projection equation can be expanded, representing the projection matrix as $\boldsymbol{P} = [\boldsymbol{A} \quad \boldsymbol{a}]$ with $\boldsymbol{A}$ and $\boldsymbol{a}$ being 3×3 matrix and a 3-vector respectively, as:

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{a}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_3 \\ 1\end{bmatrix} = \lambda \boldsymbol{x}. \tag{3.17}$$

The scalar, λ, is an arbitrary scale factor. It follows that back-projecting a ray from the image point $\boldsymbol{x}$ gives

$$\boldsymbol{X}_3 = \lambda \boldsymbol{A}^{-1}\boldsymbol{x} - \boldsymbol{A}^{-1}\boldsymbol{a}. \tag{3.18}$$

Due to the needs of the following analysis, it will be useful to expand equation (3.1) as

$$\begin{bmatrix}\boldsymbol{X}_3^T & X\end{bmatrix}\begin{bmatrix}\boldsymbol{Q}_{33} & \boldsymbol{q} \\ \boldsymbol{q}^T & q\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_3 \\ X\end{bmatrix} = 0 \tag{3.19}$$

where $\boldsymbol{Q}_{33}$ is a symmetric 3×3 matrix, $\boldsymbol{q}$ a 3-vector and q is a scalar. Substituting equation (3.18) into the quadric equation (3.19) gives a quadratic equation in λ. To simplify the notation, the substitution $\boldsymbol{c} = -\boldsymbol{A}^{-1}\boldsymbol{a}$ is made ($\boldsymbol{c}$ representing the camera center):

$$(\boldsymbol{x}^T\boldsymbol{A}^{-T}\boldsymbol{Q}_{33}\boldsymbol{A}^{-1}\boldsymbol{x})\lambda^2 + 2(\boldsymbol{x}^T\boldsymbol{A}^{-T}\boldsymbol{Q}_{33}c + \boldsymbol{x}^T\boldsymbol{A}^{-T}\boldsymbol{q})\lambda + (\boldsymbol{c}^T\boldsymbol{Q}_{33}c + 2\boldsymbol{c}^T\boldsymbol{q} + q) = 0 \tag{3.20}$$

Hence there are zero, one or two real solutions for λ, which, substituted into equation (3.18) give $\boldsymbol{X}$ (the point(s) on $\boldsymbol{Q}$ projecting to $\boldsymbol{x}$). The different solutions for λ have the following geometric interpretations:

- If there are two real solutions for λ, there must be two corresponding points, $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, from equation (3.18). Hence the ray hits the quadric surface at two points and the image point $\boldsymbol{x}$ lies within the image of the quadric.
- If there are no real solutions for λ, the ray defined by equation (3.18) does not intersect with the quadric $\boldsymbol{Q}$ at any real point. Thus, the point $\boldsymbol{x}$ does not lie within the image of the quadric. As λ is complex, the point $\boldsymbol{X}$ will be correspondingly complex and have no geometric meaning.
- If there is just one real solution (two equal roots) to equation (3.20), there is a single point of contact between the ray and the quadric surface; The ray "touches" the surface, and the point $\boldsymbol{X}$ lies on the contour generator (see Figure 3.4). Hence, the point $\boldsymbol{x}$ lies on the apparent contour or silhouette of the surface in the image.

It should be noted that equation (3.20) makes no assumption about the rank of $\boldsymbol{Q}$, and therefore holds for all quadric surfaces. If the quadric passes through the camera center $\boldsymbol{c}$, the final term, $\boldsymbol{c}^T\boldsymbol{Q}_{33}\boldsymbol{c} + 2\boldsymbol{c}^T\boldsymbol{q} + q$, disappears and $\lambda = 0$ is a solution for all $\boldsymbol{x}$, as expected.

As stated earlier, a point on the apparent contour of $\boldsymbol{Q}$ is characterized by the fact that the back-projected ray touches the surface at a single point. In this equation (3.20) gives two coincident solutions for λ. It follows that the two solutions are given by

$$\lambda_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \tag{3.21}$$

with

$$a = \boldsymbol{x}^T\boldsymbol{A}^{-T}\boldsymbol{Q}_{33}\boldsymbol{A}^{-1}\boldsymbol{x} \tag{3.22}$$

$$b = 2(\boldsymbol{x}^T\boldsymbol{A}^{-T}\boldsymbol{Q}_{33}\boldsymbol{c} + \boldsymbol{x}^T\boldsymbol{A}^{-T}\boldsymbol{q}) \tag{3.23}$$

$$c = \boldsymbol{c}^T\boldsymbol{Q}_{33}\boldsymbol{c} + 2\boldsymbol{c}^T\boldsymbol{q} + q \tag{3.24}$$

The two solutions are coincident if, and only if, the discriminant, $b^2 - 4ac$, is zero:

$$\boldsymbol{x}^T(\boldsymbol{A}^{-T}[\boldsymbol{Q}_{33}\boldsymbol{c}\boldsymbol{c}^T\boldsymbol{Q}_{33} + 2\boldsymbol{q}\boldsymbol{c}^T\boldsymbol{Q}_{33} + \boldsymbol{q}\boldsymbol{q}^T - (\boldsymbol{c}^T\boldsymbol{Q}_{33}\boldsymbol{c} + 2\boldsymbol{c}^T\boldsymbol{q} + q)\boldsymbol{Q}_{33}]\boldsymbol{A}^{-1})\boldsymbol{x} = 0 \tag{3.25}$$

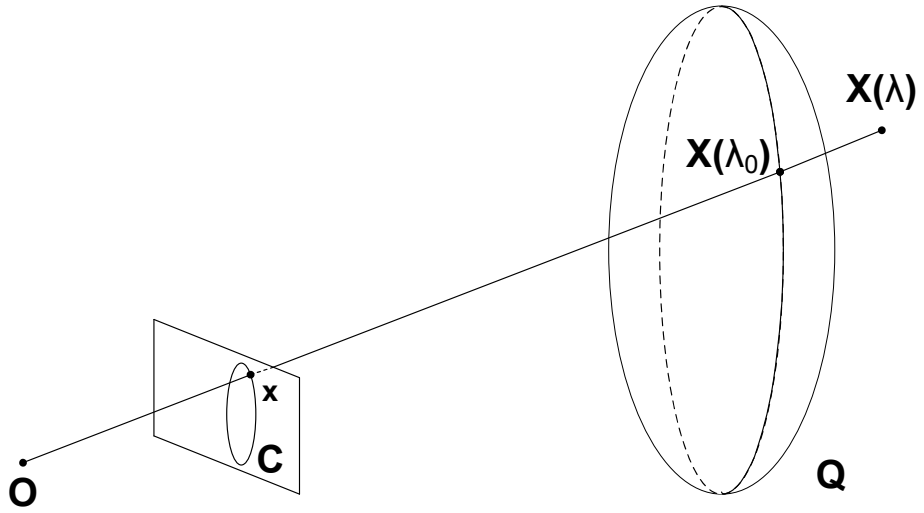


Figure 3.4: The projection of a quadric $\boldsymbol{Q}$ onto the image plane is a conic $\boldsymbol{C}$. The ray $\boldsymbol{X}(\lambda)$ defined by the camera center and the point $\boldsymbol{x}$ on the conic is parameterized by the depth λ and intersects the quadric at $\boldsymbol{X}(\lambda_0)$

It can be seen that equation (3.25) defines a conic in the image such that the image point $\boldsymbol{x}$ obeys the conic equation $\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} = 0$ where

$$\boldsymbol{C} = \boldsymbol{A}^{-T}[\boldsymbol{Q}_{33}\boldsymbol{c}\boldsymbol{c}^T\boldsymbol{Q}_{33} + 2\boldsymbol{q}\boldsymbol{c}^T\boldsymbol{Q}_{33} + \boldsymbol{q}\boldsymbol{q}^T - (\boldsymbol{c}^T\boldsymbol{Q}_{33}\boldsymbol{c} + 2\boldsymbol{c}^T\boldsymbol{q} + q)\boldsymbol{Q}_{33}]\boldsymbol{A}^{-1} \tag{3.26}$$

As an aside, it is interesting to note that this conic can be written as $\boldsymbol{C} = \boldsymbol{A}^{-T}\boldsymbol{C}'\boldsymbol{A}^{-1}$ where

$$\boldsymbol{C}' = \boldsymbol{Q}_{33}\boldsymbol{c}\boldsymbol{c}^T\boldsymbol{Q}_{33} + 2\boldsymbol{q}\boldsymbol{c}^T\boldsymbol{Q}_{33} + \boldsymbol{q}\boldsymbol{q}^T - (\boldsymbol{c}^T\boldsymbol{Q}_{33}\boldsymbol{c} + 2\boldsymbol{c}^T\boldsymbol{q} + q)\boldsymbol{Q}_{33}. \tag{3.27}$$

$\boldsymbol{C}'$ is also a conic which is only dependent on the camera center and $\boldsymbol{Q}$, whilst $\boldsymbol{C}$ is a 2-D projective transformation (homography) of $\boldsymbol{C}'$. This is an example of the more general rule that a change of the orientation and other parameters of the camera (here specified by $\boldsymbol{A}$) whilst maintaining the camera center fixed, induces a general homography in the image. It is often possible to choose one camera matrix as $\boldsymbol{P} = [\boldsymbol{I} \quad \boldsymbol{0}]$ so that the equation (3.26) reduces to

$$\boldsymbol{C} = \boldsymbol{q}\boldsymbol{q}^T - q\boldsymbol{Q}_{33}. \tag{3.28}$$

In this case, for the points $\boldsymbol{x}$ on the conic $\boldsymbol{C}$, the unique solution of equation (3.21) is

$$\lambda_0 = -\frac{q}{\boldsymbol{q}^T\boldsymbol{x}} \tag{3.29}$$

which is the depth of the point on the contour generator of the quadric $\boldsymbol{Q}$ (see Figure 3.4).

The normal vector $\boldsymbol{n}$ to the conic $\boldsymbol{C}$ at a point $\boldsymbol{x}$ can be computed in a straightforward way. The line $\boldsymbol{l}$ which is tangential to the conic at the point $\boldsymbol{x} \in \boldsymbol{C}$ is

$$\boldsymbol{l} = \boldsymbol{C}\boldsymbol{x}. \tag{3.30}$$

For a proof of the above one can refer to [HZ04]. For $\boldsymbol{l} = [l_1, l_2, l_3]^T$ its normal in Cartesian coordinates is given as $\boldsymbol{n} = [l'_1, l'_2]^T$ and is normalized to unit length.

3.1.1.3 Hand model construction

Structural model

Structurally, the hand model is composed of a set of quadrics $\{\boldsymbol{Q}_i\}_{i=1}^{N_Q}$ as proposed by [SMC01] representing the anatomy of a real human hand [DPD+07].

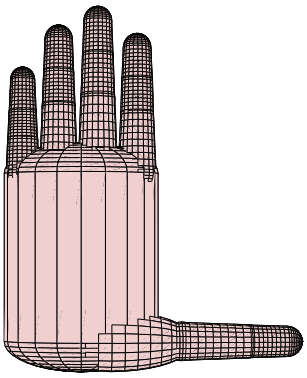


Figure 3.5: A 26 DOF 3D Hand Model composed by 35 quadrics

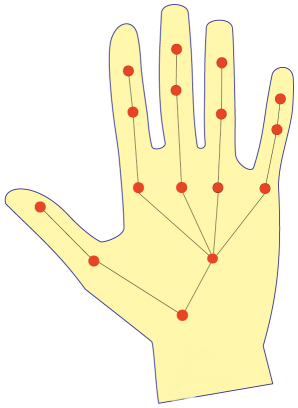


Figure 3.6: The kinematic model of a human hand. Nodes correspond to different hand parts and edges to part joints

The palm is modeled using a truncated elliptic cylinder, its top and bottom closed by half-ellipsoids. Each finger consists of three truncated cones, one for each phalanx. They are connected by truncated spheres, representing the joints. Spheres are also used for the tips of fingers and thumb, as well as for the connection of the fingers to the palm. The shape parameters of each quadric are set to be as close as possible to the anatomical concrete values (measured on an adult healthy hand) reported in [DPD+07]. The set of the employed quadrics is shown in Figure 3.5.

Kinematic model

The kinematic constraints between the different hand model components are well described by revolute joints [WH01]. Figure 3.6 shows a graph representing this kinematic structure, in which edges correspond to rigid bodies and edges to joints. The two joints connecting the phalanges of each finger and thumb have a single rotational degree of freedom, while the joints connecting the base of each finger to the palm have two degrees of freedom, representing grasping and spreading motions. Thus, twenty joint angles are required to describe the relative positions of all hand parts. The full configuration of the hand is described by these angles along with the palm's global position and orientation, giving a total of 26 degrees of freedom. Given the values for the aforementioned 26 angles, the derivation of the position and orientation of each rigid body forms a forward kinematics problem which is easily solved via a series of transformations using the Denavit-Hartenberg parameters (see [MTHC03] for more details).

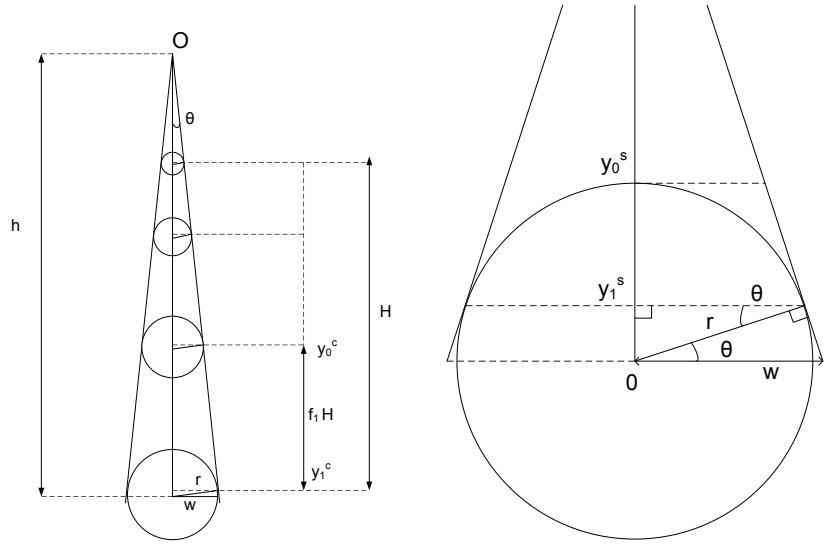


Figure 3.7: Constructing the model of a finger based on quadrics. Side-view of the finger (left) and close-up view of the bottom joint(right)

Constructing a finger - an example

Figure 3.7 illustrates how truncated quadrics can be joined together in order to design a finger of the hand. There may be several ways to represent the model shape. However, it is highly desirable to adopt a compact representation dependent on the smaller possible number of values. These values are then used to set the parameters of the quadrics. Each finger is built up based on cone segments and truncated spheres. First, the parameters of the cone are determined, given the height of the cone segment h and the radius r of the bottom joint (see figure 3.7). Let θ be the angle between the central axis of the cone and a line on the cone passing through the origin. First, $tan\theta$ is derives as

$$sin\theta = \frac{r}{h} \Rightarrow cos\theta = \sqrt{1 - \frac{r^2}{h^2}} = \sqrt{h^2 - r^2} \Rightarrow tan\theta = \frac{r}{h}\frac{h}{\sqrt{h^2 - r^2}} = \frac{r}{\sqrt{h^2 - r^2}}, \tag{3.31}$$

which is then used to obtain the width of the cone:

$$tan\theta = \frac{h}{w} \Rightarrow w = \frac{hr}{\sqrt{h^2 - r^2}}. \tag{3.32}$$

The cone segment is clipped at $y_0 = h_0 - f_1 H$ and $y_0 = h$, where H is the distance between the centers of the spheres representing the bottom joint and the center of the hemisphere on the tip of the finger, and f_1 is the ratio between the bottom finger segment to the complete length H of the finger.

The spheres are clipped so that they form a continuous shape with the conic segments.

$$sin\theta = \frac{y_1}{r} \Rightarrow y_1 = rsin\theta = \frac{r^2}{h} \tag{3.33}$$

The parameters for the other cones and spheres representing the finger can be derived in a similar way.

3.1.1.4 Contour drawing

The idea when plotting a general conic is to factorize the conic matrix $\boldsymbol{C}$ in order to obtain its shape in terms of a diagonal matrix $\boldsymbol{D}$ and a transformation matrix $\boldsymbol{V}$ such that $\boldsymbol{C} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$.

Starting from the conic equation $\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} = 0$ and by rewriting it as

$$\begin{bmatrix} x & y & w \end{bmatrix} \begin{bmatrix} a & b & d \\ b & c & f \\ d & f & g \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0 \tag{3.34}$$

$$ax^2 + 2bxy + cy^2 + 2dxw + 2fyw + gw^2 = 0 \tag{3.35}$$

and by making $w = 1$, this expression becomes the well known general expression of a planar quadratic curve

$$ax^2 + 2bxy + cy^2 + 2dx + 2fy + g = 0 \tag{3.36}$$

It can easily be verified that this equation contains terms that are not present when the conic is centered and aligned with the frame axes. So, performing the required alignment consists primarily in determining and applying the transformation that forces these terms to vanish.

The xy term can be eliminated by a suitable rotation. For a rotation of an arbitrary angle θ, the rotation is performed by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{3.37}$$

Substituting x and y into equation (3.36) and grouping items, it turns that

$$a'x'^2 + 2b'x'y' + c'y'^2 + 2d'x' + 2f'y' + g' = 0 \tag{3.38}$$

where the coefficients are

$$a' = acos^2\theta - 2bcos\theta sin\theta + csin^2\theta \tag{3.39}$$

$$b' = b(cos^2\theta - sin^2\theta) + (a - c)sin\theta cos\theta \tag{3.40}$$

$$c' = asin^2\theta + 2bsin\theta\ cos\theta + ccos^2\theta \tag{3.41}$$

$$d' = dcos\theta - fsin\theta \tag{3.42}$$

$$f' = dsin\theta + fcos\theta \tag{3.43}$$

$$g' = g \tag{3.44}$$

The term $2b'x'y'$ may vanish by forcing $b' = 0$. This implies that

$$\theta = \frac{1}{2}cot^{-1}(\frac{c - a}{2b}) \tag{3.45}$$

and the equation (3.38) becomes

$$a'x'^2 + c'y'^2 + 2d'x' + 2f'y' + g' = 0 \tag{3.46}$$

Now, it is possible to translate the conic so it becomes centered on the origin by noting that the above expression can be transformed as

$$a'x''^2 + c'y''^2 = g'' \tag{3.47}$$

where

$$x'' = x' + \frac{d'}{a'} \tag{3.48}$$

$$y'' = y' + \frac{f'}{c'} \tag{3.49}$$

$$g'' = -g' - \frac{d'^2}{a'} - \frac{f'^2}{c'} \tag{3.50}$$

So the aligned conic in matrix form is

$$\boldsymbol{D} = \begin{bmatrix} a' & 0 & 0 \\ 0 & c' & 0 \\ 0 & 0 & g'' \end{bmatrix} \tag{3.51}$$

and the transformation that makes this conic gain its original configuration is

$$\boldsymbol{V} = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\frac{d'}{a'} \\ 0 & 1 & -\frac{f'}{c'} \\ 0 & 0 & 1 \end{bmatrix} \tag{3.52}$$

We have now both the aligned conic and the transformation that can be used to produce the original one. Similarly, such transformation can be applied to any point belonging to the aligned conic bringing it to the corresponding position on the projected conic.

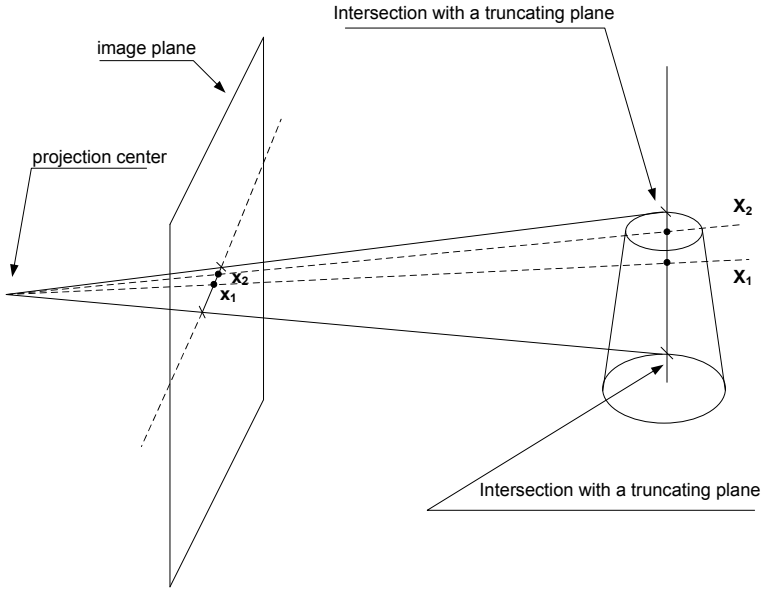


Figure 3.8: Clipping of the projected conic segments

Drawing ellipsoids

We can draw $\boldsymbol{C}$ by drawing $\boldsymbol{D}$ and transforming these points according to $\boldsymbol{V}$. From the ellipse equation $\hat{\boldsymbol{x}}^T \boldsymbol{D} \hat{\boldsymbol{x}} = 0$ the radii a and b can be found as

$$a = \sqrt{-\frac{d_{22}}{d_{00}}} \qquad b = \sqrt{-\frac{d_{22}}{d_{11}}} \tag{3.53}$$

where $\hat{\boldsymbol{x}} = [x'', y'', 1]^T$.

In order to find the ellipse points that are between the clipping planes it is more efficient to find the end points of the arches to be drawn. For this the contour generator is intersected with the clipping planes and these points are projected to the image. The contour generator is parameterized as

$$\boldsymbol{X}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \zeta(t) \end{bmatrix} \tag{3.54}$$

where $\zeta(t)$ is the inverse depth obtained from equation (3.29). One of the clipping planes is given by

$$\boldsymbol{\pi} = \begin{bmatrix} \boldsymbol{n}_\pi \\ -d_\pi \end{bmatrix} \tag{3.55}$$

The points of intersection of the contour generator and this clipping plane are given by the solution of the equation

$$\boldsymbol{X}^T(t) \quad \boldsymbol{\pi} = 0 \qquad \Leftrightarrow \qquad ak_1 cost + bk_2 sint + k_3 = 0 \tag{3.56}$$

where the vector $\boldsymbol{k}$ is defined as

$$\boldsymbol{k} = \boldsymbol{n}_\pi + \frac{\boldsymbol{q} d_\pi}{q} \tag{3.57}$$

The solution to equation (3.56) can be derived by using the sine addition formula.

Drawing cones and cylinders

Being the case where the projection reduces to a single point, rare, uninteresting, and of trivial solution, we will devote our attention to the other ones. Having obtained the expression of the centred and aligned conic, which is known to be degenerate and corresponding to two lines for the projection of cylinders or cones, two points will be easily chosen on each of these lines.

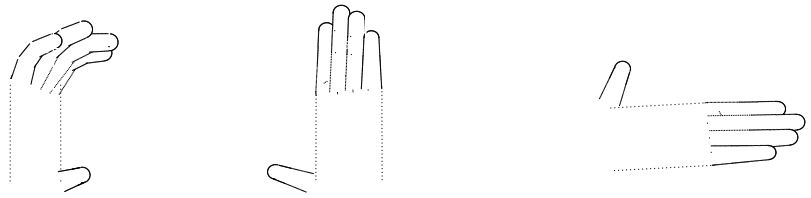


Figure 3.9: Examples of handling self-occlusions of a 3D hand model

For the parallel line cases, we known that all the points have coordinates either $y'' = \pm\sqrt{g''}$ for any x'', and $x'' = \pm\sqrt{g''}$ for any y'', for the horizontal and vertical cases respectively. So after knowing the value of the constrained coordinate, it suffices to choose any convenient value to the other.

For the concurrent lines cases, we have $x'' = \pm\sqrt{b'}y''$, so by replacing y'' with any two values will give the corresponding x'' values. After defining the lines that correspond to the projection of the degenerate quadrics, one has to find the clipping points that will become the extremities of the line segments.

Starting with two points $\{\boldsymbol{x}_i\}, i = 1, 2$ on each of the projected lines, the corresponding 3D counterparts $\boldsymbol{X}_i$ can be found as in equation (3.54).

The frontier of visibility of the quadric is the set of the two lines that project onto the silhouette contour. One of them is the line that passes through points $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. As a consequence, any other point situated on the line that they define can be written as

$$\boldsymbol{X}_n = \boldsymbol{X}_1 + \lambda \boldsymbol{X}_2 \tag{3.58}$$

Replacing the expression of the line's generic point on the truncating planes equation, results in

$$(\boldsymbol{X}_1 + \lambda \boldsymbol{X}_2)^T \boldsymbol{\Pi} (\boldsymbol{X}_1 + \lambda \boldsymbol{X}_2) = 0 \tag{3.59}$$

By rearranging and grouping the terms in λ it turns out that

$$\lambda^2 \boldsymbol{X}_2^T \boldsymbol{\Pi} \boldsymbol{X}_2 + 2\lambda \boldsymbol{X}_1^T \boldsymbol{\Pi} \boldsymbol{X}_2 + \boldsymbol{X}_1^T \boldsymbol{\Pi} \boldsymbol{X}_1 = 0 \tag{3.60}$$

As both points $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are known, the above expression is a second degree equation in λ, which can be solved giving two λ values that, once replaced in (3.58), will give the required 3D intersection points. The same procedure is repeated for the second line, and the projection of these points gives the end points of each of the line segments that correspond to the projection of the truncated cones.

3.1.1.5 Handling occlusions among hand model parts

The next step is the handling of self-occlusions, achieved by comparing the depths of points in 3D space. The depth of a point $\boldsymbol{X}(\lambda_0)$ on the contour generator of a quadric $\boldsymbol{Q}$ can be found as described in equation (3.29). In order to check if this point is visible, the equation of each of the quadrics is solved. In the general case there are two solutions λ_1^i and λ_2^i, yielding the points where the ray intersects with the quadric $\boldsymbol{Q}_i$. The point $\boldsymbol{X}(\lambda_0)$ is visible if it is closer to the camera than all the other points.

In order to speed up the computation, it is first checked whether the 2D bounding boxes of the corresponding conic segments overlap in the image. If this is not the case, the calculation of the ray and conic intersection can be omitted. Figure 3.9 shows an example of the projection of the hand model after removing the occluded segments.

3.1.1.6 Implementation

Following the previously presented analysis, an implementation was build which is able to manage models composed of truncated quadrics like cylinders, cones, and ellipsoids. These geometric primitives can be connected to other ones by rigid links or by the way of articulations.

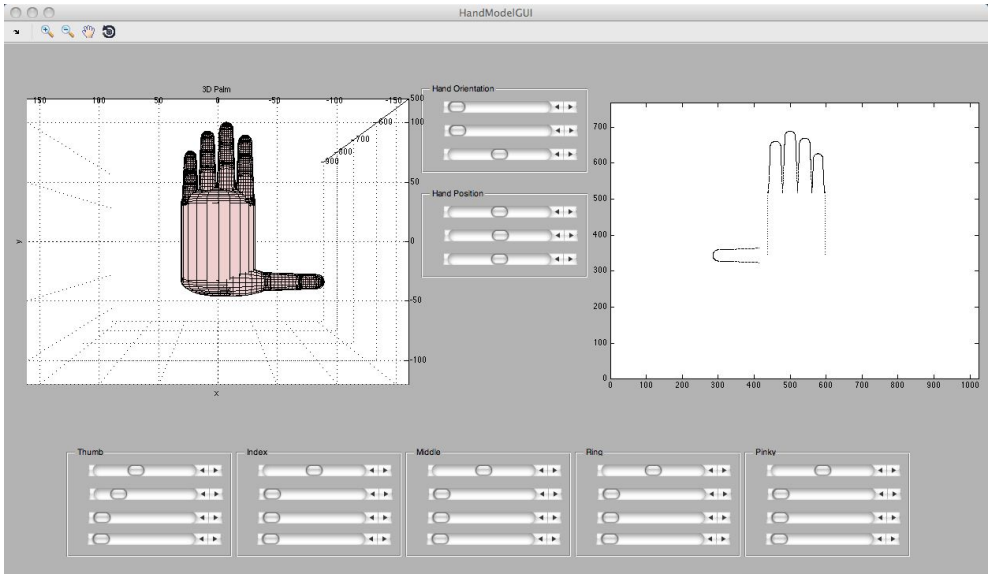


Figure 3.10: A screenshot of the developed Matlab GUI that enables the user to modify the 26 DOFs of the human hand model

The software implementation includes a Matlab GUI (see Figure 3.10), which enables the user to modify all the degrees of freedom of the 3D hand model. More specifically, the GUI presents two figures, one of the 3D hand model and another one of the projection of this model on a user selected type of camera. Furthermore, a set of sliders give to the user the ability to modify all the 26 degrees of freedom of the hand model. The user can also change the parameters of the camera (intrinsics and extrinsics) and the parameters of the quadrics representing the human hand.

3.1.2 Hand observation model

A tracker can be described as a process that follows some observable manifestation of the state of a system, commonly called measure, and infer that state from the observed manifestations. In many cases the state is composed of some set of physical quantities which are not always directly observable, requiring therefore an indirect estimation process.

There are often cases, where the observed measures are the result of some combination of the state components, possibly including nonlinear transformations. In the presence of nonlinearities at the state-measurement link, the state estimation is frequently performed by an error minimization procedure using gradient descent based methods. An alternative approach is the maximization of a likelihood function that encodes the probability of the current measurement to be a manifestation of the hypothetical state $\boldsymbol{x}$.

Furthermore, the tracking system must be robust enough to be able to infer the true state of the system using measurements that are normally corrupted by noise produced by various sources. Another aspect that must be taken into consideration, is that, for most cases, the system's state can vary dynamically and the tracking process must be able to converge to the true state values even if these are continuously changing.

The success of a tracker depends on the use of an appropriate model for the system dynamics which will guide correctly the predictions of the system state, on the ability of the tracker to explore the state space avoiding local minima and on the shape of the error or likelihood function. In fact, even if a tracker shows some weakness in one of these properties it can still exhibit a good behavior if there is a compensation effect in the other ones. In the optimal case, this function presents a single minimum on the error function and the local derivatives of this function exist for every point of the state space. Unfortunately, in many cases these conditions are not verified and multiple local maxima or minima exist which can attract and trap the tracker at locations of the state space which are far from the true state.

The interpretation of any 3D scene from 2D images obtained by perspective projection, has the intricate

ambiguities that result from losing the depth information. In other words, such ambiguities result from the fact that every 3D point located on a projective ray share the same image point. Although stereo systems are normally used to try to remove these ambiguities, their use is limited to textured zones of the scenes, where the usual Lambertian assumption can be used to establish the correspondences between pixels of the two images.

In the present work, a stereo camera pair was used to track a human hand in 3D. Without the use of the 3D reconstruction, a model is then mandatory to establish a relationship between the observed image features and the model configuration. This also requires the use of a robust approximation of the measure-state likelihood function, to guide the estimation process containing the least ambiguities possible. An approximation of this likelihood function can be made by building a cost function, which combines information from different sources. This mixture of measures aims to shape the cost surface so that it can minimize the effects of false local extrema which exist in each of the individual measures, in a way that each additional measure intends to remove the false attractors still present on the combination of the preceding ones.

The construction of such a cost function will be used in an Unscented Kalman Filter (UKF) framework and in a particle filter framework. At present, only the UKF framework has been implemented. However, as it can be seen in the future work session a particle filter framework is under development and this is the reason that we also present a probabilistic view of the observation model. Given the high dimensionality of the problem, the ill-conditioning that results from the ambiguities in the pose-appearance link and the unavoidable visual clutter that appears in a real-world application, we pay special attention to the construction of a robust likelihood function in a way to minimize the effects of these undesirable properties. This function employs a combination of parameters obtained from various cues extracted from the images of the input sequences, from values derived from the structure properties, and from the *a priori* knowledge about the behavior of some parameters.

3.1.2.1 Edge cues

Image edges provide a strong cue for the human visual system which is often sufficient for scene interpretation [Mar82, KSJ91]. Initially, a simple observation model to be used in the UKF framework will be presented. Extensions to this model and a probabilistic approach of the observation model will be also presented.

3.1.2.2 Simple observation model for the UKF tracker

The observation vector $\boldsymbol{Z}$ is obtained by detecting edges in the neighborhood of the projected hand model. Let $S_v^i = \{\boldsymbol{X}_i^{v,1}, \boldsymbol{X}_i^{v,2}, ..., \boldsymbol{X}_i^{v,N_i}\}$ be the set of visible (not occluded) points on the contour generator of $\boldsymbol{Q}_i$ as seen from camera $\boldsymbol{P}_v$, and let $\boldsymbol{C}_i^v$ be the projection of quadric $\boldsymbol{Q}_i$ on $\boldsymbol{P}_v$. The image of each point $\boldsymbol{X}^{v,j}$ is denoted by $\boldsymbol{x}_i^{v,j}$. The vector $\boldsymbol{n}_i^{v,j}$ normal to $\boldsymbol{C}_i^v$ at $\boldsymbol{x}_i^{v,j}$ can be obtained easily as described in a previous section. For each point $\boldsymbol{x}_i^{v,j}$ one looks for edges along the normal $\boldsymbol{n}_i^{v,j}$ as proposed in [SMC01, BI98]. The intensity values in the images are convolved with a derivative of Gaussian kernel and an edge in assigned to the position $\boldsymbol{e}_i^{v,j}$ with the largest absolute value. The observation vector $\boldsymbol{Z}$ is constructed by stacking the inner products $\boldsymbol{v}_i^{v,jT}\boldsymbol{e}_i^{v,j}$ to form a single vector. Each component of the innovation vector will then have the form $\boldsymbol{n}^T(\boldsymbol{e} - \tilde{\boldsymbol{x}})$, where $\tilde{\boldsymbol{x}}$ is the average position of the contour points in a single image, $\boldsymbol{e}$ is the corresponding edge in that image and $\boldsymbol{n}$ is the corresponding normal vector of the reference contour. Thus, a component of the innovation vector is the projection of the distance between the average contour point and the corresponding edge onto the direction of the normal. Therefore, the innovation vector can be interpreted as the error in pixels between the projection of the hand model on each image and the edges in that image.

An implementation issue of this approach is to find a way to keep the size of the observation vector fixed for all the sigma points of the UKF tracker. This is a non-trivial problem in general, because the number of occluded points varies and so does the dimension of the observation vector. The approach adopted is to keep the distance of the contour points per quadric fixed, having a greater point density in the fingertips to get a more precise estimate of the hand configuration.

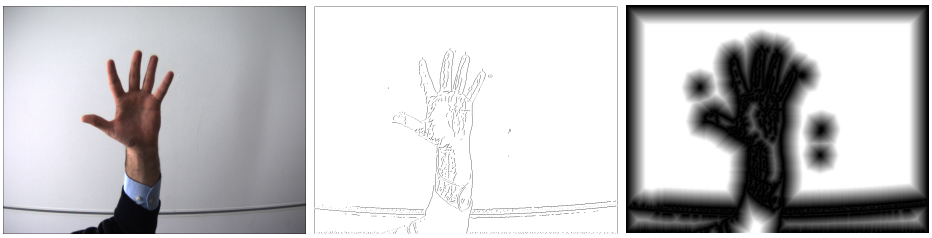


Figure 3.11: Computation of the distance transform: original image (left), edge map (middle) distance transformed edge map (right)

3.1.2.3 Approximating distances via shape matching

Performance issues may require that the overall process of computing the set of distances between the sample points of a hand model projection and the image edges is as fast as possible. In a particle filter context (UKF included), this measure needs to be computed for the contours generated by every particle. Consequently, depending on the number of particles involved, it can generate an important computational effort. Approaches, like the one in the previous section, use the derivative along the edge normals to look for image edges resulting in a more efficient algorithm than computing edges all over the image. However, these kind of approaches present a larger sensitivity to image noise, as there is no spatial relationship between one edge pixel and its neighbors. One can recall that optimal edge detectors like the one proposed by Canny [Can86] use neighboring information to extract edges.

For the cases that there is a large number of distances to be computed between image locations and the closest image edges, there is normally a computational advantage of using the so called Distance Transfrom (DT). These transforms generate a new image where each pixel contains an approximation of the distance to the closest edge of the input image. By using this kind of approximations, the operation of computing the distance between a point and an edge, may be reduced to a simple peeking of the corresponding DT pixel [GGS04]. Thus, although its initial computation represents an additional computing effort, the subsequent distance measurements are significantly alleviated, resulting in a drastic reduction of the required computing power for cases where a large number of distance measurements is performed. The distance transform, also called Chamfer distance transform, tries to approximate the distance between a pixel and the nearby edges by assuming that its value can be deduced from the distances of its neighbors.

To formalize the idea of Chamfer matching, the shape of an object is represented by a set of points $\boldsymbol{A} = \{\boldsymbol{a}\}_{i=1}^{N_a}$. In our case, this is a set of points on the projected model contour. The image edge map is represented as a set of feature points $\boldsymbol{B} = \{\boldsymbol{b}_i\}_{i=1}^{N_b}$. In order to be tolerant to small variations, any similarity function between two shapes should vary smoothly when the point locations change by small amounts. This required is fulfilled by Chamfer distance functions, which are based on a *distance transform* (DT) of the edge map. An extended survey on distance transforms can be found in [FBTdFC07]. The distance transform takes a binary feature image as input and assigns each location the distance to its nearest feature, see Fig 3.11. If $\boldsymbol{B}$ is the set of edge points in the image, the DT value at location u contains the value $\min_{b\in\boldsymbol{B}} \|u-b\|$. A number of cost functions can be defined using the distance transform [Bro88]. A common choice is the normalized squared distance between each point of $\boldsymbol{A}$ and its closest point in $\boldsymbol{B}$:

$$d_{cham}(\boldsymbol{A},\boldsymbol{B}) = \frac{1}{N_a}\sum_{a\in A}\min_{b\in B}\|a-b\|^2 \tag{3.61}$$

The distance between a template $\boldsymbol{A}$ and an edge map $\boldsymbol{B}$ can then be computed by adding the squared DT values at the template point coordinates and dividing by the number of points in N_a. The DT can be computed efficiently using a two-pass algorithm over the image as proposed in [Bro88], making the operation linear to the number of pixels. Furthermore, the DT should be computed only once for each input image and it can then be used to match more than one templates. This characteristic implies large computational savings of this method, compared to others that explicitly search for feature correspondence between template and image features. From another perspective, DT can be viewed as a smoothing operation in the feature space. In absence of any smoothing operation, the correlation of a template $\boldsymbol{A}$ with an edge map $\boldsymbol{B}$ leads to a sharply peaked cost function, which is not robust towards small perturbations of the contour. On the other hand, the value of the function $d_{cham}(\boldsymbol{A},\boldsymbol{B})$ in equation (3.61)

varies smoothly when the template $\boldsymbol{A}$ is translated away from the global minimum, therefore smoothing the cost surface. The cost function can be made even more robust to noise by applying an upper bound threshold on the DT image. This threshold reduces the effect of missing edges and is therefore more tolerant towards partial occlusions. The matching cost takes the following form in this case:

$$d_{cham}(\boldsymbol{A}, \boldsymbol{B}, \tau) = \frac{1}{N_a} \sum_{a \in A} \min \left(\min_{b \in B} \|a - b\|^2, \tau \right) \tag{3.62}$$

where τ is the upper bound threshold value. We should note that the Chamfer cost function is not a metric, as it is not a symmetric function. However it can be made symmetric by defining the distance as the sum $d_{cham}(\boldsymbol{A}, \boldsymbol{B}) + d_{cham}(\boldsymbol{B}, \boldsymbol{A})$, but in the particular application the term $d_{cham}(\boldsymbol{B}, \boldsymbol{A})$ indicates how well the edges in the background match to the model and the term is not computed as very often the background is cluttered.

In cases where we want to match templates with images that have many background edges, the DT image contains low values on average and therefore a single DT image is not sufficient to discriminate between different templates. To remedy this problem, one can make use of the gradient direction as well. The idea is to use a distance measure, which not only considers distance in translation space, but also in the gradient orientation space. This approach increases the discriminative power of the matching function, as has been demonstrated by Olson and Huttenlocher for the Hausdorff distance [Ruc96].

As it has been already mentioned, Chamfer matching avoids the explicit computation of point correspondences. However, the correspondences can easily be obtained during the DT image computation by storing for each value in the DT image the corresponding location. These can be used to incorporate higher order constraints within the cost function, such as continuity and curvature between corresponding points. Additionally, the Chamfer function is not invariant towards transformation of the template points, such as translation, rotation and scaling. Thus, each of thse cases needs to be handled by searching over the parameter space. In order to much a large number of templates efficiently, hierrarchical search methods have been in suggested in [Bro88, Gav00].

3.1.2.4 Skin color cues

Colors, textures and patterns are also important characteristics of objects. Objects have, frequently, parts with very distinctive colors or color patterns, which apart from the decorative role may play an important role for a visual process. In nature, colors also play important role making the identification of group members a simple task, as in almost every species of animals or plants their members share the same colors. This is also the case for humans where frequently the head, the hair or the hands have colors, or color patterns, that differ markedly from those of the work clothes. Consequently, for a body part, the local color or the local color distribution, can be an important and discriminative source of information that can be used in a tracking context. If the target we intend to track presents some distinctive color characteristics, then it may be possible to identify it or parts of it, by simple application of color segmentation techniques seperating the regions formed by pixels which present some characteristics from the other ones.

Related Work

Skin color in images has been exploited by systems for tracking or detecting hands or faces [Bir98, IB98, LF02, Shi01]. Jones and Rehg [JR02] carried out a thorough study of skin color models. A data set of 13.640 images was used to obtain distributions of skin color and non-skin color values. A Bayesian classifier, treating each pixel independently, was defined using different representations of the distribution. It was shown that an RGB-histogram representation, quantized to 32^2 bins leads to better generalization performance, compared to other quantizations. Furthermore, it was shown that this scheme outperforms a classifier based on a Gaussian mixture model.

Color spaces

Many authors have chosen to work in color spaces where the brightness value is seperated from the hue, such as the HSV and YUV color spaces [Bir98, TvdM01]. The intensity value is then discarded

or weighted less relative to the hue. Another option is to adopt the intensity normalized (r, g)-space, where $r = \frac{R}{R+G+B}$ and $g = \frac{G}{R+G+B}$. Yang et al. [YLW97] suggest modeling the distributions as a Gaussian in (r, g)-space, being able to easy update when necessary. Adapting color models to changes in illumination can yield significant improvements over static models, particularly over very long image sequences [B.02, SS00].

3.1.2.5 Segmentation of skin colored regions

In the present work a Gaussian distribution in normalized (r, g)-space is used. The normalized (r, g)-space introduces invariance to brightness changes, however, it does not compensate for changes in illumination color. The distribution parameters were estimated from skin colors in the first frame of a sequence, obtained by manual segmentation and no adaptation has been performed.

A cost function based on color values can be constructed in a number of ways. One approach is to color segment the image and subsequently match a silhouette template to this binary feature map. The alternative approach would be a probabilistic approach by deriving a likelihood function $p(\boldsymbol{z}^{col}|\boldsymbol{x})$ based on the image observation $\boldsymbol{z}^{col}$, the color vectors in (r, g)-space in the whole image. Given a state vector $\boldsymbol{x}$, corresponding to a particular hand pose, the pixels in the image are partitioned into a set of locations within the hand silhouette $\{k : k \in S(\boldsymbol{x})\}$ and outside the hand silhouette $\{k : k \in \bar{S}(\boldsymbol{x})\}$. Assuming pixel-wise independence the likelihood function for the whole image can be factored as follows:

$$p(\boldsymbol{z}^{col}|\boldsymbol{x}) = \prod_{k \in S(\boldsymbol{x})} \frac{p^s(I(k))}{p^{bg}(I(k))} \prod_{k \in S(\boldsymbol{x}) \cup \bar{S}(\boldsymbol{x})} p^{bg}(I(k)) \tag{3.63}$$

where $I(k)$ is the color vector at location k in the image, and p^s and p^{bg} are the skin color and background color distributions, respectively. When taking the log-likelihood this becomes a sum:

$$logp(\boldsymbol{z}^{col}|\boldsymbol{x}) = \sum_{k \in S(\boldsymbol{x})} (logp^s(I(k)) - logp^{bg}(I(k))) + \sum_{k \in S(\boldsymbol{x}) \cup \bar{S}(\boldsymbol{x})} logp^{bg}(I(k)) \tag{3.64}$$

The independence assumption in equation (3.63) does not hold in general, because the color values of neighboring pixels are not independent. Jedynak et al. [JZD03] suggest using a first order model by considering a pixel neighborhood. A different way to remedy the problem is to apply filters with local support at points on a regular grid on the image. The filter outputs are then approximately independent [IM01, SBR00]. However the pixel-wise independence assumption gives reasonable approximation and leads to a form of the likelihood function that is efficient to evaluate.

3.1.2.6 Fusion of edge and color cues

In the previous sections similarity measures based on edge and color cues have been described. When combining the two features, they can be stacked into a single 2D observation vector $\boldsymbol{z} = (\boldsymbol{z}^{edge}, \boldsymbol{z}^{col})^T$. This observation vector can be used in the update step of the tracker module.

3.1.3 3D hand tracking

3.1.3.1 Recursive Bayesian estimation

Tracking a 3D hand model amounts to the estimation of the state of a system that changes over time, using a sequence of noisy measurements made on some variables that are related to this system. Given the characteristics of the problem to be solved, a state-space model for the dynamic system written using a discrete-time formulation seems the most appropriate choice. Thus, the evolution of the system is modeled using difference equations, and the measurements are assumed to be available at discrete times. Particular focus is given on the state vector which should contain all the relevant information to describe the system. In a tracking problem, this information is normally related to the kinematic characteristics of the target, whereas the measurement vector represents noisy observations that are somehow related to the state vector.

Two models are required to analyze and infer the state of dynamic system. The first one, known as the **system model** describes the evolution of the state with time and the second one relates to the noisy measurements of the current state, known as the **measurement model**. We assume that both of these models are available in a probabilistic form.

The probabilistic state-space formulation and the requirement for updating the state of the system upon the reception of each new measurement are well suited to a Bayesian approach. In such an approach, an attempt is made to construct the posterior probability density function (pdf) of the state given all the available information, which includes the set of the received measurements. In case that an estimate must be obtained whenever a measurement is received, a recursive filter is an adequate solution. The filtering procedure is normally divided in two steps: **prediction** and **update**. In the prediction step, the system model is applied to predict the state pdf at the next time instant, given the previous one. Due to the presence of noise (which models the unknown disturbances that affect the system) the predicted pdf is generally translated, deformed and spread. The reception of a new measurement permits the adjustment of this pdf during the update operation. This is performed, using the Bayes theorem as the mechanism to update the knowledge about the system state upon the reception of new information.

3.1.3.2 Bayesian tracking

The term **tracking** is commonly associated with a process of following a signal or some object that moves in a space. However, it is no more than the estimation of a set of variables that can be used to describe the behavior of a process accordingly to some model of the same process. The use of an estimator is justified either by the fact that some of the variables appearing in the model cannot be measured directly, or because some of the variables form an abstraction to the approximation of the process and thus they do not exist in the physical sense. It is considered that

$$\boldsymbol{x}_k = f_k(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \tag{3.65}$$

represents the state evolution from time $k-1$ to time k, where $f_k()$ is a (possibly non-linear) function that may evolve over time, $\boldsymbol{x}_k$ represents the state at time k and $\{\boldsymbol{u}_{k-1}\}, k \in \mathbb{N}$ is an independent and identically-distributed (i.i.d.) process noise sequence. The goal is to estimate the current state $\boldsymbol{x}_k$ using the previous state information and all the input data received so far. Due to the fact that the state is not directly accessible, the data, which can be used to infer it, is normally composed of measurements of some physical manifestation of this state. Thus, a model that relates the output of this system and its internal state is required. The available input data are then related to the state by

$$\boldsymbol{z}_k = h_k(\boldsymbol{x}_k, \boldsymbol{n}_k) \tag{3.66}$$

where $h_k()$ is a function (possibly non-linear) and $\boldsymbol{n}_k$ represents the measurement noise.

A tracking process can be also seen from a probabilistic point of view. More specifically, the state can be represented as a random vector, modeled by a probability density function. In this case, the whole process is modeled through the use of pdf's and their evolution in time. The process model is replaced by the conditional distribution $p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{z}_1, ..., \boldsymbol{z}_{k-1})$ and the measurement model by $p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{z}_1, ..., \boldsymbol{z}_{k-1})$. Denoting $\boldsymbol{z}_{1:k} = [\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_k]$ for the prediction step and starting from the prior $p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})$, we can use the Chapman-Kolmogorov equation to perform the prediction as

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1}) d\boldsymbol{x}_{k-1} \tag{3.67}$$

where $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{z}_{1:k-1}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ represents a first order Markov process.

The state evolution described by the probabilistic model $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ is defined by the system model and the noise $\boldsymbol{u}_{k-1}$ statistics. The update step is performed when the input measurements $\boldsymbol{z}_k$ become available at time k and for this we can use the Bayes rule as

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})}{p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} \tag{3.68}$$

where

$$p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1}) = \int p(\boldsymbol{z}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) d\boldsymbol{x}_k \tag{3.69}$$

depends on the likelihood function $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ and the statistics of $\boldsymbol{n}_k$. The measurement $\boldsymbol{z}_k$ is used on the update stage to compute the likelihood value of the previous state estimate to modify the prior density and obtain current state posterior density.

Unfortunately, these equations cannot be solved in closed form for the general case. However, for the restricted case of linear models and Gaussian noise an optimal solution can be obtained using the Kalman Filter [Kal60]. For the nonlinear cases, the Extended Kalman Filter addresses the nonlinearity by linearizing through a Taylor expansion and using the first term. This method uses the assumption that the state is a Gaussian random variable which is propagated through the first order linear approximation of the nonlinear system. This can introduce large errors in the true posterior mean and variance that may lead to suboptimal performance or even divergence of the filter [WM00].

The Unscented Kalman filter proposed by Julier et al. [JU97] is another extension that addresses this problem by using a deterministic sampling approach. The state distribution is still modeled by a Gaussian distribution but this time represented by a covariance matrix propagated through sigma points. These points are propagated through the true nonlinear system and capture the posterior mean accurately to the third order (Taylor series expansion) for any nonlinearity. This filter still has the limitation that it requires the estimated quantities to be well described by Gaussian distributions. Finally, the Particle filter based algorithms, use a set of samples to approximate the distribution but don't impose the Gaussian restriction any longer.

3.1.3.3 Kalman filter

In case that the involved densities are Gaussian, they can be fully described by using solely the respective means and covariances. As a consequence, the recursive estimation process can be constructed by predicting the evolution of the mean and covariance of the system state and then correcting them by integrating the measurement information. The Markovian dynamic systems for which the transition and observation probability distributions are Gaussian, are linear systems with additive white Gaussian noise. The system equation is

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{u}_{k-1}, \tag{3.70}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ is the state vector at time instant k. $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the state transition matrix, describing the system dynamics. The linear observation equation is given by

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{w}_k, \tag{3.71}$$

where $\boldsymbol{H} \in \mathbb{R}^{m\times n}$ is the measurement matrix.

The Kalman filter estimates the state using the process model, the observation model and a sequence of observations. The filter equations are often formulated as time-ordered pair of recursive equations, given by a prediction step and a measurement update step. After the initialization, these two steps are executed in each time step. In the **prediction step**, a state estimate $\hat{\boldsymbol{x}}$ at time step k is determined given the knowledge of the process prior to time k. Let $\boldsymbol{Z}_k = (\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_k)$ be the sequence of all observation vectors up to time step k. The prediction of the state at time k is the expected value of the true state given all observations prior to time k [BS87, RP92]:

$$\hat{\boldsymbol{x}}_{k|k-1} = E[\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}]. \tag{3.72}$$

The estimation error $\boldsymbol{e}$ of the state estimate is defined as

$$\boldsymbol{e}_{i|j} = \boldsymbol{x}_i - \hat{\boldsymbol{x}}_{i|j} \tag{3.73}$$

and the associated error covariance matrix is

$$\boldsymbol{P}_{i|j} = E[\boldsymbol{e}_{i|j}\boldsymbol{e}_{i|j}^T|\boldsymbol{Z}_j]. \tag{3.74}$$

In the **measurement update step**, the state estimate $\hat{\boldsymbol{x}}_k$ is obtained by a linear combination of the predicted state estimate and a correction term incorporating the new observation $\boldsymbol{z}_k$:

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - \hat{\boldsymbol{z}}_{k|k-1}). \tag{3.75}$$

The vector $\boldsymbol{v}_k$ is called *innovation* and is equal to the difference between the actual observation $\boldsymbol{z}_k$ at time

k and the predicted observation $\hat{\boldsymbol{z}}_{k|k-1}$. The observation prediction is obtained by taking the conditional mean:

$$\hat{\boldsymbol{z}}_{k|k-1} = E[\boldsymbol{z}_k|\boldsymbol{Z}_{k-1}]. \tag{3.76}$$

The matrix $\boldsymbol{K}$ is called *Kalman gain matrix*. It is chosen to minimize the mean squared error in the state estimate $E[\boldsymbol{e}_{k|k-1}\boldsymbol{e}_{k|k-1}^T|\boldsymbol{Z}_{k-1}]$. This is equivalent to minimizing the trace of $\boldsymbol{P}_k$ [Kal60, May79] and the solution is:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}^{\boldsymbol{ev}}(\boldsymbol{P}_{k|k-1}^{\boldsymbol{vv}})^{-1} \tag{3.77}$$

where the matrices $\boldsymbol{P}_{k|k-1}^{\boldsymbol{ev}}$ and $\boldsymbol{P}_{k|k-1}^{\boldsymbol{vv}}$ are the corresponding covariance matrices. $\boldsymbol{P}_{k|k-1}^{\boldsymbol{vv}}$ represents the covariance of the innovation vector with its self and the $\boldsymbol{P}_{k|k-1}^{\boldsymbol{ev}}$ represents the covariance between the innovation vector and the state estimation error vector. The state estimation error is

$$\boldsymbol{e}_{k|k} = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k} \tag{3.78}$$

$$= \boldsymbol{x}_k - (\hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\boldsymbol{v}_k) \tag{3.79}$$

$$= \boldsymbol{e}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{v}_k. \tag{3.80}$$

Taking the outer product and expectation gives a recursive formula for $\boldsymbol{P}_{k|k}$:

$$\boldsymbol{P}_k = \boldsymbol{P}_{k|k-1} - \boldsymbol{P}_{k|k-1}^{\boldsymbol{ev}}\boldsymbol{K}_k^T - \boldsymbol{K}_k\boldsymbol{P}_{k|k-1}^{\boldsymbol{ev}} + \boldsymbol{K}_k\boldsymbol{P}_{k|k-1}^{\boldsymbol{vv}}\boldsymbol{K}_k^T. \tag{3.81}$$

Substituting the expression for $\boldsymbol{K}$ in equation (3.77) back into equation (3.81) gives the update equation for the error covariance matrix:

$$\boldsymbol{P}_k = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{P}_k^{\boldsymbol{vv}}\boldsymbol{K}_k^T. \tag{3.82}$$

An overview of the recursive Kalman filtering algorithm is given in algorithm 1. One can see that

Algorithm 1 Kalman Filtering Algorithm

Initialization

- $\boldsymbol{P}_{0|0} = \boldsymbol{R}_0$ and $\hat{\boldsymbol{x}}_{0|0} = E[\hat{\boldsymbol{x}}_0]$

Prediction Step

- Prediction of state and error covariance matrix

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1|k-1} \tag{3.83}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{A}\boldsymbol{P}_{k-1|k-1}\boldsymbol{A}^T + \boldsymbol{Q}_k \tag{3.84}$$

- Prediction of observation

$$\hat{\boldsymbol{z}}_{k|k-1} = \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1} \tag{3.85}$$

Measurement Update Step

- Compute the innovation

$$\boldsymbol{v}_k = \boldsymbol{z}_k - \hat{\boldsymbol{z}}_{k|k-1} \tag{3.86}$$

$$\boldsymbol{P}_{k|k-1}^{\boldsymbol{vv}} = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R}_k \tag{3.87}$$

$$\boldsymbol{P}_{k|k-1}^{\boldsymbol{ev}} = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T \tag{3.88}$$

- Computation of the Kalman gain matrix

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{R}_k^{-1} \tag{3.89}$$

- Update state estimation and error covariance matrix

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\boldsymbol{v}_k \tag{3.90}$$

$$\boldsymbol{P}_k = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{P}_{k|k}^{\boldsymbol{vv}}\boldsymbol{K}_k^T \tag{3.91}$$

the Kalman gain is determined by the predicted error covariance matrix, expressing confidence in the model, and the observation covariance matrix, expressing confidence in the current measurement. If the confidence in the state prediction is high relative to the measurement, the entries in the Kalman gain matrix will be small, and more weight will be given to the prediction. On the other hand, if the confidence in the measurement is relatively high, the innovation, including the new measurement, is weighted more heavily.

3.1.3.4 Extended Kalman filter

In many practical cases the system equation and the observation equation are nonlinear. The Extended Kalman Filter (EKF) approach to this situation is to apply the Kalman filter for linear systems by continuously updating a linearization of the system and observation function. The nonlinear functions are linearized about the predicted values and the mean values of the noise random variables. The linearization assumes that the differences between the true values and the estimated values are small enough to justify a first-order Taylor series expansion. Truncating the series after the first order introduces second and higher order errors, but by assumption, these are negligible. The system equation can be expanded to

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{v}_k, k-1) \tag{3.92}$$

$$\approx \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, 0, k-1) + \nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) + \nabla_{\boldsymbol{v}}\boldsymbol{f}\boldsymbol{v}_k \tag{3.93}$$

$$= \tilde{\boldsymbol{x}}_k + \nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) + \nabla_{\boldsymbol{v}}\boldsymbol{f}\boldsymbol{v}_k \tag{3.94}$$

where $\tilde{\boldsymbol{x}}_k$ is the state prediction using the previous estimate $\hat{\boldsymbol{x}}_{k-1}$ and assuming that the unknown value of $\boldsymbol{v}_k$ is equal to its mean, zero. $\nabla_{\boldsymbol{x}}\boldsymbol{f}$ is the Jacobian matrix of $\boldsymbol{f}$ with respect to the state $\boldsymbol{x}$, and $\nabla_{\boldsymbol{v}}\boldsymbol{f}$ is the Jacobian of $\boldsymbol{f}$ with respect to the process noise vector. Both matrices are evaluated at the point of linearization.

Similarly, the observation equation can be linearized:

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{w}_k, k) \tag{3.95}$$

$$\approx \boldsymbol{h}(\tilde{\boldsymbol{x}}_k, 0, k) + \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k) + \nabla_{\boldsymbol{w}}\boldsymbol{h}\boldsymbol{w}_k, \tag{3.96}$$

where $\nabla_{\boldsymbol{x}}\boldsymbol{h}$ and $\nabla_{\boldsymbol{w}}\boldsymbol{h}$ are the Jacobians of $\boldsymbol{h}$ with respect to the state $\boldsymbol{x}$, and the observation noise $\boldsymbol{w}$ respectively.
The predicted state is the conditional mean

$$\hat{\boldsymbol{x}}_{k|k-1} = E[\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}] \tag{3.97}$$

$$\approx E[\tilde{\boldsymbol{x}}_k + \nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) + \nabla_{\boldsymbol{v}}\boldsymbol{f}\boldsymbol{v}_k] \tag{3.98}$$

$$= E[\tilde{\boldsymbol{x}}_k] + E[\nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1})] + E[\nabla_{\boldsymbol{v}}\boldsymbol{f}\boldsymbol{v}_k] \tag{3.99}$$

$$= \tilde{\boldsymbol{x}}_k \tag{3.100}$$

where the last equation holds due to the assumption that both the error term $(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1})$ and the noise term $\boldsymbol{v}_k$ are random variables with zero mean. The prediction error made by this approximation is

$$\boldsymbol{e}_{k|k-1} = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1} \tag{3.101}$$

$$\approx \nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) + \nabla_{\boldsymbol{v}}\boldsymbol{f}\boldsymbol{v}_k \tag{3.102}$$

Taking the outer product and expectation yields the recursion for the predicted error covariance:

$$\boldsymbol{P}_{k|k-1} = \nabla_{\boldsymbol{x}}\boldsymbol{f}\boldsymbol{P}_{k-1}\nabla_{\boldsymbol{x}}\boldsymbol{f}^T + \nabla_{\boldsymbol{v}}\boldsymbol{Q}_k\nabla_{\boldsymbol{v}}\boldsymbol{f}^T \tag{3.103}$$

Similarly, expressions for the observation prediction and other error covariance matrices can be found:

$$\hat{\boldsymbol{z}}_{k|k-1} = \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, 0, k-1) \tag{3.104}$$

$$\boldsymbol{P}^{\boldsymbol{vv}}_{k|k-1} = \nabla_{\boldsymbol{x}}\boldsymbol{h}\boldsymbol{P}_{k|k-1}\nabla_{\boldsymbol{x}}\boldsymbol{h}^T + \nabla_{\boldsymbol{w}}\boldsymbol{h}\boldsymbol{R}_k\nabla_{\boldsymbol{w}}\boldsymbol{h}^T \tag{3.105}$$

$$\boldsymbol{P}^{\boldsymbol{ev}}_{k|k-1} = \boldsymbol{P}_{k|k-1}\nabla_{\boldsymbol{x}}\boldsymbol{h}^T \tag{3.106}$$

When these equations are used in the Kalman filter algorithm 1, the filter is known as *Extended Kalman Filter* (EKF), because it extends the filter to nonlinear systems. However, a limitation of this approach is that the nonlinear system has to be linearizable, and it is not clear in which cases this is possible. Indeed, the second and higher-order terms may be significant and the introduced errors can lead to the divergence of the filter. One way to approach this problem is to explicitly include more terms in the Taylor expansions. However, the implementation of such filters is often impractical due to their computational complexity.

3.1.3.5 Unscented Kalman filter

When the system and the observation equations are linear, the best state estimate in the minimum mean squared error (MMSE) sense can be obtained with the Kalman filter algorithm. For nonlinear stochastic systems the optimal estimator in the sense of minimizing the mean-square error $E[(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k)^2|\boldsymbol{Z}_k]$ is the mean of the state at time k conditioned on the observation up to time instant k [BS87, Jaz70]:

$$\hat{\boldsymbol{x}}_k^{MMSE} = E[\boldsymbol{x}_k|\boldsymbol{Z}_k] = \int p(\boldsymbol{x}_k|\boldsymbol{Z}_k)\boldsymbol{x}_k d\boldsymbol{x}_k \tag{3.107}$$

Computing the conditional state distribution $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ is not trivial in general, since the observation sequence grows with time and an infinite number of parameters (e.g. moments) would be required. Often the state estimation problem does not have an explicit solution. However, under the assumption of white and independent noise sequences the optimal estimator may be computed using a recursive formula for the conditional densities. Bayes's rule specifies an expression for updating the conditional probability density of the current state $\boldsymbol{x}_k$:

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k) = c\ p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) \tag{3.108}$$

where the equation follows from the assumption that the measurement noise is white and independent in time, so that $p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{Z}_{k-1}) = p(\boldsymbol{z}_k|\boldsymbol{x}_k)$. c is a normalization constant. The prior state distribution can be written as:

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) = \int p(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})d\boldsymbol{x}_{k-1} \tag{3.109}$$

$$= \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{Z}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})d\boldsymbol{x}_{k-1} \tag{3.110}$$

$$= \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})d\boldsymbol{x}_{k-1}, \tag{3.111}$$

where the last equation is valid under the assumption that the process noise sequence is white and independent of the measurement noise. Plugging equation (3.111) into equation (3.108) yields the recursive formula:

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k) = c\ p(\boldsymbol{z}_k|\boldsymbol{x}_k) \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})d\boldsymbol{x}_{k-1}. \tag{3.112}$$

It is often convenient to evaluate this expression using a time-ordered pair of recursive equations:

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})d\boldsymbol{x}_{k-1} \tag{3.113}$$

and

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k) = c\ p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_{k-1}, \boldsymbol{Z}_{k-1}). \tag{3.114}$$

In the linear Gaussian case, the above equations lead directly to the Kalman filter equations.

Various methods have been suggested for the nonlinear case. The EKF handles nonlinearity by linearizing the system and observation equations and applying the Kalman filter algorithm as in the linear case. Instead of approximating the equations by terms in the Taylor series, the problems can also be approached by approximating the prior distribution of $\boldsymbol{x}_k$ directly. For this, a number of discrete samples is drawn at random, the model is applied to each of these samples, and the statistics of the transformed samples are computed. This technique forms the principle of Monte-Carlo methods [Han70]. However, the convergence rate for these methods are slow and a large number of samples is required. If n is the number of samples, the covariance converges according to $\sqrt{n}$. These problems motivated the development of a new filter by Julier and Uhlmann [Jul97, JUD95, JU97, WM00]. This filter approximates the state

distribution, but the samples are chosen in a deterministic way. The resulting algorithm is known as *Unscented Kalman Filter* (UKF).

The underlying principle of the UKF is the unscented transformation. The unscented transformation is a technique for computing the statistics of a random variable after a nonlinear transformation. More specifically, given an n dimensional random variable $\boldsymbol{x}_{k-1}$ with mean $\hat{\boldsymbol{x}}_{k-1}$ and covariance matrix $\boldsymbol{P}_{k-1}$ the following procedure is undertaken:

- A set of $2n+1$ points with the same mean $\hat{\boldsymbol{x}}_{k-1}$ and covariance $\boldsymbol{P}_{k-1}$ is selected.
- The nonlinear function is applied to each of these points.
- The mean and the covariance for the transformed points are computed.

The points are chosen according to the following scheme:

$$\boldsymbol{X}^i_{k-1} = \begin{cases} \hat{\boldsymbol{x}}_{k-1} & i = 0 \\ \hat{\boldsymbol{x}}_{k-1} - \sigma^i_{k-1} & i = 1, ..., n \\ \hat{\boldsymbol{x}}_{k-1} + \sigma^{i-n}_{k-1} & i = n+1, ..., 2n \end{cases} \tag{3.115}$$

where σ^i_{k-1} is the ith column of the matrix $\sqrt{(n+\kappa)\boldsymbol{P}_{k-1}}, \kappa \in \mathbb{R}$. The square root of $\sqrt{A}$ a symmetric positive definite matrix A is defined as follows: If $BB^T = A$, then $B = \sqrt{A}$. The square root is determined up to orthonormal transformations (if B is square root of A, so is BU, if $UU^T = I$). A valid square root of a matrix can efficiently be computed using Cholesky decomposition. Additionally, the points are weighted according to the following weights:

$$w^i = \begin{cases} \frac{\kappa}{n+\kappa} & i = 0 \\ \frac{1}{2(n+\kappa)} & i = 1, ..., 2n+1 \end{cases} \tag{3.116}$$

and $\sum_i w^i = 1$. These weighted points have the same mean, covariance and all higher odd ordered moments at the Gaussian distribution of $\boldsymbol{x}_{k-1}$ [Jul97]. Each point is transformed through the process model:

$$\boldsymbol{X}^i_{k|k-1} = \boldsymbol{f}(\boldsymbol{X}^i_{k-1}, k), \;\; i = 0, ..., 2n \tag{3.117}$$

and the predicted mean and covariance matrix is computed as:

$$\hat{\boldsymbol{x}}_{k|k-1} = \sum_{i=0}^{2n} w^i \boldsymbol{X}^i_{k|k-1} \tag{3.118}$$

$$\boldsymbol{P}_{k|k-1} = \sum_{i=0}^{2n} w^i [\boldsymbol{X}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}][\boldsymbol{X}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}]^T \tag{3.119}$$

It can be shown that this prediction of the moments is correct up to the third order [Jul97].

The selection of points can also be extended to include process noise. For doing so, the noise terms are concatenated to the state vector to yield an *augmented state vector* of dimension $n+q$. This method ensures that the effects of the process noise on the mean and the covariance are introduced with the same order of accuracy as the uncertainty in the state.

The same principle is followed to predict the state observation $\hat{\boldsymbol{z}}_{k|k-1}$ and the innovation covariance matrix $\boldsymbol{P}^{\boldsymbol{vv}}_{k|k-1}$, including effects of observation noise.

$$\boldsymbol{Z}^i_{k|k-1} = \boldsymbol{h}(\boldsymbol{X}^i_{k|k-1}, k-1), i = 0, ..., 2n \tag{3.120}$$

$$\hat{\boldsymbol{z}}_{k|k-1} = \sum_{i=0}^{2n} w^i \boldsymbol{Z}^i_{k|k-1} \tag{3.121}$$

$$\boldsymbol{P}^{\boldsymbol{vv}}_{k|k-1} = \sum_{i=0}^{2n} w^i [\boldsymbol{Z}^i_{k|k-1} - \hat{\boldsymbol{z}}_{k|k-1}][\boldsymbol{Z}^i_{k|k-1} - \hat{\boldsymbol{z}}_{k|k-1}]^T + \boldsymbol{R}_k \tag{3.122}$$

$$\boldsymbol{P}^{\boldsymbol{ev}}_{k|k-1} = \sum_{i=0}^{2n} w^i [\boldsymbol{X}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}][\boldsymbol{Z}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}]^T \tag{3.123}$$

The parameter κ is weighting the mean relative to the other sample points. This does not affect the

Algorithm 2 Unscented Kalman Filtering Algorithm

- **Initialization**

$$\boldsymbol{X}^i_{k-1} = \begin{cases} \hat{\boldsymbol{x}}_{k-1} & i = 0 \\ \hat{\boldsymbol{x}}_{k-1} - \sigma^i_{k-1} & i = 1, ..., n \\ \hat{\boldsymbol{x}}_{k-1} + \sigma^{i-n}_{k-1} & i = n+1, ..., 2n \end{cases} \tag{3.124}$$

$$w^i = \begin{cases} \frac{\kappa}{n+\kappa} & i = 0 \\ \frac{1}{2(n+\kappa)} & i = 1, ..., 2n+1 \end{cases} \tag{3.125}$$

- **Prediction step:**

$$\boldsymbol{X}^i_{k|k-1} = \boldsymbol{f}(\boldsymbol{X}^i_{k-1}, k), i = 0, ..., 2n \tag{3.126}$$

$$\hat{\boldsymbol{x}}_{k|k-1} = \sum_{i=0}^{2n} w^i \boldsymbol{X}^i_{k|k-1} \tag{3.127}$$

$$\boldsymbol{P}_{k|k-1} = \sum_{i=0}^{2n} w^i [\boldsymbol{X}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}][\boldsymbol{X}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}]^T \tag{3.128}$$

$$\boldsymbol{Z}^i_{k|k-1} = \boldsymbol{h}(\boldsymbol{X}^i_{k|k-1}, k-1), i = 0, ..., 2n \tag{3.129}$$

$$\hat{\boldsymbol{z}}_{k|k-1} = \sum_{i=0}^{2n} w^i \boldsymbol{Z}^i_{k|k-1} \tag{3.130}$$

$$\boldsymbol{P}^{\boldsymbol{vv}}_{k|k-1} = \sum_{i=0}^{2n} w^i [\boldsymbol{Z}^i_{k|k-1} - \hat{\boldsymbol{z}}_{k|k-1}][\boldsymbol{Z}^i_{k|k-1} - \hat{\boldsymbol{z}}_{k|k-1}]^T + \boldsymbol{R}_k \tag{3.131}$$

$$\boldsymbol{P}^{\boldsymbol{ev}}_{k|k-1} = \sum_{i=0}^{2n} w^i [\boldsymbol{X}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}][\boldsymbol{Z}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}]^T \tag{3.132}$$

- **Update step:**

$$\boldsymbol{K}_k = \boldsymbol{P}^{\boldsymbol{ev}}_{k|k-1} (\boldsymbol{P}^{\boldsymbol{vv}}_{k|k-1})^{-1} \tag{3.133}$$

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \boldsymbol{v}_k \tag{3.134}$$

$$\boldsymbol{P}_k = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{P}^{\boldsymbol{vv}}_{k|k} \boldsymbol{K}^T_k \tag{3.135}$$

mean but scales the distributions of the points, since they are found from $\sqrt{(n+\kappa)\boldsymbol{P}}$. This scaling can lead to higher accuracy in the higher order moments of the distribution.

3.1.3.6 Comparison of EKF and UKF

The UKF has several advantages over the EKF. These can be summarized as follows:

- **Accuracy:** The UKF has a lower expected error for all continuous nonlinear transformations. The errors in the EKF prediction are of second order, the UKF is accurate up to the third order.
- **Applicability:** Linearization in the EKF algorithm can produce very unstable filter performance if the time steps are not sufficiently small. In case of non-differentiable functions, the EKF is not applicable at all, whereas the UKF can be applied.
- **Computational Complexity:** The selection of points in the UKF algorithm can be achieved efficiently using Cholesky decomposition. It is not necessary to evaluate Jacobian matrices in the UKF, compared to the EKF. For many problems, the derivation of Jacobians is non-trivial and the computational cost to compute them may be significant.

- **Empirical Results:** In the experiments the UKF has been shown to outperform the EKF in a number of problems such as vehicle control, parameter estimation for neural networks and satellite navigation [Hay01].

3.1.3.7 Limitations of the UKF

The major limitation of the UKF is the Gaussian assumption. Although performing a better approximation than the EKF, is still relies on the hypothesis that the prior distribution is Gaussian and that the posterior is still well approximated by such type of distribution. In fact, even if we accept the prior Gaussian restriction, after transforming the random variable through a nonlinear function the result will no longer be Guassian. In order to generalise to arbitrary distributions, the ideas of particle filtering and the unscented transform have been combined to construct a new filtering algorithm [WDF00].

3.1.4 Experimental results

Real data experiments were designed to test the proposed tracking algorithm. A single-view and a stereo sequence of 200, 1024×768 color images of a hand in front of a white background have been acquired as a first test dataset. The parameters of the hand model were manually set to match the pose of the hand in the first frame of the sequence. In the experiments six degrees of freedom were given to the motion of the hand corresponding to $x-$, $y-$ and $z-$direction translation and rotation about the $x-$, $y-$ and $z-$ axis. The dynamics of the hand were modeled using a second order process, using position, velocity and acceleration. Therefore the state vector was:

$$\boldsymbol{X} = [x, y, z, \theta_x, \theta_y, \theta_z, \dot{x}, \dot{y}, \dot{z}, \dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z, \ddot{x}, \ddot{y}, \ddot{z}, \ddot{\theta}_x, \ddot{\theta}_y, \ddot{\theta}_z]^T \tag{3.136}$$

The results of the tracking algorithm are presented in the Figure 3.12, in which the projection of the 3D hand model is shown superimposed to the hand images. It can be verified that the system is accurate enough, given the fact that only 6 degrees of freedom are allowed to move, succeeding in obtaining the correct pose of the hand.

3.1.5 Conclusions and future work

In the previous sections a 3D model-based hand tracking system was presented. The use of quadrics to build the 3D model yields a practical and elegant method for generating the contours of the model, which are then compared with the image data. The results of this comparison are used in an Unscented Kalman Filter (UKF) framework to estimate the current motion and the configuration parameters of the hand. The experimental results demonstrate the efficiency of the proposed methods. The system is generic, in the sense that it can be easily expanded to use multiple views and can be applied without any modifications to rigid and articulated models. Finally, keeping the computational load at a bearable level, a real-time implementation is expected to be achieved for an optimized tracker that could exploit all the processing power of a conventional computer (including CPU and GPU).

Ongoing and future work with respect to 3D hand tracking addresses the following issues:

- **Use of different filters** The efficiency of the UKF and its superiority over the Extended Kalman Filter (EKF) have already been demonstrated. However, the UKF assumes a uni-model noise model, which probably is not appropriate for the hand tracking problem (especially for movements such as grasping). Particle filters seem to be a solution to this problem, although the introduce an extensive computational load. There have also been studies that propose the use of particle filters in a graphical model framework, demonstrating good tracking results, in the presence of self occlusions. However, the computational load remains too much also in this case, posing problems to real-time implementations. The fact that the hand tracking system will be assigned to track specific movements (grasping movements) has also to be taken under serious consideration in the choice of another tracking system.
- **Dimensionality Reduction** The hand tracking problem is a high dimensional inference problem and it would be desirable to reduce the number of dimensions that a system has to track, thus making the tracking algorithms perform faster and more robustly. One has to consider that a

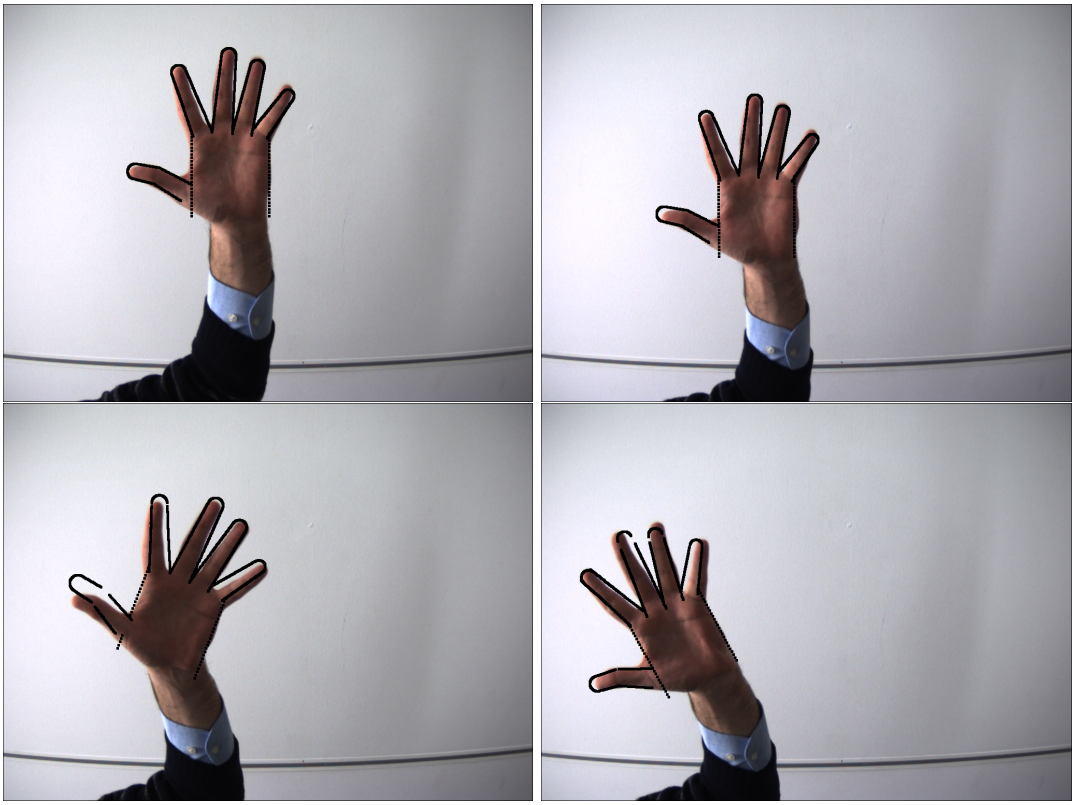


Figure 3.12: Hand tracking results for frames 10 (upper left), 40 (upper right), 88 (bottom left) and 185 (bottom right) of the test data set.

number of couplings exist in the motion of the fingers. A number of methods for dimensionality reduction have been proposed and it has to be made further research to choose one of them as the most appropriate for this problem.

- **Analysis of failure cases.** At the moment it is unclear which are the limits of the proposed contour-based approach. It has been though demonstrated that the handling of self-occlusions makes the tracking in these situations feasible. Extensive testing of image sequences is desired, containing out of plane rotations and opening-closing of the hand.

- **Improvement of the observation model** Many features have been proposed to the hand tracking problem. These include, edges, skin color and optical flow. Currently edges and skin-clor are used in the framework described in the corresponding chapter. However, it has been reported that some slight modifications in the treatment of these features can make the observation model more robust. Specifically, chamfer distances can be used in different orientations and oriented edges could be used to improve the performance of the system. Furthermore, an adaptive skin-color model could be used to improve the performance of the current skin-color model.

- **Automatic initialization of the configuration of the hand** It is very important to have a refined shape of the model at the first frame so that a better alignment can be achieved between the projection of the model and the edges in the image. A good initialization will make possible the increase of the number of model parameters to be tracked. The initialization of the model is currently done by hand. However, an optimization framework could be used to estimate the shape of the model from a set of images as an off-line preprocessing step before the tracking begins.

- **Performance and accuracy evaluation** The accuracy of the model can be evaluated using ground truth measurements. This will be done as soon as there are some measurements from the experiments that take place in LMU, in which sensors are placed in the fingertips of the hand. Finally, performance evaluation will be obtained as soon as there is an optimized final version of the tracker.

3.2 Grasp type classification using binary latent variables

3.2.1 Modeling human motion using Restricted Boltzmann Machines

Boltzmann Machines is a type of stochastic artificial neural network that is considered to be the stochastic version of a Hopfield network. They are composed of probabilistic binary units symmetrically connected to each other. Learning in Boltzmann Machines of unrestricted connectivity is slow and renders them unusable to practical problems. If, however, the connectivity is restricted only among different layers (i.e. no connections within the same level), learning becomes much more efficient. This type of ANN is called a *Restricted Boltzmann Machine* (RBM).

RBMs have been shown to be successful in a number of practical applications, ranging from character recognition to full motion modeling. Taylor et al [THR06, SH06] have used RBMs to model different kinds of human motion (running, walking, sitting etc). Their model proved successful in effortlessly reproducing different kinds of motion and alternating between them.

In a restricted Boltzmann machine there are two layers of nodes: a visible layer that represents the observables during both learning and generation of motion, and a hidden layer of stochastic binary nodes. The network (see Fig.3.13) alternates between 2 phases:

- The positive phase, where the hidden states are being updated with the probability of a hidden unit being dependent on the input that the hidden unit receives:

$$p(h_j = 1|\mathbf{v}) = f(b_j + \sum_i v_i w_{ij})$$

- The negative phase, where the machine generates the visible data from the hidden state. The visible nodes in a typical RBM are binary, but, if we are modeling continuous variables, as is the case, they need to be replaced by linear, real valued units. The probability of visible states is thus:

$$p(v_i|\mathbf{h}) = N(c_i + \sum_j h_j w_{ij}, 1)$$

  where $N()$ is a Gaussian distribution with standard deviation 1.

The learning rule can be approximated accurately by following the gradient of a function called "Contrastive Divergence" [Hin02]. The learning rule is:

$$\Delta w_{ij} \propto < v_i h_j >_{data} - < v_i h_j >_{recon},$$

where the $< . >_{recon}$ denotes expectation with respect to the reconstructed data after running the machine for 1 step.

3.2.2 Conditional RBMs

When modeling human motion, temporal information needs to be incorporated. One way to achieve this is to treat the visible variables of previous time slices as additional fixed inputs, as suggested by Taylor

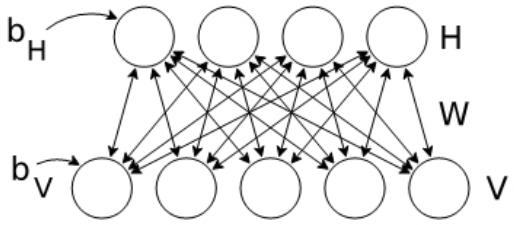


Figure 3.13: Restricted Boltzmann Machine (RBM).

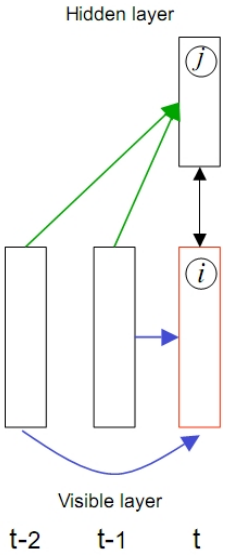


Figure 3.14: Use of cRBMs to model temporal data

et al.[THR06]. Thus, directed connections are added as additional biases from the previous n time slices to the current visible and hidden layers (see Fig. 3.14). The learning rule for the directed connections from the previous time slices is:

$$\Delta {d_{ij}}^{(t-q)} \propto {v_i}^{t-q}(< {h_j}^t >_{data} - < {h_j}^t >_{recon})$$

for the visible-to-hidden biases and

$$\Delta {a_{ki}}^{(t-q)} \propto {v_k}^{t-q}({v_i}^t - < {v_i}^t >_{recon})$$

for the visible-to-visible biases.

3.2.3 Modeling the motion of human hands

We can use the methods described above to model the motion of human hands when executing reaching and grasping actions. Our data comprises of hand movements when the subject executes a variety of grasping and reaching actions. In preliminary data sets acquired by LMU, a Polhemus tracker has been used to record the positions of three fingers plus that of the palm of a subject that has been performing certain reaching and grasping actions. These trajectories were concatenated in a 12 dimension observation vector for each time frame. Each trajectory is 100 frames long. We trained a cRBM with 150 hidden units using the observation data for various actions for 200 epochs using a learning rate of 10^{-3}. The cRBM was 3 levels deep, that is, there were directed connections from the past 3 visible vectors to the current visible and hidden vectors.

We first used the trained network to reproduce the learned trajectories, learning one trajectory at a time. Initializing the machine with the 3 first frames of the learned movement, the network is able to reproduce an action very similar to the learned trajectory (see Fig.3.15). For the generation of subsequent frames, the machine proceeds as follows: the visible layer is initialized using the previous frame data, adding Gaussian noise. Then the hidden and visible layers are updated iteratively with alternating Gibbs sampling using 30 iterations.

The same network can be used to model and reproduce more than one type of actions. The actions are reproduced successfully by starting the network with the 3 first frames of each action (see Fig. 3.16).

We can examine the robustness of generating an action by initializing the machine with frames away from the beginning of the action. The generation of action is robust when initialized with frames no further than 30 frames from the beginning of the action (see Fig. 3.17). Initialization with subsequent frames

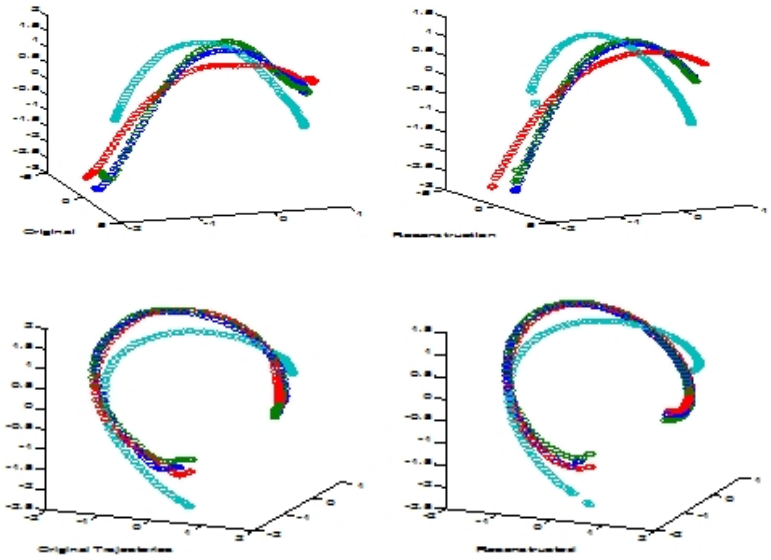


Figure 3.15: Examples of reconstructed finger positions: original trajectories (left) and reconstructed ones (right)

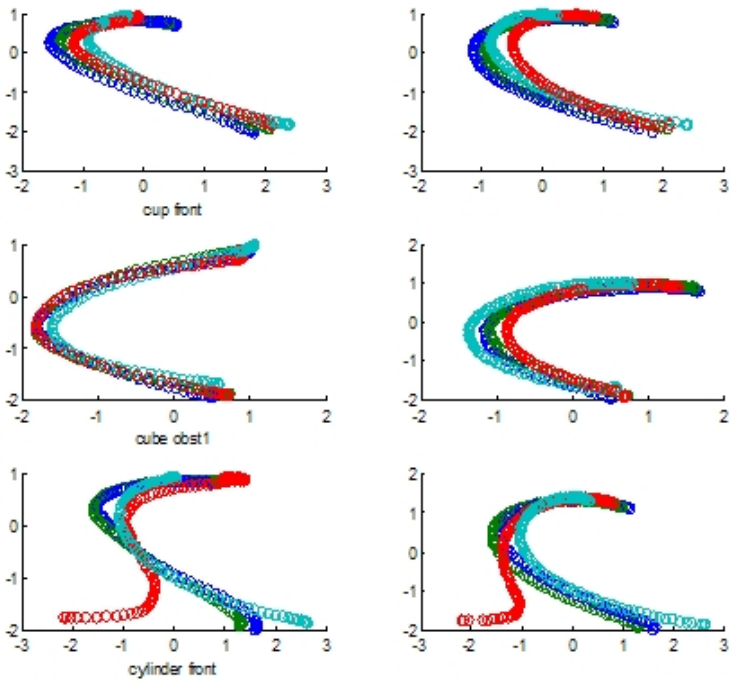


Figure 3.16: **Trajectory reconstruction results**

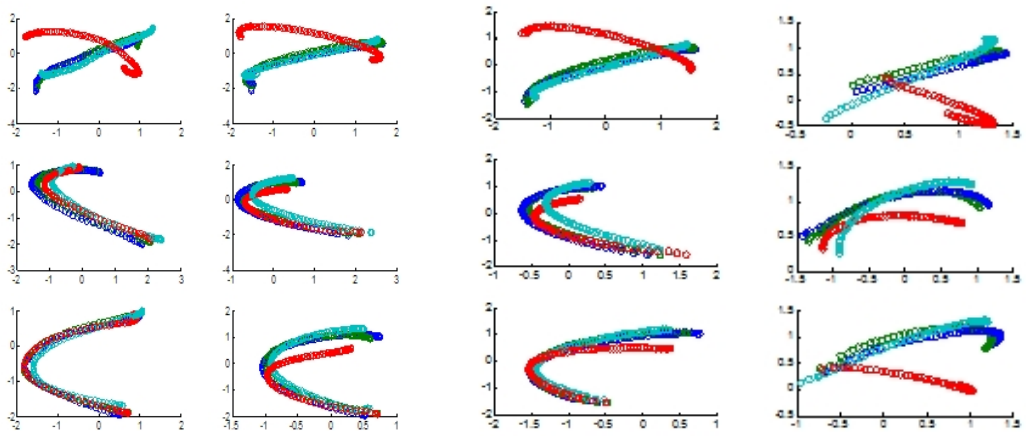


Figure 3.17: **Trajectory reconstruction results**

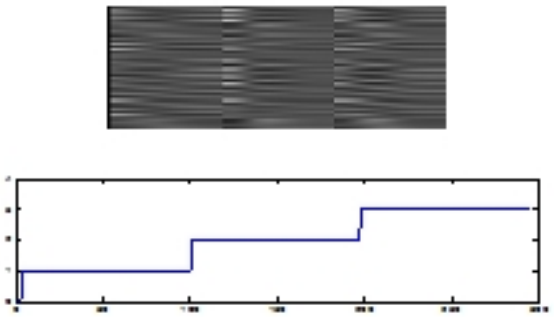


Figure 3.18: **Activation of hidden units (top) and classification output (bottom)**

seems to generate false trajectories, depending on the kinds of actions learned. The figure below shows actions generated by initializing the machine at various time points.

3.2.4 Classification of grasping actions

We performed preliminary experiments to investigate the ability of an RBM to classify accurately various types of grasps. As the cRBM hidden layer nodes model the features of the motion, used the activation probabilities of the hidden nodes during the course of the action to infer the category of the action that is provided in the visible layer. We thus trained an RBM network in 3 subsequent actions. The activation probabilities of the hidden units during the course of all the actions were used as input to a simple feedforward backpropagation network. The network has a single linear real-valued output unit which gives the current estimate for the class of action observed. The network was trained to output an integer (1..3) corresponding to the action being observed. We used the default BP network implementation of MATLAB with 3 hidden units. The network was trained to recognize the patterns of activation in different actions as shown if Fig. 3.18.

Using this network, we can attempt to classify actions of the same class that were executed by different persons. Although the networks seems to discriminate between different actions, it sometimes seems to be confused among different classes of actions (see Fig. 3.19).

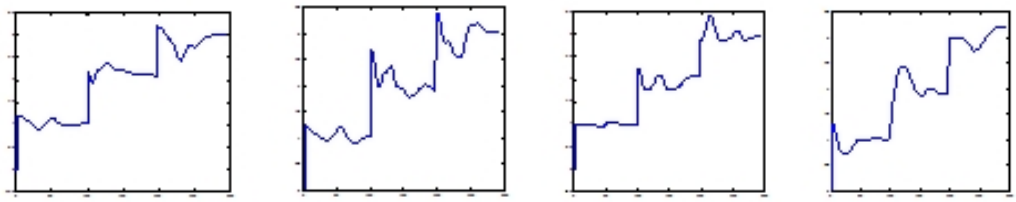


Figure 3.19: **Classification results**

3.2.5 Future work

The purpose of this work was to investigate the ability of RBM networks to generate and classify different kinds of 3D hand trajectories. These RBMs can then be used to model the regularities of complex hand and finger movements that would be hard to model using a physical model of the hand. This can prove useful in on-line prediction and guidance of action. The output of the system can also be used to infer the kind of action being performed. This is useful for the recognition of action by a higher-level process. Importantly, this recognition mechanism can be used to aid the segmentation of an action to its parts. One research direction that we plan to investigate is the exploitation of RBMs trained in grasping movements for dimensionality reduction in the context of the developed framework for 3D hand tracking.

3.3 Acquisition of grasping sequences

In a collaboration of Ludwig-Maximilians-University (Germany), FORTH (Greece) and UniKarl (Germany) this activity of GRASP studies human grasping behaviour with an interdisciplinary approach. The rationale is to track the human hand during grasp movements and to simultaneously record high-speed motion pictures of these operations. The motion pictures will then be used to assist computer algorithms to detect and recognize the human hand in the scene and to classify from the appearance of the hand predefined posture primitives. This is an essential step in implementing robots that can learn from a human model. Further, the simultaneously recorded multidimensional information about the actual trajectories and dynamics of several hand parts will be used to evaluate and optimize the visual detection algorithm.

Two systems are implemented in the setup. An electromagnetic Polhemus Liberty system provides three dimensional information about the position and orientation of eight sensors which are attached to the grasping hand (see Figure 3.20).

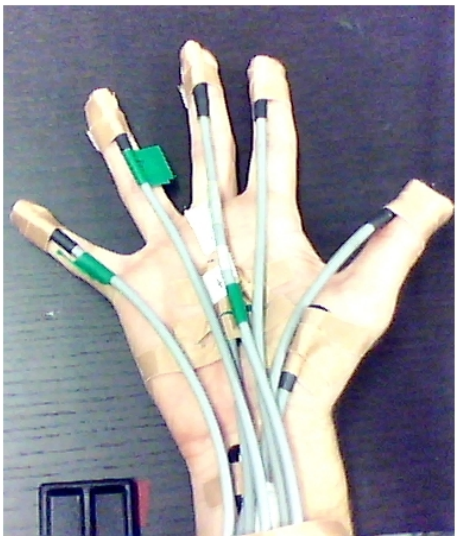


Figure 3.20: The placement of Polhemus sensors on a human hand.

The system continuously records data with 240 Hz. These recordings can also be used to animate a virtual 3D hand model. The experimental procedure in every trial is controlled by Matlab software (Mathworks) using psychophysics toolbox. This program also controls Polhemus Liberty recording system and sends online triggers to the camera system that runs on a different platform.

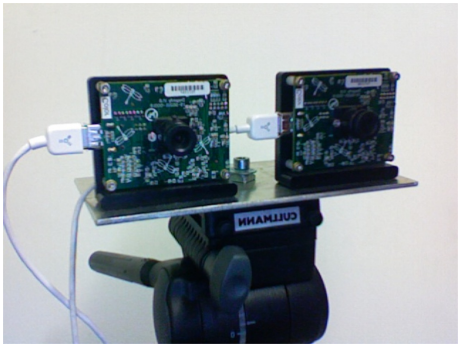


Figure 3.21: The stereo camera system observing grasping activities.

A binocular camera system (figure 3.21) records the ongoing grasping movements at a rate of 30 frames per second from an external observation point. This observation point is similar to the viewpoint of a humanoid robot that observes a human in natural environments. The recorded image sequences (see an example in figure 3.22) will be used by FORTH to train an algorithm that can recognize and detect the human hand and extract a sequence of hand postures (posture primitives). The exact synchronization of

both recordings is very important. Later on, the synchronized hand tracking data will help to validate the outcome of the hand detection and tracking algorithm.

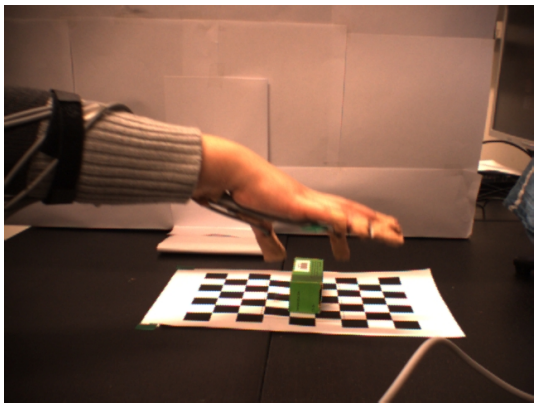


Figure 3.22: A frame acquired by one of the cameras of the stereo head observing a grasping activity.

In several experimental studies human participants execute grasping movements for various objects of everyday use. The objects of study are a cup, a cylinder, a box and a small matchbox like item. Each object will be accessed by different types of grasps (e.g., spherical, cylindrical, pinch etc.). In a typical experimental trial the hand will start from an open start posture in front of the object with the palm facing upwards. Upon an acoustical go-signal the human actor is required to grasp the object with a comfortable, unspeeded and stable grasp. For each object and each grasp type 20 trials will be recorded in a total of five human participants. The analysis of the recorded hand trajectories will also provide grand-averaged data about typical trajectories when grasping a certain kind of object (see figure 3.23). We will extract from these trajectories important via-points (i.e., intermediate goals where the hand path runs through by default). Another focus in the analysis lies on the spatial and temporal variations within and between subjects in order to determine factors that make grasping movements unstable or prone to failure.

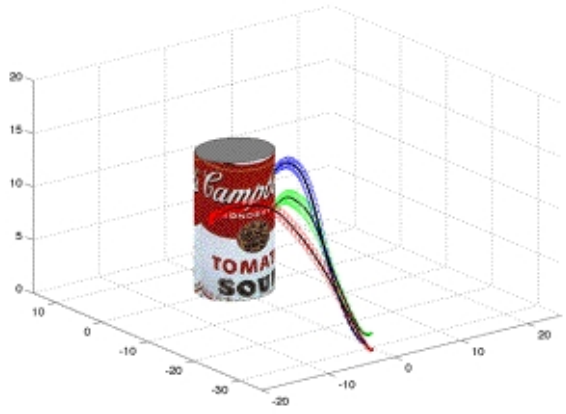


Figure 3.23: Example trajectories recorded by the Polhemus system while a person has been reaching and grasping a cylindrical object.

Chapter 4

Future work

Deliverable D3 presented the current status and results of GRASP research on human upper body tracking and 3D hand tracking. With respect to 3D hand tracking, a number of issues have been identified in section 3.1.5, including:

- Accuracy evaluation and analysis of failure cases
- Use of different filters
- Dimensionality reduction
- Improvement of the observation model
- Automatic initialization of the configuration of the hand

Of particular interest, is the exploitation of the work on RBMs section 3.2) together with probabilistic inference for developing a 3D hand tracker tailored for grasping. The availability of image sequences of grasping activities that are annotated with synchronized ground truth data acquired through a Polhemus tracker (collaboration of LMU, UniKarl and FORTH, section 3.3) is expected to prove very valuable in this direction.

Finally, the integration of the developed systems for upper body tracking need to be properly integrated with efforts on 3D hand tracking and the overall system needs to be evaluated in terms of robustness and with respect to real time performance issues.

Chapter 5

References

[AL04] Antonis A. Argyros and Manolis I. A. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 368–379, Prague, Czech Republic, May 2004.

[AS03] Vassilis Athitsos and Stan Sclaroff. Estimating 3d hand pose from a cluttered image. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:432, 2003.

[AUAD07] P. Azad, A. Ude, T. Asfour, and R. Dillmann. Stereo-based markerless human motion capture for humanoid robot systems. In *ICRA*, pages 3951–3956, 2007.

[B.02] Martinkauppi B. *Face colour under varying illumination - analysis and applications.* PhD thesis, University of Oulu, 2002. Dissertation. Acta Univ Oul C 171, 104 p + App.

[BALT08] H. Baltzakis, A. Arhyros, M. Lourakis, and P. Trahanias. Tracking of human hands and faces through probabilistic fusion of multiple visual cues. In *Proc. International Conference on Computer Vision Systems (ICVS)*, Santorini, Greece, May 2008.

[BCB09] Ricardo Backes, Dalcimar Casanova, and Odemir Martinez Bruno. A complex network-based approach for boundary shape analysis. *Pattern Recognition*, 42(1):54 – 67, 2009.

[BET09] E. Baseski, A. Erdem, and S. Tari. Dissimilarity between two skeletal trees in a context. *Pattern Recognition*, 42(3):370 – 385, 2009.

[BI98] Andrew Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[Bir98] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 232, Washington, DC, USA, 1998. IEEE Computer Society.

[BKMVG04] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for 3d hand tracking. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 675–680, 2004.

[BMP02] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

[Bro88] G. Brogefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.

[BS87] Y. Bar-Shalom. *Tracking and data association.* Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[Can86] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[CFH+09] M. Cui, J. Femiani, J. Hu, P. Wonka, and A. Razdan. Curve matching for open 2d curves. *Pattern Recognition Letters*, 30(1):1 – 10, 2009.

[DBR00] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, volume 2, pages 126–133 vol.2, 2000.

[DDH04] Guillaume Dewaele, Frédéric Devernay, and Radu P. Horaud. Hand motion from 3d point trajectories and a smooth surface model. In T. Pajdla and J. Matas, editors, *Proceedings of the 8th European Conference on Computer Vision*, volume I of *LNCS 3021*, pages 495–507. Springer, May 2004.

[dlGPF08] M. de la Gorce, N. Paragios, and D.J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *CVPR08*, pages 1–8, 2008.

[DPD+07] Doina Dragulescu, Veronique Perdereau, Michel Drouin, Loredana Ungureanu, and Karoly Menyhardt. 3d active workspace of human hand anatomical model. *BioMedical Engineering OnLine*, 6(1):15, 2007.

[dSTF07] R. da S. Torres and A.X. Falcao. Contour salience descriptors for effective image retrieval and analysis. *Image and Vision Computing*, 25(1):3 – 13, 2007.

[EAACC08] Yasser Ebrahim, Maher Ahmed, Wegdan Abdelsalam, and Siu Chung-Chau. Shape representation and description using the hilbert curve. *Pattern Recognition Letters*, In Press, Accepted Manuscript:–, 2008.

[EBN+07] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, 2007.

[FBTdFC07] R. Fabbri, O. M. Bruno, J. C. Torelli, and L. da F. Costa. 2d euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys*, 40(1):2:1–2:44, 2007.

[FGD99] Lerasle F., Rives G., and M. Dhome. Tracking of human limbs by multiocular vision. *Comput. Vis. Image Underst.*, 75(3):229–246, 1999.

[FHD09] P. Azad F. Hecht and R. Dillmann. Markerless human motion tracking with a flexible model and appearance learning. In *ICRA*, 2009.

[FS07] P.F. Felzenszwalb and J.D. Schwartz. Hierarchical matching of deformable shapes. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[Gav00] Dariu Gavrila. Pedestrian detection from a moving vehicle. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 37–49, London, UK, 2000. Springer-Verlag.

[GGS04] J. Giebel, D. M. Gavrila, and C. Schnrr. A bayesian framework for multi-cue 3d object tracking. In *In Proceedings of European Conference on Computer Vision*, pages 241–252. SpringerVerlag, 2004.

[Han70] J. E. Handschin. Monte carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica*, 6(4):555 – 563, 1970.

[Hay01] Simon S. Haykin. *Kalman Filtering and Neural Networks*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[HH96] T. Heap and D. Hogg. Towards 3d hand tracking using a deformable model. *Automatic Face and Gesture Recognition, IEEE International Conference on*, 0:140, 1996.

[Hin02] G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[IB98] Michael Isard and Andrew Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 893–908, London, UK, 1998. Springer-Verlag.

[IM01] M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. *Computer Vision, IEEE International Conference on*, 2:34, 2001.

[Jaz70] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, April 1970.

[JR02] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *Int. J. Comput. Vision*, 46(1):81–96, 2002.

[JU97] S. J. Julier and J. K. Uhlmann. New extension of the Kalman filter to nonlinear systems. In I. Kadar, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 3068 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 182–193, July 1997.

[JUD95] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of the American Control Conference*, volume 3, pages 1628–1632, June 1995.

[Jul97] Simon J. Julier. Process models for the navigation of high-speed land vehicles, 1997.

[JZD03] Bruno Jedynak, Huicheng Zheng, and Mohamed Daoudi. Statistical models for skin detection. In *IEEE Workshop on Statistical Analysis in Computer Vision, in conjunction with CVPR 2003*, 2003.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[KH94] J. J. Kuch and T. S. Huang. Human computer interaction via the human hand: a hand model. In *Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on*, volume 2, pages 1252–1256 vol.2, 1994.

[KSJ91] Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell. *Principles of Neural Science, 3rd Ed.* Appleton & Lange, Norwalk, CT, 1991.

[Lat] Longin Jan Latecki. Shape data for the mpeg-7 core experiment ce-shape-1. http://www.cis.temple.edu/ latecki/TestData/mpeg7shapeB.tar.gz.

[LF02] R. Lockton and A. W. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *Proceedings, British Machine Vision Conference*, 2002.

[LH04] John Y. Lin and Thomas S. Huang. 3d model-based hand tracking using stochastic direct search method. In *In Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, page 693, 2004.

[Lie05] C.C. Lien. A scalable model-based hand posture analysis system. *MVA*, 16(3):157–169, May 2005.

[LJ07] Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):286–299, 2007.

[LMWY07] L.J. Latecki, V. Megalooikonomou, Q.A. Wang, and D. Yu. An elastic partial shape matching technique. *Pattern Recognition*, 40(11):3069–3080, November 2007.

[Mar82] D. Marr. *Vision: a computational investigation into the human representation and processing of visual information.* W. H. Freeman, San Francisco, 1982.

[May79] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering.* Academic Press, Inc, 1979.

[MTHC03] Ivana Mikić, Mohan Trivedi, Edward Hunter, and Pamela Cosman. Human body model acquisition and tracking using voxel data. *IJCV*, 53:2003, 2003.

[NSMO96] Kenichi Nirei, Hideo Saito, Masaaki Mochimaru, and S. Ozawa. Human hand tracking from binocular image sequences. In *22th International Conference on Industrial Electronics, Control, and Instrumentation*, pages 297 – 302, August 1996.

[OH99a] Hocine Ouhaddi and Patrick Horain. 3d hand gesture tracking by model registration. In *Proc. IWSNHC3DI99*, pages 70–73, 1999.

[OH99b] Hocine Ouhaddi and Patrick Horain. 3d hand gesture tracking by model registration. In *Proc. IWSNHC3DI99*, pages 70–73, 1999.

[Pet] E. Petrakis. Shape datasets and evaluation of shape matching methods for image retrieval. http://www.intelligence.tuc.gr/ petrakis.

[QO01] Delamarre Quentin and Faugeras Olivier. 3d articulated models and multiview tracking with physical forces. *Comput. Vis. Image Underst.*, 81(3):328–357, 2001.

[RASS01] Rmer Rosales, Vassilis Athitsos, Leonid Sigal, and Stan Sclaroff. 3d hand pose reconstruction using specialized mappings. In *In ICCV*, pages 378–385, 2001.

[RK94] J. Rehg and T. Kanade. Visual tracking of high dof articulated structures: An application to human hand tracking. In *Proceedings of the 3rd European Conference on Computer Vision (ECCV '94)*, volume II, pages 35–46, May 1994.

[RP92] Brown R.G. and Hwang P.Y.C. *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley & Sons, New York, second edition, 1992.

[Ruc96] William Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[SBF00] Hedvig Sidenbladh, Michael J. Black, and David J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *In European Conference on Computer Vision*, pages 702–718, 2000.

[SBR00] Josephine Sullivan, Andrew Blake, and Jens Rittscher. Statistical foreground modelling for object localisation. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 307–323, London, UK, 2000. Springer-Verlag.

[SFC07] Frank R. Schmidt, Dirk Farin, and Daniel Cremers. Fast matching of planar shapes in sub-cubic runtime. In *ICCV*, pages 1–6. IEEE, 2007.

[SH06] I. Sutskever and G. E. Hinton. Learning multilevel distributed representations for high-dimensional sequences. In *Tech. Rep. UTML TR 2006-003, University of Toronto*, 2006.

[Shi01] N. Shimada. Real-time 3d hand posture estimation based on 2-d appearance retrieval using monocular camera. In *RATFG'01*, 2001.

[SLSO03] Dimitris Metaxas Shan Lu, Dimitris Samaras, and John Oliensis. Using multiple cues for hand tracking and model refinement. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages II: 443–450, 2003.

[SMC01] B. Stenger, P. R. S. Mendonça, and R. Cipolla. Model based 3D tracking of an articulated hand. In *Proc. CVPR*, volume II, pages 310–315, Kauai, HI, December 2001.

[SMFW04] Erik B. Sudderth, Michael I. Mandel, William T. Freeman, and Alan S. Willsky. Visual hand tracking using nonparametric belief propagation. *Computer Vision and Pattern Recognition Workshop*, 12:189, 2004.

[SS00] Leonid Sigal and Stan Sclaroff. Estimation and prediction of evolving color distributions for skin segmentation under varying illumination. In *In CVPR*, pages 152–159, 2000.

[SSKM98] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura. Hand gesture estimation and model refinement using monocular camera - ambiguity limitation by inequality constraints. *Automatic Face and Gesture Recognition, IEEE International Conference on*, 0:268, 1998.

[ST03] Cristian Sminchisescu and Bill Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 22:2003, 2003.

[STTC04] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Hand pose estimation using hierarchical detection. In *in Intl. Workshop on Human-Computer Interaction*, pages 102–112. Springer, 2004.

[THR06] G. Taylor, G. Hinton, and S. Roweis. Modeling human motion using binary latent variables. *Neural Information Processing Systems (NIPS'06)*, 19:1345–1352, 2006.

[TM91] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.

[TSTC03] A. Thayananthan, B. Stenger, P.H.S. Torr, and R. Cipolla. Learning a kinematic prior for tree-based filtering. In *BMVC03*, 2003.

[TvdM01] J. Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1449–1453, 2001.

[WDF00] E. Wan, A. Doucet, and N. De Freitas. The unscented particle filter, 2000.

[WH00] Ying Wu and Thomas S. Huang. View-independent recognition of hand postures. In *In CVPR*, pages 88–94, 2000.

[WH01] Ying Wu and T.S. Huang. Hand modeling, analysis and recognition. *Signal Processing Magazine, IEEE*, 18(3):51–60, May 2001.

[WM00] Eric A. Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control*, pages 153–158, 2000.

[WTYY09] Angus Wu, P.W.M. Tsang, T.Y.F. Yuen, and L.F. Yeung. Affine invariant object shape matching using genetic algorithm with multi-parent orthogonal recombination and migrant principle. *Applied Soft Computing*, 9(1):282 – 289, 2009.

[YLW97] Jie Yang, Weier Lu, and Alex Waibel. Skin-color modeling and adaptation. In *ACCV '98: Proceedings of the Third Asian Conference on Computer Vision-Volume II*, pages 687–694, London, UK, 1997. Springer-Verlag.

[ZBA08] X. Zamboulis, H. Baltzakis, and A. Argyros. Vision-based hand gesture recognition for human-computer interaction. In C. Stefanides, editor, *The Universal Access Handbook*. Lawrence Erlbaum Associates, Inc. (LEA), 2008.

[ZH03] Hanning Zhou and Thomas S. Huang. Tracking articulated hand motion with eigen dynamics analysis. *Computer Vision, IEEE International Conference on*, 2:1102, 2003.

Appendix A

Attached papers

[A] 'Markerless Human Motion Tracking with a Flexible Model and Appearance Learning', F. Hecht, P. Azad, R. Dillmann, IEEE ICRA 2009 (accepted)

Markerless Human Motion Tracking with a Flexible Model and Appearance Learning

Florian Hecht, Pedram Azad and Rüdiger Dillmann

***Abstract*— A new approach to the 3D human motion tracking problem is proposed, which combines several particle filters with a physical simulation of a flexible body model. The flexible body model allows the partitioning of the state space of the human model into much smaller subsets, while finding a solution considering all the partial results of the particle filters. The flexible model also creates the necessary interaction between the different particle filters and allows effective semi-hierarchical tracking of the human body. The physical simulation does not require inverse kinematics calculations and is hence fast and easy to implement. Furthermore the system also builds an appearance model on-the-fly which allows it to work without a foreground segmentation. The system is able to start tracking automatically with a convenient initialization procedure. The implementation runs with 10 Hz on a regular PC using a stereo camera and is hence suitable for Human-Robot Interaction applications.**

I. INTRODUCTION

Finding and tracking the posture of a person over time is fundamental to several applications. It is used extensively in the animation industry to capture the performance of actors for films and computer games. The pose of a person is also very important in Human-Robot Interaction (HRI). When humans and robots interact, it is expected that the robots can understand the human body language, that is, they should be able to recognize certain actions such as waving or pointing. Also to teach a robot new actions it would be very helpful, if the actions could be taught by demonstration, where the robot learns the specific motions by observing the human performing them. To do all that the robot needs to have a notion of the body posture of the persons it is interacting with.

Motion capture applications in films and games use a large number of high-speed cameras in a studio environment to capture the performance of an actor in a tight suit with carefully placed markers. This expensive and complicated setup allows very precise measurements of the performed motions. For the application on a robot we cannot require the person being tracked to wear special clothing with markers or sensors. The robot also has only a limited view with a mono or stereo-camera and not a set of conveniently placed cameras around the person. With these restrictions it becomes much harder to track the movements of a person.

Institute for Technical Informatics, University of Karlsruhe, Germany

F. Hecht is a CS Diplom student, at the University of Karlsruhe florian.hecht@ira.uka.de

P. Azad is with the Institute for Technical Informatics, University of Karlsruhe azad@ira.uka.de

R. Dillmann is Professor of the IAIM chair at the Institute of Technical Informatics, University of Karlsruhe dillmann@ira.uka.de

It would be beneficial to have sophisticated camera-based tracking systems that have only few requirements. Such a system should work with few regular cameras and should not require any preparations, neither of the person to be tracked nor of the environment. Such a system would be a lot cheaper and would not be restricted to certain locations. A person could work with such a system without special preparation or training. Ideally, such systems should have good performance in the following criteria at the same time: Robustness, precision and computational effort.

A. Previous Work

There is a plethora of different approaches to human motion capture that differ in the sensors used (accelerometers, cameras, depth cameras), the number of sensors, the model that is constructed (2D, 3D, with appearance, etc.) and the underlying algorithms. Since the proposed method uses particle filters and a 3D model with a single perspective view of the person, we will focus on previous work in these areas.

The use of short-baseline stereo cameras gives some additional 3D information about the scene [1], [8], [9] compared to a single camera, but has the same difficulties as monocular systems [3], [15] as only one fundamental perspective is available. These systems have to handle occlusions and the fact that motion in the depth direction is not directly visible.

There are several different approaches to determine the posture from a given set of images. For monocular 2D-3D registration, where the actual 3D pose is determined from a single view, a precise model and optimization algorithms are use to find the correct pose, as done in [15]. Systems that use stereo cameras usually extract the 3D location of key body parts like the head and hands and use other methods to solve the position of elbows and other body parts [1], [8]. Algorithms that use more cameras can extract 3D information from the different views and then fit an articulated model to this 3D data [9], [2], [4], [16], [11]. A different approach is to project a model into each view and determine the change of the model from each view either by optimization [3], [7], [15], filtering or sampling.

Particle filters are used in several algorithms [1], [5], [6], [8], [12], [14]. The biggest problem with human motion tracking with particle filters is the exponential growth of the needed number of particles with the increase in dimensions of the search space. For most whole body human models in 3D we have about 30 DoF, which is infeasible to solve with a regular particle filter. One of the tasks to solve when using particle filtering for human motion capture is to deal with

this high dimensionality. Several different approaches have been proposed:

One approach is to split the search space into smaller search spaces in combination with hierarchical search or the localization of certain body parts by other means, like the head, hands or the the dominant axis of the torso [12]. One big criticism of these approaches is that, say the two arms, are treated independently from each other, where in fact the position of the one-side influences the position of the other. In [6], an automatic partitioning scheme is proposed that reduces the needed number of particles while still creating the needed interactions.

In [14], a strong motion model is used to predict the state of the skeleton in the next frame. This learned motion model reduces the number of needed particles since the particles will be relatively close to the actual position. This system is limited to tracking one type of motion at a time (walking in that paper) and is therefore not universally applicable, but showed that a good prediction can significantly improve tracking results.

Another approach to deal with the high dimensional search space is the *annealed particle filter* as proposed in [5]. Similar to the optimization technique *simulated annealing*, several filtering runs are performed, with increasing detail in the weighting function. This guides the particle set to coarse peaks first, and then optimizes the result with finer details. With this approach the method can find the correct optimum with a reduced number of particles. The annealed particle filter is analyzed for application in non-studio-like environments in [13] and was found to depend on relatively noise free measurements for reliable results. In [6], extensions to this method are proposed including decreased noise that is dependent on the variance of each state space variable. The purpose of this is to focus attention to variables which are not yet determined precisely and to prevent losing an already precise localization of variables due to added noise.

Another algorithm is presented in [15], where *covariance scaled sampling* is proposed, which is a generalization of the variance-based noise extension of the annealed particle filter. The state space distribution is represented as a mixture of Gaussians. Samples are generated from each Gaussian, with a distribution that captures the dimensions with the most variance computed by eigen decomposition of the covariance matrix.

None of [14], [5], [15] achieve real-time results and are thus not applicable for use on a robot.

B. Outline

This paper presents a new method that was developed for the purpose of HRI. The focus here is on performance and robustness under certain conditions, like a short-baseline stereo camera and a frame rate of at least 10 Hz on a regular PC. The proposed method is not restricted to the application on a robot, but can be used in multi-camera scenarios with increased precision. The flexible model that represents the human pose is introduced in Section II. The integration of the particle filters into the complete tracking system is described in Section III, which is followed by experimental results in Section IV. Concluding remarks and ideas for future work can be found in Section V.

II. Flexible Model

The flexible model used in this application is a mass-spring system as illustrated in Fig. 1. Unlike a classical kinematic model, which uses a hierarchy of joints, we have a set of point masses, which are connected through springs. The springs represent abstract bones for the extremities, but are also used to construct a two-part torso, which is not completely rigid. The state of the model consists of the positions of the 16 points, which means the model has $16 \cdot 3 = 48$ DoF, which is more than the usual 30 DoF for a kinematic model, but still less than the $11 \cdot 6 = 66$ DoF for a model consisting of connected individual body parts[1].

Another thing to note is that the model does not represent rotations explicitly. Some rotations can be recovered from the positions of the mass-points, but the rotations around the arms' axes for instance cannot be derived. If the 24 springs in the system where completely rigid, then the number of DoF of the complete system would be $48-24 = 24$. But the springs are not 100% rigid and hence give the model more flexibility. Parametrizing the human model this way seems a bit of a waste, but enables a locality of change, since movements in a certain point can be implemented simply by moving that mass-point and do not require an inverse kinematics calculation. Note that the 48 DoF are estimated in a semi-hierarchical way as will be explained in Section III. Also this model allows splitting of the state space, while still retaining interaction between the subparts, which enables a good estimate for one part to fix the bad estimate for another part, thus greatly improving the robustness of the whole estimate.

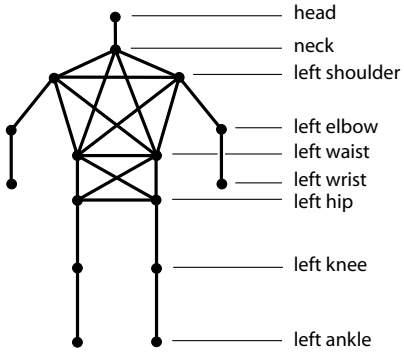


Fig. 1. Mass-Spring model of a human showing the mass-points and the springs.

The model is enhanced with additional constraints to limit the angular motions of the limbs. Since rotations are not modeled, several coordinate frames are derived from sets of the three positions of mass-points. In these frames the

[1]This assumes two parts per extremity, two for the torso and one for the head, thus 11 parts, where each part has 6 DoF as a rigid body.

rotational limits are defined. Further constraints prevent the self-penetration of the arms and the torso (by using an anisotropically scaled cylinder collision), as well as cross-over situations through distance constraints between the legs. This limits the number of states the model can get into further. The model is suspended in the air with gravity dangling from the position of the head.

The mass-spring system is solved in a Verlet framework [10], [17], which makes the implementation very simple and the physics simulation very stable. The system of constraints is solved by iteratively applying them individually several times, similar to Gauss-Seidel iterations for linear systems. The Verlet integration step is

$$\boldsymbol{x}_{t+1} = 2\,\boldsymbol{x}_t - \boldsymbol{x}_{t-1} + \boldsymbol{a}_t \Delta t^2,$$

where $\boldsymbol{x}_t$ denotes the position of a point at time t, $\boldsymbol{a}_t$ is the acceleration which is computed from the accumulated forces during a frame, and Δt is the time step. The significant difference to the regular Euler integration is the absence of the velocity, which is implicitly calculated by the difference to the previous position. This implicit velocity makes the solution stable, since position and velocity cannot get out of sync. It also simplifies the implementation of various constraints, since the current position is simply projected to a valid state without having to calculate a new velocity, which is implicitly handled by the previous position.

The weights of the mass points are chosen so that the torso points are heavier and the points of the extremities are lighter. The head has an infinite weight, which makes it immovable, i.e it is only moved by assigning the position measured by the head tracker. The strengths of the springs have been chosen in a way to reflect the movability of the mass-points with regard to their corresponding joints in humans, which gives the shoulder points a certain amount of play. The springs also allow the model to adapt to a limited amount of size change of the tracked person.

The state of the mass-spring model is used to calculate the state of a cylinder model, which consists of two cylinders for each extremity, two for the torso and one cylinder for the head. This model is used for projections in the measurement model of the particle filters, as well as for the occlusion model, and for visualizations. See also Fig. 2.

The various parameters of this model have been determined empirically and may not be realistic with regard to a real human, but the simulation produces quite realistic results and enables good predictions. This model can be used for people of a similar stature, but the lengths and diameters would have to adapted for persons of different stature (depending on the body height). The constraints and spring coefficients can stay the same.

III. Tracking with a Flexible Model

The tracking of the body takes place in a two-step semi-hierarchal way: The head is tracked by a special particle filter-based face tracker, using skin color segmentation. This is the only place where the stereo-camera is actually required, as the rest of the system could be using only one of the images. Unfortunately, the 3D estimate of the particle filter is not precise enough for a robust 3D localization of the head. Therefore we use stereo correspondences with features on the face to get a better depth estimate. The head point of the mass-spring system is moved to the head position and the rest of the points are moved 80% of the translation that the head point moved.

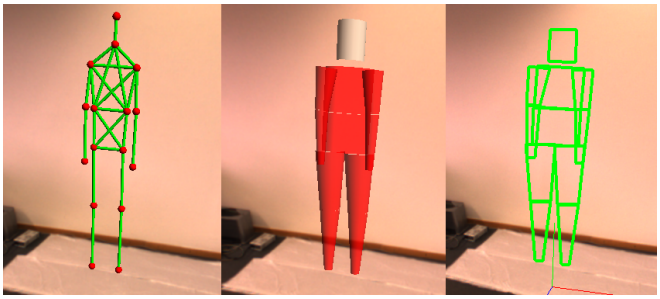


Fig. 2. Overlays of the model over the rectified image. From left to right: The mass-spring system, the cylinder model and the simplified cylinder projection.

When the model is moved with the head, the rest of the body pose is determined by tracking the extremities with particle filters and solving the mass-spring system, which creates an implicit solution for the torso. The approach to deal with the *curse of dimensionality* is to split the problem into smaller sub parts and to integrate the partial solutions into an overall solution. However, one cannot simply split the state space into smaller parts without considering the interactions between the sub-parts. The needed interaction is achieved by the mass-spring system, which influences and is influenced by the four particle filters of the extremities, as shown in Fig. 3. The particle filter for a limb has a formal state space with $3{\cdot}3 = 9$ dimensions, combining the positions of the three mass-points that define the state of that limb, e.g. shoulder, elbow, and wrist position for an arm filter. However, for the same reason as before, the actual DoF is lower, since with rigid springs only 4 DoF would be used for each extremity. So with not completely rigid springs, as was used here, we have more than 4 DoF but much less than the formal 9 DoF.

The estimation of the particle filter for each extremity is put into the mass-spring system by overwriting the position of the points that are filtered. The mass m_i of the i-th point is changed to reflect the confidence in the estimate of the particle filter, which is based on the variance σ_i of the i-th point in the particle filter. While unrealistic in a physical simulation sense, it works very well as way to integrate the 4 weighted results into the complete state estimate.

$$m_i = m_{base} + m_{variable} e^{s\sigma_i},$$

with m_{base} being the base mass and a variable part $m_{variable}$, and $s < 0$. It hence gives more weight to points it is confident about and less weight to points for which the estimate is not reliable. The solution of the mass-spring system will then determine a state which reflects these confidence measures. The four extremity particle filters are run independently from each other and use the current state of the mass-spring system

in their motion models. The estimates are put into the mass-spring system at the same time.

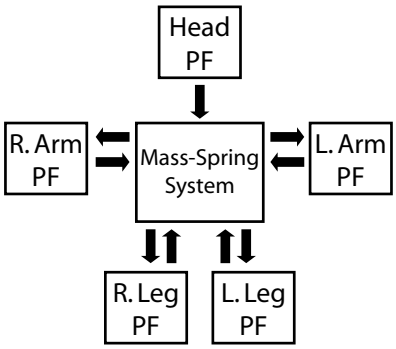


Fig. 3. The particle filter (PF) of the head influences the mass-spring system, while there is a two-way coupling of the extremity PFs and the physics simulation.

A. Motion Model

Each particle of the particle filters consists of three points $(\boldsymbol{p}_1^t, \boldsymbol{p}_2^t, \boldsymbol{p}_3^t)$, representing an alternative state to the one in the mass-spring system at time t. The points are first moved with a constant velocity model

$$\bar{\boldsymbol{p}}_i^{t+1} = \boldsymbol{p}_i^t + \eta(\boldsymbol{p}_i^t - \boldsymbol{p}_i^{t-1}),$$

where η is a factor that determines the trust in this constant velocity model (empirically set to $50\% - 80\%$). The previous position is stored for all particles and points. After that, noise is added, which is a mixture of two Gaussian distributions: a process noise and a "depth" noise that is intended to push particles into the direction that is not visible in the camera image to improve the search for the correct state, similar to the use of covariance scaled sampling in [15]. The new position of the point is drawn according to the probability distribution of the new position $p(\boldsymbol{p}_i^{t+1})$, which is modeled as:

$$p(\boldsymbol{p}_i^{t+1}) = (1-\alpha)\,\mathcal{N}(\bar{\boldsymbol{p}}_i^{t+1}, \sigma_i \mathbf{I}) + \alpha\,\mathcal{N}(\bar{\boldsymbol{p}}_i^{t+1}, \boldsymbol{\Sigma}_i),$$

where $\bar{\boldsymbol{p}}_i^{t+1}$ is the prediction from the constant velocity model, $\mathbf{I}$ is the identity matrix and σ_i the variance of the first normal distribution $\mathcal{N}(\bar{\boldsymbol{p}}_i^{t+1}, \sigma_i \mathbf{I})$. The second normal distribution has a covariance matrix instead of a uniform variance. The mixture weight α determines how many particles are drawn in average by the second distribution. We use a value of $\alpha = 0.25$. The first variance σ_i, which represents the unknown motion, consists of three components: a base noise level, one that depends on the variance of the point positions[2], and one that depends on the motion in the image at the projected position of the particle. The amount of motion is sampled from a motion image for each of the three points of the particle. For this purpose the motion image is a down-sampled and heavily blurred thresholded difference image between the previous and current image. If more computational power is available, this could be improved by the use of optical flow.

[2]This is the same variance that is used to determine the mass of the points in the mass-spring system, that are estimated by the particle filters.

The second part of the noise distribution is the depth component with the covariance matrix $\boldsymbol{\Sigma}_i$. This matrix is constructed as follows:

$$\boldsymbol{\Sigma}_i = \mathbf{T}\,\mathbf{R} \begin{pmatrix} \sigma_x & & \\ & \sigma_y & \\ & & \sigma_z \end{pmatrix} \mathbf{R}^T\,\mathbf{T}^T$$

The diagonal covariance matrix that expresses the different scale factors in a frame where the z-axis is the depth direction. This covariance matrix is rotated into camera coordinates by $\mathbf{R}$ and then into world coordinates by $\mathbf{T}$. The transformation into world coordinates $\mathbf{T}$ comes from the camera projection. The rotation matrix $\mathbf{R}$ is constructed for each point, as a coordinate frame which has the z-axis in the depth direction. The scale factors are chosen $\sigma_x = \sigma_y = \epsilon$ to be very small and the σ_z is chosen differently for each point of a particle, to give the end of the extremities, wrists and ankles, more depth noise than the other points[3]. Since the noise in the x- and y-directions in the rotated frame are small and since the construction has to be done three times for every particle, an approximation to this Gaussian distribution is used, which is simply Gaussian noise along the depth direction.

At this point we have combined a constant velocity model with a model for the assumed and estimated random distribution of the particles. But since we estimate these values independently for the three points of the particle, the particles might have reached positions which do not correspond to physically or anatomically correct states of the extremity. To force the particles back to valid states as defined by the constraint system of the body model, the three points are put into the mass-spring system and the active constraints that affect these points are applied to them for several iterations. The other points in the mass-spring system are assigned an infinite mass and do not move (See Fig. 4). This is the other direction of the coupling between the mass-spring system and the particle filters, since the state of the mass-spring system influences the state the particles can get into.

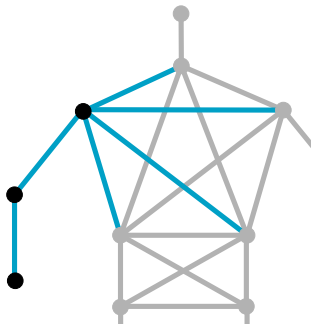


Fig. 4. Constrains in the particle filter of the right arm — points from the particle are black, the active constraints are light blue and the disabled constraints and mass points are gray. Only the spring constraints are displayed.

[3]Empirically set to $\sigma_z = 1.0m/0.5m/0.15m$ for wrists/elbows/ shoulders. Same for the legs.

B. Measurement Model

The measurement model is using the projection of cylinders into the images to calculate two scores per particle, an edge score and a surface score. The edge score is calculated by comparing the projected long edges of the cylinders against an edge image (like in [5]), which is a blurred version of the combination of a thresholded difference image and a thresholded gradient image. This combination has the benefit of having a higher score for moving edges that are assumed to belong to the moving person. The second benefit is that this scheme does not depend on a foreground segmentation. The drawbacks are that it is not as clean as silhouette edges from a segmented image and that it also features edges that do not belong to the person, which is compensated to an extent through the moving edges.

The surface score is calculated by sampling the surface of the simplified cylinder projection with a regular grid, as illustrated in Fig. 5. One possibility for calculating the score is counting the number of foreground samples, as it is coming from a foreground segmentation. But this requires a good segmentation, which is hardly possible with motion segmentation. Therefore we use an appearance model instead, where each sample on the grid gets a reference color, which is compared to the actual color in the current image (sum of absolute differences). The color model for each body part (left/right upper arm, left/right lower arm, etc.) consists of ten reference colors along the cylinder for that part and assumes that the color is constant on a ring around the surface of the limb. The colors are linearly interpolated to get the reference color for a sample, by using the normalized distance from the base of the cylinder as an index.

The color model is learned during the first couple of frames, with an running average scheme. The color in a frame is sampled on a regular grid and the points that are classified as foreground (from the segmented image) are averaged together per ring of the cylinder and learned over several frames. Once the learning is complete, a foreground segmentation is no longer needed and an active head could start moving again.

The whole measurement model makes use of an occlusion model, which is based on the current state of the mass-spring system. The occlusion model consists of the convex quadrilaterals of the simplified projections of the cylinders. The quadrilaterals have a minimum and maximum depth and are indexed through a spatial grid, which makes the query whether or not a sample point is visible very fast. All samples, for the edge and the surface score, are tested and receive special default scores if a sample is occluded or is outside of the view. These default values have to be chosen carefully to prevent a preference for occluded or unoccluded states.

C. Initialization

The system is initialized by taking the first frame as a background model, which is used for the background subtraction. The head tracker is looking for a suitable skin blob to track as the head. When a head is found, the model is moved to the measured position of the head and the physics simulation starts — but at this time without the particle filters, which results in a relaxed state of the model with arms and legs simply dangling. The configuration from the physical simulation is run through the measurement model and the edge score is used to determine when a match between the relaxed state and the image occurs. When a match is found an initial appearance model is captured and the tracking with the filters starts. During the next frames the color model is updated as already explained. The learning rate depends on the local confidence values, which are interpolated along the extremities. When the learning stops, the appearance model is fixed and the foreground segmentation is no longer needed, allowing the camera to be moved.

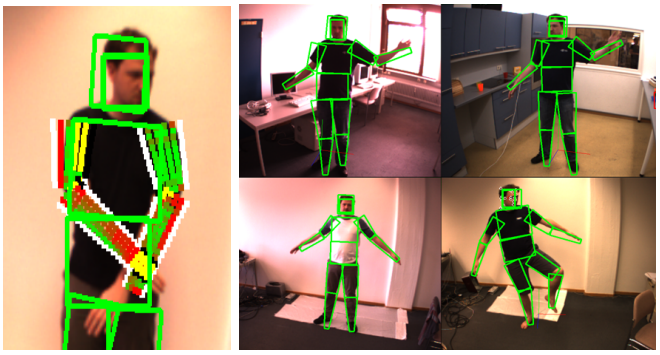


Fig. 5. The left image is showing the sampling grid for the arms displayed on the mean estimates. The surface samples show the quality of the match in a red to green scale. Occluded sample points are yellow. Edge samples are white or black (occluded). The right image shows different backgrounds and clothes.

Note that this initialization procedure assumes that the person will get into a pose that is close to the relaxed state at the beginning.

IV. Experimental Results

A. System Parameters

The single threaded tracking application runs on a Pentium 4 3.2 GHz CPU with 10 Hz. The calibrated stereo camera consists of two Dragonfly cameras by Point Grey Research Inc. Image processing is performed on 640×480 color images. The used lenses are 4 mm M12 micro lenses, which have a significant radial distortion, but provide a big enough field of view, to see the complete human at a distance of 2–3 m. The software was built using the *Integrating Vision Toolkit*[4] which offers a clean camera abstraction and a generic camera model.

The particle filters used 200 particles per extremity and 100 particles for the head. The processing time is ≈35 ms for the image processing and ≈50 ms for the filtering, where the motion model is consuming about half the processing power.

B. Range of Motions

The tracking of the arms is able to follow the motions of a single arm very robustly, even through complicated

[4]http://ivt.sourceforge.net

and ambiguous situations. This is achieved through the combination of the angular constraints, the occlusion model and the motion model, which guide the estimate through ambiguous situations. The tracking is even able to detect if parts of a limb are hidden behind the body or the head, due to the occlusion model. The interactions of the two arms are tracked as long as both arms are visible to a certain extent, cross-over situations are successfully tracked due to the occlusion model. Fig. 6 shows tracked arms positions. Furthermore folded arms are captured, but the estimate does not reflect which one of the arms is visible, since it yields fluctuating depth estimates. When opening the folded arms, the tracking can get confused, but usually recovers through the opening motion.

The legs are tracked individually very well, including the detection of knee bends and folded up calves, which works also when standing on one leg. As with the arms the interaction of the two legs is more complicated to track. As they are closer to another than the arms, they have a tendency to both capture the same leg. To compensate that, a plane separation constraint forces the points from the two legs away from each other similar to the self penetration constraint, which improves the tracking, but makes it impossible to capture natural cross-over situations. When a person turns side ways, one leg is completely occluding the other, which can cause significant confusion for the filters.

As the tracking depends strongly on the localization of the head, more precisely the face, the person cannot turn away more than 90° from the camera. The suspension of the head together with the simulation of gravity implies that the person has to be standing. The waist area is also relatively stiff at the moment, and thus does not model the flexibility of a human. The constraint framework allows easy implementation of external constraints, like collisions with objects or furniture, which should allow the integration of specific knowledge into sitting and other special motions, but this has not been explored yet.

C. Precision

To measure the precision of the estimate one needs ground truth to compare to. Unfortunately, a motion capture system was not available, so the tracking of an object in the right hand, which was localized with the stereo camera, is compared to the estimate of the right wrist point of the human motion tracking application. This is a substitute for better ground truth and is not very precise, but confirms that most of the error – as one would expect – is in the depth direction and that the correct estimation of bent limbs needs a certain amount of foreshortening of the limbs before the model will capture it. Comparing the 3D renderings of the estimates to images gives enough insight into how good the estimate is and where the precision deteriorates. See also the accompanying video. The focus of the proposed system is not on precision, but on speed and robustness, since the estimated body posture is used for the recognition of states and gestures and not the capturing of performances for animations.

Fig. 6. A series of pictures from a tracking session. The left side shows the simplified cylinder projection, while the right side shows a 3D overlay over the rectified image. The 3D model is transparent and uses the color from the appearance model. The head and torso color are derived from the left arm.

D. Tracking Failures

Unfortunately, the system has problems when the person is seen from the side, since two limbs are not visible. The estimate for the parts that are not visible will deteriorate rapidly which will affect the estimates of the other limbs, since they are connected through the flexible model. Turning back to face the camera again, does usually not recover the correct estimate.

If two limbs are close to each other and one is moving while the other is still, the moving limb can affect the estimate of the still limb, by pulling the estimate away, since moving edges yield a higher score than still gradients. Particles from the still limb will follow the moving edges.

Naturally, motions which are too fast for the 10 Hz processing rate of the system cannot be tracked

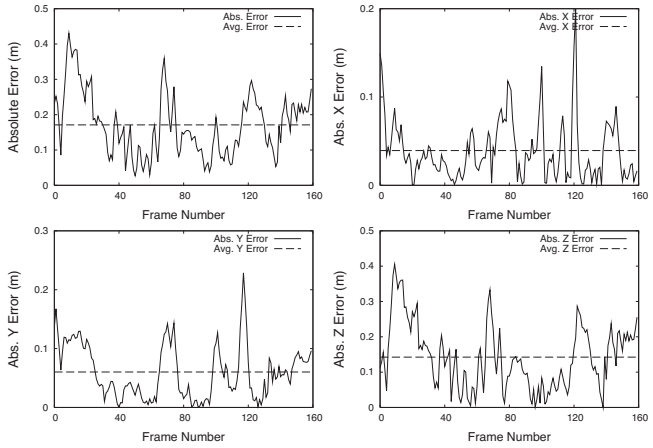


Fig. 7. Plot of the absolute error between the tracked object position and the wrist estimate of the human motion tracking, showing that the biggest error is in the depth direction, which also correlates the most with the overall error. Note the different scales!

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

We have presented a new way to approach the human motion tracking problem, by coupling a physical mass-spring simulation with probabilistic particle filters to get an estimate for the pose of a human. Due to the reduced dimensionality of the particle filters achieved by semi-hierarchical decomposition, the system can run with 10 Hz on a regular PC, which allows real-time tracking of the motions of a person. The flexible model enables the tracking system to adapt to the person being tracked and creates a natural interaction between different parts of the body model.

A simple appearance model with reference colors is learned on-the-fly and allows the system to work without background subtraction, which enables the camera to be moved after the appearance model has been learned. The use of a strong motion model, which takes motion cues from the images and enforces the constraints of the system, allows tracking through ambiguous situations. Together with the occlusion model this creates a robust system that is able to track through complicated situations. All this is done with a stereo camera, where the stereo is only used for the head tracking. The filtering is basically monocular[5].

B. Future Works

Particle filtering is very well-suited for a parallel implementation. The speedup from such an implementation should be substantial and would increase the range of motions that can be tracked, due to the current speed limit.

The current mass-spring system is flat and has been chosen because it is the simplest model for the purpose, which results in few springs. A more elaborate mass-spring skeleton could enable more realistic deformations for the waist area and more freedom for the hip.

The current system uses only a stereo-camera and is therefore suitable for application on a humanoid robot head. However, we plan to evaluate how a system with several cameras can benefit from the proposed flexible body model.

[5]We actually alternate between the two images, but the benefits are minimal.

VI. ACKNOWLEDGMENTS

The work described in this paper was partially conducted within the EU Cognitive Systems projects GRASP (FP7-215821) and PACO-PLUS (FP6-027657) and the German Humanoid Research project SFB588 funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft).

REFERENCES

[1] P. Azad, A. Ude, T. Asfour and R. Dillmann, "Stereo-based Markerless Human Motion Capture for Humanoid Robot Systems", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3951–3956, 2007

[2] F. Caillette and T. Howard, "Real-Time Markerless Human Body Tracking Using Colored Voxels and 3-D Blobs", *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 266–267, 2004

[3] Y. Chen, J. Lee, R. Parent and R. Machiraju, "Markerless Monocular Motion Capture Using Image Features and Physical Constraints", *Computer Graphics International*, June, pp. 36– 43, 2005.

[4] G. K. M. Cheung, S. Baker and T. Kanade, "Shape-From-Silhouette of Articulated Objects and its Use for Human Body Kinematics Estimation and Motion Capture", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 77–84, 2003

[5] J. Deutscher, A. Blake and I. Reid, "Articulated Body Motion Capture by Annealed Particle Filtering", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 126–133, 2000

[6] J. Deutscher, A. Davidson and I. Reid, "Automatic Partitioning of High Dimensional Search Spaces associated with Articulated Body Motion Capture", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 669–676, 2001

[7] T. Drummond and R. Cippola, "Real-time Tracking of Highly Articulated Structures in the Presence of Noisy Measurements", *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pp. 315–320, 2001

[8] M. Fontmarty, F. Lerasle and P. Danès, "Data Fusion within a modified Annealed Particle Filter dedicated to Human Motion Capture", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3391–3396, 2007

[9] P. Fua, A. Gruen, N. D' Apuzzo and R. Plänkers, "Markerless Full Body Shape and Motion Capture From Video Sequences", *Symposium on Close Range Imaging, International Society for Photogrammetry and Remote Sensing*, Corfu, Greece, 2002

[10] T. Jakobsen, "Advanced Character Physics", *Game Developer Conference 2001*, www.teknikus.dk/tj/gdc2001.htm, link verified June, 26th, 2008

[11] S. Knoop, S. Vacek, R. Dillmann, "Sensor Fusion for 3D Human Body Tracking with an Articulated 3D Body Model", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1686–1691, 2006

[12] M. W. Lee, I. Cohen and S. K. Jung, "Particle Filter with Analytical Inference for Human Body Tracking", *Proceedings of the Workshop on Motion and Video Computing (MOTION)*, p. 159ff, 2002

[13] P. Peursum, "On the Behaviour of Body Tracking with the Annealed Particle Filter in Realistic Conditions", *Technical Report*, November, 2006

[14] H. Sidenbladh, M. J. Black and D. J. Fleet, "Stochastic Tracking of 3D Human Figures Using 2D Image Motion", *Proceedings of the 6th European Conference on Computer Vision*, Part II, pp. 702–718, 2000

[15] C. Sminchisescu and Bill Triggs, "Estimating Articulated Human Motion With Covariance Scaled Sampling", *International Journal of Robotics Research*, Vol. 22, No.6, pp. 371–393, 2003

[16] A. Sundaresan and R. Chellappa, "Markerless Motion Capture using Multiple Cameras", *Proceedings of the Computer Vision for Interactive and Intelligent Environment (CVIIE)*, pp. 15–26, 2005

[17] L. Verlet, "Computer Experiments on Classical Fluids", *PhysRev. Vol. 159*, No. 98, July 1967