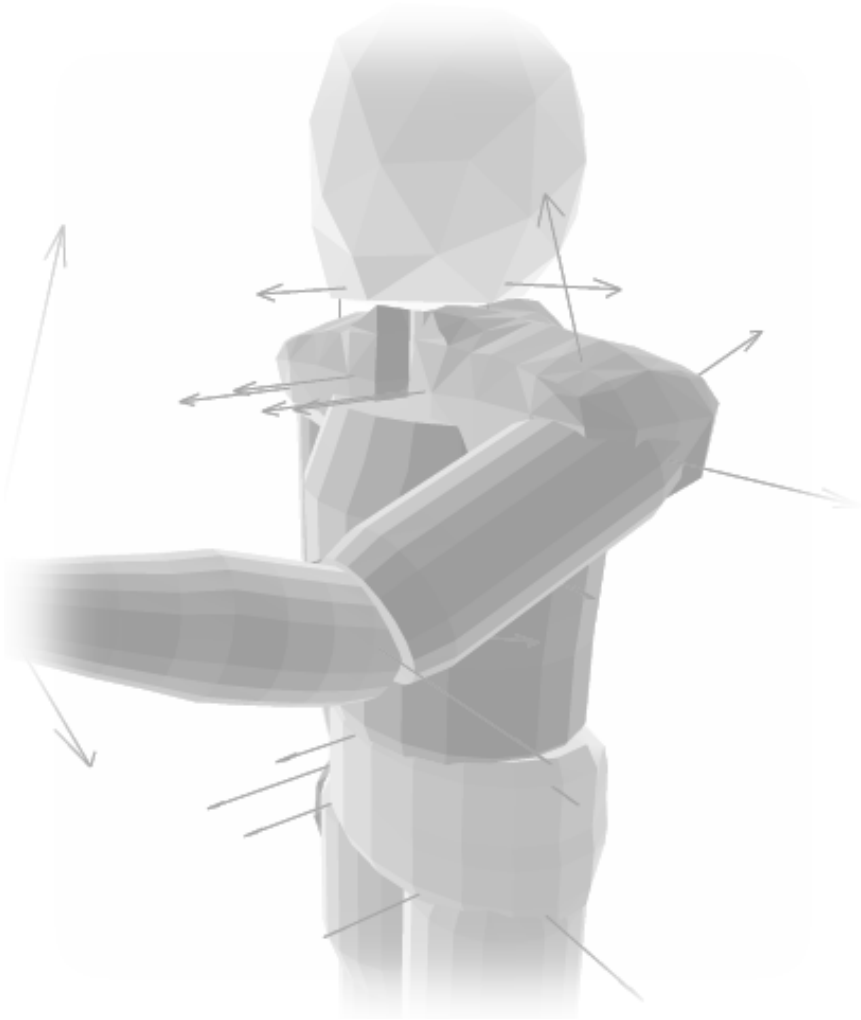
Marker-Free Human Motion Capture in Dynamic Cluttered Environments from a Single View-Point

PhD thesis

by Daniel Grest



Kiel, Germany, 2007

Marker-Free Human Motion Capture in Dynamic Cluttered Environments from a Single View-Point

Dissertation

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät der Christian-Albrechts-Universität zu Kiel

Dipl.-Ing. Daniel Grest

Kiel
2007

1. Gutachter	Prof. Dr.-Ing. Reinhard Koch
2. Gutachter	Prof. Dr.-Ing. Marcus Magnor
Datum der mündlichen Prüfung	14.Dez. 2007

Abstract

Human Motion Capture is a widely used technique to obtain motion data for animation of virtual characters. Commercial optical motion capture systems are marker-based. This thesis is about marker-free motion capture.

The pose and motion estimation of an observed person is carried out in an optimization framework for articulated objects. The motion function is formulated with kinematic chains consisting of rotations around arbitrary axes in 3D space. This formulation leads to a Nonlinear Least Squares problem, which is solved with gradient-based methods. With the formulation in this thesis the necessary derivatives can be derived analytically. This speeds up processing and increases accuracy. Different gradient based methods are compared to solve the Nonlinear Least Squares problem, which allows the integration of second order motion derivatives as well.

The pose estimation requires correspondences between known model of the person and observed data. To obtain this model, a new method is developed, which fits a template model to a specific person from 6 posture images taken by a single camera.

Various types of correspondences are integrated in the optimization simultaneously without making approximations to the motion or optimization function, namely 3D-3D correspondences from stereo algorithms and 3D-2D correspondences from image silhouettes and 2D point tracking. Of major importance for the developed methods is the processing time and robustness to cluttered and dynamic background.

Experiments show, that complex motion with 24 degrees of freedom is track-able from a single stereo view until body parts get totally occluded. Further methods are developed to estimate pose from a single camera view with cluttered dynamic background. Similar to other work on 2D-3D pose estimation, correspondences between model and image silhouette of the person are established by analyzing the gray value gradient near the predicted model silhouette. To increase the accuracy of silhouette correspondences, color histograms for each body part are combined with image gradient search.

The combination of 3D depth data and 2D image data is tested with depth data from a PMD camera (Photonic Mixer Device), which measures the depth to scene points by the time of flight of light.

Contents

Chapter 1

Introduction

Human motion capture is the analysis of a person in motion, that is giving rise to some mathematical representation of the observed motion. In this work the representation of motion are joint angles over time or similar data, which allows animation of virtual characters.

In the past various commercial systems for human motion capture and analysis have been developed. All rely on some kind of markers or other equipment attached to the body. Marker-free systems are not commercially available yet and are an intensively studied research topic.

Electromechanical systems, e.g. [88], give very accurate joint angle values directly by use of potentiometers and sliding rods, but are restrictive in their weight and freedom of motion as visible in the left of Figure 1.1 The measuring devices, which are attached to the body can disturb the actor, e.g. if he gets in contact with other objects or persons. Wearing such clearly noticeable equipment will make it difficult to move naturally as desired for example in medical or sport motion analysis.

Electromagnetic systems measure the position and orientation of markers in real-time. Although less cumbersome than Electromechanical systems, the markers are noticeable by the actor, because of their weight and their cable connection to a control station. A recent system [121] comes with a wireless control station, wearable by the actor as a small backpack as shown in the middle of Figure 1.1. However all wires have to be attached to the person in some way, which can be cumbersome and disturbing for the actor. Their main advantage over optical systems is their price and easier setup. A disadvantage in studio setups is their sensitivity to metal in the near vicinity of the markers, which disturbs the measurements. Also the range of possible motions is restricted to an approximately 6 by 6 meter area near the device, which generates the magnetic field (the small black box in the lower left of the image).

Video-based optical systems, e.g. [95, 120], rely on markers, which are attached to the person as visible in the right image of Figure 1.1. These markers are then detected in the images of multiple cameras, which surround the person. Complete calibration of the cameras together with the 2D-marker positions in the image allow triangulation of the marker position in 3D. These 3D-marker positions per frame are the output of the

Figure 1.1: Left: A commercial electromechanical system from metamotion [88] Middle: A commercial electromagnetic system from Ascension Technologies [121] Right: Actor Andy Serkis with reflective infrared markers on his suit. The small white balls in the middle of the black patches are the reflective markers. The actors motion was recorded by several surrounding cameras, which are not visible here.

system. State-of-the-art products achieve up to 1000 frames per second. The 3D positions of markers are then used in a next step to estimate the motion of a virtual model, which has to fit in size to the actor. The main advantage of these systems compared to the former two are the less disturbing markers for the actor, because they are lightweight and almost unnoticeable during action. The motion area can be significantly larger of up to 50 by 50 meters, depending on how well the cameras cover the area.

A popular example of video-based motion capture is the performance of the creature 'Gollum' in the award winning movie 'The Lord of the Rings'. Motions of the actor Andy Serkis have been recorded in a standard studio environment. The flexibility of the reflective markers allowed the actor to roll and creep down a artificial river bed in the movie studio, which would not have been possible with the other two systems.

Although animation of virtual characters for movies, music videos or games is the most popular application of motion capture systems, one can point out the following application areas:

Animation , e.g. in the film and games industry. Commercial systems [95, 120] are marker-based and their output is mainly used for animation of virtual characters. [80, 138, 93, 84]

Surveillance Tracking people's position, crowd behavior [117]or analysis and recognition of events [39]. Captured motion information is usually a floor or image position. A survey is available in [64].

Control applications Human motion acts most often as machine input (e.g. computer mouse). In order to control applications, gestures [99] and hand motion [104]can be an efficient tool for control of applications.

Motion Analysis The goal of the analysis can be detection of abnormal motion [40] or improving the course of motion for athletes [22].

Commercial optical marker-based systems have advantages, but they have also disadvantages:

- Markers can disturb the naturalism of motion.
- Inaccuracies can appear due to manual marker-displacement or a misfit shape of the body model, which does not reflect the observed person's shape accurate enough.
- Deformations like muscle bulges or cloth deformations, can only be captured with a large amount of markers.
- Restrictive capture environments. Illumination has to be controlled, e.g. with artificial infrared lighting. No other infrared reflective objects should be visible in the images.
- Multiple views of surrounding cameras are necessary.

Therefore a lot of research, especially in the last 15 years, is devoted to overcome these limitations. A recent survey [91] gives an impression by citing more than 600 papers from 2000 to 2005.

Some marker-free motion capture systems proved already to be advantageous over marker-based systems as in [97], where a marker-free approach gives more accurate results in gait-analysis.

Other accurate marker-free systems rely on images that allow segmentation of the person in the foreground. Because the person can wear normal clothing, the appearance of the person can be captured together with the motion [123, 124, 77]. The accuracy of such approaches is comparable to marker based systems [111, 40]. However, the segmentation step in Shape-From-Silhouette [23, 90, 128, 73, 5] approaches makes strong restrictions to the capture environment, e.g. homogenous clothing or background, multiple views and controlled lighting.

It is not only desirable to capture motion without markers, but further benefits will result from systems able to capture motion

- with less cameras, ideally from a single view
- without artificial lighting, ideally in natural indoor and outdoor environments
- with dynamic, cluttered and changing background
- in real-time, e.g. more than 25 motion measurements per second

The first aspect is very important in robot-human-interaction [74, 69], because a human-like robot will have only a single view on the person, although it may be equipped with multiple cameras. If the motion of the person can be captured correctly, the estimated motion can act as a basis for understanding and interpreting the motion. The second aspect of changing lighting conditions is important in outdoor scenarios and in interaction environments like the CAVE [28] or the one shown in Figure 1.2 (more details in [53]), where a large amount of light is emitted from displays.

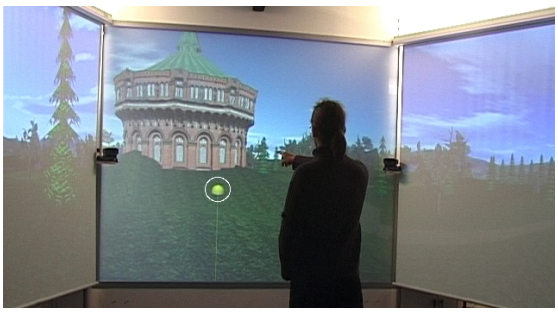


Figure 1.2: An interaction area with three displays. A stereo back projection in the middle and two standard projections left and right.

The third aspect is of general importance for various applications. In sport games the in-game analysis of motion from players will exhibit an image background that is dynamic and cluttered. In most applications, where the environment can not be controlled as in a laboratory, the background will be cluttered and dynamic.

Taking these view further, the ideal motion capture system would be able to capture motion of a person moving in a group with high accuracy from a single view in real-time without markers, e.g. from standard movie footage. Current research is still far away from this goal. However, single aspects have been partially solved as discussed in the next *Related Work* section.

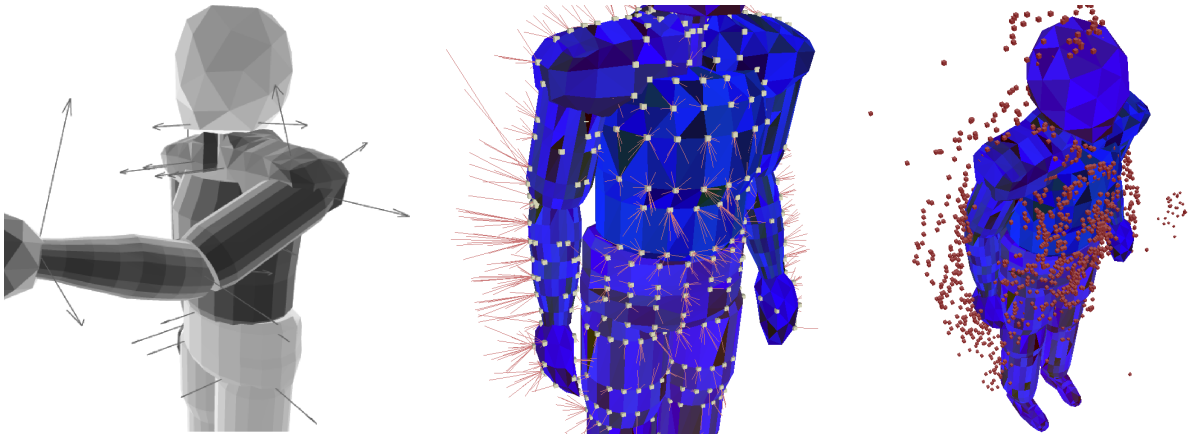
This thesis addresses all topics above. The approach taken and the methods developed pay attention to these problems. Though not solving them completely, the developed approach tries to push the limitations of previous methods with respect to these topics a bit further.

Generally, an optical motion capture system has to address the following aspects:

1. Acquisition of data. This includes choosing appropriate hardware, e.g. which cameras, calibration of the hardware, synchronization of cameras and choosing appropriate lighting.

2. Segmentation and processing of data. In this step data for the pose estimation is extracted from the recorded images. Typical image processing techniques are thresholding, grey value gradient evaluation, color processing, stereo, active contours, template matching and optical flow.

3. Pose Estimation. In this step optimization methods are applied to estimate the motion parameters from the data of step two. The pose estimation normally includes typical constraints of human motion, like fixed bone lengths, body parts are connected by joints, limits of joint angle motion and constraints, that the shape of the person is not varying much during motion except for muscle bulges and cloth deformations. The type of optimization method may also differ, e.g. correspondence based gradient descent or Monte-Carlo methods.



(a) (b) (c)

Figure 1.3: (a) The body model with its motion capabilities (rotation axes) shown as arrows. (b) The red lines indicate the difference between real observed data and synthesized data. (c) 3D points from a stereo algorithm shown as red boxes.

Marker-based systems put most effort into the acquisition step in order to simplify the segmentation and pose estimation step, e.g. by controlling the lighting conditions and background, such that data segmentation and processing simplifies to image thresholding.

A marker-free system, which is expected to handle cluttered and dynamic environments, has to develop better methods for the second and third step. These steps are the focus of this thesis.

The human body model plays an important role within these two steps. It incorporates the motion parameterization and motion limits and the appearance of the person by its shape and colors. Figure 1.3 (a) shows such a body model used in this work. The motion capabilities are shown by arrows, where each arrow represents a rotation axis.

In *Analysis-by-Synthesis* approaches , e.g. [75], the body model is used to generate synthetic data from the motion parameters, e.g. a synthetically rendered image. Then, the pose estimation analyzes how well the synthesized data fits to the real observed data, e.g. how similar the synthesized and the real image are.

Figure 1.3 (b) shows an example from this work for 3D data obtained from a stereo algorithm [43]. The value, which expresses how well the synthesized data (the person model in a specific pose) fits to the real data (the red boxes in Figure 1.3 (c)), is the sum of the lengths of the red lines. This value is then optimized in the pose estimation step.

One important aspect of all motion capture systems is the mathematical way to model and estimate the motion. All approaches solve an optimization problem, where motion parameters are estimated from data. However the optimization method, the motion parameterization and the type of data vary.

The motion parameterization is often formulated as a pose estimation problem of articulated objects. This is a well studied topic, for example in robotics [98, 26]. However, approaches to marker-free motion capture often use different optimization methods and

formulations of the motion in order to solve the problem. Especially the integration of different data types, e.g. depth data and silhouette data, within the estimation is addressed in numerous ways. Because the problem of pose estimation is nonlinear and combining different data types is complex and difficult, approximations have to be made in order to solve the problem efficiently.

The pose estimation method developed in this thesis combines 3D-point-data from stereo algorithms (Figure 1.3 c), 2D-point-data and 2D-line-data from silhouettes and optical flow methods. The optimization framework is formulated without making approximations to the motion model or the optimized error function. Additionally, the necessary derivatives for optimization are derived analytically and are calculated efficiently. The *Nonlinear Least Squares* (see section 2.3.2) approach allows also to include second order motion derivatives into the pose estimation, whose advantages and disadvantages are analyzed.

However, the *Nonlinear Least Squares* approach requires a starting point for optimization as all nonlinear optimization methods. This starting point is the pose of the person in the first frame of the analyzed video sequence. From this starting point the relative motion from frame to frame is estimated by pose estimation of an articulated object. Because only relative motion is estimated, this approach is also known as *motion tracking*.

Methods for comparison, different body models and optimization approaches are discussed in the next section.

1.1 Related Work

Human motion capture has been intensively studied in the past, especially in the last 15 years, when human motion capture started to play a more important role in the film and game industry.

An overview on human motion capture in the last 15 years can be found in the following surveys: in Aggarwal [6] up to 1997, in Moeslund & Granum [92] up to 2001 and in Moeslund,Hilton & Krüger [91]up to 2006.

Free-Viewpoint video systems [77] are a research field in computer graphics related to motion capture. There, the main interest is generally an accurate realistic look of the captured motion [123, 124], while the image analysis can be kept simple by enforcing heavy constraints on the capture environment, e.g. cameras surround the person, static homogenous background, non-changing studio lighting etc. The input data for the pose estimation step is often a visual hull of the person [23, 90, 128, 73, 5, 89], which is obtained from multiple synchronized silhouette images of the person (also known as shape-from-silhouette).

Pose Estimation in general and 2D-3D Pose Estimation specifically have been topics of research long since. A recent overview on monocular rigid pose estimation can be found in [79]. Rigid pose estimation can be extended to articulated objects, which is often used for human motion capture. Articulated objects and inverse kinematics are part of robotics research and various textbooks are available e.g. [98, 26].

In the following, literature is discussed, which has similar goals as this work. It is in contrast to surveillance and recognition approaches. A categorization of methods as done in the survey [91] is based on whether the approaches address the subproblems *Initialization, Tracking, Pose Estimation* and *Analysis*.

Initialization Initialization includes:

- Acquisition of the body model. The body model reflects the shape of the observed person, joint positions and the motion capabilities. The model can be obtained by a in advance with a offline procedure either automatic [5, 128, 110] or semi-automatic [106]. Motion capture and shape fitting simultaneously has also been investigated in [102, 124, 128].

- Finding the initial model state. Many motion tracking systems assume the initial pose of the person is given (often manually) [38, 102, 111, 34]. It is also possible to detect the person and its pose roughly from single images [10].

Tracking Tracking can be understood as

- Extracting relevant data for motion capture from the complete data, that is obtained from images or other sensors. In order to achieve this, segmentation of the person from irrelevant background data is necessary. This can be done by using silhouettes extracted from the image by thresholding [111] as for visual hulls or by using grey

value gradients to find edges of body parts as in [102, 38, 72, 48]. Because silhouettes only provide information at the boundary of the person's image, optical-flow-like methods have been utilized as well to track human motion as in [49, 17]. Rich information can be obtained from stereo algorithms, which provide a 3D-surface of the observed person. Fitting a template model to these 3D-point data has been done in [102, 67, 16, 34]. Another useful cue is color, e.g. of the skin or of the clothes of the observed person [58, 16, 50, 100, 17].

- Bringing the model in relation to the data, for example by finding temporal correspondences in past image frames or correspondences between model and observed 3D points. Different approaches are discussed in the remainder of this section.

Pose Estimation Pose Estimation is the task of evaluating the motion parameters of interest from the data. Three approaches can be generally distinguished [91]

- *Model-Free.* No a-priori model is used to estimate parameters. Previous approaches find motion parameters of a person for example by labeling the body parts [131].

- *Indirect Model-Use* In these approaches no a-priori body model is given, but is obtained during the measurement process. Popular examples for this approach are visual-hull reconstruction systems, which first build a 3D surface of the observed person from multiple camera views and afterwards look for characteristic positions on the hull in order to find body parts and fit joint positions [82, 128, 5].

- *Model-Based* Analysis-by-synthesis approaches use a template model of the person. The model includes the motion capabilities, joint positions and shape of the person. The accuracy of the shape varies from simple cylinders for each body part [34] over multiple ellipsoidals for each part to linear blend skinned triangle surfaces [16], which model skin deformations and muscle bulges. These template models can be fitted to quasi-visual hulls [124, 123] or partial hulls like obtained from multiple stereo cameras [67]. There can be two further methods distinguished: *Correspondence-Based* and *Correspondence-Free* Pose Estimation, which will be discussed in the following.

Analysis This includes identification of people, recognition of actions, classification of diseased motions etc. Analysis is not addressed in this thesis.

In this thesis the focus lies on tracking and pose estimation.

1.1.1 Correspondence-Free Pose Estimation

As mentioned above there are *Correspondence-Based* and *Correspondence-Free* pose estimation methods. Correspondence-free approaches are either model-free [131] or make an indirect use of a model [82, 128].

Analysis-by-Synthesis approaches that estimate pose and motion without correspondences often solve the optimization problem with *Monte-Carlo* or *Particle Filter* [83, 66, 37] based methods. These methods 'guess' solutions (called particles or samples) systematically and then evaluate how well the guessed solution fits to the observed data. This is achieved by approximating the probability distribution in the motion parameter space by samples. The advantage of these methods is, that they are unlikely to get stuck in local minima. However, to be sure that the 'interesting' region of a high-dimensional parameter space is properly sampled, a large amount of samples is usually necessary [83].

Essential for particle filter approaches [36, 78] is the likelihood or measurement probability, which assigns to a specific parameter set a value that reflects how well the parameter set fits the data. In order to achieve this, the parameter set together with the model is used to synthesize artificial data (usually a rendered image of the 3D body model). The analysis step then assigns a likelihood probability. The guesses of likely parameter sets are predicted from previous estimates by a motion model [69].

Correspondence-free methods are less popular than correspondence-based ones, probably because it is difficult to design a suitable likelihood function and to constrain the state space sufficiently, such that results can be estimated with a small number of samples in acceptable time.

1.1.2 Correspondence-Based Pose Estimation

Given in this section is an overview of work that is more closely related to this thesis. These are approaches that concentrate on correspondence-based marker-free full body motion capture.

They provide data similar to marker based systems, which is joint data over time for reanimation of artificial characters, as needed in computer games or movies. Included are methods that capture joint data only for subparts of the human body, e.g. hands or upper body motion.

These approaches have in common, that they are correspondence based, that means the pose estimation relies on correspondences between image data and body model. Therefore the main challenge of the tracking step is how to find reliable correspondences. In the pose estimation step most of these approaches rely on gradient based methods, e.g. [73, 111, 40, 54, 102].

Closely related work is now discussed with respect to the tracking step (how correspondences are found) and the pose estimation method. They are ordered with respect to the kind of correspondences used in the optimization.

3D-point-3D-point correspondences This correspondence type consists of one 3D-point on the model and a target 3D point from the observed data, e.g. depth points from stereo or a visual hull.

- Demirdjian [34] developed a system that is able to estimate body pose in near real-time with 6-10Hz on a Pentium4 2GHz. The estimation of articulated

relative motion from frame to frame is based on 3D data points calculated by a dense real-time stereo algorithm [35]. For each body segment an Iterative Closest Point algorithm finds the relative rigid motion. The additional constraint of linked body parts is applied afterwards by projecting the estimated motion to the subspace of the possible articulated motion. The authors state themselves in [34] *'Constraint projection methods may give a sub-optimal solution but are usually easier to implement and, when used in an iterative fashion, provide solutions very close to the estimation provided by direct methods.'*. A further motivation for their method is, that *'the size of the system involved in the body motion estimation is very small'* without specifying it further or giving comparisons. The disadvantage of this pose estimation method is that the motion of each body part is first estimated independently of other parts without taking into account that the parts are connected by joints. Only in a second step the connection is enforced by projection. Problems that can arise with this approach are illustrated in [87].

The estimation from 3D stereo data alone is extended in [33] to include view based information also. Views are defined by key-frames, each key-frame consists of an image, the associated pose, a number of 2D feature points, and a matrix that converts feature displacements (2D optical flow) to a pose change for the associated pose. Combination of 3D stereo data and view based estimation is done by estimating a new pose separately for each method and taking that one, which better fits the 3D stereo data.

- In Bray & van Gool [16] motion of one hand is captured using 3D Stereo data from a structured light sensor. The hand model is obtained from 3D scans with a structured light sensor [42] with 26 internal degrees of freedom. Animation of this model yields accurate skin deformations, because a linear blend skinning approach is taken [84, 93], where the motion of one point of the model skin is dependant not only from the previous parent joint, but also from the next joint in the kinematic chain.

  A variant of *Gradient Descent*, namely *'Stochastic Meta Descent' (SMD)*, is proposed as a new algorithm, that is able to to track motion in high-dimensional spaces. The efficiency is shown in the results and compared with other popular optimization methods. All optimization techniques rely on numerical derivatives of the objective and the motion function. Estimation of hand pose with 26 DOF using 45 3D points is done in 4.7 seconds per frame on a 2Ghz Pentium IV.

  Although the pose estimation method models the motion of points on the model with a chain of rotations, the necessary partial derivatives are calculated numerically. Because SMD is reported to perform more efficiently than other optimization methods, we compare the SMD optimization method with *Nonlinear Least Squares* solved by *Gauss-Newton* (see Section 2.3.2).

  The SMD optimization method is applied in [73] to full body tracking of humans. The pose of a template model, which is already fitted in size to the observed

person, is estimated using 3D data from a visual hull of the person. The 3D visual hull is calculated from up to 4-8 camera views based on fore/background segmentation with a voxel representation. Reconstruction and texturing of the visual hull is done in 90ms. The pose estimation with SMD is done by taking 5 random points from 4000-8000 surface voxel points in each iteration.

The comparison in chapter 3.8 discusses the effect of random sub-sampling the complete data set and the size of this subset.

- Ziegler et al. [67] estimate the pose of the upper body with an unscented Kalman Filter. The input to their system are depth images from 4 stereo views. The raw stereo data is preprocessed to eliminate invalid depth values by an edge detection filter. Further background segmentation based on the distance values yields the final point cloud for pose estimation. An ICP algorithm then yields model and point cloud correspondences as in this thesis. Minimization of differences between point cloud and model is done with an unscented Kalman Filter, which is not in the need of partial derivatives (the Jacobian matrix).

  A person can be tracked over 6000 image frames, with less than one second processing time per frame. Mean errors over that sequence for the orientation of the upper and lower arm are at 22-25deg. Problematic poses where the arms are too close to the body are detected and the hands are moved away from the body to regain valid ICP correspondences.

  The disadvantage of this method is that, that the unscented Kalman Filter approximates the covariance and mean of the state probability density by samples around the current assumed solution. This is very similar to calculating partial derivaties numerically. The advantage of their apporach is, that they can track long sequences without getting lost, which is due to the multiple views.

3D-point-2D-line Here the 3D point on the model is in correspondence with a 2D-line. This 2D-line can be a part of the silhouette or the combination of spatial image gradient and image difference.

- Malik & Bregler [17] estimate the motion of a human person from monocular images. Walking sequences of the famous Muybridge recordings from 1884 are the input to their motion capture algorithm. Motion of arms and legs can be successfully estimated using an ellipsoidal body model. Instead of establishing explicit correspondences between image and 3D model the optical flow assumption of a locally linear grey value gradient is combined with the dynamic constraints of the articulated body. As shown in this thesis, their pose estimation method is equal to 3D-point-2D-line correspondences.

  Motion is expressed as concatenated rotations around known axes like in this thesis. The axis/angle description of rotation is formulated with twists inspired from robotics [98] and lead to the same partial derivatives as in this thesis. The resulting linearized rotation equations are applied in a Nonlinear Least Squares

approach. The camera projection is modeled as an orthographic projection and its scale is estimated also. Motion estimation in the depth direction is therefore not possible from a single view.

Each body segment has a 2D support map, which adds weights for each pixel in the optimization. The weights are calculated from a 2D Gaussian distribution with its maximum at the center of the body part. This support map is updated after each frame using an Expectation Maximization (EM) algorithm following the approach of [70]. How the necessary likelihood is calculated is not quite clear, but it depends on the difference between the motion vector of each region and the motion vector of each pixel as computed from the spatial image gradient and the temporal image gradient at that pixel position.

The same approach is extended in [18] to multiple views. In addition to joint angle estimation, the length of body parts is included in the optimization.

- In Drummond & Cipolla [38] motion of a human body is estimated based on gray value gradients, assuming that the outline of the person has a significant contrast to the background. This silhouette information is used to estimate arm motion of a person in front of cluttered unknown background and further results show estimated arm and leg motion of a wooden toy-mannequin.

  Due to multiple views arm motion in all directions can be estimated. The estimation method uses Gaussian statistics for each body segment, which is modeled as a 3D-ellipsoid. The relative motion from frame to frame is linearized using the twist motion (rotations with axis/angle) yielding an analytic Jacobian. The dynamics of the kinematic chain are not directly modeled in the motion function as in [17, 111, 102] or in this thesis, but the statistics for the relative rigid body motion of each body segment are propagated back and forth through the chain to incorporate the constraints of linkage between body parts. These approximation to the correct kinematic motion allows estimation of ca. 20 degrees of freedom with 25Hz from PAL images. Silhouette information is established by searching along the normal for maximal gradient values from specific points on the model surface. These distance measurements make up 3D-2D point correspondences. From the result images it can be assumed that approximately 60-80 correspondences are used for each of the three views .

Multiple Correspondence Types Here multiple correspondence types are combined namely 3D-point on the model and 3D target point or target 2D image point or 2D-line.

- This thesis was partially inspired by the work on pose estimation of Rosenhahn [109]. The correspondence based pose estimation is addressed in his work within a *Geometric Algebra* framework. The Geometric Algebra framework allows the estimation of other moving primitives as well [112]. The pose estimation methods are extended to free-from objects in [113], which allow low-pass approximation of the object based on Fourier descriptors.

In [111] it is shown, that marker-free motion capture based on silhouette information can compete with marker-based systems in accuracy of joint angle estimation. The silhouette correspondences are established by searching for nearest neighbors between the image silhouette and the silhouette of the projected model. An Iterative Closes Point (ICP) approach then estimates the pose of the model. The model's shape is fitted before capturing to the observed person as in [110].

The advantage of the pose estimation method is, that multiple moving entities on the model can be handled, e.g. points or lines. The disadvantage is that all errors are optimized in 3D-space, which is less accurate than the optimization in the image plane in case of noisy measurements as the author experimentally showed in [127].

- Plaenkers & Fua [102] present a system that estimates the pose of a human body using Stereo data in combination with extracted silhouettes. A Nonlinear Least Squares approach estimates the joint angles and the size of a human body, which consists of 3D-ellipsoids (metaballs). Results for upper body pose estimation, which involves arm motion are presented. The distance between observed 3D-points from a stereo algorithm and the model surface is minimized with a modified LM optimizer, that can handle sparse matrices [103]. Silhouette information is included in the optimization by taking the 3D-point on the line of sight from a silhouette pixel that is closest to the current model configuration. The distance of the model surface to this point is minimized under the constraint, that the line of sight lies within the tangential plane of the surface at that point. The partial derivatives needed for the Jacobian are given analytically in [103].

  The silhouette is found by first subtracting the background through segmentation of 3D points, assuming that no scene objects are near to the person, and second, applying this segmentation to the gradient image. After thresholding and filling holes with morphological operators an *Active Contour [15]* is fitted to the remaining outline of the binary image. The time needed to process one image is not given and it can be assumed that it is far from real-time, because the scaling parameters of the metaballs increase the state space dramatically in contrast to pure joint angle estimation.

Summary Related Work

The correspondence based pose estimation methods developed in the literature, as for example those above, show a large variety, even though the pose estimation problem of articulated objects is a topic of research long since. They differ in the following aspects:

- The constraint, that body parts are connected by joints is sometimes enforced by approximative methods, e.g. by regularization [40] or by projection [33]
- The optimization methods differ, e.g. Gradient Descent, unscented Kalman Filter, Nonlinear Least Squares, Propagation of Gaussian Statistics

- The types of correspondences differ, which are used for pose estimation.

One might ask: Why do they differ so much in spite of a very common problem ?
Possible answers to this question are

- some methods are easier to implement than others [33]
- partial derivatives are not necessary [67]
- some approximative optimization methods might be faster than others (usually no direct comparison is made) [33, 38, 16]
- sometimes a specific functional description for the shape of body parts is used, which can be better suited for a specific optimization function, e.g. ellipsoids modelled by 3D Gaussian probabilities [38, 102]

Another question one might ask: Does the literature suggest, that there is a common or optimal way to estimate pose of articulated objects ?

This is difficult to answer. However, a commonality that occurs in multiple approaches is a pose estimation method, which is Gradient-based, namely Gradient Descent [16, 73, 48] or Nonlinear Least Squares [113, 17, 103, 102]. Additionally, some of these approaches model the constraints of connected body parts directly without approximative methods [113, 17, 103, 102]. Further it can be noted, that it is possible to formulate the optimization function, such that necessary partial derivatives are analytically available [113, 17, 103, 102, 38] rather than numerically [16, 73]. Many approaches, which use analytical partial derivatives, model the motion of the articulated object by a rotational chain, where each rotation is modeled by a rotation axis and an angle (twist) [113, 17, 38]. Others use matrix notation [103, 102] to derive necessary analytical derivatives.

1.2 Contribution of this Thesis

There are two main contributions of this thesis, one is the developed pose estimation method and the other is the way to establish correspondences between model and data for tracking.

Pose Estimation The pose estimation developed in this thesis is based on motion of 3D points within in a kinematic chain. The underlying motion constraints, that human body parts are connected by joints, are directly included in the motion function without making approximations [33] or enforcing joint connections by regularization [97].

One important aspect of the developed pose estimation method is the possibility to use correspondences of different entities at once, namely 3D-point-3D-point, 3D-point-3D-line, 3D-point-2D-point and 3D-point-2D-line. Here the entity on the body model is always a 3D-point and image entities can be points or lines. It is also shown, that the optimization function for 3D-point-2D-lines is equal to the KLT-equations [12, 125] brought to articulated objects, which is equal to the approach with twists and exponential maps of *Malik & Bregler* [17].

The optimization framework for pose estimation of articulated objects can be seen as an enhancement of previous approaches to pose estimation. Very similar is the pose estimation method of *Rosenhahn* [112, 111, 109], because the resulting equation system is equal for 3D-point-3D-point and 3D-point-3D-line correspondences. For 3D-point-2D-point and 3D-point-2D-line correspondences the optimization function used in this work makes no approximation, because the error is optimized in the image plane rather than in 3D space. Additionally the internal parameters of the camera can be estimated simultaneously. Another difference is the mathematical framework, in which the motion function is formulated. While *Rosenhahn* uses Geometric Algebra [62] in this thesis a standard optimization method namely *Nonlinear Least Squares* estimates motion parameters.

Another very related approach is that of *Lowe* [81] or more specifically the extension of it in [7]. The pose estimation method in this work can be seen as an enhancement of *Lowe*'s pose estimation for rigid objects extended to articulated objects.

Both approaches above have in common with this thesis, that the partial derivatives of the motion function are calculated analytically, which is in contrast to many other correspondence-based motion capture approaches, which use numerical derivatives.

A very important aspect in practice is the possibility to use *dampening* within the estimator. Though this is a standard way in optimization, e.g. in the *Levenberg-Marquardt* algorithm (see section 2.3.4), other motion capture approaches do not utilize it.

The formulation within the *Nonlinear-Least Squares* optimization allows to include second order motion derivatives, which can be interpreted as the acceleration of points moving on the human body. To our knowledge this is investigated for the first time.

Tracking Tracking is in the context of this thesis, the methods developed in order to find correspondences between observed data and body model of the human. The processing of data is carried out, such that reliable correspondences can be found in spite of cluttered dynamic background and changing lighting.

Depth data from stereo algorithms is invariant to both aspects. Therefore a method is developed, that estimates pose from a single stereo view. Occlusion and visibility are efficiently estimated by rendering the model with OpenSG [108] and the correspondence search is accelerated by ordering visible data points within an associative array. The advantage of the implementation is, that no functional description of the body model is necessary. Any model build with professional modeling tools can be used as long as it is in VRML [20, 133] description. The only additional necessary information is, which MPEG4 animation id [4] is associated with each joint in the body model.

A comparison with similar approaches to motion tracking from depth data is made and shows, that challenging upper body motion can be estimated with 5fps on a Pentium IV 3Ghz.

To get more information about the observed motion from a single view, silhouette information and optical flow is utilized to build 3D-point-2D-point and 3D-point-2D-line correspondences. The novel method combines a gray value gradient search along the model silhouette with color histograms from each body part. With this combination silhouette correspondences are found in spite of dynamic and changing background. In order to get information about motion, which is not visible in the silhouette from one view, optical flow point tracking provides further correspondences.

Concluding results for arm tracking from depth data and silhouette data show, that the different cues complement each other and increase the accuracy and stability.

1.3 Overview of this Thesis

This thesis is organized in 5 chapters. In this first chapter an introduction to motion capture in general and to video-based optical motion capture in particular is given. Related work is discussed and the contribution of the thesis is stated. The second chapter gives the basic mathematical fundamentals of optimization, which are used in the following chapters.

Chapter 3 and 4 deal with the two contributions of this thesis the Pose Estimation and the Tracking respectively. In chapter 3 the optimization algorithms are used to estimate pose of a rigid object from different geometric correspondences. Namely 3D-point-3D-point, 3D-point-2D-point and 3D-point-2D-line correspondences. The pose estimation is then extended to articulated objects, which is straightforward with the chosen representation of motion.

In Chapter 4 the pose estimation methods are applied to track motion of a human body without markers. First a method to obtain a body model for a specific person is described.

This model is then used in various experiments to track human motion and a comparison with related optimization algorithms is given.

In chapter 5 the thesis is concluded by summarizing the achieved results and open problems are discussed.

Chapter 2

Basics of Optimization

Of specific interest in this thesis is Nonlinear-Least-Squares, although a common optimization method [24], it is not paid much attention in motion capture approaches. This thesis tries to show that Nonlinear-Least-Squares is more efficient than other methods and easy to implement. Further interesting optimization aspects for motion capture in practice like *dampening, robust estimation* and *constrained estimation* are difficult to find in a single textbook. Therefore the theory is given in this chapter before the pose estimation algorithm itself.

2.1 Notation

$s, \theta, x \in \mathbb{R}$	a scalar value
$\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n$	a column vector or point, usually $n = 3$
$\boldsymbol{p'} \in \mathbb{R}^m$	a projection of $\boldsymbol{p}$ to a lower dimension or a point with $m = 2$
$A, B, J \in \mathbb{R}^{m \times n}$	a matrix with m rows and n columns
J_{ij}	the matrix element in the i-th row and j-th column
$J_{k,t}$	indices with different meaning, e.g. the Jacobian for the k-th point at iteration t
p_x, p_y, p_z	indices x, y, z always indicate components of a vector or point, e.g. $\boldsymbol{p} = (p_x, p_y, p_z)^T$

Parameter Estimation

$\theta, \psi \in \mathbb{R}$	a single scalar paramter to estimate
$\boldsymbol{\theta}, \boldsymbol{\psi} \in \mathbb{R}^n$	a vector of multiple parameters to estimate
$p = \dim \boldsymbol{\theta}$	the amount of parameters
$\tilde{\boldsymbol{p}} \in \mathbb{R}^n$	an observed point or measurement
$\hat{\boldsymbol{\theta}} \in \mathbb{R}^n$	the minimizer of an optimization problem

Special Terms and Functions

$\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})$	motion function for a point $\boldsymbol{p}$ regarding pose parameters $\boldsymbol{\theta}$
$(\boldsymbol{\omega}, \boldsymbol{q})$	a line (rotation axis) with direction vector $\boldsymbol{\omega}$ and a point $\boldsymbol{q}$ on that line
$R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta, \boldsymbol{p})$	rotation of a point around an axis defined by $(\boldsymbol{\omega}, \boldsymbol{q})$ with angle θ
$r(\boldsymbol{\theta})$	the residuum function (an error value)

2.2 Linear Least Squares

Linear Least Squares is a common and known technique for parameter estimation and can be looked up in detail in textbooks about optimization e.g.[24]. A short description is given here, because this linear technique is the basis for the nonlinear estimation methods and important in this work.

Consider a system of linear equations

$$A\boldsymbol{\theta} = \boldsymbol{b} \tag{2.1}$$

where $A \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m$ are known and $rank\ A = n$. If $m = n$ there is exactly one solution. If $m > n$ the problem is over determined and there is usually no exact solution. In that case one may instead find the best $\hat{\boldsymbol{\theta}}$ that minimizes

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} ||A\boldsymbol{\theta} - \boldsymbol{b}||^2 \tag{2.2}$$

This minimization problem can be rewritten by minimizing a quadratic function

$$\begin{aligned} f(\boldsymbol{\theta}) &= ||A\boldsymbol{\theta} - \boldsymbol{b}||^2 && (2.3)\\ &= (A\boldsymbol{\theta} - \boldsymbol{b})^T (A\boldsymbol{\theta} - \boldsymbol{b}) && (2.4)\\ &= \frac{1}{2}\boldsymbol{\theta}^T(2A^TA)\boldsymbol{\theta} - \boldsymbol{\theta}^T 2A^T\boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b} && (2.5) \end{aligned}$$

The objective is then to minimize $f(\boldsymbol{\theta})$. Therefore $f(\boldsymbol{\theta})$ is called the *objective function*. The necessary condition for $\hat{\boldsymbol{\theta}}$ minimizing $f(\boldsymbol{\theta})$ is, that the first order derivative vanishes:

$$\begin{aligned} \mathbf{0} = \nabla f(\hat{\boldsymbol{\theta}}) = 2A^TA\hat{\boldsymbol{\theta}} - 2A^T\boldsymbol{b} \iff && (2.6)\\ 2A^TA\hat{\boldsymbol{\theta}} = 2A^T\boldsymbol{b} \iff && (2.7)\\ \hat{\boldsymbol{\theta}} = (A^TA)^{-1}A^T\boldsymbol{b} && (2.8) \end{aligned}$$

Because $rank\ A = n$ the rank of A^TA equals also n. Therefore $G = (A^TA)$ is invertible. The matrix $G = (A^TA)$ is also called the *Gram matrix* or the *Grammian*.

2.2.1 Parameter Estimation from Data Points

Least Squares is an optimal method in case of parameter estimation from Gaussian distributed data points, i.e. the data points are disturbed by Gaussian noise. If the number of data points is sufficiently high and their distribution is exactly Gaussian, the estimated solution would be equal to an estimate from non disturbed data points. In fact this is true for bounded symmetric distributions with zero mean.

Consider the problem of fitting a third degree polynomial to a number of data points $(x_i, y_i), i = 1, .., m$. Polynomial fitting can be used for example for smoothing motion estimates.

$$f(\boldsymbol{\theta}) = \sum_i \left(\theta_0 x_i^3 + \theta_1 x_i^2 + \theta_2 x_i + \theta_3 - y_i\right)^2 \tag{2.9}$$

Estimating the best fit $\hat{\boldsymbol{\theta}}$ of f is then equal to

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} ||A\boldsymbol{\theta} - \boldsymbol{b}||^2$$

where the i-th row of $A \in \mathbb{R}^{m\times 4}$ is $A_i = (x_i^3, x_i^2, x_i, 1)$ and $\boldsymbol{b} = (y_0, .., y_m)^T$. The estimated solution however is optimal if only the values y_i are disturbed by Gaussian noise. If the values x_i are additionally disturbed by noise the *Total Least Squares* can be applied.

2.2.2 Total Least Squares

The idea of this approach is to minimize not only the distance in y direction, but to minimize the Euclidean distance of the data points $(x_i, y_i)^T$ to the desired function, whose parameters are to be estimated. In a total least squares sense the minimization reads as follows:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_i \left| \begin{pmatrix} u_i(\boldsymbol{\theta}, x_i, y_i) \\ v_i(\boldsymbol{\theta}, x_i, y_i) \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right|^2 \tag{2.10}$$

where $(u_i, v_i)^T$ is the nearest point to $(x_i, y_i)^T$ on the estimation function. The *Total Least Squares* problem corresponding to a standard *Linear Least Squares* problem is not necessarily also linear. However, even if u_i and v_i are nonlinear functions the minimization problem may be solved using nonlinear methods. For more details on Total Least Squares see [31].

2.3 Nonlinear Optimization

So far only linear problems were considered. The following sections now give a brief introduction of nonlinear optimization methods. Their main differences to linear parameter estimation are, that nonlinear methods require a starting point (an initial guess of parameters) and need multiple iterations to find the desired parameters. As there is a wide variety of algorithms, we cannot give a complete description of all methods here, but will focus on two of the most important ones. These are *Gradient Descent* and *Nonlinear Least Squares*. Many other methods are just an extension (e.g. *Steepest Descent, Stochastic Meta Descent*) or approximation (e.g. *Quasi-Newton, BFGS*) of these methods. For more details refer to [105, 24]

2.3.1 Gradient Descent

The idea of *Gradient Descent* methods is quite simple. If a sufficiently small step (in parameter space) is taken in the direction of the negative gradient $-\nabla f_i(\boldsymbol{\theta})$, then the value of the objective function has to decrease. This is done iteratively until a minimum is found, where the necessary condition is, that the gradient vanishes, i.e. $\nabla f_i(\boldsymbol{\theta}) = \mathbf{0}$. Of course only the next local minimum will be found with this approach.

The sequence of parameters $\{\boldsymbol{\theta_0}, .., \boldsymbol{\theta_k}\}$ leading to $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta_k}$ is calculated by

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \alpha \nabla f_i(\boldsymbol{\theta}) \tag{2.11}$$

The iteration usually stops if the change of the error is below a certain threshold. Alternatively the iteration may stop if the length of the gradient vector is below a certain value, as this is the necessary condition. The step size α controls the rate of convergence. Ideally α is as large as possible. If taken too large the parameters may jump out of the locally convex region of the objective function leading to a false local minimum or to no convergence at all. If taken too slow, the optimization may need too many iterations to converge in an acceptable time scale.

Therefore there exists a variety of methods to choose α optimally. Usually the scalar value α is extended to a vector $\boldsymbol{\alpha}$ to have a different step size for each parameter. This is for example due to different orders of magnitude of the gradient components or because the error surface has different curvature in different directions. The *Stoachstic Meta Descent [16]* discussed later in this work is a variant of this *Gradient Descent*.

2.3.2 Nonlinear Least Squares

Nonlinear Least Squares is a special class of optimization problems and can be solved efficiently using *Newton's Method*.

A *Nonlinear Least Squares* problem reads as follows:

$$\min_{\theta} \sum_{i=1}^{m} (r_i(\boldsymbol{\theta}))^2 \tag{2.12}$$

where $r_i(\boldsymbol{\theta}) : \mathbb{R}^n \to \mathbb{R}, i = 1, .., m$ are real valued nonlinear functions, which must have in common the same parameterization $\boldsymbol{\theta}$. We call $f(\boldsymbol{\theta}) = \sum_{i=1}^{m} (r_i(\boldsymbol{\theta}))^2$ the *objective function*. A possible solution to this problem can be found with *Newton's* method as described in the next section. Newton's method requires more functions (also called *residui*) than parameters, such that $m >= n$. If $m < n$, then a variation of Newton's method, called *Dampening*, or a Gradient Descent method can be applied (see following sections).

The minimizer is found by iteratively solving a simpler problem starting from initial parameters $\boldsymbol{\theta_0}$. Under the assumption that the objective function is locally convex around $\boldsymbol{\theta_0}$ the next local minimum will be found at the end of the iteration. The iteration may stop, if the change of the error is below a certain threshold. Alternatively the iteration may stop, if the length of the gradient vector is below a certain value, as the necessary condition for a minimum is, that the gradient equals zero.

Newton's Method

The idea of *Newton's Method* or sometimes called *Newton-Rhapson* is to approximate the objective function by a quadratic function in each step of the iteration. For a quadratic function $\mathbb{R}^n \to \mathbb{R}$ with n parameters the optimal solution of the minimization can be

given in closed form and evaluated in a single step. For an over determined problem (there are more observations than parameters) this is known as *(Linear) Least Squares*. The approximating quadratic function $f_\Delta(\boldsymbol{\theta})$ is the *Taylor series* expansion of the objective function at some point $\boldsymbol{\theta_t}$ neglecting terms of third order and higher:

$$f(\boldsymbol{\theta}) \approx f_\Delta(\boldsymbol{\theta}) = f(\boldsymbol{\theta_t}) + \boldsymbol{\Delta_\theta}^T \nabla f(\boldsymbol{\theta_t}) + \frac{1}{2}\boldsymbol{\Delta_\theta}^T H(\boldsymbol{\theta_t})\boldsymbol{\Delta_\theta} \tag{2.13}$$

where $\boldsymbol{\Delta_\theta} = (\boldsymbol{\theta} - \boldsymbol{\theta_t})$. The gradient

$$\nabla f(\boldsymbol{\theta_t})_i = \left.\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i}\right|_{\boldsymbol{\theta_t}}$$

is evaluated at $\boldsymbol{\theta_t}$ and $H_{ij} = \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j}$ is the matrix of second derivatives (the Hessian). The term $H(\boldsymbol{\theta_t})$ denotes also, that the second derivatives are evaluated at $\boldsymbol{\theta_t}$, i.e.

$$H(\boldsymbol{\theta_t})_{ij} = \left.\frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j}\right|_{\boldsymbol{\theta_t}}$$

The objective is now to minimize $f_\Delta(\boldsymbol{\theta})$ with respect to $\boldsymbol{\Delta_\theta}$, which gives the parameters for the next iteration $\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \hat{\boldsymbol{\Delta}}_{\boldsymbol{\theta}}$. The necessary condition for $\hat{\boldsymbol{\Delta}}_{\boldsymbol{\theta}}$ being a minimizer of $f_\Delta(\boldsymbol{\theta})$ is:

$$\begin{aligned} \nabla f_\Delta(\boldsymbol{\theta}) = \mathbf{0} \quad &\Leftrightarrow \nabla f(\boldsymbol{\theta_t}) + H(\boldsymbol{\theta_t})\hat{\boldsymbol{\Delta}}_{\boldsymbol{\theta}} = \mathbf{0} \\ &\Leftrightarrow \hat{\boldsymbol{\Delta}}_{\boldsymbol{\theta}} = -H(\boldsymbol{\theta_t})^{-1}\nabla f(\boldsymbol{\theta_t}) \end{aligned} \tag{2.14}$$

Please note that the Hessian is a symmetric matrix. In the next section is shown how the gradient and the Hessian look like for a *Nonlinear Least Squares* problem.

Newton's Method for Nonlinear Least Squares

To apply *Newton's Method* to *Nonlinear Least Squares* the gradient and the *Hessian* (the matrix of second derivatives) of the objective function are needed. The j-th value of the gradient of the objective function is:

$$(\nabla f(\boldsymbol{\theta}))_j = \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_j} = \frac{\partial \sum_{i=1}^m (r_i(\boldsymbol{\theta}))^2}{\partial \theta_j} = 2\sum_{i=1}^m r_i(\boldsymbol{\theta})\frac{\partial r_i(\boldsymbol{\theta})}{\partial \theta_j} \tag{2.15}$$

By introducing the *Jacobian* $J \in \mathbb{R}^{n\times m}$ the matrix of first derivatives

$$J(\boldsymbol{\theta})_{ij} = \frac{\partial r_i(\boldsymbol{\theta})}{\partial \theta_j} \tag{2.16}$$

we can write the gradient of f as:

$$\nabla f(\boldsymbol{\theta}) = 2J(\boldsymbol{\theta})^T\, \boldsymbol{r}(\boldsymbol{\theta}) \tag{2.17}$$

with $\boldsymbol{r}(\boldsymbol{\theta}) = (r_1(\boldsymbol{\theta}), .., r_m(\boldsymbol{\theta}))^T$.

The needed second derivatives of f (the Hessian) are obtained by taking the derivative of equation 2.15 with respect to θ_k:

$$\frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_k\,\partial\theta_j} = 2\sum_{i=1}^{m}\frac{\partial\left(r_i(\boldsymbol{\theta})\frac{\partial r_i(\boldsymbol{\theta})}{\partial\theta_j}\right)}{\partial\theta_k} \tag{2.18}$$

$$= 2\sum_{i=1}^{m}\left(\frac{\partial r_i(\boldsymbol{\theta})}{\partial\theta_k}\frac{\partial r_i(\boldsymbol{\theta})}{\partial\theta_j} + r_i(\boldsymbol{\theta})\frac{\partial^2 r_i(\boldsymbol{\theta})}{\partial\theta_k\,\partial\theta_j}\right) \tag{2.19}$$

By introducing a matrix $S(\boldsymbol{\theta})$ with

$$S_{kj} = \sum_{i=1}^{m} r_i(\boldsymbol{\theta})\frac{\partial^2 r_i(\boldsymbol{\theta})}{\partial\theta_k\,\partial\theta_j}$$

we write the Hessian as:

$$H(\boldsymbol{\theta}) = 2\left(J(\boldsymbol{\theta})^T J(\boldsymbol{\theta}) + S(\boldsymbol{\theta})\right) \tag{2.20}$$

Inserting these results into equation 2.14 gives the final update rule for one iteration:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \left(J(\boldsymbol{\theta_t})^T\, J(\boldsymbol{\theta_t}) + S(\boldsymbol{\theta_t})\right)^{-1}\, J(\boldsymbol{\theta_t})^T\, \boldsymbol{r}(\boldsymbol{\theta_t}) \tag{2.21}$$

It is quite common to neglect the matrix S, because its components are negligible small or because second derivatives of r_i are not available or are too time consuming to calculate. In that case *Newton's method* becomes what is commonly called the *Gauss-Newton method*:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \left(J(\boldsymbol{\theta_t})^T\, J(\boldsymbol{\theta_t})\right)^{-1}\, J(\boldsymbol{\theta_t})^T\, \boldsymbol{r}(\boldsymbol{\theta_t}) \tag{2.22}$$

Please note in the last two equations, that $J(\boldsymbol{\theta_t})$ consists of the first derivatives evaluated at $\boldsymbol{\theta_t}$.

In figure 2.1 three iterations of the Gauss-Newton method are visualized for a sinusoidal error function $sin(\theta)^2$, which is plotted as a solid thick line. The starting value is $\theta_0 = 1.0rad$. The Gauss-Newton approximation of the first iteration is shown as a dotted line leading to $\theta_1 = -0.56$. The second iteration is shown as a dashed line and yields its minimum at $\theta_2 = 0.066$. The quadratic function of the third iteration is plotted as a thin solid line, its minimum is so near to the correct minimum at zero, that further iterations are not drawn.

2.3.3 Dampened Newton's Method

The idea of dampening is to make the Hessian positive definite, which reduces the parameter change in each iteration step. The larger the dampening value is, the smaller the step is and the more similar the method becomes with Gradient Descent. To achieve this

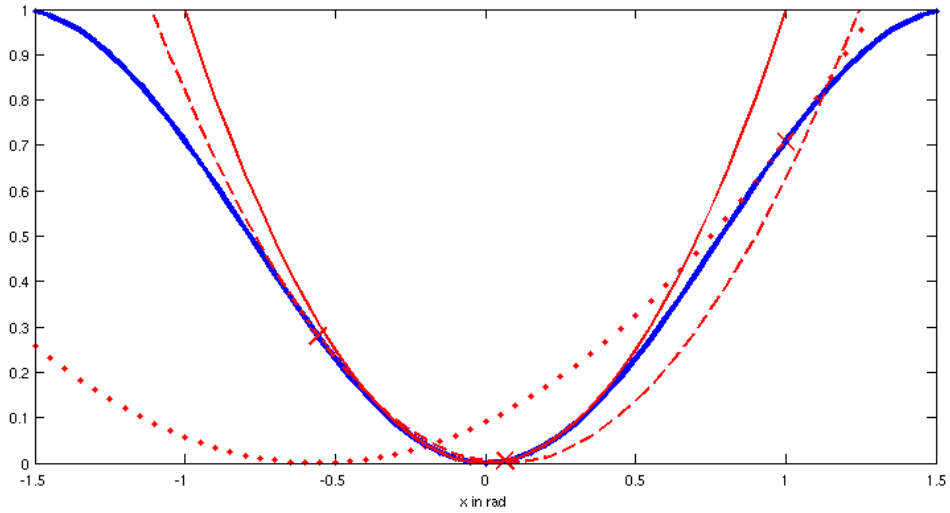


Figure 2.1: Illustration of Newton's method for a sinusoidal function $sin(\theta)^2$

the original minimization function is extended by a regularization term in each iteration, which penalizes large changes in the parameters:

$$\min_{\theta} \sum_{i=1}^{m} (r_i(\boldsymbol{\theta}))^2 \; + \; \lambda |\boldsymbol{\Delta_\theta}|^2 \tag{2.23}$$

Important here is to note, that $\boldsymbol{\Delta_\theta} \in \mathbb{R}^n$ refers to a parameter change in each iteration. That way, the final minimizer $\hat{\boldsymbol{\theta}}$ is the same as for the original problem, there are just more iterations necessary to reach it.

The regularization term $\lambda|\boldsymbol{\Delta_\theta}|^2$ is equal to changing the minimization problem by adding n additional functions $r_j(\boldsymbol{\theta}) = \lambda \Delta\theta_j, \;\; j = 1, .., n$. Therefore the Jacobian of the original minimization problem has to be extended by the partial derivatives of the additional residuum functions r_j, which is a scaled identity matrix $\lambda I^{n \times n}$.

The additional residuals (error values) at the current position $\boldsymbol{\theta_t}$ equal zero, because $\boldsymbol{\Delta\theta} = \boldsymbol{\theta} - \boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_t = 0$. Therefore a similar effect is achieved, if the Hessian is varied by adding $\lambda I^{n \times n}$ to it, which requires less operations in solving for $\hat{\boldsymbol{\Delta\theta}}$. Changing the Hessian in that way is done in the Levenberg-Marquardt method.

2.3.4 Levenberg-Marquardt (LM)

The Levenberg-Marquardt modification of Newton's method can be seen as an automatic way to find the regularizing value λ for dampening. In Newton's method the Hessian is not necessarily positive definit, in that case the optimization may not go in a descent direction. To ensure that the parameter change is in a descent direction, the Hessian is modified to be positive definite by adding a scaled identity matrix λI, such that the new Hessian H_{LM}

of the dampened minimization problem is:

$$H_{LM} = H + \lambda I$$

For Newton's method the value lambda has to be at least as large as the absolute value of the smallest negative eigenvalue of H. In the Gauss-Newton method the Hessian is always positive semi-definite, to ensure that it is positive definite the same modification is performed.

A larger value λ shifts the Gauss-Newton method more towards a Gradient Descent method with small step sizes. Consider the modified update rule for LM:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} \; - \; \left(J(\boldsymbol{\theta_t})^T \; J(\boldsymbol{\theta_t}) + \lambda I\right)^{-1} \; J(\boldsymbol{\theta_t})^T \; \boldsymbol{r}(\boldsymbol{\theta_t}) \tag{2.24}$$

If λ is large, the entries of the Hessian $H = J^T J$ can be neglected, and the update rule becomes:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} \; - \; \frac{1}{\lambda} \; J(\boldsymbol{\theta_t})^T \; \boldsymbol{r}(\boldsymbol{\theta_t}) = \boldsymbol{\theta_t} \; - \; \nabla f(\boldsymbol{\theta_t}) \tag{2.25}$$

Therefore one iteration step with sufficiently large λ has to decrease the error value of the objective function.

An automatic way to find a suitable regularizing value for λ is as follows: Start with a manually chosen initial λ, make one update step and check if the error has decreased. If so, decrease the regularizing value, because it might be possible to make larger steps. If it has not decreased, increase λ by a fixed factor, e.g. 10 and repeat until the error decreases.

2.4 Robust Estimation

Explained here is the estimation of parameters, which is robust with respect to outliers in the data. The assumption for parameter estimation is, that the data is disturbed by noise, whose distribution has zero mean and is bounded. The standard least squares solution is unstable, if the data contains outliers, which do not follow the assumed distribution of noise. In that case, the objective function may be altered, such that data points with large errors contribute less to the estimated solution.

Instead of minimizing the quadratic error

$$\min_{\theta} \sum_{i=1}^{m} (r_i(\boldsymbol{\theta}))^2 \tag{2.26}$$

a different error function ϕ is minimized:

$$\min_{\theta} \sum_{i=1}^{m} \phi(r_i(\boldsymbol{\theta})) \tag{2.27}$$

The minimization of the robust error function ϕ can be performed by weighting each data point (residual) depending on its current absolute value. Details about the derivation

and common robust error functions can be found in [135](also available as html [136]). Instead of minimizing the altered robust error function directly, an iterative re-weighted least-squares problem is solved by introducing a weight w_i for each residual:

$$\min_{\theta} \sum_{i=1}^{m} w_i(r_i)(r_i(\boldsymbol{\theta}))^2 \tag{2.28}$$

Each weight w_i depends on the kind of robust error function and the current error value of the residual in the current iteration. Given in figure 2.2 are the quadratic error function and two robust error functions, which are very popular, namely *Huber* and *Tukey*[135]. Their advantage is, that the error threshold, above which data points are assumed as outliers, corresponds directly to the parameter $k > 0$ in the error function. For the graphs in the figure the parameter was chosen as $k = 1$. The corresponding error and weighting

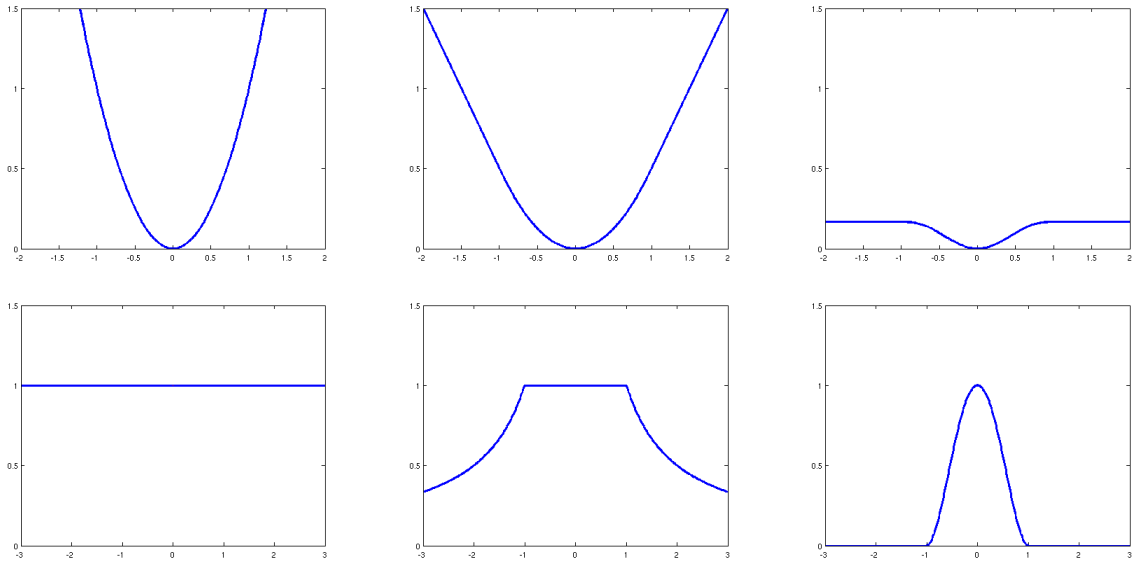


Figure 2.2: Top row: Error functions. Bottom row: weight functions. From left to right: Quadratic, Huber, Tukey.

functions are given in table 2.1.

The Huber function does not eliminate the influence of gross errors completely, because the error function is linear outwards the k-defined interval. The Tukey function eliminates gross errors completely, but has the disadvantage, that if either k is chosen too small, or if the starting point for optimization is too far away from the desired minimum, then important data may not be used for optimization at all. Therefore it may be practical to iterate a few times with a non-bounded function like the *Huber*-function and afterwards refine the solution with a bounded function like *Tukey*.

The weights can be included in the estimation by scaling the rows of the Jacobian and the corresponding values in the residual vector $\boldsymbol{r}$ with $\sqrt{w}$.

	Quadratic	Huber	Tukey
Error function	r^2	$for\ \|r\| \leq k:\ \frac{r^2}{2}$ $for\ \|r\| > k:\ k\|r\| - \frac{k^2}{2}$	$for\ \|r\| \leq k:\ \frac{k^2}{6}\left(1 - \left[1 - \frac{r}{k}^2\right]^3\right)$ $for\ \|r\| > k:\ \frac{k^2}{6}$
Weight function	1	$for\ \|r\| \leq k:\ 1$ $for\ \|r\| > k:\ \frac{k}{\|r\|}$	$for\ \|r\| \leq k:\ \left(1 - \frac{r}{k}^2\right)^2$ $for\ \|r\| > k:\ 0$

Table 2.1: Error functions for robust estimation (top row) for Least Squares, Huber and Tukey and their corresponding weighting functions (bottom row)

2.5 Constrained Optimization

In this work only inequality constraints are of importance. In general these *constrained nonlinear optimization* problems can be expressed like:

$$\min_{\boldsymbol{\theta}}\ m(\boldsymbol{\theta}) \tag{2.29}$$

subject to

$$h(\boldsymbol{\theta}) \geq \mathbf{0} \tag{2.30}$$

where $m : \mathbb{R}^p \to \mathbb{R}$ and $h : \mathbb{R}^p \to \mathbb{R}^c$. The function h constraints the parameters, please note that $h(x) \geq 0$ is to be interpreted component-wise; that is, $h(x) \geq 0$ if and only if $h_i(x) \geq 0$ for each $i = 1, 2, \ldots, c$.

If, for example, the parameters are constrained to lie in a certain interval, such that $a_i \leq \theta_i \leq b_i$ than $h : \mathbb{R}^p \to \mathbb{R}^c$ $(c = 2p)$ would be defined as:

$$h(\boldsymbol{\theta}) = \begin{pmatrix} \theta_1 - a_1 \\ \theta_2 - a_2 \\ \vdots \\ \theta_p - a_p \\ b_1 - \theta_1 \\ b_2 - \theta_2 \\ \vdots \\ b_p - \theta_p \end{pmatrix}.$$

Assume m nonlinear given functions $r_i(\boldsymbol{\theta}), i = 1, .., m$. The *constrained nonlinear least squares* problem can then be solved by:

$$\boldsymbol{\theta}^{\#} = arg\min_{\boldsymbol{\theta}}\ \sum_{i=1}^{m}(r_i(\boldsymbol{\theta}))^2 - \mu\sum_{i=1}^{c} ln(h_i(\boldsymbol{\theta})) \tag{2.31}$$

which is equal to:

$$\min_{\boldsymbol{\theta}}\ \frac{1}{2}\left[\sum_{i=1}^{m}(r_i(\boldsymbol{\theta}))^2 + \mu\sum_{i=1}^{c} ln(h_i(\boldsymbol{\theta})^{-1})\right] = \min_{\boldsymbol{\theta}}\ C(\boldsymbol{\theta}) \tag{2.32}$$

with $C : \mathbb{R}^p \to \mathbb{R}$ and c additional residual functions, which represent the constrained equations. If the problem is solved iteratively for decreasing μ, then the solution of the minimization $\boldsymbol{\theta}^{\#}$ is within a decreasing small region next to the real solution $\hat{\boldsymbol{\theta}}$. A discussion on the convergence rate of logarithmic barrier methods can be found in [132].

2.5.1 Jacobian for the Constrained Problem

To apply Newton's method the first and second derivatives are needed. The j-th component of the gradient is

$$\begin{aligned}
(\nabla\, C(\boldsymbol{\theta}))_j &= \sum_{i=1}^{m} \frac{\partial r_i(\boldsymbol{\theta})}{\partial \theta_j} r_i(\boldsymbol{\theta}) + \frac{\mu}{2} \sum_{i=1}^{c} \frac{\partial h_i(\boldsymbol{\theta})^{-1}}{\partial \theta_j} h_i(\boldsymbol{\theta}) \Leftrightarrow \\
(\nabla\, C(\boldsymbol{\theta}))_j &= \sum_{i=1}^{m} \frac{\partial r_i(\boldsymbol{\theta})}{\partial \theta_j} r_i(\boldsymbol{\theta}) + \frac{\mu}{2} \sum_{i=1}^{c} \frac{-1}{h_i(\boldsymbol{\theta})^2} \frac{\partial h_i(\boldsymbol{\theta})}{\partial \theta_j} h_i(\boldsymbol{\theta}) \Leftrightarrow \\
(\nabla\, C(\boldsymbol{\theta}))_j &= \sum_{i=1}^{m} \frac{\partial r_i(\boldsymbol{\theta})}{\partial \theta_j} r_i(\boldsymbol{\theta}) + \sum_{i=1}^{c} \frac{-\mu}{2h_i(\boldsymbol{\theta})^2} \frac{\partial h_i(\boldsymbol{\theta})}{\partial \theta_j} h_i(\boldsymbol{\theta})
\end{aligned} \tag{2.33}$$

Minimization can then again take place as above by solving each iteration a linear least squares problem involving the Jacobian. Now the Jacobian at iteration t also includes the constraining function h:

$$J_c = \begin{bmatrix}
\frac{\partial r_1(\boldsymbol{\theta^t})}{\partial \theta_1} & \frac{\partial r_1(\boldsymbol{\theta^t})}{\partial \theta_2} & \cdots & \frac{\partial r_1(\boldsymbol{\theta^t})}{\partial \theta_p} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\partial r_m(\boldsymbol{\theta^t})}{\partial \theta_1} & \frac{\partial r_m(\boldsymbol{\theta^t})}{\partial \theta_2} & \cdots & \frac{\partial r_m(\boldsymbol{\theta^t})}{\partial \theta_p} \\
\frac{-\mu}{2h_1(\theta^t)^2} \frac{\partial h_1(\boldsymbol{\theta^t})}{\partial \theta_1} & \frac{-\mu}{2h_1(\theta^t)^2} \frac{\partial h_1(\boldsymbol{\theta^t})}{\partial \theta_2} & \cdots & \frac{-\mu}{2h_1(\theta^t)^2} \frac{\partial h_1(\boldsymbol{\theta^t})}{\partial \theta_p} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{-\mu}{2h_c(\theta^t)^2} \frac{\partial h_c(\boldsymbol{\theta^t})}{\partial \theta_1} & \frac{-\mu}{2h_c(\theta^t)^2} \frac{\partial h_c(\boldsymbol{\theta^t})}{\partial \theta_2} & \cdots & \frac{-\mu}{2h_c(\theta^t)^2} \frac{\partial h_c(\boldsymbol{\theta^t})}{\partial \theta_p}
\end{bmatrix} \tag{2.34}$$

In the special case that the parameters are constrained to lie in a certain interval, such that h is defined as in the previous section, the Jacobian simplifies to:

$$J_c = \begin{bmatrix}
\frac{\partial r_1(\boldsymbol{\theta^t})}{\partial \theta_1} & \frac{\partial r_1(\boldsymbol{\theta^t})}{\partial \theta_2} & \cdots & \frac{\partial r_1(\boldsymbol{\theta^t})}{\partial \theta_p} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\partial r_m(\boldsymbol{\theta^t})}{\partial \theta_1} & \frac{\partial r_m(\boldsymbol{\theta^t})}{\partial \theta_2} & \cdots & \frac{\partial r_m(\boldsymbol{\theta^t})}{\partial \theta_p} \\
\frac{-\mu}{2h_1(\theta^t)^2} & 0 & \cdots & 0 \\
0 & \frac{-\mu}{2h_2(\theta^t)^2} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \frac{-\mu}{2h_p(\theta^t)^2} \\
\frac{\mu}{2h_{p+1}(\theta^t)^2} & 0 & \cdots & 0 \\
0 & \frac{\mu}{2h_{p+2}(\theta^t)^2} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \frac{\mu}{2h_c(\theta^t)^2}
\end{bmatrix} \tag{2.35}$$

If we now define $c(\boldsymbol{\theta}) = (r_1(\boldsymbol{\theta}), .., r_m(\boldsymbol{\theta}), h_1(\boldsymbol{\theta}), .., h_c(\boldsymbol{\theta}))^T$, then the gradient may be written as:

$$\nabla\, C(\boldsymbol{\theta}) = J_c^T c(\boldsymbol{\theta}) \tag{2.36}$$

With this expressions the Gauss-Newton may be applied to solve the constrained problem.

Chapter 3

Pose Estimation

In this chapter the pose estimation of rigid and articulated objects is described. Pose estimation of articulated objects is very similar to solving inverse kinematic problems, which are well known from robotics. However, in robotics one is usually interested in calculating joint angles for a given end effector position and orientation (pose). While there exist closed-form analytic solutions for robots with revolute and prismatic joints with up to 6 degrees of freedom [26], configurations with more degrees of freedom require usually iterative optimization methods, which are a research field of their own.

In this chapter pose estimation methods are presented that are more related to problems in computer vision, where pose is to be estimated from different kind of data, which is calculated from images. At first pose estimation of rigid bodies is explained and extended in later sections to articulated objects. The chosen motion function and optimization method, allows to include second order motion derivatives within the pose estimation, whose benefits are analyzed. The chapter is concluded by a comparison of different optimization methods. Applied experimental results are given in the next chapter *Tracking*.

3.1 Rigid Bodies

A rigid body motion (RBM) in $\mathbb{R}^3$ is a transformation of an object, that keeps the distances between all points of the object constant. There are different formulations for rigid body motions, e.g. twists [98, 17], which use an exponential term e^{ψ}, or rotors [111], which may be seen as an extension of quaternions. Most common are transformation matrices $T \in \mathbb{R}^{4\times4}$ in homogenous coordinates, which are described in the Appendix B.1. Given here is another description that allows a straightforward application within *Nonlinear Least Squares*.

3.1.1 Rotation around Arbitrary Axis

Of importance in this work are rotations around arbitrary axes in space. The motion of a point around an arbitrary axis in space is a circular motion on a plane in 3D space as

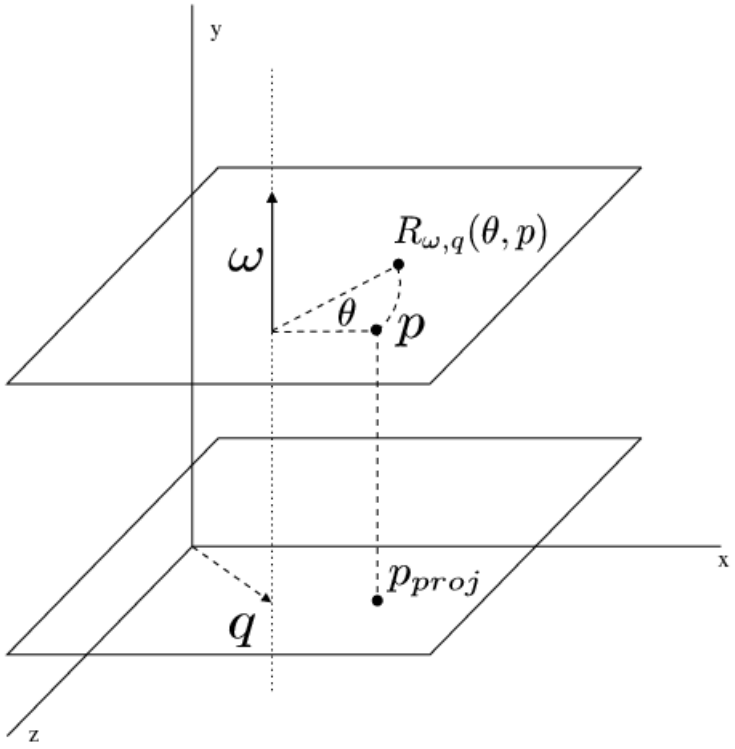


Figure 3.1: Rotation around an arbitrary axis in space.

shown in Figure 3.1. Consider the normal vector $\boldsymbol{\omega} \in \mathbb{R}^3$, which describes the direction of the axis, and the point $\boldsymbol{q} \in \mathbb{R}^3$ on the axis, which has the shortest distance to the origin, i.e. $\boldsymbol{q}$ lies on the axis and $\boldsymbol{q}^T\boldsymbol{\omega} = 0$, refer to Figure. The rotation of a point $\boldsymbol{p}$ around that axis with angle θ may then be written as

$$\begin{aligned} R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta, \boldsymbol{p}) &= \boldsymbol{p} + \sin(\theta)(\boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{q})) + (1 - \cos(\theta))(\boldsymbol{q} - \boldsymbol{p_{proj}}) \\ &= \boldsymbol{p} + \sin(\theta)(\boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{q})) + (\boldsymbol{q} - \boldsymbol{p_{proj}}) - \cos(\theta)(\boldsymbol{q} - \boldsymbol{p_{proj}}) \end{aligned} \tag{3.1}$$

where $\boldsymbol{p_{proj}} = \boldsymbol{p} - (\boldsymbol{p}^T\boldsymbol{\omega})\boldsymbol{\omega}$ is the projection of $\boldsymbol{p}$ onto the plane through the origin with normal $\boldsymbol{\omega}$. Note that $\boldsymbol{q}$ is also on that plane. A very similar equation for a rotation around an axis, which goes through the origin, is known as the *Rodriguez' formula* [98]. This expression is very useful as the derivative $\frac{\partial R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta)}{\partial \theta}$ is easy to calculate:

$$\frac{\partial R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta)}{\partial \theta} = \cos(\theta)(\boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{q})) + \sin(\theta)(\boldsymbol{q} - \boldsymbol{p_{proj}}) \tag{3.2}$$

with the special derivative at zero:

$$\left.\frac{\partial R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta)}{\partial \theta}\right|_{\theta=0} = \boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{q}) = \boldsymbol{\omega} \times \boldsymbol{p} - \boldsymbol{\omega} \times \boldsymbol{q} \tag{3.3}$$

The cross product $\boldsymbol{\omega} \times \boldsymbol{q}$ is also known as *momentum*. Please note that $\boldsymbol{q}$ can be any point on the axis as it does neither change the direction nor the magnitude of the momentum.

The derivative at zero gives a vector, which is the velocity of the rotating point (for an angular rotation velocity of one). It has the same direction as the the tangent on the circle at the point's position.

3.1.2 Pose and Rigid Motion

The position and orientation of an object can be described by a rotation and a translation with respect to some coordinate system. Let $\boldsymbol{p_{obj}}$ be an object point in object coordinates. If the orientation of the object coordinate system is given in Euler angles (α, β, γ) and the position is $\boldsymbol{t} = (\theta_x, \theta_y, \theta_z)^T$, then the position of $\boldsymbol{p_w}$ with respect to the world coordinate system is given by

$$\boldsymbol{p_w} = (\theta_x, \theta_y, \theta_z)^T + \left(R_{e_x}(\alpha) \circ R_{e_y}(\beta) \circ R_{e_z}(\gamma)\right)(\boldsymbol{p_{obj}}) \tag{3.4}$$

where $\boldsymbol{e_x}, \boldsymbol{e_y}, \boldsymbol{e_z}$ denote the axes of the world coordinate system and the operator $\circ$ denotes function concatenation. The rotations around the axes of the world coordinate system can be described by a rotation matrix, such that the function concatenation becomes a matrix multiplication.

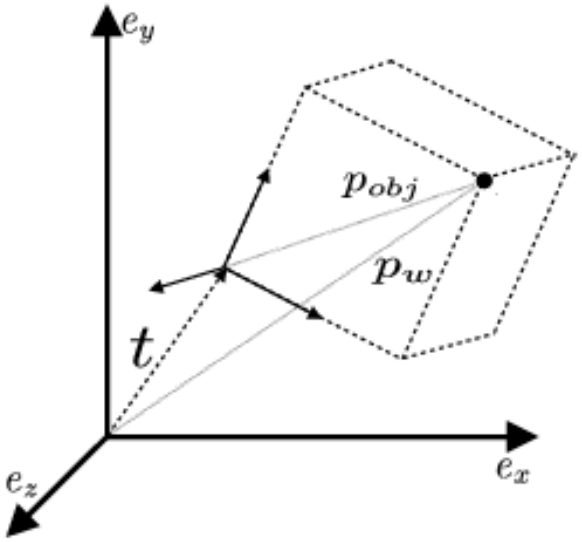


Figure 3.2: Pose of an rigid object: Object and world coordinate systems

Figure 3.2 illustrates this relation. This coordinate transform may also be interpreted as a motion: The angles α, β, γ give the rotation of a coordinate system, that is aligned with the world, into the coordinate system of the object. The rotation of such a third

coordinate system is with respect to the world coordinate axes. The order of rotation is in that case, first around the z-axis, second around the fixed y-axis of the world and third around the fixed x-axis. With respect to this definition $\boldsymbol{t}$ and α, β, γ define the *pose* of an object.

Now assume the object moves in space to another position and orientation. Let $\boldsymbol{t_0}, \alpha_0, \beta_0, \gamma_0$ be the initial pose and $\boldsymbol{t_1}, \alpha_1, \beta_1, \gamma_1$ the new pose. The motion of an object point in world coordinates from the initial pose to the new pose can be described by

$$\boldsymbol{p_{w1}} = (\theta_x, \theta_y, \theta_z)^T + \left(R_{\omega_x}(\theta_\alpha) \circ R_{\omega_y}(\theta_\beta) \circ R_{\omega_z}(\theta_\gamma)\right)(\boldsymbol{p_{w0}}) \tag{3.5}$$

Here $\omega_x, \omega_y, \omega_z$ denote that the center of rotation is not necessarily the origin of the world coordinate system. If the rotation axes are arbitrary, the function concatenation denoted by the operator $\circ$ can not be replaced by a rotation matrix, but rather by a rotation and translation or the expression of Equation 3.1.

In this work the estimation of pose is always with respect to an initial known pose. Estimating the new pose is done by estimating the relative motion of the object leading to the new pose. In the following the *motion function* will be

$$\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_{w0}}) = (\theta_x, \theta_y, \theta_z)^T + \left(R_{\omega_x}(\theta_\alpha) \circ R_{\omega_y}(\theta_\beta) \circ R_{\omega_z}(\theta_\gamma)\right)(\boldsymbol{p_{w0}}) \tag{3.6}$$

where the operator $\circ$ denotes function concatenation. Additionally it is always assumed in the following, that points are given in world coordinates with respect to a known pose. Therefore, $\boldsymbol{p_{w0}}$ will be just written as $\boldsymbol{p}$.

3.2 Rigid Motion by Nonlinear Least Squares

In this work estimation of pose relies on correspondences and its parameters are found by minimizing an objective function. The minimization is a Nonlinear Least Squares problem and is solved with Newton's method (see previous chapter). Given in the following are the objective functions for different types of input data and their partial derivatives (Jacobians), which are necessary for the minimization. At first, estimation from 3D-point-3D-point correspondences $(\boldsymbol{p_i}, \tilde{\boldsymbol{p}}_{\boldsymbol{i}})$ is given (Figure 3.3). Pose estimation means here to find the rigid motion, which transforms all points $\boldsymbol{p_i}$ as close as possible to their corresponding points $\tilde{\boldsymbol{p}}_{\boldsymbol{i}}$. For a given set of correspondences the *Nonlinear Least Squares method* (compare with Equation (2.12)) can be applied to estimate that motion. A general *Nonlinear Least Squares Problem* reads:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \sum_{i=1}^{m} (r_i(\boldsymbol{\theta}))^2$$

In case of pose estimation from corresponding 3D data points $(\boldsymbol{p_i}, \tilde{\boldsymbol{p}}_{\boldsymbol{i}})$, as shown in Figure 3.3, the *residual* functions r_i are of the form:

$$\boldsymbol{r_i}(\boldsymbol{\theta}) = (\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{p}}_{\boldsymbol{i}}) \tag{3.7}$$

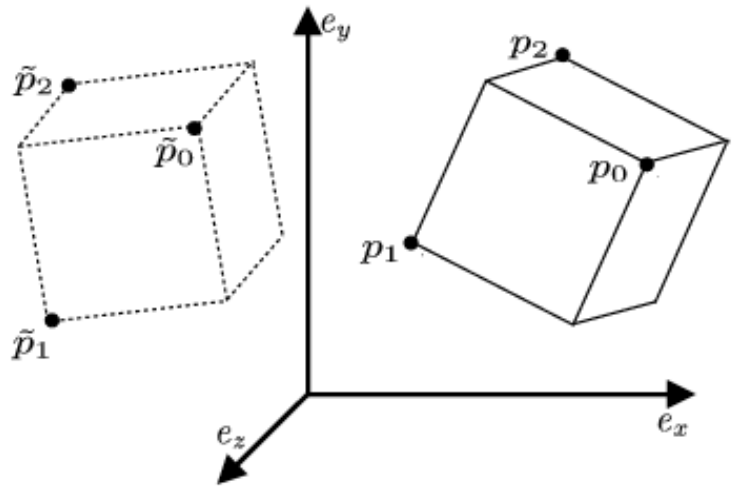


Figure 3.3: Rigid motion of an object and corresponding points.

with $\boldsymbol{r_i}(\boldsymbol{\theta}) = (r_{ix}, r_{iy}, r_{iz})^T$ and $\boldsymbol{\theta} = (\theta_x, \theta_y, \theta_z, \theta_\alpha, \theta_\beta, \theta_\gamma)^T$ there are $3m$ real-valued *residual* functions for minimization. The motion function $\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_i})$ of the point $\boldsymbol{p_i}$ is given in Equation (3.6). *Newton's method* can be applied to find the minimizer $\hat{\boldsymbol{\theta}}$, which requires the first and second derivatives of the residual functions.

3.2.1 Jacobian for the Rigid Motion

The Jacobian matrix of partial derivatives is given in the following for a point set with one correspondence $(\boldsymbol{p}, \tilde{\boldsymbol{p}})$. For point sets with m points the Jacobian is simply extended by additional rows for the additional correspondences.

The derivative for the residual of rigid motion is:

$$\frac{\partial \boldsymbol{r}(\boldsymbol{\theta})}{\partial \theta_j} = \frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_j} = \frac{\partial (\theta_x, \theta_y, \theta_z)^T + \left(R_{\omega_x}(\theta_\alpha) \circ R_{\omega_y}(\theta_\beta) \circ R_{\omega_z}(\theta_\gamma)\right)(\boldsymbol{p})}{\partial \theta_j} \tag{3.8}$$

Therefore, the Jacobian for one point-point correspondence is

$$J = \begin{bmatrix} 1 & 0 & 0 & \frac{\partial m_x}{\partial \theta_\alpha} & \frac{\partial m_x}{\partial \theta_\beta} & \frac{\partial m_x}{\partial \theta_\gamma} \\ 0 & 1 & 0 & \frac{\partial m_y}{\partial \theta_\alpha} & \frac{\partial m_y}{\partial \theta_\beta} & \frac{\partial m_y}{\partial \theta_\gamma} \\ 0 & 0 & 1 & \frac{\partial m_z}{\partial \theta_\alpha} & \frac{\partial m_z}{\partial \theta_\beta} & \frac{\partial m_z}{\partial \theta_\gamma} \end{bmatrix} \tag{3.9}$$

and for the derivatives at zero:

$$\begin{aligned}
\left.\frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_\alpha}\right|_{\boldsymbol{\theta}=\mathbf{0}} &= \boldsymbol{\omega_x} \times (\boldsymbol{p} - \boldsymbol{q_x}) \\
\left.\frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_\beta}\right|_{\boldsymbol{\theta}=\mathbf{0}} &= \boldsymbol{\omega_y} \times (\boldsymbol{p} - \boldsymbol{q_y}) \\
\left.\frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_\gamma}\right|_{\boldsymbol{\theta}=\mathbf{0}} &= \boldsymbol{\omega_z} \times (\boldsymbol{p} - \boldsymbol{q_z})
\end{aligned} \tag{3.10}$$

Here $\boldsymbol{\omega_x}, \boldsymbol{q}_x$ denote rotation around an arbitrary point in space and $\boldsymbol{\omega_x}, \boldsymbol{\omega_y}, \boldsymbol{\omega_z}$ are assumed orthogonal. If $\boldsymbol{\omega_x}, \boldsymbol{\omega_y}, \boldsymbol{\omega_z}$ are the world coordinate axes then the three Equations above are equal to the linearized rotation matrix as commonly used, e.g. like in [76].

The minimizer is then found by iteratively solving

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} \; - \; \left(J^T \; J\right)^{-1} \; J^T \; \boldsymbol{r}(\boldsymbol{\theta_t}) \tag{3.11}$$

3.2.2 Relative Motion and Derivative at Zero

The Jacobian above is only valid, if the derivative is taken at zero. This can be achieved by estimating only relative motion in each iteration step, such that $\boldsymbol{\theta_t} = 0$ and $\boldsymbol{\theta_{t+1}} = \Delta\boldsymbol{\theta_t}$.

Therefore, recalculation of the world coordinates of the point $\boldsymbol{p}$ at each iteration is required. The corresponding observed point $\tilde{\boldsymbol{p}}$ stays unchanged. If the rotation center is $\boldsymbol{c}$, which is the intersection of $\boldsymbol{\omega_x}, \boldsymbol{\omega_y}, \boldsymbol{\omega_z}$, the final motion is obtained by

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{t+1}} = \hat{\boldsymbol{\theta}}_{\boldsymbol{t}} + \Delta\boldsymbol{\theta_t} \tag{3.12}$$

and

$$\boldsymbol{p_{t+1}} = \left(R_{\boldsymbol{\omega_x}}(\hat{\theta}_{\alpha,t+1}) \circ R_{\boldsymbol{\omega_y}}(\hat{\theta}_{\beta,t+1}) \circ R_{\boldsymbol{\omega_z}}(\hat{\theta}_{\gamma,t+1})\right)(\boldsymbol{p_{t_0}} - \boldsymbol{c}) + \boldsymbol{c} \tag{3.13}$$

where $\boldsymbol{p_{t_0}}$ is the initial point coordinate $\boldsymbol{p}$. In the last Equation it is more efficient to use rotation matrices for evaluating the new points than Equation (3.1).

3.2.3 Geometric Interpretation

Newton's Method approximates the objective function by a quadratic function and iterates until convergence. Shown here is the development of the iterative approach for a rotation with angle θ around a fixed axis $(\boldsymbol{\omega}, \boldsymbol{q})$ and a known 3D-point-3D-point correspondence $(\tilde{\boldsymbol{p}}, \boldsymbol{p})$.

Because a nonlinear problem is considered of estimating the rotation angle around a fixed known axis in 3D space, the motion of a point on that axis is a motion on a plane. Shown in the following Figures is only the development on that plane. The error vector $\boldsymbol{r}(\boldsymbol{\theta}) = (\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p}) - \tilde{\boldsymbol{p}})$ is not within that plane. However for optimization only the projection onto the plane is of importance, because the resulting minimizing point lies in that plane.

The problem is shown in Figure 3.4, where the vector $\boldsymbol{v} = \boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{q})$ is the velocity vector of the rotational motion of the model point $\boldsymbol{p}$. Consider now the projection of $\boldsymbol{r}$ onto $\boldsymbol{v}$, which reads:

$$\boldsymbol{r}'_{\boldsymbol{v}}(\theta) = \frac{\boldsymbol{r}^T\boldsymbol{v}}{\boldsymbol{v}^T\boldsymbol{v}}\,\boldsymbol{v} \tag{3.14}$$

One iteration of the Gauss-Newton results in the parameter change:

$$\Delta\theta = -(J^TJ)^{-1}J^T\,\boldsymbol{r}(\theta) = -(\boldsymbol{v}^T\boldsymbol{v})^{-1}\boldsymbol{v}^T\boldsymbol{r}(\theta) = -\frac{\boldsymbol{r}^T\boldsymbol{v}}{\boldsymbol{v}^T\boldsymbol{v}} \tag{3.15}$$

Hence $\Delta\theta$ is the scale value of $\boldsymbol{v}$ that gives the projection of the error $\boldsymbol{r}$ onto $\boldsymbol{v}$ and $\hat{\boldsymbol{p}} = \boldsymbol{p} - \boldsymbol{r}'_{\boldsymbol{v}}(\theta) = \boldsymbol{p} + \Delta\theta\,\boldsymbol{v}$.

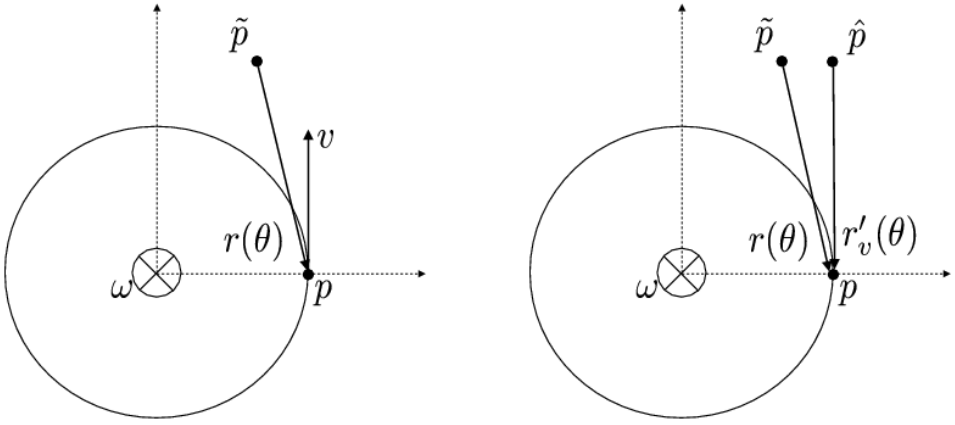


Figure 3.4: Estimating the rotation angle for a 3D-point-3D-point correspondence

Inserting $\Delta\theta$ in the motion Equation (3.1) gives the starting point p_2 for the next iteration as shown in Figure 3.5 left. In the example shown $\Delta\theta_1$ is 1.36 *rad* or 78 *deg*.

In the second iteration the projection of $\boldsymbol{r_2}$ onto $\boldsymbol{v}$ leads to the minimizer $\hat{\boldsymbol{p}}_{\boldsymbol{2}}$ that is almost on the circle. After insertion of $\Delta\theta_2$, which is -0.27 *rad* or -15.6 *deg*, the next starting point $\boldsymbol{p_3}$ for iteration 3 is found as shown in Figure 3.5 right. This point is so near to the final correct solution, that a drawing of further iterations is not done here.

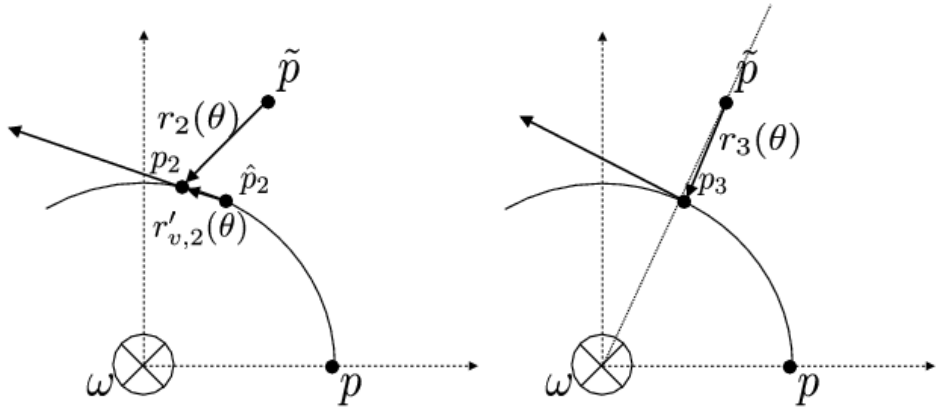


Figure 3.5: Estimating the rotation angle for a 3D-point-3D-point correspondence. Iteration 2 and 3.

3.3 Pose under Projection

In this section the pose of a rigid object is calculated by estimating the relative motion of 3D points. Here the 3D points are observed by a camera, such that the corresponding points are 2D points in the image plane. The approach can be applied also to a moving camera and a fixed object (or scene) as explained in the section *Camera Calibration*. The objective functions and their Jacobians are given first for 3D-point-2D-point correspondences and later for 3D-point-2D-line correspondences.

3.3.1 Rigid Motion under Projection

Assume the point $\boldsymbol{p} \in \mathbb{R}^3$ is observed by a pinhole camera and its projection onto the 2D image plane is $\boldsymbol{p'} \in \mathbb{R}^2$. If the camera is positioned at the origin and aligned with the world, such that the optical axis is the world z-axis, the combination of camera projection and 3D rigid motion of the point can be written as :

$$\boldsymbol{m'}(\boldsymbol{p}, \boldsymbol{\theta}) = \begin{pmatrix} s_x \frac{m_x(\boldsymbol{p},\boldsymbol{\theta})}{m_z(\boldsymbol{p},\boldsymbol{\theta})} + c_x \\ s_y \frac{m_y(\boldsymbol{p},\boldsymbol{\theta})}{m_z(\boldsymbol{p},\boldsymbol{\theta})} + c_y \end{pmatrix} \tag{3.16}$$

where $\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p}) = (m_x, m_y, m_z)^T$ is the motion function from Equation (3.6), s_x, s_y is the focal length of the camera in x- and y-direction expressed in pixel units and $(c_x, c_y)^T$ is the principal point of the camera. Assumed here is, that the rows and columns of the image sensor are orthogonal to each other (skew is zero).

3.3.2 Pose Estimation from 2D-3D Correspondences

If an object with known geometry moves in space from a known pose to a new pose and its known 3D points $\boldsymbol{p_i}$ are observed in an image at $\tilde{\boldsymbol{p}}_{\boldsymbol{i}}'$, its relative motion can be estimated

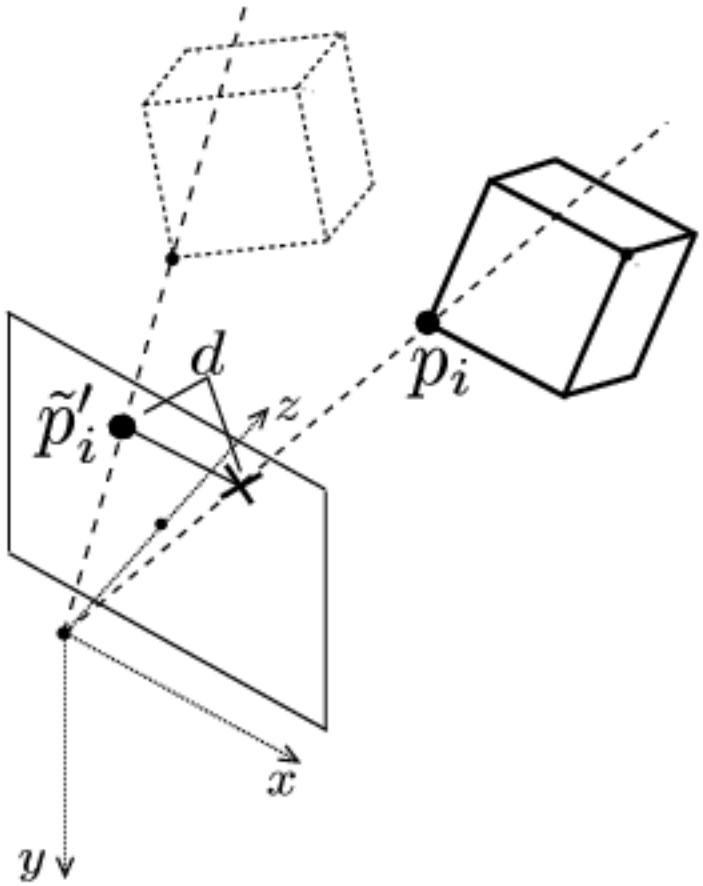


Figure 3.6: Object motion and 2D-3D correspondences

by Nonlinear Least Squares using the 2D-3D correspondence $(\tilde{\boldsymbol{p}}_{\boldsymbol{i}}', \boldsymbol{p}_{\boldsymbol{i}})$ with the following residual. See also Figure 3.6.

$$d = \boldsymbol{r}_{\boldsymbol{i}}(\boldsymbol{\theta}) = (\boldsymbol{m}'(\boldsymbol{p}_{\boldsymbol{i}}, \boldsymbol{\theta}) - \tilde{\boldsymbol{p}}_{\boldsymbol{i}}') \tag{3.17}$$

The necessary Jacobian for the optimization is given now for a single correspondence. For m correspondences the Jacobian is simply extended by additional rows. With $\boldsymbol{\theta} = (s_x, s_y, c_x, c_y, \theta_x, \theta_y, \theta_z, \theta_\alpha, \theta_\beta, \theta_\gamma)^T$ the Jacobian reads

$$\begin{aligned} J &= \begin{bmatrix} \frac{\partial m'}{\partial s_x} & \frac{\partial m'}{\partial s_y} & \frac{\partial m'}{\partial c_x} & \frac{\partial m'}{\partial c_y} & \frac{\partial m'}{\partial \theta_x} & \frac{\partial m'}{\partial \theta_y} & \frac{\partial m'}{\partial \theta_z} & \frac{\partial m'}{\partial \theta_\alpha} & \frac{\partial m'}{\partial \theta_\beta} & \frac{\partial m'}{\partial \theta_\gamma} \end{bmatrix} \\ &= \begin{bmatrix} \frac{m_x}{m_z} & 0 & 1 & 0 & \frac{s_x}{m_z} & 0 & s_x \frac{-m_x}{m_z^2} & \frac{\partial m'_x}{\partial \theta_\alpha} & \frac{\partial m'_x}{\partial \theta_\beta} & \frac{\partial m'_x}{\partial \theta_\gamma} \\ 0 & \frac{m_y}{m_z} & 0 & 1 & 0 & \frac{s_y}{m_z} & s_y \frac{-m_y}{m_z^2} & \frac{\partial m'_y}{\partial \theta_\alpha} & \frac{\partial m'_y}{\partial \theta_\beta} & \frac{\partial m'_y}{\partial \theta_\gamma} \end{bmatrix} \end{aligned} \tag{3.18}$$

and

$$\frac{\partial \boldsymbol{m}'}{\partial \theta_j} = \begin{pmatrix} \frac{\partial\left(s_x \frac{m_x}{m_z}\right)}{\partial \theta_j} \\ \\ \frac{\partial\left(s_y \frac{m_y}{m_z}\right)}{\partial \theta_j} \end{pmatrix} = \begin{pmatrix} \frac{s_x\left(\frac{\partial m_x}{\partial \theta_j} m_z - m_x \frac{\partial m_z}{\partial \theta_j}\right)}{m_z^2} \\ \\ \frac{s_y\left(\frac{\partial m_y}{\partial \theta_j} m_z - m_y \frac{\partial m_z}{\partial \theta_j}\right)}{m_z^2} \end{pmatrix} \tag{3.19}$$

The partial derivatives $\frac{\partial \boldsymbol{m}}{\partial \theta_j}, j \in \{\alpha, \beta, \gamma\}$ are given in Equation (3.10). The Jacobian above does not only allow estimation of the pose of an object, but in addition the estimation of the internal camera parameters. Additionally the formulation allows to estimate only a subset of the parameters, if some of them are known, e.g. rotation only or fixed principal point etc.

As the Jacobian above requires derivatives at zero, the estimation should be done relative to the previous estimate iteratively as described in section 3.2.2.

This analytically derived Jacobian for the problem of camera calibration was already published 1996 as an extension of Lowe's pose estimation algorithm [7]. However it is given here in detail, because its extension to articulated objects is a contribution of this work and is given in the next sections.

3.3.3 Approximation by 3D-point-3D-line Correspondences

An alternative approach to estimate pose from 2D-3D correspondences is a minimization of 3D-point-3D-line distances as in [111], this approximates the observed error in the image plane by a distance in 3D space. However, it has disadvantages in certain cases as explained below.

If the inverse projection function is known, as e.g. for a calibrated pinhole or fisheye camera, it is possible to calculate the 3D viewing ray for the known 2D image point. Assume the viewing ray is described by its normalized direction vector $\boldsymbol{\omega}$ and a point $\boldsymbol{q_\omega}$ on the ray. The distance d between a point p and the viewing ray is

$$d = |\, \boldsymbol{\omega} \times \boldsymbol{p} - \boldsymbol{\omega} \times \boldsymbol{q_\omega}| \tag{3.20}$$

Using this distance the pose estimation problem by *Nonlinear Least Squares* with correspondences $((\tilde{\boldsymbol{\omega}}_{\boldsymbol{i}}, \tilde{\boldsymbol{q}}_{\boldsymbol{\omega},\boldsymbol{i}}), \boldsymbol{p_i})$ is:

$$\min_{\boldsymbol{\theta}} \sum_i |\, \tilde{\boldsymbol{\omega}}_{\boldsymbol{i}} \times \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{\omega}}_{\boldsymbol{i}} \times \tilde{\boldsymbol{q}}_{\boldsymbol{\omega},\boldsymbol{i}}|^2 \tag{3.21}$$

Because the cross product $\tilde{\boldsymbol{\omega}}_{\boldsymbol{i}} \times \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_i})$ can also be described by a matrix multiplication (see Appendix A.2), it is obvious that this minimization problem is very close to that of Equation (3.7).

Let $[\tilde{\omega}\times]^{3x3}$ be the cross product matrix of one correspondence. The necessary Jacobian J for the Gauss-Newton Method then is:

$$J = [\tilde{\omega}\times]^{3x3} J_{3.9} \tag{3.22}$$

where $J_{3.9}$ is the Jacobian for 3D point - 3D point correspondences from Equation (3.9). For additional correspondence the Jacobian is extended by 3 additional rows for each correspondence.

The minimization in 3D space is less accurate compared to the minimization in 2D space, if the 2D points are disturbed by noise [127], because correspondences of 3D-points, which are far away from the projection center, have more influence on the solution. In [127] a down weighting of these correspondences is proposed, which is similar to the correct solution as given in this thesis.

Therefore, the minimization in 3D space should be applied only, if the projection function is difficult to derive, e.g. for fisheye cameras. In case of pinhole cameras the minimization in the image plane, where the error is observed, is more accurate.

However, the inaccuracies of minimization in 3D space are only significant in the presence of noisy correspondences and if the extent of the object in depth is large compared with the distance from object to projection center (camera position) [127].

3.3.4 Camera Calibration

So far the motion of an object has been estimated. The same approach can be used to calibrate the internal and external parameters of a pinhole camera. If the camera moves and the object is standing still, the approach above, which estimates the object's motion, can still be applied, because the object motion is the inverse of the camera's.

In Equation (3.16) it is assumed that the camera is aligned with the world coordinate system. If this isn't the case, the object points have to be transformed in the following way.

If the pose of the camera coordinate system is given by a rotation matrix R_c and a center point $\boldsymbol{c}$, the points $\boldsymbol{p_i}$ have to be transformed by $\boldsymbol{u_i} = R_c^T \boldsymbol{p_i} - R_c^T \boldsymbol{c}$. Then the Gauss-Newton method can be applied with the Jacobian as above to the points $\boldsymbol{u_i}$ and the same corresponding 2D image points. The estimated motion $R_r, \boldsymbol{t_r}$ is the motion of an object. Because the case described here is motion of the camera, the resulting motion $R_r, \boldsymbol{t_r}$ of the object has to be taken inverse and transformed back, such that the new pose $\hat{R}_c, \hat{\boldsymbol{c}}$ for the camera is:

$$\hat{R}_c = R_c R_r^T \tag{3.23}$$

$$\hat{\boldsymbol{c}} = \boldsymbol{c} - \hat{R}_c \boldsymbol{t_r} \tag{3.24}$$

For numerical stability during the optimization it is necessary to have a Jacobian matrix with entries of about the same absolute value. Depending on the unit of the points position, e.g. meters or centimeters, it may happen, that the derivative for the rotation is very large or very small compared to the derivative for the translation, which equals one.

To overcome this problem the 3D point set $\boldsymbol{u_i}$ is scaled, such that its mean $\bar{\boldsymbol{u}}$ has a distance of one to the camera. Let $s = |\bar{u}|$ be the necessary scale, then the resulting translation is $\hat{\boldsymbol{c}} = \boldsymbol{c} - s\hat{R}_c \boldsymbol{t_r}$ and the rotation $\hat{R}_c$ is the same as above.

The optimization method, which estimates the pose under projection, needs an initial guess of the motion. This initial pose is important, because the optimization may converge

into local minima that are not the desired pose. For projection with pinhole cameras, the projection equation does not incorporate whether the object is in front or behind the camera. Therefore, a valid pose with a local minima is usually found also if the object is positioned behind the camera.

There is a linear method available, that calculates an initial pose, whose orientation is close to the real one and whose position is in front of the camera. The method can be found in more detail in the work of Perwass [101], where a description in Geometric Algebra is used. For completeness a description in Euclidean space is given in the Appendix C. In comparison to other pose estimation algorithms, e.g. the POSIT algorithm [32, 29], the approach here estimates a valid pose from only three correspondences, if the internal camera parameters are known, while POSIT needs four non-coplanar points and estimates only a scaled orthographic projection instead of the full perspective one. However, if the focal length of the camera is not known, the approach given here needs also 4 non-coplanar points and an initial guess of the focal length.

3.3.5 Estimation from 3D-point-2D-line Correspondences

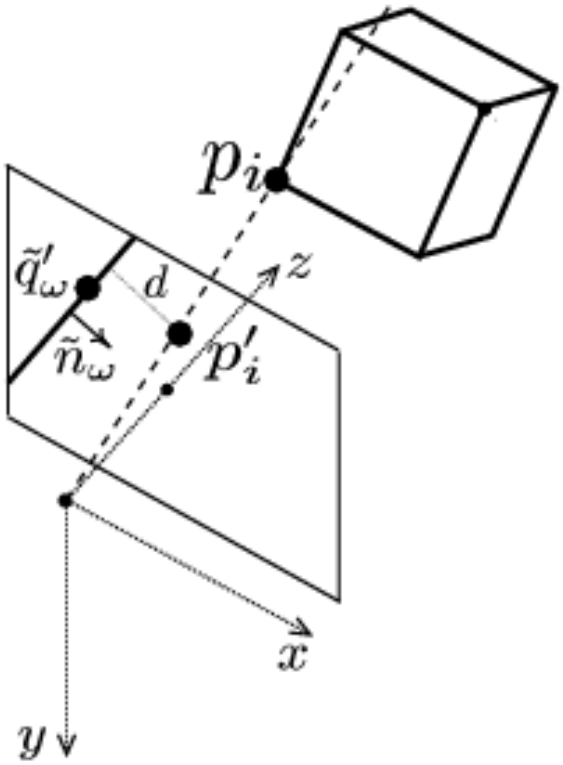


Figure 3.7: Pose estimation from corresponding 3D-point and 2D-line

Let $\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}} \in \mathbb{R}^2$ be the normal vector of a 2D line and $\tilde{\boldsymbol{q}}'_{\boldsymbol{\omega}} \in \mathbb{R}^2$ one point on this line (as

described in Appendix A). If the distance of a projected point $\boldsymbol{p'_i} = \boldsymbol{m'}(\mathbf{0}, \boldsymbol{p_i})$ to this line is to be minimized as shown in Figure 3.7, then the objective function for minimization is

$$\min_{\boldsymbol{\theta}} \sum_i \left(\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}{}^T \left(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega}} \right) \right)^2 \tag{3.25}$$

with correspondences $((\tilde{\boldsymbol{n}}_{\boldsymbol{\omega},\boldsymbol{i}}, \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega},\boldsymbol{i}})\ ,\ \boldsymbol{p_i})$.

The residual function is the distance of $\boldsymbol{p_i}$ to the line in the direction of the normal:

$$\begin{aligned} r_i(\boldsymbol{\theta}) &= \tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}{}^T \left(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega}} \right) \Leftrightarrow & (3.26)\\ &= \tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}{}^T \boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}\ \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega}} \Leftrightarrow & (3.27)\\ &= (\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}})_x\ m'_x(\boldsymbol{\theta}, \boldsymbol{p_i}) + (\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}})_y\ m'_y(\boldsymbol{\theta}, \boldsymbol{p_i}) - d_\omega & (3.28) \end{aligned}$$

with $d_\omega = \tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}\ \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega}}$.

Applying the Gauss-Newton Method requires the first derivative of the residual function, which reads:

$$\begin{aligned} \frac{\partial r_i(\boldsymbol{\theta})}{\partial \theta_j} &= (\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}})_x\ \frac{\partial m'_x(\boldsymbol{\theta}, \boldsymbol{p_i})}{\partial \theta_j} + (\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}})_y\ \frac{\partial m'_y(\boldsymbol{\theta}, \boldsymbol{p_i})}{\partial \theta_j} & (3.29)\\ &= \tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}{}^T \frac{\partial \boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i})}{\partial \theta_j} & (3.30) \end{aligned}$$

The 2D-line is described by one point on the line and its normal. The normal can point towards the origin or away from it, in both cases the same line is described. One may question, if the direction of this normal is important for the optimization. This is discussed in the following.

With the Jacobian $J_{ij} = \frac{\partial r_i(\boldsymbol{\theta})}{\partial \theta_j}$ the gradient of the objective function for a single correspondence is:

$$\begin{aligned} \nabla f &= J^T r(\boldsymbol{\theta}) & (3.31)\\ &= \left(\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}{}^T \frac{\partial \boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i})}{\partial \theta_j} \right) \left(\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}{}^T \left(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega}} \right) \right) & (3.32) \end{aligned}$$

Because the first and the last term in the last Equation both change signs, if the normal is negated, it is not of importance whether the normal $\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}}$ points towards or away from the origin.

In other words: The distance of the point to the line is given in the direction of the normal in the residual function. The derivative of the point motion is also with respect to that normal. Therefore, the minimization makes steps in the correct direction, regardless in which direction the normal is pointing.

3.3.6 Rigid Motion from Image Gradients

The following sections require a basic understanding of image processing, convolution and edge detection. Readers unfamiliar with these topics are referred to the textbook [68].

KLT Corner Tracking

The abbreviation KLT is short for the authors *Kanade, Lucas and Tomasi*. Related literature is [125, 116, 12]. Given here is a short overview of important aspects.

The basis for tracking is the *Image Brightness Constancy*, meaning that 3D scene points project with the same brightness to image points, regardless of the point of view. Or in other words, if the camera or the object of interest moves, its projected 3D points have the same image values. This assumption does of course not reflect reality, but it approximates reality sufficiently, if the motion of the camera or object is small and lighting changes only insignificantly. The additional assumption is, that all 3D scene points visible in the first image are also visible in the next images, neglecting occlusion and image borders.

The derivation of the Kanade-Lucas-Tomasi-Tracking Equation is a further application of the Gauss-Newton method (see chapter 2) as described now. Let $\boldsymbol{p}$ be an image point in the current image I_t, whose displacement $\hat{\boldsymbol{\theta}} = (\hat{\theta}_x, \hat{\theta}_y)^T$ to the next image I_{t+1} is to be found, such that $I_t(\boldsymbol{p}) = I_{t+1}(\boldsymbol{p} + \boldsymbol{\theta})$, which is equal to $I_t(\boldsymbol{p} - \boldsymbol{\theta}) = I_{t+1}(\boldsymbol{p})$. The KLT tracker now assumes that the image region $\mathcal{R}$ around point $\boldsymbol{p}$ looks similar in the next image around $\boldsymbol{p} + \hat{\boldsymbol{\theta}}$ with the additional assumption, that all points within the region have the same displacement.

Therefore, the following error is minimal for the correct displacement $\hat{\boldsymbol{\theta}}$:

$$\hat{\boldsymbol{\theta}} = \min_{\boldsymbol{\theta}} \sum_{\mathcal{R}} \left(I_t(\boldsymbol{p} - \boldsymbol{\theta}) - I_{t+1}(\boldsymbol{p}) \right)^2 \tag{3.33}$$

This *Nonlinear Least Squares problem* can be solved with the Gauss-Newton method, which iteratively finds the correct displacement $\hat{\boldsymbol{\theta}}$. The initial displacement $\boldsymbol{\theta}_{init}$ in the first iteration of Gauss-Newton may be taken as zero, or may result from prediction using knowledge from previous displacements.

Here there is one residual function for each image point $\boldsymbol{p}_i \in \mathcal{R}$ in the region. The Jacobian for all image points in the region with the current displacement $\boldsymbol{\theta}_t$ is:

$$J_{ij} = \begin{bmatrix} \frac{\partial I(\boldsymbol{p}_i - \boldsymbol{\theta}_t)}{\partial \theta_x} & \frac{\partial I(\boldsymbol{p}_i - \boldsymbol{\theta}_t)}{\partial \theta_y} \end{bmatrix}$$

with $j \in \{x, y\}$. The spatial image gradient $\boldsymbol{g} = \left(\frac{\partial I(\boldsymbol{p} - \boldsymbol{\theta}_t)}{\partial \theta_x}, \frac{\partial I(\boldsymbol{p} - \boldsymbol{\theta}_t)}{\partial \theta_x} \right)^T$ may be calculated by applying the *sobel operator* [68]. The displacement $\Delta\boldsymbol{\theta}_t$ for one iteration is then found by solving

$$\Delta\boldsymbol{\theta}_t = (JJ^T)^{-1} J^T \boldsymbol{r}(\boldsymbol{\theta_t}) \tag{3.34}$$

The *coefficient matrix* JJ^T is also known as the *structure tensor* $G \in \mathbb{R}^{2\times 2}$, which reads:

$$G = \sum_{p \in \mathcal{R}} \begin{bmatrix} g_x g_x & g_x g_y \\ g_y g_x & g_y g_y \end{bmatrix} \tag{3.35}$$

Please note, that the type of motion for the region around the tracked point can be changed from pure translation like above, to an affine 2D transformation like in [116] or to 3D motion of articulated objects like mentioned in the following sections.

Corner Tracking and Optical Flow for Rigid Objects

In some applications the texture or color of the object is also known in addition to its geometry (as in Figure 3.8). In that case, there is the possibility to use the texture information of the model to establish correspondences with the current image. It is not necessary to have a complete textured model, it is sufficient to have 3D-points, which have an associated image value (e.g. RGB-color or grayscale) like in [71]. The associated image intensity value, will be called color in the following.

The idea is to move the object such that the projection of it is most similar to the current image of the object. To achieve this, an assumption is made, that the development of grey values in the near vicinity of the correct point is linear. Or in other words, the image intensity function is approximated by a first order Taylor series expansion. This assumption is the same as for the KLT tracker [125]. However, this assumption requires, that the distance between the projected model point and its position in the current image is small, such that the linearisation holds. If the displacement is large a multi scale approach is necessary as in [76]. In that case the current image is smoothed, e.g. with a Gaussian kernel and sub sampled. Increasing sizes of the kernel are applied to get different levels of smoothing. This results in faster image processing for the lower resolution images. The estimation is then applied iteratively to the smoothest image first, giving a rough estimate of the pose. Higher resolution images give more accurate pose estimates down to the original image, which yields the most accurate pose. In Figure 3.9 three different levels of smoothing are shown from the closeup of the marked rectangular area of the textured box in Figure 3.8. The different levels of smoothed images are also called a image pyramid [8].

Let $\boldsymbol{p'} \in \mathbb{R}^2$ be the projection of the model point for the last estimated pose from the last image (see Figure 3.11). And let $\tilde{\boldsymbol{g}} \in \mathbb{R}^2$ be the grey value gradient vector at position $\boldsymbol{p'}$ and let $d = c_i(\boldsymbol{p'}) - c_m(\boldsymbol{p'})$ be the grey value difference between the color of the projected model $c_m(\boldsymbol{p'})$ point and the image value $c_i(\boldsymbol{p'})$ at the same position in the current image. Then the objective function for minimization is

$$\min_{\boldsymbol{\theta}} \sum_i \left| \tilde{\boldsymbol{g}} \ (\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \boldsymbol{p'}) - d \right|^2 \tag{3.36}$$

The sum denotes that the minimization takes place for multiple correspondences at once.

The geometric interpretation of the objective is given in the following. Assume the object moved to the left and a little upwards resulting in a displacement vector as shown on the left in Figure 3.11 (the dotted line). This displacement is unknown and depends on the motion parameters $\boldsymbol{\theta}$ of the object, which are to be estimated. The point $\boldsymbol{p'}$ equals the projected model point from the last estimated pose $\boldsymbol{p'} = \boldsymbol{m'}(\boldsymbol{0}, \boldsymbol{p})$. Shown in the left is the projected model point $\boldsymbol{p'}$ and the image gradient at that position. The goal of minimization is to move the object model point in 3D-space, such that the image color at its projected position equals its associated color. The right image in Figure 3.11 shows the new projected position $\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p})$ of the model point after a small motion $\boldsymbol{\theta}$.

If the development of color values would be linear the necessary displacement is $\frac{d}{|\tilde{\boldsymbol{g}}|}$ in direction of the gradient vector, where d is the difference of color values between the

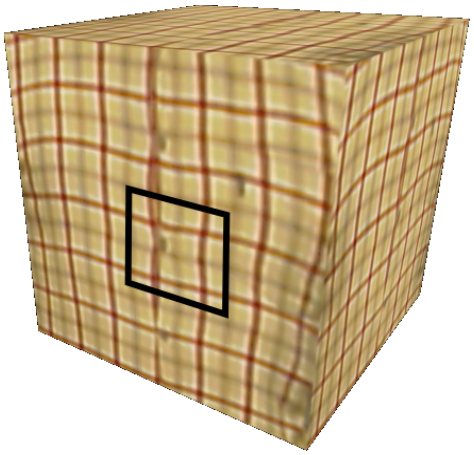


Figure 3.8: Pose estimation using the model's color

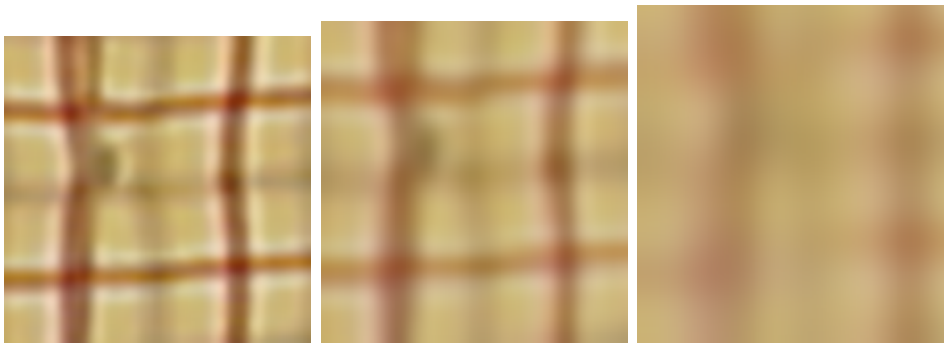


Figure 3.9: A closeup from the textured box, with different levels of smoothing.

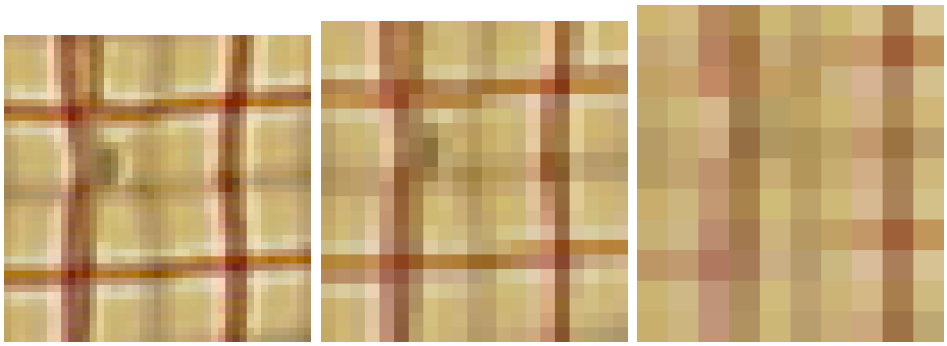


Figure 3.10: Sub-sampled closeups from the textured box (applied after smoothing)

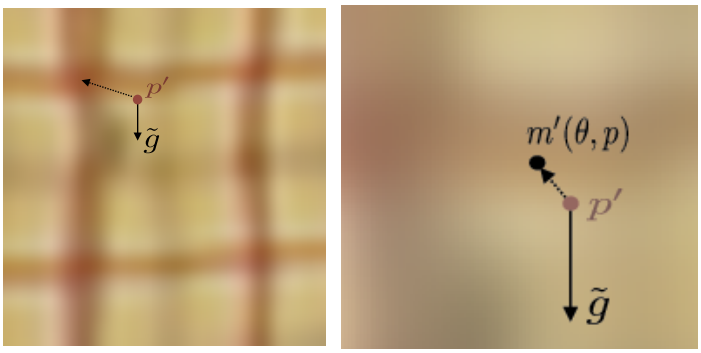


Figure 3.11: Left:Projection of model point and gradient. Right: Closeup of the motion.

old position from the previous frame and the new position at $\boldsymbol{p'}$. In the example shown d is negative because the image is brighter at the new position. Please note, that the objective function changes only, if the projected point position $\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p})$ moves parallel to the gradient. A motion perpendicular to the gradient direction does not change the error.

Relation to 3D-point-2D-line Correspondences

The above objective function for minimization is very similar to 3D-point-2D-line correspondences as in the previous section. By replacing the terms in (3.36) appropriately, the minimization function becomes equal to that for 3D-point-2D-line correspondence (3.25). The corresponding line has the normal $\tilde{\boldsymbol{n}}_{\boldsymbol{g}} = \frac{1}{|\tilde{\boldsymbol{g}}|}\tilde{\boldsymbol{g}}$ and a point $\boldsymbol{q'_g}$ on the line is found by starting from $\boldsymbol{p'}$ and taking the point with distance $\frac{d}{|\tilde{\boldsymbol{g}}|}$ in the opposite direction of the gradient vector resulting in $\boldsymbol{q'_g} = \boldsymbol{p'} + \frac{d}{|\tilde{\boldsymbol{g}}|}\tilde{\boldsymbol{n}}_{\boldsymbol{g}}$ as shown in Figure 3.12. It is the opposite gradient direction, because the distance d of color values is negative in this example. The objective function reads with this formulation:

$$\min_{\boldsymbol{\theta}} \sum_{i} \left| \tilde{\boldsymbol{n}}_{\boldsymbol{g}} \left(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p}_{\boldsymbol{i}}) - \tilde{\boldsymbol{q}}'_{\boldsymbol{g}} \right) \right|^2 \tag{3.37}$$

with correspondences $((\tilde{\boldsymbol{n}}_{\boldsymbol{g},\boldsymbol{i}}, \tilde{\boldsymbol{q}}'_{\boldsymbol{g},\boldsymbol{i}})\ ,\ \boldsymbol{p}_{\boldsymbol{i}})$.

In [17] a formulation with twists results in similar equations and are also solved using Gauss-Newton optimization. The difference here is, that the full perspective camera model is used, while in [17] a scaled orthographic projection modeled the camera.

KLT Equations for Rigid Motion

An alternative derivation for the equations above, which allow pose estimation from model color, is given now. It leads to the same equations as in Equation (3.37). This is achieved by applying the KLT Equations (see section 4.5.2).

As in the section above, the idea is, that the projected model point has the same image brightness for different poses, also known as *image brightness constancy*. A similar

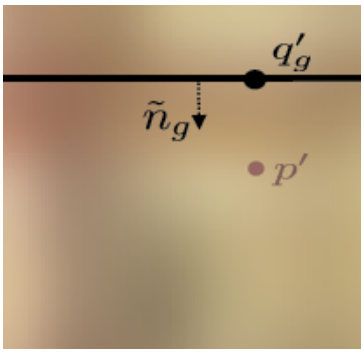


Figure 3.12: Formulation as point line correspondence

approach was made in [49], where the original KLT Equations are applied to track the motion of image regions and to segment regions by the assumption that each limb of human undergoes a rigid motion. A hierarchical RBF network smoothes the motion vector field within a region.

In this section the KLT-equations are changed, such that the underlying motion dynamics of the human are directly incorporated, resulting in image displacements that are smooth over the body part region reflecting the exact 3D motion of the body part. The resulting equations are similar to that of Malik & Bregler [17], except that here the full perspective projection is included, where in [17] a scaled orthography is used.

If the point $\boldsymbol{p'}$ in the image I_t corresponds to a known 3D object, whose pose is known, the corresponding 3D point $\boldsymbol{p}$ for that image point can be calculated by intersection of the viewing ray with the 3D object. Now it is possible to find the image displacement to the next image I_{t+1}, which minimizes the brightness difference under the constraint, that the point belongs to a rigid object, which performed a rigid motion from one image to the next. The objective function for multiple points belonging to the object is now with respect to the pose parameters. If the relative motion to the next image is found, the corresponding image displacements can be easily calculated.

Let $I_{t+1}(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}))$ be the image value at the projected 3D point position, such that $\boldsymbol{m'}(\boldsymbol{0}, \boldsymbol{p_i}) = \boldsymbol{p'_i}$. The objective function is then

$$\min_{\boldsymbol{\theta}} \sum_i \left(I_{t+1}(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i})) - I_t(\boldsymbol{p'_i}) \right) \tag{3.38}$$

The necessary Jacobian for solving this Nonlinear Least Squares problem consists of the partial derivatives of the residuals:

$$J_{ij} = \frac{\partial I_{t+1}(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}))}{\partial \theta_j} \tag{3.39}$$

Important is now, that the derivatives are taken at a specific parameter position $\boldsymbol{\theta}_t$. Ap-

plication of the chain rule leads to:

$$\left.\frac{\partial I_{t+1}(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}))}{\partial \theta_j}\right|_{\boldsymbol{\theta}_t} = \left.\frac{\partial I_{t+1}(\boldsymbol{q'})}{\partial \boldsymbol{q'}}\right|_{\boldsymbol{\theta}_t} \left.\frac{\partial \boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i})}{\partial \theta_j}\right|_{\boldsymbol{\theta}_t} \tag{3.40}$$

where $\frac{\partial I_{t+1}(\boldsymbol{q'})}{\partial \boldsymbol{q'}} = (g_x, g_y)^T$ is the spatial image gradient evaluated at $\boldsymbol{m'}(\boldsymbol{\theta_t}, \boldsymbol{p_i})$, which equals $\boldsymbol{p'_i}$, if θ_t is zero. As visible, the Jacobian equals those of (3.30) for 3D-point-2D-line correspondences.

3.4 Articulated Objects

In this section the estimation of pose or rigid motion as explained in the previous section is extended to kinematic chains. A kinematic chain is a concatenation of transformations. An articulated object consists of one or multiple kinematic chains and has a shape or geometry in addition, like the arm of the human model in Figure 3.13. The application of pose estimation algorithms to human bodies is described in chapter *Tracking*.

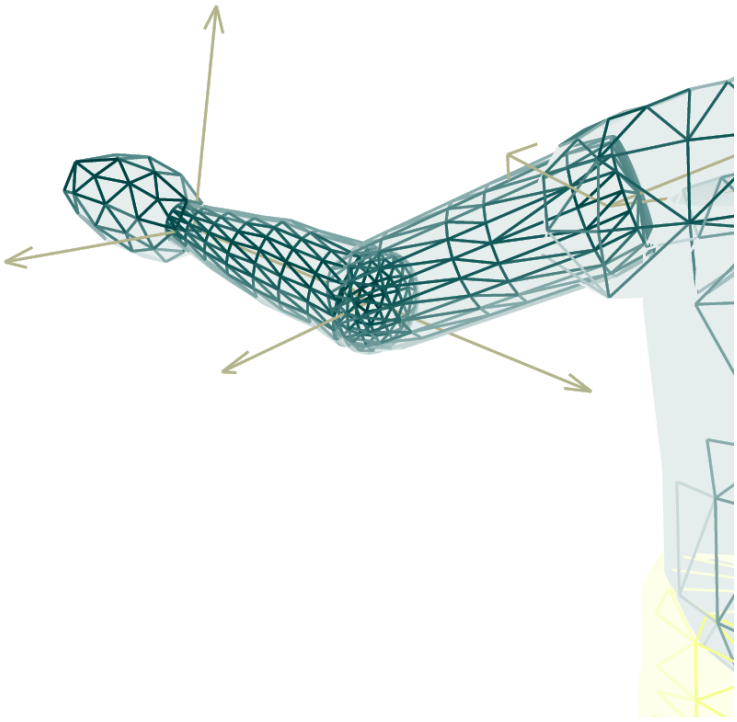


Figure 3.13: The arm of a human is an example of an articulated object.

3.4.1 Motion in a Kinematic Chain

In this work kinematic chains consist only of rotational transformations around known axes at arbitrary positions. The axes for the arm are shown as arrows in Figure 3.13. There is a fixed hierarchy defined for the rotations, e.g. a rotation around the shoulder abduction axis (the one pointing forward) changes the position and orientation of all consecutive axes. In addition to the rotational transformations, there is the position and orientation of the base, which undergoes rigid motions, e.g. the upper body in the Figure, which is partially visible on the right.

Consider now the motion of a point $\boldsymbol{p}$ on the model of the arm with p rotational axes, e.g. a point on the hand. Its motion can be described using the rotation description from

Equation (3.1) and the rigid motion from Equation (3.6) resulting in:

$$\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p}) = (\theta_x, \theta_y, \theta_z)^T + \left(R_{\omega_x}(\theta_\alpha) \circ R_{\omega_y}(\theta_\beta) \circ R_{\omega_z}(\theta_\gamma) \circ R_{\omega_1,q_1}(\theta_1) \circ \cdots \circ R_{\omega_p,q_p}(\theta_p)\right) \boldsymbol{p} \tag{3.41}$$

The coordinates of the axes $(\boldsymbol{w_j}, \boldsymbol{q_j})$ may be given relative to their parent axis (the next one upwards in the chain) or all coordinates of all axes may be given in the same coordinate system, i.e. the same where the base transform is defined, e.g. the world coordinate system. If not stated otherwise $(\boldsymbol{w_j}, \boldsymbol{q_j})$ are assumed to be in world coordinates.

If $\boldsymbol{\theta}$ equals $\mathbf{0}$ the position and orientation of the axes define the kinematic chain and also the relative transformations between the axes. To obtain the world coordinates of the axes for a specific $\boldsymbol{\theta}_t$ the direction vector $\boldsymbol{\omega_j}$ and the point on the axis $\boldsymbol{q_j}$ are transformed by applying the chain transform up to axis $(\boldsymbol{w_{j-1}}, \boldsymbol{q_{j-1}})$ to them. Therefore, $(\boldsymbol{\omega}_j, \boldsymbol{q_j})$ are different depending on the current $\boldsymbol{\theta}_t$, if a description in world coordinates is used.

For a specific pose $\boldsymbol{\theta}_t$ a relative motion $\Delta\boldsymbol{\theta}$ can be imagined in the following way: The point $\boldsymbol{p}$, e.g. on the hand, is rotated at first around the closest axis $(\boldsymbol{\omega}_p, \boldsymbol{q}_p)$ in the hierarchy with angle $\Delta\theta_p$, then around the fixed second axes $(\boldsymbol{\omega}_{p-1}p, \boldsymbol{q}_{p-1})$ and so on up to the base, which adds the rigid motion.

Estimating the pose of a kinematic chain from given 3D-3D correspondences $(\boldsymbol{p_i}, \tilde{\boldsymbol{p}}_{\boldsymbol{i}})$ may be done with *Newton's method*, which means here estimating the joint angles and the global orientation and position. The minimization problem is the same as in Equation (3.7), while the motion function $\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})$ includes here the joint angles as well:

$$\min_{\boldsymbol{\theta}} \sum_i \left|\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{p}}_{\boldsymbol{i}}\right|^2 \tag{3.42}$$

The Jacobian for each residual function is necessary in order to find the minimizer with the iterative *Gauss-Newton* method.

3.4.2 Jacobian of Articulated Motion

The partial derivatives of the motion function, which gives the Jacobian, can be derived in the same way as for the rigid motion using the description of rotations around known axes in 3D space from Equation (3.1). If the current pose is $\boldsymbol{\theta}_t$ and only relative motion is estimated the Jacobian is:

$$J = \begin{bmatrix} 1 & 0 & 0 & \frac{\partial m_x}{\partial \theta_\alpha} & \frac{\partial m_x}{\partial \theta_\beta} & \frac{\partial m_x}{\partial \theta_\gamma} & \frac{\partial m_x}{\partial \theta_1} & .. & \frac{\partial m_x}{\partial \theta_p} \\ 0 & 1 & 0 & \frac{\partial m_y}{\partial \theta_\alpha} & \frac{\partial m_y}{\partial \theta_\beta} & \frac{\partial m_y}{\partial \theta_\gamma} & \frac{\partial m_y}{\partial \theta_1} & .. & \frac{\partial m_y}{\partial \theta_p} \\ 0 & 0 & 1 & \frac{\partial m_z}{\partial \theta_\alpha} & \frac{\partial m_z}{\partial \theta_\beta} & \frac{\partial m_z}{\partial \theta_\gamma} & \frac{\partial m_z}{\partial \theta_1} & .. & \frac{\partial m_z}{\partial \theta_p} \end{bmatrix} \tag{3.43}$$

The derivatives at zero are:

$$\begin{aligned}
\left.\frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_\alpha}\right|_{\boldsymbol{\theta}=\mathbf{0}} &= \boldsymbol{\omega_x} \times (\boldsymbol{p} - \boldsymbol{q_x}) \\
\left.\frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_\beta}\right|_{\boldsymbol{\theta}=\mathbf{0}} &= \boldsymbol{\omega_y} \times (\boldsymbol{p} - \boldsymbol{q_y}) \\
\left.\frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_\gamma}\right|_{\boldsymbol{\theta}=\mathbf{0}} &= \boldsymbol{\omega_z} \times (\boldsymbol{p} - \boldsymbol{q_z}) \\
\left.\frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_j}\right|_{\boldsymbol{\theta}=\mathbf{0}} &= \boldsymbol{\omega_j} \times (\boldsymbol{p} - \boldsymbol{q_j})
\end{aligned} \tag{3.44}$$

where $j \in \{1, .., p\}$ and $\boldsymbol{\omega_x}, \boldsymbol{q}_x$ denote rotation around an arbitrary point in space and $\boldsymbol{\omega_x}, \boldsymbol{\omega_y}, \boldsymbol{\omega_z}$ are assumed orthogonal.

Given here is the special Jacobian for $\boldsymbol{\theta} = \mathbf{0}$, which is valid if all axes are given in world coordinates and only relative motion is estimated. A general derivation of the partial derivatives can be found in Appendix D, where the second partial derivatives are derived.

3.4.3 Jacobian of Articulated Motion under Projection

If the motion of the articulated object is observed by a pinhole camera, and 3D-2D point correspondences $(\boldsymbol{p}, \tilde{\boldsymbol{p}}')$ are given, the estimation of joint angles and global pose can be done in the same way as in section 3.3 by Nonlinear Least Squares (solved with Gauss-Newton). The optimization problem reads:

$$\min_{\boldsymbol{\theta}} \sum_i \left| \boldsymbol{m}'(\boldsymbol{\theta}, \boldsymbol{p}) - \tilde{\boldsymbol{p}}' \right|^2 \tag{3.45}$$

The necessary Jacobian is similar to the partial derivatives of Equation (3.17). Here the additional partial derivatives for the rotation around the joint axes give additional columns in the Jacobian.

3.4.4 Estimation from other Correspondence Types

To estimate the pose of the articulated object from other kinds of correspondences, the same optimization with the according objective function as for the rigid motion can be applied. The only difference is the special motion function, that now also includes the joint angles of the kinematic chain. The necessary partial derivatives can be looked up in the two previous sections.

Important to note is, that for each model point it is necessary to know, to which joint of the articulated object the point belongs, e.g. a point on the lower arm between the wrist and elbow can give no information about the joint angles of the wrist. Therefore, the Jacobian entries for that point will be zero in the column with the wrist angles.

For more complex models like linear blend skinned, a point belongs even to multiple joints in a weighted manner [84, 93].

3.4.5 Geometric Illustration of the Gauss-Newton Method

The Gauss-Newton method approximates the objective function by a quadratic function and iterates until convergence. Shown here is the development of the pose of an articulated object during the iterations for a robot arm with two rotational joints $(\boldsymbol{\omega}_1, \boldsymbol{q}_1)$ and $(\boldsymbol{\omega}_2, \boldsymbol{q}_2)$. Both rotation axes have the same orientation in this example, namely the z-axis. The first axis is positioned at the origin and the second axis at $(1, 0, 0)^T$. The length of both parts of the arm equals one. The end effector (model point) is at $\boldsymbol{p}_1 = (1.9, 0.5, 0)^T$ in the initial state as shown in Figure 3.14.

The first iteration of joint angle estimation is as follows, which minimize the distance between the end effector and the target point $\tilde{\boldsymbol{p}} = (1.8, 0.2, 0)^T$.

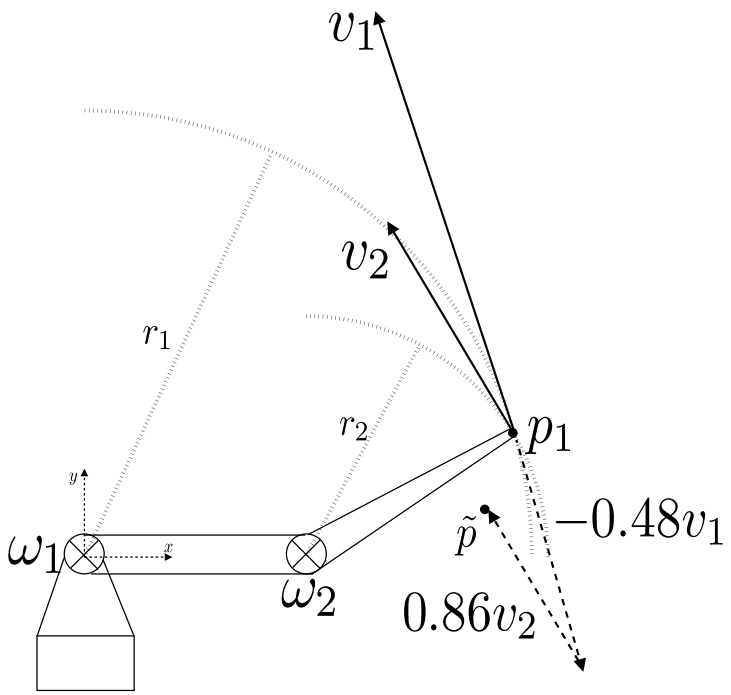


Figure 3.14: Example of an articulated object with two rotational joints. Illustration of the Gauss-Newton method: Iteration 1

The partial derivatives are the velocity vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. They are the tangents on the circle with radius r_1 and r_2 respecively. One step in the Gauss-Newton estimation finds the linear combination of these vectors, which best matches the error vector $\boldsymbol{r} = (\boldsymbol{p}_1 - \tilde{\boldsymbol{p}})$. The resulting change in the rotation angle is $\Delta\theta_1 = -0.48$ and $\Delta\theta_2 = 0.86$. The same is true if the target point would lie outside the x-y plane at a different z-value, in that case the linear combination would be found that moves the model point as close as possible to the target point.

As the linear velocity vectors are only an approximation of the rotational motion, further iterations are necessary. Inserting the change in angles lead to the next arm con-

figuration for the next iteration as shown in Figure 3.15. Shown is the second iteration starting with the model point $\boldsymbol{p}_2$ and the same target point. The resulting change gives the point $\boldsymbol{p}_3$ for the next iteration. This point is already so close to the target point, that further illustrations of the iterations are not shown.

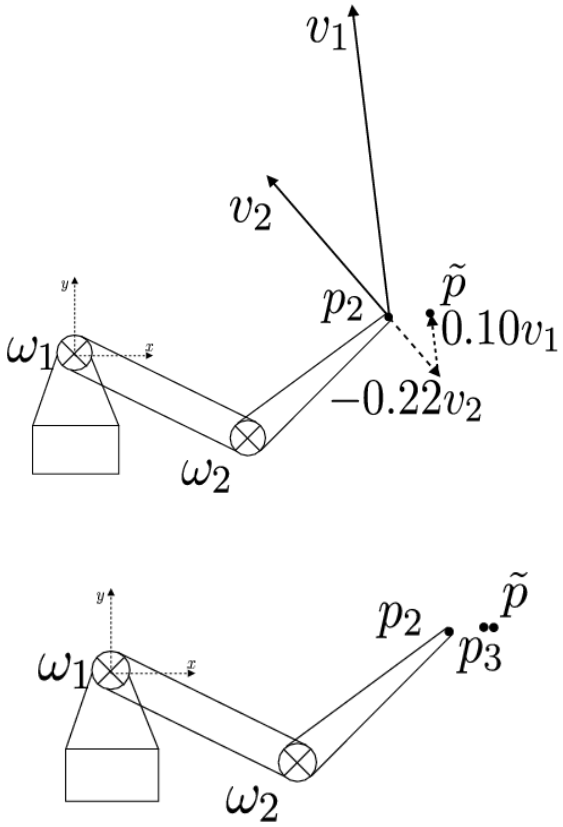


Figure 3.15: Illustration of the Gauss-Newton method: Iteration 2 and 3

As visible the result estimated in one iteration is the best linear combination of the partial derivative vectors (the velocities), which minimizes the difference to the current error vector. The same is true for estimation of the global orientation, where the three axes of rotation are orthogonal to each other and are positioned at the same point in 3D space.

3.5 Second Order Motion Derivatives

Given in the following are the second order motion derivatives of rotational motions as they appear in an articulated object. With these derivatives the correct Hessian for Newton's method is available and is compared with the Gauss-Newton method, which neglects these derivatives. In that way the acceleration of points moving in a rotational chain are included directly in the pose estimation process.

The *Gauss-Newton* method neglects partial derivatives of second order by approximating the Hessian with $J^T J$. The correct Hessian is $H = J^T J + S$ (details in section 2.3.2) with

$$S_{kj} = \sum_{i=1}^{m} r_i(\boldsymbol{\theta}) \frac{\partial^2 r_i(\boldsymbol{\theta})}{\partial\theta_k \, \partial\theta_j}$$

Given here are the second partial derivatives at zero. The general derivation can be found in Appendix D.

The second derivatives of the residual for the motion function of a kinematic chain are

$$\begin{aligned} j \leq i: \quad & \left.\frac{\partial^2 m}{\partial\theta_i \, \partial\theta_j}\right|_{\boldsymbol{\theta}=\mathbf{0}} = \boldsymbol{\omega_j} \times \ (\boldsymbol{\omega}_i \times (\boldsymbol{p} - \boldsymbol{q}_i)) \\ j > i: \quad & \left.\frac{\partial^2 m}{\partial\theta_i \, \partial\theta_j}\right|_{\boldsymbol{\theta}=\mathbf{0}} = \boldsymbol{\omega_i} \times \ (\boldsymbol{\omega}_j \times (\boldsymbol{p} - \boldsymbol{q}_j)) \end{aligned} \tag{3.46}$$

with $i, j \in \{\alpha, \beta, \gamma, 1, .., p\}$. The axis that is nearer in the chain to the point $\boldsymbol{p}$ gives the first derivative, while the axis further away gives the second derivative.

Hence the second partial derivatives follow *Clairaut's theorem*, that states if for a function $f : \mathbb{R}^n \to \mathbb{R}$ the second partial derivatives exist at every point and are continuous then

$$\frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_i \, \partial\theta_j} = \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_j \, \partial\theta_i}$$

Which as a result gives a symmetric Hessian H.

The remaining second derivatives for the translational part are:

$$\frac{\partial^2 m}{\partial\theta_{x,y,z} \, \partial\theta_i} = \frac{\partial^2 m}{\partial\theta_i \, \partial\theta_{x,y,z}} \ = \mathbf{0} \tag{3.47}$$

with $i \in \{\alpha, \beta, \gamma, 1, .., p\}$.

3.5.1 The Benefit of Second Derivatives

In this section the use of second partial derivatives is analyzed. Various arm configurations, target points and model points are compared for the arm with two rotational joints shown in Figure 3.16.

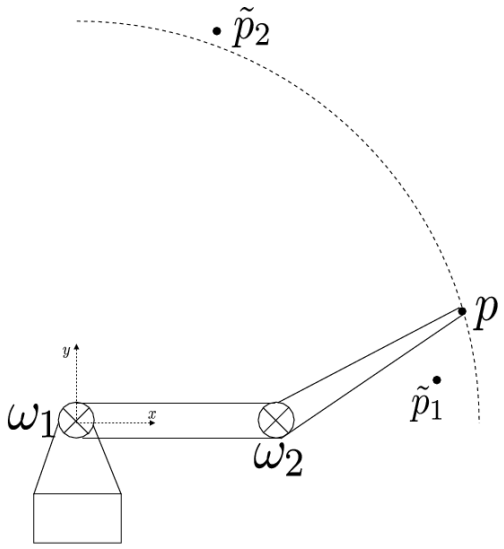


Figure 3.16: Example of an articulated object with two rotational joints.

Consider the residual $\boldsymbol{r}(\boldsymbol{\theta_t}) = (\boldsymbol{m}(\boldsymbol{\theta_t}, \boldsymbol{p}) - \tilde{\boldsymbol{p}})$. The approximating function for a configuration with two joints like in Figure 3.16 is the Taylor approximation up to second order with $\boldsymbol{\Delta_\theta} = (\boldsymbol{\theta} - \boldsymbol{\theta_t})$:

$$f(\boldsymbol{\theta}) \approx f_\Delta(\boldsymbol{\theta}) = f(\boldsymbol{\theta_t}) + \boldsymbol{\Delta_\theta}^T \nabla f(\boldsymbol{\theta_t}) + \frac{1}{2}\boldsymbol{\Delta_\theta}^T H(\boldsymbol{\theta_t})\boldsymbol{\Delta_\theta} \tag{3.48}$$

$$= |\boldsymbol{r}(\boldsymbol{\theta})|^2 + \boldsymbol{\Delta_\theta}^T J(\boldsymbol{\theta_t})^T \boldsymbol{r}(\boldsymbol{\theta_t}) + \frac{1}{2}\boldsymbol{\Delta_\theta}^T H(\boldsymbol{\theta_t})\boldsymbol{\Delta_\theta} \tag{3.49}$$

$$\tag{3.50}$$

where $H(\boldsymbol{\theta_t}) = 2(J(\boldsymbol{\theta_t})^T J(\boldsymbol{\theta_t}) + S(\boldsymbol{\theta_t}))$ and S is the matrix with second derivatives of the residual as described in section 2.3.2. The Gauss-Newton method neglects the second derivative matrix S, therefore the approximating function is analyzed firstly without S and afterwards the advantage of including second derivatives is shown.

Approximation of the Gauss-Newton Method

Consider now the case where $\boldsymbol{\theta_t} = \mathbf{0}$ as in Figure 3.16. Let $\boldsymbol{v_1} = \boldsymbol{\omega_1} \times (\boldsymbol{q_1} - \boldsymbol{p})$ and $\boldsymbol{v_2} = \boldsymbol{\omega_2} \times (\boldsymbol{q_2} - \boldsymbol{p})$ be the first partial derivatives of the residual function, that build the Jacobian. Then the approximating function reads:

$$f_\Delta(\boldsymbol{\theta}) = |\boldsymbol{r}(0)|^2 + \boldsymbol{\theta}^T V^T \boldsymbol{r}(\mathbf{0}) + \boldsymbol{\theta}^T V^T V \boldsymbol{\Delta_\theta} \tag{3.51}$$

$$= |\boldsymbol{p} - \tilde{\boldsymbol{p}}|^2 + \boldsymbol{\theta} V^T (\boldsymbol{p} - \tilde{\boldsymbol{p}}) + \boldsymbol{\theta}^T V^T V \boldsymbol{\theta} \tag{3.52}$$

with $V \in \mathbb{R}^{2\times 2} = [\boldsymbol{v_1}\ \boldsymbol{v_2}]$.

A plot of this approximating function and a plot of the real error function for a configuration with two joints as in Figure 3.16 is given in Figure 3.17. The arm parameters are:

- Joint center at $q_1 = (0,0,0)^T, q_2 = (1,0,0)^T$.
- Rotation axis $\omega_1 = (0,0,1)^T, \omega_2 = (0,0,1)^T$
- model point at $p = (1.9, 0.5, 0)^T$

The target point for this plot is $\tilde{\boldsymbol{p}}_{\mathbf{2}} = (0.7, 2.0, 0)^T$, which is outside the reachable circle. Therefore, the minimum error is found for an outstretched arm configuration at ($\theta_1 = 70.3, \theta_2 = -30deg$).

In the real error function the maximum on the left corresponds to a arm configuration outstretched in the other direction. Near the minimum the error surface is very close to a quadratic surface like the error surface on the right of the Gauss-Newton method. As can be expected for rotations, the error surface is more quadratic for small angles, when the model point and target point are close to each other.

Approximation of Newton's Method

The approximation by Newton's method takes into account the second derivatives of the motion function also, which are the acceleration of the point moving on the circle around the joint axis. The function is the same as for the Gauss-Newton method above, but the Hessian is now $H = J^T J + S$. A plot of the Newton approximation error function is shown in Figure 3.18.

In contrast to the Gauss-Newton approximation without second order derivatives the Newton approximation yields a minimum like the real error function.

It can be expected that the Newton approximation is closer to the real error function. To evaluate this, both approximations are compared with the real error function by taking the absolute difference. These absolute differences are then subtracted by taking the Gauss-Newton difference minus the Newton approximation. Let e_r be the real error function, e_G the Gauss-Newton approximation and e_N the Newton approximation with second derivatives. The difference e_d used for comparison is:

$$e_d = |e_r - e_G| - |e_r - e_N| \tag{3.53}$$

The difference e_d is shown in Figure 3.19. Values greater zero indicate, that the Newton approximation is closer to the real error function and values smaller zero show, that the Gauss-Newton approximation is closer to the real error function. As visible in the Figure, the Newton approximation is in fact closer to the real error function in most regions, where it isn't, the Gauss-Newton approximation is only slightly better. An explanation, why the Gauss-Newton approximation is better close to the minimum is given as follows.

The second derivatives can be understood as the acceleration of a circular motion. This acceleration can be imagined as a force vector pointing from the point on the circle

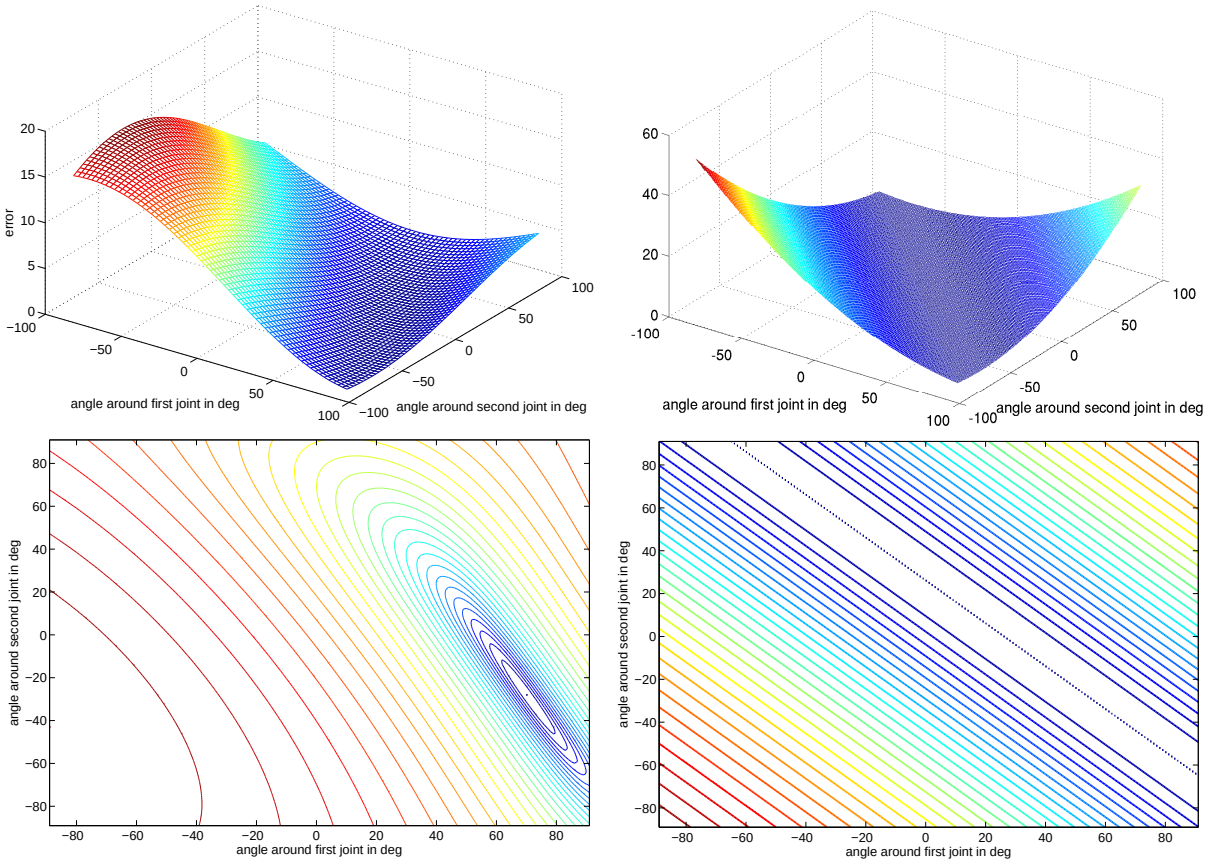


Figure 3.17: Error function for two joints. Left: Real Error function. Right:Approximation by Gauss-Newton. Bottom row: The functions shown with isolines.

towards the center of the circle. The effect of adding these to the Hessian is a change of the tangential vectors (compare Figure 3.14). Their direction is changed slightly to point more towards the center of the circle. This new direction is a better approximation to the real circular movement for large angles, however, it can be worse for very small angles as visible in the difference e_d as shown in Figure 3.19.

From this result, it can be expected that one step in the Newton iteration will lead to a better parameter estimate than the approximation without the second order derivatives, if the change in angle is sufficiently large. If the change is very small the second derivatives are not expected to give a recognizable benefit.

In Figure 3.20 10 iterations of the optimization with both methods are shown with lines and markers. The background contour isolines are the real error function.

The Gauss-Newton method does not reach the correct minimum in the case without dampening (see section 2.3.3), because an arm configuration is reached after a few iterations, where both velocity vectors are almost parallel (when the arm is outstretched). The partial derivatives lead then to a Hessian with determinant near to zero, which results in

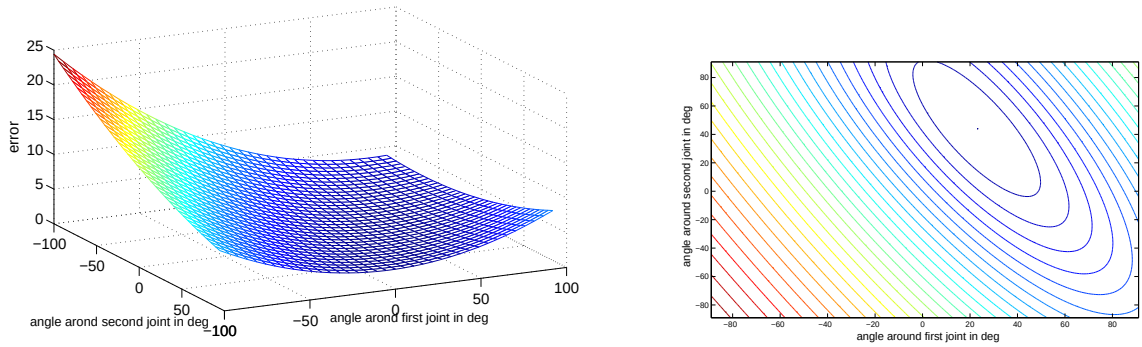


Figure 3.18: Error function for two joints. Left: Newton Error approximation. Right: The same function with isolines.

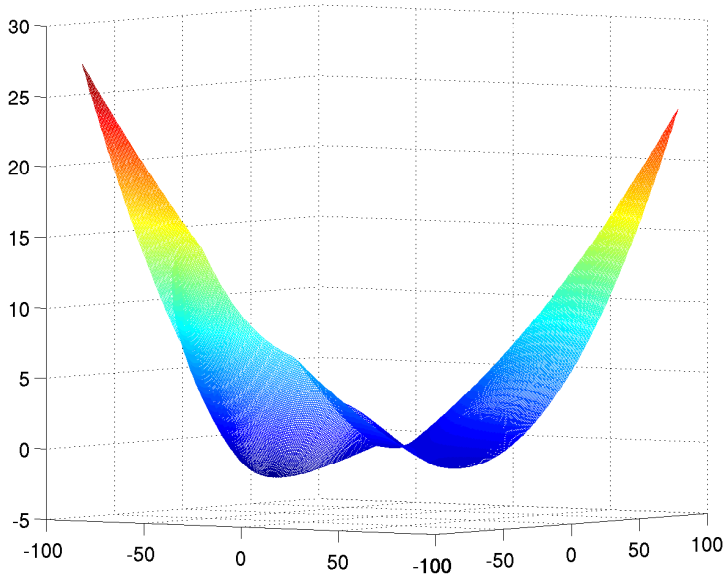


Figure 3.19: Difference e_d to the real error function: Gauss-Newton minus Newton Error approximation. Values greater zero indicate, that the Newton approximation is closer to the real error function.

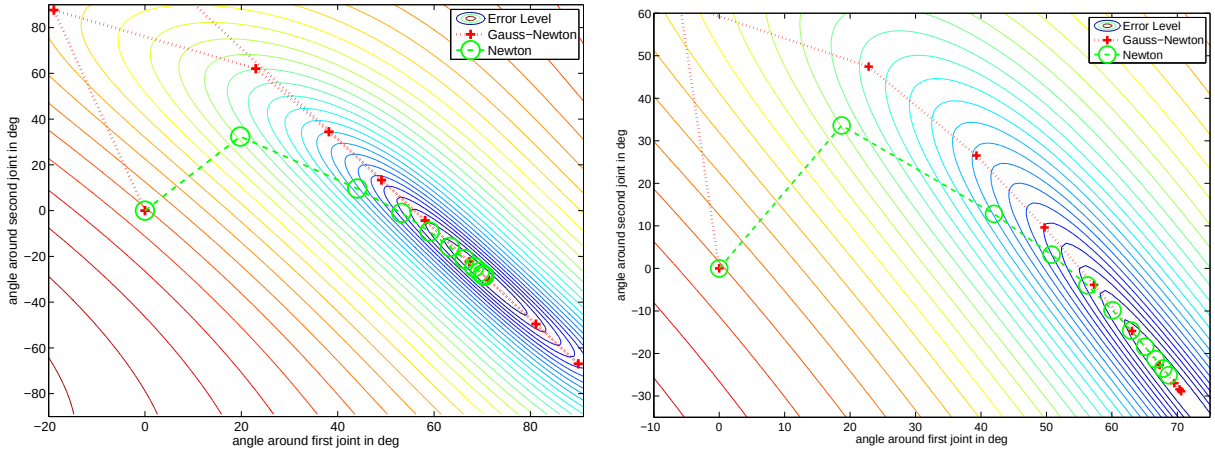


Figure 3.20: 10 iterations of both methods. Left: Without Dampening. Right: Damped estimate with 0.05

large almost arbitrary jumping. In the dampened case on the right both methods are near to the correct minimum after 10 iterations.

Different Arm Configuration

A plot of the approximating function and a plot of the real error function for a configuration with two joints as in Figure 3.21 is given in Figure 3.22. The arm parameters are:

- Joint center at $q_1 = (0,0,0)^T, q_2 = (1,0,0)^T$.
- Rotation axis $\omega_1 = (0,0,1)^T, \omega_2 = (0,0,1)^T$
- model point at $p = (2,0,0)^T$

The target point for this plot is $\tilde{\boldsymbol{p}}_{\mathbf{1}} = (1.8, 0.2, 0)^T$, therefore the minimum error is found for an arm configuration with $(\theta_1 = -20deg, \theta_2 = 50deg)$.

The main difference to the previous arm configuration is, that here both velocity vectors are parallel in the beginning resulting in a non-invertible Hessian for the Gauss-Newton case. Additionally there are two minima now, because the target point is inside the circle of reachable points as visible in Figure 3.22 on the left. The approximating error function of Newton's method shows the two minima as well. Accordingly the minimization without dampening converges to the saddle point as visible on the left in Figure 3.23. The Hessian of Newton's method is not positive definite in that case. Introducing a dampening value larger than the smallest eigenvalue of the Hessian leads to an optimization as shown on the right, where the minimization converges to the next minimum. While both minima represent a valid arm pose, the minimization with dampening converges to the next minimum, similar to a Gradient Descent.

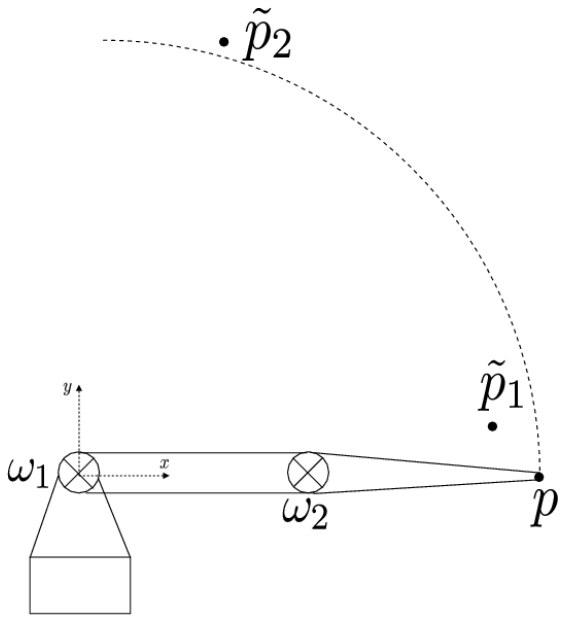


Figure 3.21: Example of an articulated object with two rotational joints in a different configuration.

3.5.2 Summary Second Derivatives

Though the above experiments are rather simple compared to complete pose estimation of the human body, the advantages and problems of including second order motion derivatives already occurred. The results of the above experiments and Figures are summarized in Table 3.1.

As expected from the Taylor series approximation, Newton's method gives an error approximation that is closer to the real error function than the Gauss-Newton approximation in most regions around the current parameter position. In the iterative process of finding the next minimum, Newton's method converges faster than the Gauss-Newton method, if the same dampening value is chosen and if the dampening value is sufficiently large for Newton's method. However, for different dampening values the Gauss-Newton method can outperform Newton's method regarding the time until convergence.

In the second example (Figure 3.21) the Hessian of the Gauss-Newton method isn't invertible, while the Hessian with second derivatives of the Newton's method is invertible and minimization gives the correct minimum even without dampening.

Appropriate dampening values are important for both methods, because there are configurations, that don't lead to the correct result with Newton's method without appropriate dampening. This is either because the Newton error function is too flat resulting in too large steps, or the Hessian is not positive definite resulting in convergence towards a saddle point like in the first example (figure 3.16).

The Gauss-Newton method is less sensitive to the dampening value. It is only impor-

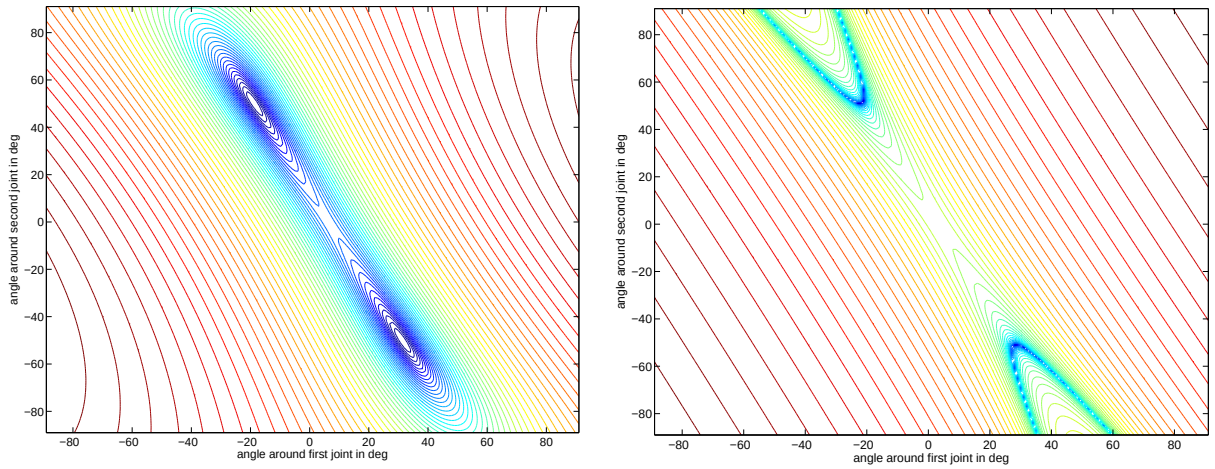


Figure 3.22: Left: Real error function with contour levels. Two mininima are present. Right: Newton approximation with contour levels showing both minima as well and a saddle point in between.

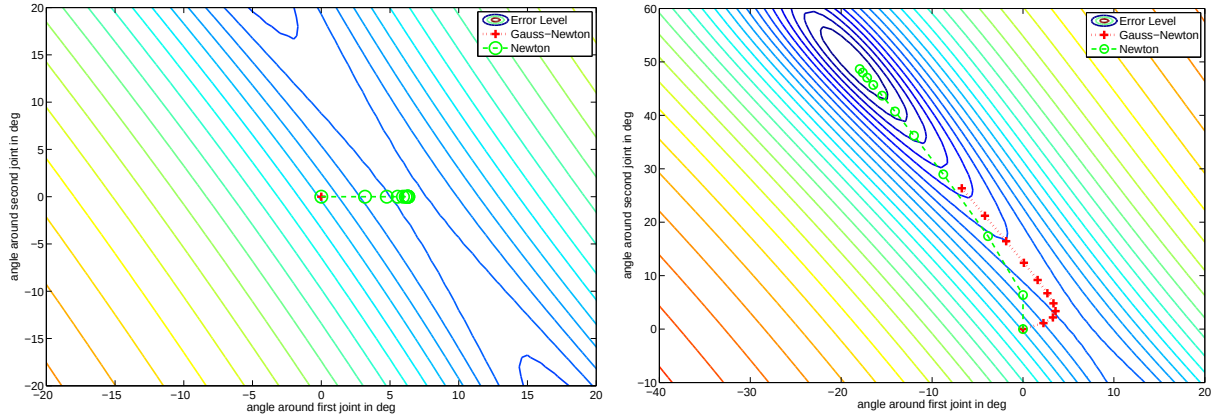


Figure 3.23: Left: 10 iterations without dampening. Convergence towards the saddle point. Right: Dampening with 0.2 leads to convergence towards the next minimum.

Property	with Second Order Derivatives	without Second Order Derivatives
Error approximation	better in most regions	worse, except for very small angles
Computational cost for Hessian	$c_1 \cdot M \cdot P + c_2 \cdot M \cdot P^2$ M points, P parameters c_1, c_2 constants $c_1 < c_2$	$c_1 \cdot M \cdot P$ operations M points, P parameters c_1 constant
Convergence without dampening	can convergence towards saddle points	no convergence possible
Convergence with appropriate dampening	faster high sensitivity to dampening value	slower low sensitivity to dampening value
Appropriate dampening value	by Eigenvalues	can be chosen fixed

Table 3.1: Summary of the experiments with second order motion derivatives

tant in cases where the Hessian isn't invertible, because of rank deficiency due to parallel derivative vectors. Introducing a small dampening value is sufficient then.

As the error surface of Newton's method is not necessarily convex, minimization without dampening may result in convergence towards a saddle point. An appropriate dampening value has to be as large as the smallest eigenvalue of the Hessian. However the convergence rate showed to be very sensitive to the dampening value. If the dampening value is only slightly larger than the smallest eigenvalue, large jumps can occur leading outside the locally convex region.

An appropriate dampening can be found by using the LM-algorithm (see section 2.3.4 or by calculating the eigenvalues of the Hessian. If there are negative eigenvalues, the dampening should be at least as high as the absolute value of the smallest eigenvalue to have a positive definite Hessian matrix. In practice the calculation of eigenvalues may be too time consuming and the initially chosen dampening value for the LM algorithm may be too small resulting in a large initial step outwards the locally convex region.

Newton's method is the better one concerning error approximation and stability. If computation time is the most important issue and errors and parameter changes are small, the Gauss-Newton estimation is more appropriate, because the dampening value is less critical and can be chosen manually and left fixed. The Newton method requires a dampening value, which has to be calculated each iteration from the eigenvalues of the Hessian plus a manually chosen offset. Because Newton's method is very sensitive to the dampening value, it can be expected, that the advantage of better error approximation made by Newton's method will be hard to observe in real applications. In this thesis the Gauss-Newton method was prefered, because of the lower computational cost. Additionally the dampening value is rather easy to chose, because the intervals and sizes of the expected errors and derivatives are approximately known for human motion estimation. In that

case the effect of manually chosen dampening values is easy to interpret and predict. The experiments showed, that the estimation is not sensitive to the chosen dampening value, which is fixed for most experiments.

3.6 Singularities and the Gimbal Lock problem

There are two types of singularities, that can occur during optimization, which lead to a non-invertible Hessian matrix. Both problems are solved by Dampening (see section 2.3.3) the Gauss-Newton estimate, which makes the Hessian invertible.

The first arises, when all derivatives of all correspondences for one joint are zero, as described in the previous section. The solution by dampening was also explained there.

The second type of singularities arise, when two axes become identical, which is also called the gimbal lock problem. This happens, if the second axis of a ball and socket joint is rotated by 90 degrees from the initial position, then the first and third axes become parallel. This can occur for the human body only for the hip and shoulder. While this is a serious problem for people, who animate virtual characters manually, in the experiments of this work it was never encountered to be problematic.

The reason is, that both axes become not perfectly identical in practice, where pose is estimated from noisy data. However, cases occur, where the axes are nearly identical. In that case, the Hessian gets nearly singular. Then a large parameter change will occur, that can lead outside the locally convex region of the correct minimum. However, by dampening the Hessian gets invertible and the estimation is stabilized resulting in small reasonable parameter changes.

Additionally it should be noted, that the gimbal lock problem can be avoided by estimating joint angles in the same way as for the global rotation. However, it is then not possible any more to constrain the motion as explained in the next section, because the joint angles can not be interpreted like abduction, flexion and twisting.

3.7 Constrained Estimation

The constraints implied by the dynamics of human bodies, e.g. connection of body parts by joints and bones with fixed length, are included in the motion function (3.42). If in addition the motion of each joint is limited, e.g. as it is for humans, these constraints can be included by adding limits to the possible joint angles. For ball-and-socket joints of human bodies this a more complex topic and is covered in the next chapter.

There are two methods to incorporate additional motion constraints of the articulated object into the estimation. The first one, easier to implement, is limiting the joint angles to certain intervals after each iteration of the pose estimation. The second is to add regularizing barrier functions within the pose estimation.

3.7.1 Limiting Joint Angles

To ensure that parameters are inside certain intervals one solution is to set the parameters after each iteration in the estimation back to the allowed limit, if they are outside the interval. Though easy to implement, this approach contradicts the idea of optimization to make as large steps as possible in each iteration. After large changes in the parameters a position may be reached that is far outside the valid region. Setting back the parameters to their specific interval may result in no further progress towards the correct minimum. This was reported to be problematic for estimation of hand pose in [16], where gradient descent methods were compared with Gauss-Newton like methods. The conclusion made is, that the gradient descent is advantageous, because it makes smaller steps along the border of the feasible region.

However, in general their is no guarantee, that making smaller steps avoids the problem of getting stuck. The characteristic of gradient descent methods to make smaller steps can be achieved by introducing a dampening factor in the optimization with the Newton or Gauss-Newton method as well.

3.7.2 Barrier Functions

Another way to incorporate constraints in the optimization is the use of barrier functions. They add a regularization term to the objective function, which penalizes parameter values outside the feasible region. These so called barrier functions have a strong increase in their value at the boundaries.

A detailed description how to include these regularization terms is given in section 2.5. Important in practice is, that the additional regularization has to be assigned an initial weight, which is then decreased in each iteration. The lower this value, the nearer to the border of the feasible region the solution is. A higher weight draws each parameter estimate more towards the center of its interval and therefore, the position in parameter space more towards the center of the feasible region, which may lay outside the locally convex region, where the sought minimum is.

The barrier function method gives theoretically the correct result, but the initial weight for the regularization term has to be chosen sensibly, such that the estimate in the first iteration is not outside the convex region. An approach like in the LM-algorithm can help to avoid these problem by lowering the weight until the error decreases, but needs additional time consuming iterations.

3.8 Comparison with Gradient Descent

Throughout the literature on body pose estimation from 3D-point-3D-point correspondences there are various optimization techniques that try to minimize the difference between observed 3D points and the surface of the model. Very often this optimization problem is solved with a Gradient Descent method and numerical derivatives, e.g. numerical derivatives are used in [30, 16, 73]. Gradient Descent is still very popular, e.g. in [72] Euler-Lagrange equations are integrated over time, which is similar to standard Gradient Descent with a fixed step size.

The Gradient Descent method has two main problems

- Slow convergence in flat regions near the final solution, because the change in parameters is proportional to the length of the gradient.
- Step sizes have to be chosen manually for fast convergence, often different step sizes are needed for each parameter.

Therefore, there exists a variety of methods to choose the step sizes optimally, Steepest Descent, Conjugate gradient methods etc. (see also chapter 2). Often there are different step sizes for each parameter. This is for example due to different orders of magnitude of the gradient components or because the error surface has different curvature in different directions.

It can be expected that the Gauss-Newton method solves the problem of pose estimation more efficiently, cause of the following reasons.

The Gauss-Newton method increases in performance, the better the residual functions can be approximated with a linear function, because the Gauss-Newton method assumes, that the objective function is quadratic, which is true if the residuals are linear. If they are, the solution is found in a single step. The residual functions for pose estimation in this thesis describe the motion of points in 3D space. These motions are rotations around known axes and a translation, which are known to be more linear the smaller the angle is. The linear approximation is quite good, because relative transforms are estimated and the motion between frames is assumed to be small. This is illustrated in sections 3.4.5 and 3.5.1. Therefore, the convergence rate of the Gauss-Newton method should be higher then a Gradient Descent method, especially close to the minimum.

One disadvantage of the Gauss-Newton and Newton-Rhapson methods is their need for sufficient amount of data, because in each iteration a Least Squares problem is solved. If the data amount is small and the Hessian becomes nearly singular, the steps become larger. An efficient solution to this problem is dampening, which requires one control parameter to be set manually, as explained in sections 2.3.3 and 3.5.1.

Because Gradient Descent (GD) converges very slowly in flat regions, there are various approaches to increase the convergence rate. In the work of Bray [16, 73] a new Gradient Descent method is proposed, called 'Stochastic Meta Descent' (GDSMD). The GDSMD method is applied to estimate the motion and pose of a human. The motion function is modeled similar as in this work by rotations around known axes, however optimization

relies on numerical derivatives. Additionally in both publications a comparison with other common optimization methods was made, these were the Levenberg-Marquardt extension of Gauss-Newton in [73] and BFGS and Powell's method in [16]. The proposed GDSMSD was reported to work faster and more accurate. Therefore, a detailed comparison with Gauss-Newton is made here.

3.8.1 Stochastic Meta Descent

We describe here briefly the GDSMD method. The stochastic character of this approach is due to random sub sampling of the complete data in each iteration.

A Gradient Descent method updates parameters in one iteration using the update formula

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \boldsymbol{\alpha} \cdot \nabla f_i(\boldsymbol{\theta}) = \boldsymbol{\theta_t} - \boldsymbol{\alpha} \cdot J^T \boldsymbol{r}(\boldsymbol{\theta}_t) \tag{3.54}$$

where t is the current time step or iteration number, the operator '$\cdot$' indicates a componentwise product, such that each parameter may have a different step size.

According to [73] the idea of GDSMD is to update the step sizes automatically depending on the steepness and direction of the gradient in last time steps. Especially if the error function exhibits long narrow valleys, a gradient descent might alternate from one side of the valley to the other. To model this behavior, the stepsize is increased for parameters, whose gradient had the same sign in the last steps and is decreased, if the sign changed. In detail this is:

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} \cdot \max\left(\frac{1}{2}, \boldsymbol{\mu} \cdot \boldsymbol{v_t} \cdot \nabla f_t(\boldsymbol{\theta}_t)\right) \tag{3.55}$$

where $\boldsymbol{\mu}$ is a vetor of meta step sizes, which control the rate of change of the original stepsizes and $\boldsymbol{v}$ is the exponential average of the effect of all past stepsizes, which reads:

$$\boldsymbol{v}_{t+1} = \lambda \boldsymbol{v}_t + \boldsymbol{a}_t \cdot (\nabla f_t(\boldsymbol{\theta}_t) - \lambda H_t \boldsymbol{v}_t) \tag{3.56}$$

where H is the Hessian of the objective function. If λ is zero as for the results presented in [73], this simplifies to

$$\boldsymbol{v}_{t+1} = \boldsymbol{a}_t \cdot \nabla f_t(\boldsymbol{\theta}_t) \tag{3.57}$$

Therefore, λ was also chosen to be zero in the experiments conducted in the following.

Constraints on possible motion of the human are imposed by mapping back each parameter to its allowed interval in each iteration. Because the resulting parameter change is smaller in that case, the meta step sizes are updated accordingly.

3.8.2 Experimental Results for Global Transform

In this experiment the global transform is estimated, while only the translation in x-direction and the rotation around the height axis are changed. Figure 3.24 shows on the left the starting pose for the estimation together with the observed data point cloud, which is generated from the pose shown on the right. To each point position Gaussian noise is

added with a standard deviation of 5cm. In section 4.6 a detailed description, how the correspondences between model and data are established. Here, only the convergence rate of the optimization methods is of interest. The difference between start and target pose is

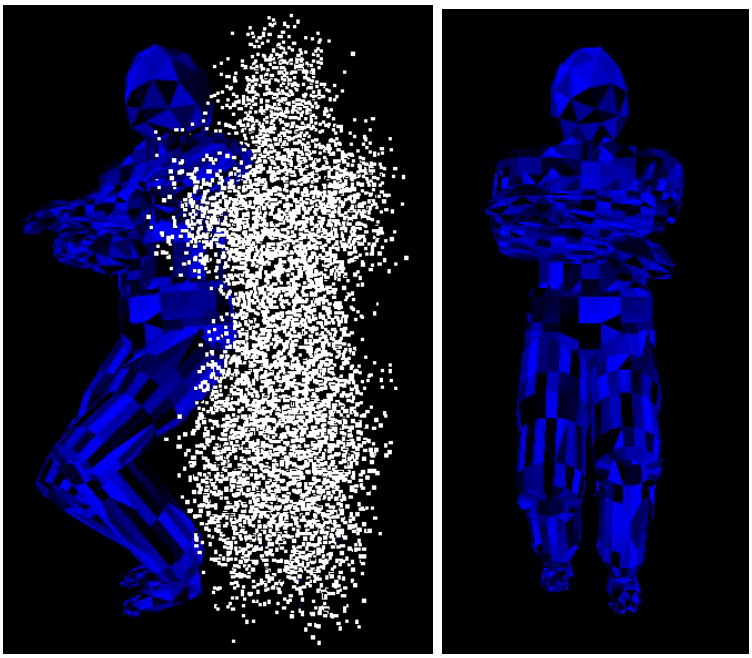


Figure 3.24: Starting pose on the left with the disturbed point cloud of the target pose (noise deviation 0.05m) and target pose on the right.

rotation around the y-axis with -40 degrees and translation in x-direction with 0.35m.

Contour levels of the error surface are shown in Figure 3.25 with the minimum at $(0, 0)$. The error surface of the global transform is calculated by changing the rotation from -40 to +40 degrees by 0.5 degrees and the translation in x-direction from -0.4m to +0.2m by 0.02cm. For each model pose the error is calculated setting up correspondences by Closest Point using all 6500 observed points. The plotted error is the average distance between model and observed points. As visible, the error surface is almost quadratic, which can be expected, because the translation in x-direction is linear giving a quadratic error for the translation and the linear approximation of the rotation is good for small angles. It is visible also, that the quadratic approximation of the error function is valid, even when the correspondences change in each iteration, which is due to the nearest neighbor correspondence building.

Compared are in the following *Gauss-Newton* (GN), Gradient Descent (GD) and the *Stochastic Meta Descent* (GDSMD) by estimating the 6 parameters of the global transform. The x position and the rotation angle around the y-axis are plotted for each three methods. From the total set of 6500 observed depth points 1000 points were chosen randomly for each iteration. The estimation is stopped, if the parameter change is below a certain threshold

or at a maximum of 200 iterations. The result for GN is shown in Figure 3.25 as a green solid line, for GD as blue crosses, and for GDSMD as a dashed red line. The initial step sizes for GD and GDSMD are 0.1 for translation and 1 for rotation. The additional control parameters for GDSMD are $\mu = 1000$ and $\lambda = 0$.

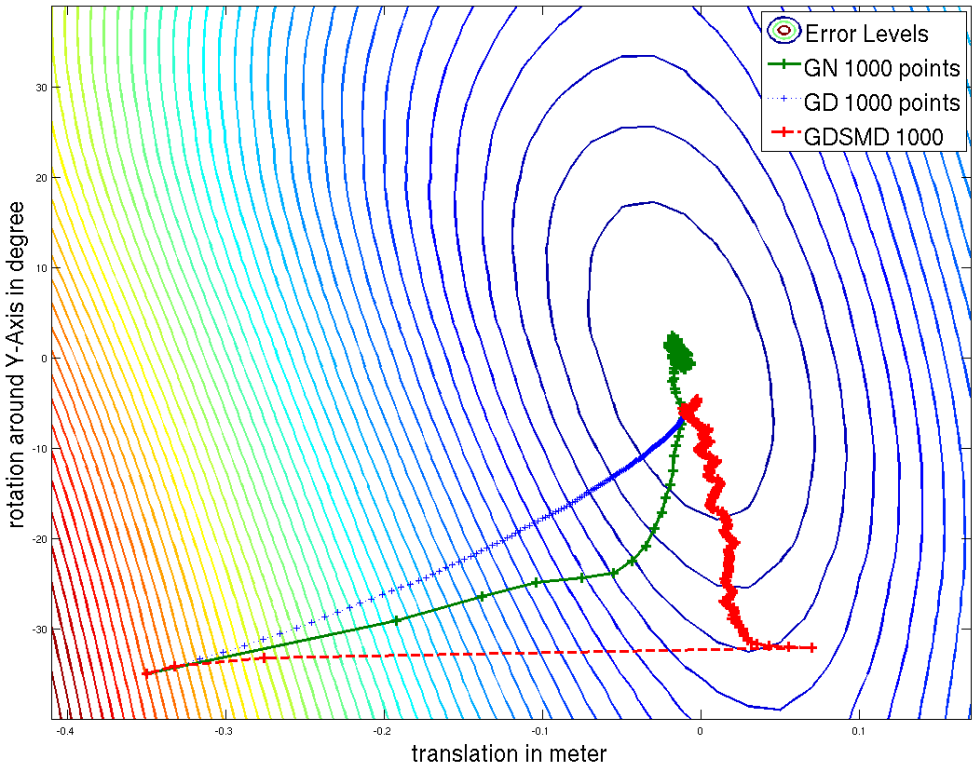


Figure 3.25: Estimation of the global transform for 1000 correspondences. Plotted is the error surface and the iterations for Gauss-Newton (GN)), Gradient Descent (GD) and Meta Descent (GDSMD). Real minimum at $(0, 0)$

The Gradient Descent methods don't reach the minimum at $(0, 0)$ within 200 iterations. The main reason for the bad performance of the GD methods is their characteristic to make small steps on flat surfaces, because the parameter change in each iteration is proportional to the size of the Jacobian (the gradient), while the GN assumes an quadratic error function resulting in larger steps on smooth surfaces.

The GDSMD approach tries to overcome the slow convergence of the GD in flat regions by adjusting the stepsizes according to previous parameter changes. However, in the case shown, this feature has also the effect to increase the step size dramatically in steep regions. This dramatic increase in the step sizes made the GDSMD method in the shown case even slower than the standard GD without varying step sizes. Possibly better chosen values of $\mu = 1000$ and $\lambda = 0$ and better initially step sizes could overcome the effect. In contrast, the GN approach doesn't need any kind of control parameters, making it much easier to

use.

Using the the same starting point and the same control parameters for GD and GDSMD the estimation is repeated using only 10 random correspondences. Results are shown in Figure 3.26. As visible the GN starts to jump very erratic, because of the small amount of data. The GD methods are able to reach a point near the minimum, but the larger effect of noise due to the small data amount is also visible.

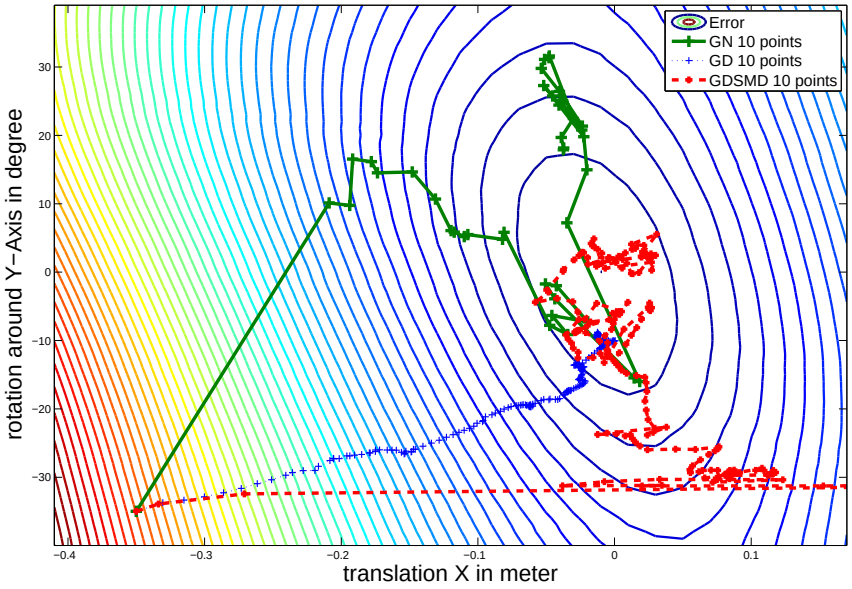


Figure 3.26: Estimation of the global transform for 10 correspondences. Initial step size of GD and GDSMD was 0.1 for translation and 1 for rotation.

With a smaller initial step size for the Gradient Descent methods it could be expected, that the resulting steps in each iteration are smaller and less erratic. Results using 0.01 for translation and 0.1 for rotation are shown in Figure 3.27. In this case the GDSMD is stuck far outside the minimum, showing that it is not necessarily better than the standard GD.

3.8.3 Experimental Results for Shoulder and Elbow Motion

The error surface of the previous experiment is nearly quadratic. To have a more difficult and less smooth error surface another experiment is carried out. The global position and rotation of the model is unchanged, but the shoulder twist and the elbow flexion are changed. Due to the Closest Point correspondences, which are changing significantly for the lower arm, when it is moving near to the body, the error surface is less smooth. Also the amount of correspondences is lower (around 650), because correspondences were only established for observed points, whose closest corresponding model point is on the upper

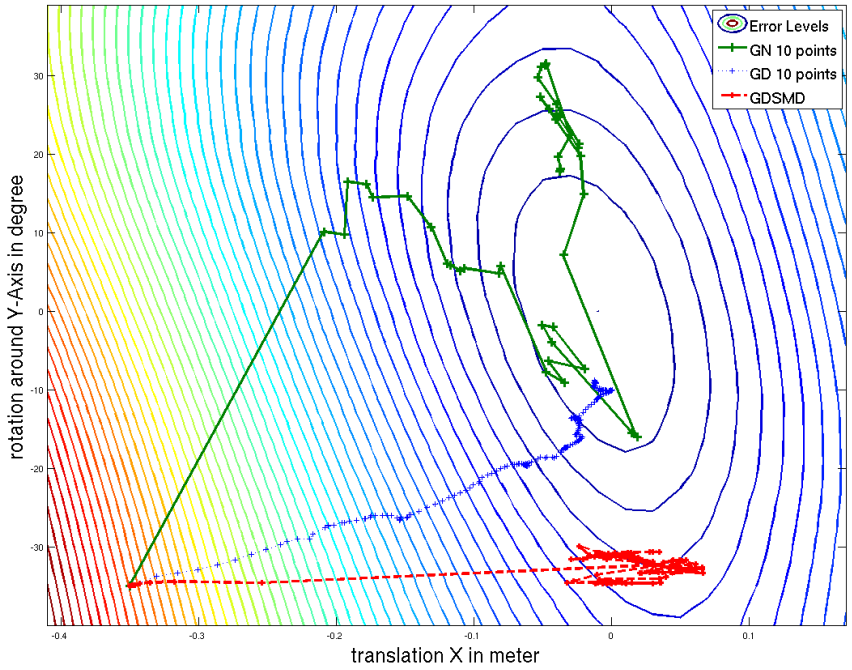


Figure 3.27: Estimation of the global transform for 10 correspondences. Initial step size of GD and GDSMD was 0.01 for translation and 0.1 for rotation.

or lower right arm. The smaller amount of correspondences makes the error surface less smooth.

The estimation is performed using an initial step size of 20 for GD and GDSMD. The additional meta step sizes for GDSMD were $\mu = 1000$ and $\lambda = 0$. Figure 3.28 shows the development of shoulder twist and elbow flexion together with error contour levels. Within 200 iterations all methods reach the correct minimum while performing similarly.

Shown in Figure 3.29 are two runs of the same minimization with 10 random correspondences in each iteration. In the second run, the standard GD is not shown for better visibility. As visible, the noise and the nearest neighbor correspondences lead to an erratic development of estimated angles. Smaller initial step sizes for the GD methods smoothes the development of values, but also increases the possibility to get stuck at a local minimum as in the plot above, where the GDSMD method got stuck in the beginning. On the other hand the small amount of data can lead to large changes in the wrong direction for the GN method.

3.8.4 Summary of Experiments with Synthetic Data

Comparing the Gradient Descent (GD) and the Stochastic Meta Descent (GDSMD) directly, the GDSMD performs not necessarily better than standard GD with fixed step sizes. However with changes in the step sizes and additional meta step sizes it is possible to tune

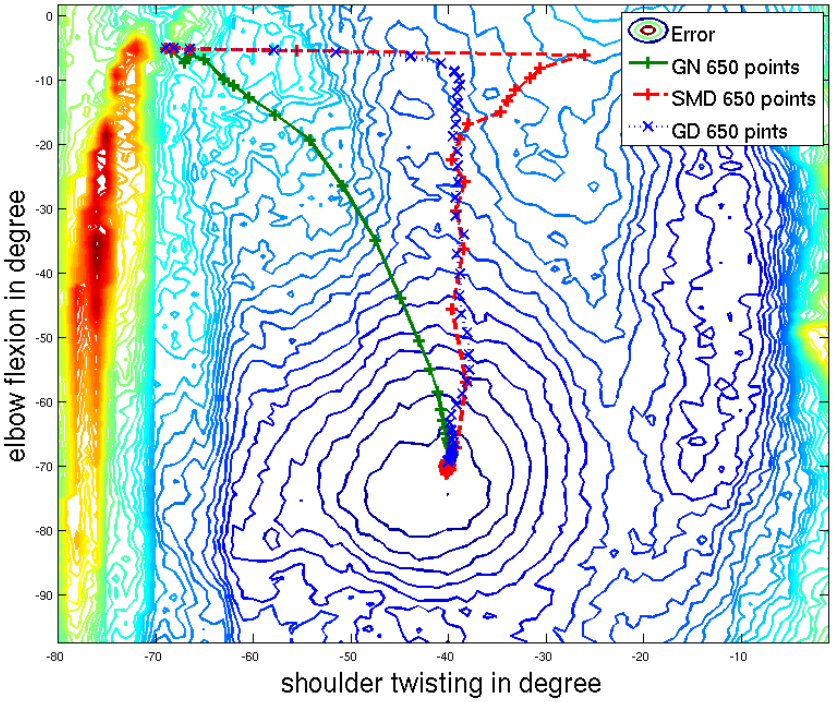


Figure 3.28: Estimation of the shoulder twisting and elbow flexion for 650 correspondences. Initial stepsizes are 20 for both angles.

the control parameters, such that GDSMD needs less iterations than standard GD.

The GD methods have the advantage in comparison with Gauss-Newton (GN), that even a small amount of data is sufficient to find a minimum. However, in the presence of noisy measurements averaging smoothes the effect of noise, such that it is beneficial to use more data points at once.

The Gauss-Newton method is not in the need of additional control parameters and converges faster in flat regions. The drawback of the GN approach is the need for many data points. There should be at least as much data than parameters to estimate, otherwise it is not possible to calculate a solution in the Least Squares iteration step, because the Hessian matrix $J^T J$ isn't invertible. Important to note here is the possibility of *dampening* (see Section 2.3.3) by introducing a single dampening value, which is added as regularizer to the objective function. The *Levenberg-Marquardt* extension of GN is an automatic method to calculate the dampening value. However, if the fraction of data amount to parameters is small, a manual dampening value is more appropriate, because the initial automatic value used in LM might be too small, which can result in a jump out of the locally convex region.

The experiments showed (as can be expected), that optimization in the presence of noise is more accurate and stable (with respect to local minima), if more data is used simultaneously. If only a subset of the whole data is used in each iteration, processing time is decreased, but additional iterations are necessary, because step sizes have to be small.

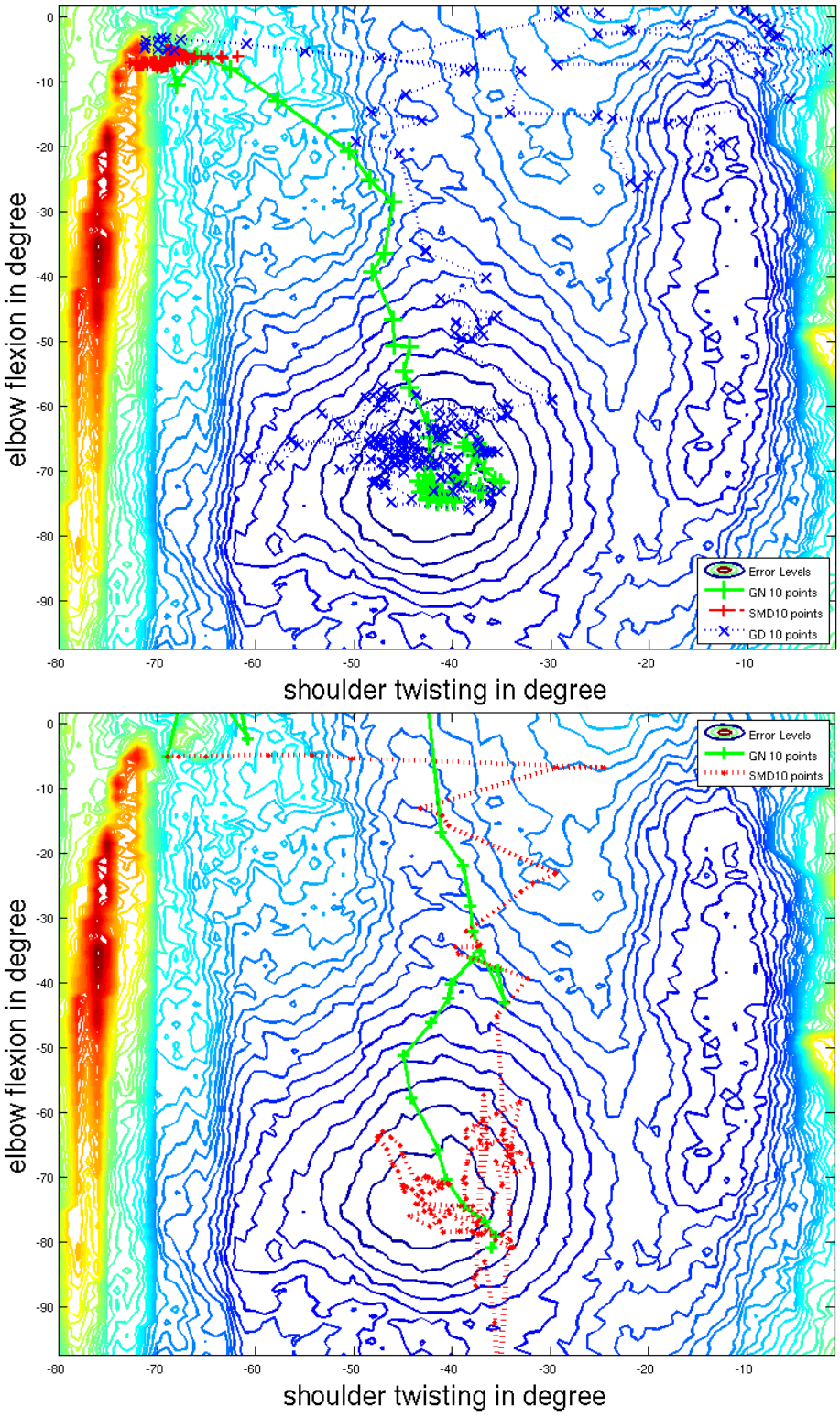


Figure 3.29: Estimation of the shoulder twisting and elbow flexion for 10 correspondences. Two runs with the same control parameters.

Method	GN	GD	GDSMD
Complexity M points, P DOF	$O(MP^3)$ (matrix inversion)	$O(MP)$	$O(MP)$
Time for 24 DOF 1 iteration			
3400 points	70 ms	4.3 ms	5.5 ms
1700 points	35 ms	2.1 ms	2.8
850 points	21 ms	1.2 ms	1.5
Necessary iterations	3-5	20-200	20-180
Control parameter	1 dampening value	P stepsizes	P stepsizes and P meta-stepsizes
Convergence Speed	fast, especially in flat regions	slow in flat regions	faster than GD in flat regions
Sensitivity to control parameter	very low	high	very high

Table 3.2: Summary of the comparison between Gauss-Newton(GN), Gradient Descent (GD) and Stochastic Meta Descent (GDSMD). Values are taken from the results of section 4.6.4 for motion estimation with 24 degrees of freedom (DOF).

An argument in [16], which explains the reported worse behavior of GN like methods, is their characteristic to make large steps in each iteration. If the range of possible motion for each joint angle is constrained by min-max values, it is possible, that the minimization does not improve significantly in one iteration, because a large step may lead far outside the feasible region and is then set back by the min-max limiting. However, for GN it is possible to reduce the step size by increasing the dampening value, which shifts the GN more to a GD. For GD methods it is also not guaranteed, that the small steps overcome the problem of getting stuck at the border of the feasible region. In the experiments made in this work the problem of getting stuck on the border with GN did not occur.

3.8.5 Comparison for Real Data

The summary above, shows that the GN optimization is preferable.

As long as the number of estimated parameters is low, the increased complexity of the GN method (matrix inversion) is not noticeable. However, if the number increases, as given here for 24 parameters the increased complexity of GN is clearly recognizable.

In Table 3.2 the properties of the three optimization methods Gauss-Newton(GN), Gradient Descent (GD) and Stochastic Meta Descent (GDSMD) are shown. The results are taken from section 4.6.4, where motion with 24 degrees of freedom (DOF) was estimated from a real video sequence using depth data from a stereo algorithm. The higher complexity

of GN is due to the necessary matrix inversion of the Hessian, which has a size of $P \times P$, where P is the number of estimated DOF. Given is the time in ms for one iteration of GN, GD and GDSMD on a 2Ghz Intel Core2 Duo (only 1 CPU used). The difference of GD and GN is not equal to the different complexity, because a large amount of time is spent on other calculations rather than matrix inversion, e.g. calculation of new model points after applying the estimated relative motion angles. As visible, the time depends linearly on the number of points. The GD are significantly faster, however they need usually more iterations to reach the minimum, because they converge slowly in flat regions. The GDSMD converges faster, if control parameters are chosen appropriately. As already explained in the previous section, this is achieved by adjusting the additional meta-stepsizes.

The number of necessary iterations varies for different control parameter settings and also varies according to how large the motion from frame to frame is. For small motions, less numbers of iterations are necessary.

The sensitivity of the methods increases with the number of adjustable control parameters. Sensitivity means, that the convergence speed varies significantly with small changes in the control parameters and that the optimization may not converge at all if the stepsizes are chosen too large.

However, it is possible to tune the control parameters of the GDSMD method, such that the overall optimization is up to 2 times faster than GN for a large number of DOF (more than 10). Because these tuning of control parameters is time consuming and has to be repeated, if the input data or the model changes, we favor the GN method.

Chapter 4

Tracking Human Motion

The previous chapter discussed the theoretical background of articulated pose estimation assuming known body models. In this chapter their application to tracking of human motion is presented. The methods to find correspondences between body model and observed data (images) are described and experimental results are given.

At first body models and methods to obtain them for a specific person are described. This is followed by implementation details about articulated pose estimation, methods to find necessary correspondences and how to do this efficiently for various kinds of data. These are namely depth range data from stereo algorithms, silhouette information from gradient, color histograms and optical flow data from a corner tracker. Experimental results for each data type are given concluded by multi-cue integration.

4.1 Body Models

A body model for human motion capture has different important aspects, which have to be distinguished. These are in detail:

- **The geometry or skin.** The most important aspect of the model is the skin surface. This skin model does not include any aspects of motion or animation. It just describes the outlook of the human for one specific pose. For example each body part is a cylinder or the whole body consisting of one triangle mesh as obtained from laser scanners.

- **The skeleton**, which defines the motion capabilities. The skeleton gives the degrees of freedom of the motion model by defining a hierarchy of joints and specific bone lengths.

- **Joint positions.** This defines the relation of skeleton and surface skin model. It can be thought of 'where in the skin' the skeleton is positioned, like shown in Figure 4.1. In addition to the joint positions, it has to be specified for each element of the skin model to which joint it belongs. For blend skinned models, each skin surface element even belongs to multiple joints with different weights.

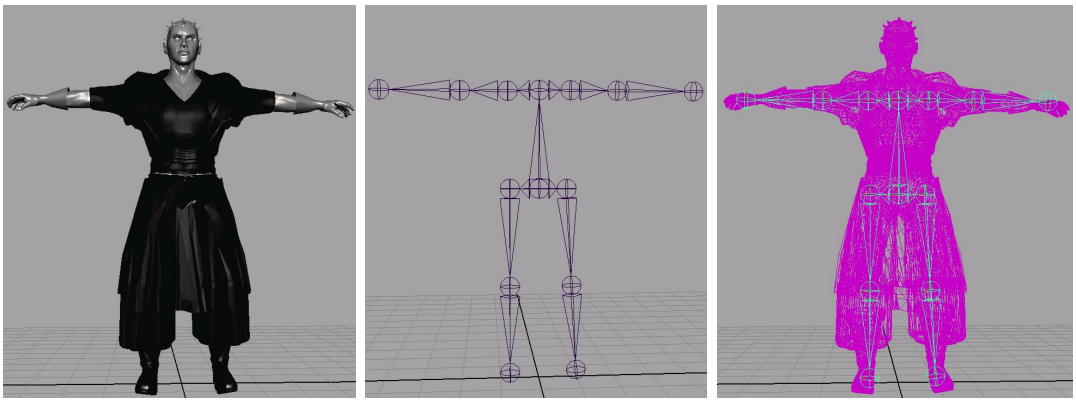


Figure 4.1: Left: Skin surface model. Middle: skeleton. Right: Combination

These three parts define how a point on the surface moves with respect to joint parameters, e.g. joint angles as in this work. If the three parts are given, it is possible to evaluate skin point positions for given joint values, which is also referred to as *'skinning'* in modeling tools like Maya's 'MotionBuilder' [86], cinema4D's 'MOCCA' [85] or Blender [27]. The calculation of skin point positions with known model parts is known in robotics as *forward kinematics*, because it allows to calculate the pose of a specific object part, for example the end effector, if joint angles are given.

The inverse kinematics is necessary for parameter estimation from observed data with known correspondences as in this work, or for example estimating pose from 3D marker data as provided by professional marker systems , e.g. from Vicon [120] or MetaMotion [95]. While the inverse kinematics is usually understood as estimating joint angles from a given end-effector pose, in motion capture joint angles have to be estimated from different kind of data, e.g. skin surface positions, which is referred to as pose estimation of articulated objects.

The ideal motion capture system would be able to do the *'inverse skinning*', which is estimation of all the mentioned aspects above, namely skin surface, skeleton, joint positions and joint angles.

4.1.1 Two-Step Approach

Most systems simplify the task by making the assumption, that all three aspects are known for the observed human and the problem of motion capture is reduced to estimating joint angles per frame. The size, shape and skeleton of the body model is estimated in a previous step before the capturing of motion data.

This approach has the following advantages:

- The skeleton is nearly the same for healthy humans, except for the bone lengths. Therefore most motion capture systems assume a similar skeleton for all subjects and often apply knowledge about relative bone lengths from anthropometric statistics.

- Joint positions and skin geometry can be obtained from a specific pose, e.g. a T-shaped like. Often a time consuming accurate off-line method to obtain skin model and joint positions is applicable, like in sport motion analysis or motion capture for animation in movies or games. In this step manual interaction is often tolerable, because it has to be done only once for each person.

- Implementation of this two step approach is much simpler than optimizing all parameters at once for a video sequence.

- Optimization of joint angles alone for one image frame is much faster and less sensitive to noisy measurements.

The main drawback of estimating only joint angles per frame, is the limited accuracy of motion, pose and surface reconstruction. One example, that tries to estimate skin geometry and joint angles at once for each video frame is the work of Plaenkers & Fua [102]. Their goal was to very accurately reconstruct the appearance of the observed subject in each frame, e.g. including muscle bulges or skin deformations.

In motion capture systems there exists a wide variety of models used for estimation. Their complexity and accuracy mainly depends on aspects like kind of input data, achieved degree of accuracy, application environment etc. Examples are from simple to complex: stick-figures (just a skeleton without skin), cylindrical body parts [34], ellipsoidal models [17, 102], arbitrary rigid body parts [111], linear blend skinned models [16]. The complexity of the skeleton varies accordingly, with up to over 30 DOF for hand models [16].

In this thesis the template models consist of rigid fixed body parts of arbitrary shape. Their advantage is, that they can be represented by a scene graph, stored in VRML [20] format and can be effectively rendered on graphics hardware using OpenGL.

The motion capabilities of the models are consistent with the MPEG4 specification with up to 180 DOF, while the maximum number of estimated DOF in the experiments is 28.

The two step approach (off-line model fitting before the capturing and estimation of joint angles during capturing) is also taken here, because reconstruction accuracy isn't the main goal.

4.1.2 MPEG4 Body Model

The ISO/IEC-14496 Standard MPEG-4 [96, 57] describes algorithms for compression of audio and video data. It is the successor of MPEG-2 and in addition to the video- and audio-compression methods various other modules for animation are included like extended VRML [20] support, or digital rights management etc. The standard is designed, such that it is possible for developers to implement only a subset of all the features.

Important in this work is the animation framework of MPEG4 [4]. There are definitions for facial animation and body animation. The body animation part defines 180 body animation parameters (BAP), which are joint angles around defined axes. The MPEG-4 description of motion capabilities was chosen in this work, because it allows to capture

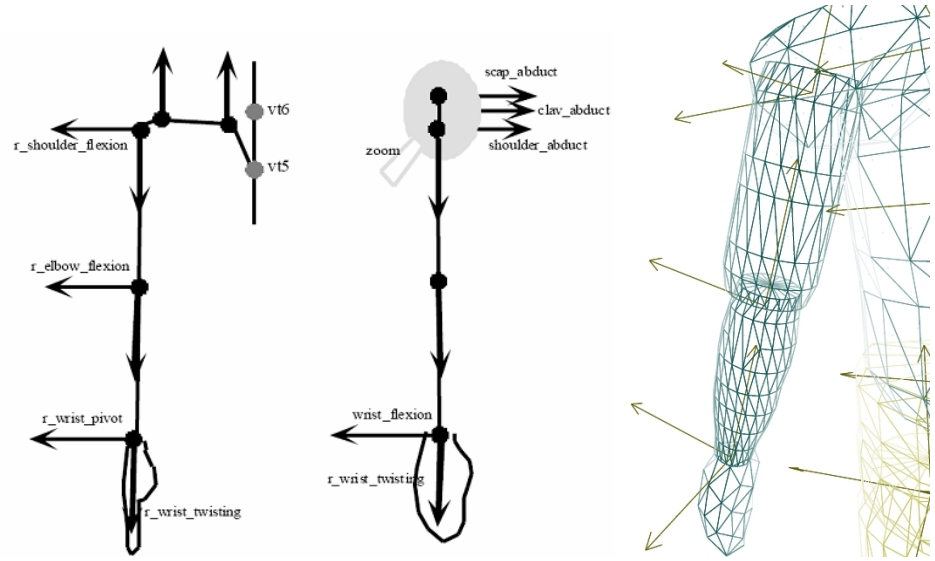


Figure 4.2: Left: Motion capabilities of the arm as defined in the MPEG4 standard. Right: One arm model with MPEG4 conformal joints as used in this work.

motion in a standard format, can be easily converted to other formats and allows reanimation of other models. Additionally the motion definition is very close to the real motion capabilities of humans. However, a MPEG4-compatible format of model and motion capture data is not implemented, but the description of motion dynamics is consistent with MPEG-4. An example model is shown in Figure 4.2, where the rotation axes (MPEG4 animation BAP IDs) are shown as arrows.

The human models are stored in the VRML format, which are represented in a scene graph after loading with the *openSG* [108] library. The successor of VRML is X3D [133], which extends the VRML description language. The relation between MPEG-4 BAP numbers and scene graph nodes is given for each model in an additional file.

4.2 Model Acquisition

There are two main approaches to model fitting of human bodies. The first are shape-constructing methods like 3D-shape-by-silhouette [82, 128, 5], laser range scanning (as used for obtaining the digital 3D hull of handmade models) or shape from dense disparity maps etc. These methods obtain first a surface skin model of the person and estimate or fit the underlying motion capabilities (the skeleton) in a second step to the estimated skin model. These approaches rely on perfect segmentation of the person and background and are sensitive to noisy measurements, but do not need a template model and are therefore well suited to fit anomalies like unusual clothes etc. An example for a shape-by-silhouette approach is [128], where an individual is captured by 16 calibrated cameras in a box with homogenous background. In [82] a real-time silhouette approach using 4 cameras is able to track roughly the motion of a person and recognizes different postures.

The second approach is to start from a template model, usually a combination of skin and skeleton, and to adapt the template to the currently observed person by minimizing the difference of some calculated features between model and images. Contour or silhouette information is usually among them. A good example is the work of Pascal Fua, e.g. [102], where a body model consisting of scalable spheres is fitted to silhouette and depth information. Another recent approach, which gives very good fitting results, but relies on manual interaction is given in [106]. In that work a template skin and skeleton model is fitted to 6 synchronized camera views, where only the internal camera parameters are known.

In this section a method is presented, which is able to construct a 3D skin and skeleton model of a person by analyzing images from a single camera. Assuming a known static background, the person is captured in 6 roughly predefined poses. A template model is then fitted to the person's contour in the 6 images, by scaling each body segment differently in size. The Levenberg-Marquardt (LM) method is used to find the scale values, which minimize the difference between the model's projected contour and the person's segmented contour. The fitting for one pose is visible in Figure (4.3). On the left the segmented contour is shown in grey, the middle image shows the template model and on the right the fitting result.
Additional 3D pose parameters like joint angles and the transformation between model and camera are optimized simultaneously, which allow a good fitting even if the person does not match the pose adequately.

The main advantage of this method is the simple setup: Only a single calibrated low-cost camera is needed and standard room lighting is sufficient, because the segmentation is invariant to casted shadows and the fitting algorithm is very robust to small errors in the segmented contours.

4.2.1 Segmentation

The fitting process relies on the contour difference between observed person and projected model. To obtain the contour of the observed person a segmentation of the image fore-

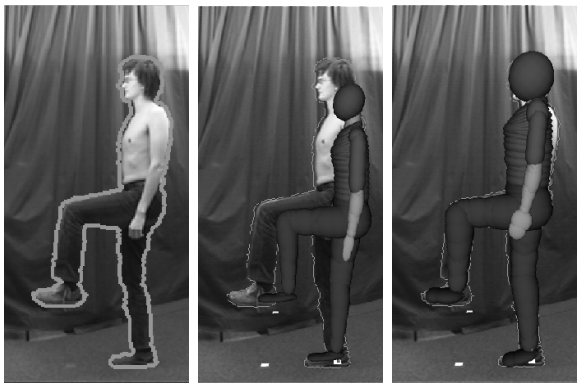


Figure 4.3: The grey contour in the left image is used to fit the template (middle image). Fitted model on the right.

Figure 4.4: Segmentation result

and background is necessary. We assume here that the background is static and known, which is achieved by taking images from the background without the person.

The segmentation should have the following properties:

- The image noise should be taken into account,
- cast shadows of the person should be segmented as background,
- the background may be any static scene.

The segmentation itself is basically done by thresholding a difference image. For each pixel the variance and mean is calculated on a sequence of the static scene without the person to reflect the image noise. This modeling of person and background regions is very similar to that of [63].

To treat shadows, the original image RGB values are converted into the HSL (hue, saturation, lightness) space, which de-couples color and brightness and represents the color as hue and saturation. Ideally shadows don't change the color of the covered scene part, but only their brightness, therefore a representation in the HSL space is well suited to make the segmentation invariant to shadows. Experiments showed, that these invariance is at least achieved to some degree. Problematic image regions are pixels of the person, that have similar color with the background pixels at the same image position, because

segmentation of those pixels has to be done by their brightness difference. In case that the person can be well distinguished from the background by color, because there are only highly saturated colors present, the segmentation is indeed invariant to shadows. Correct segmentation of shadows could be improved by a similarity measure like in [55], but was not necessary in the setup used here.

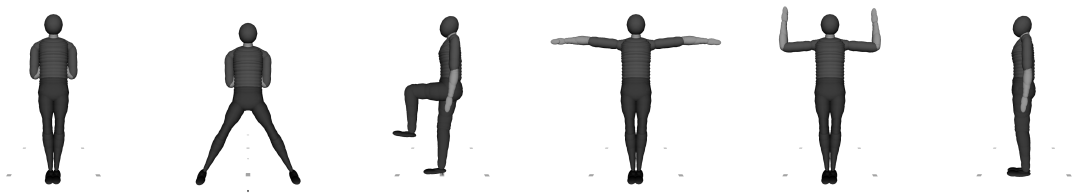


Figure 4.5: The 6 poses used for optimization

Features like color and brightness aren't sufficient for a background of arbitrary appearance. However the use of color and brightness allows for more varying backgrounds than a simple grey value thresholding as visible in Figure (4.7).

To decide whether a pixel belongs to the background or to the person, the *Mahalanobis distance* (page 35 of [13]). between HSL-pixel value and HSL-mean is thresholded. Therefore it is necessary to convert the polar coordinates of the HS channels into Cartesian coordinates (written here as hsL).

Let Σ be the covariance matrix at an image position and $\boldsymbol{\mu} = (\bar{h}, \bar{s}, \bar{L})$ the corresponding mean value. Then a pixel with values (h, s, L) belongs to the person if

$$\sqrt{\begin{pmatrix} h-\bar{h} \\ s-\bar{s} \\ \gamma(L-\bar{L}) \end{pmatrix}^{\top} \Sigma^{-1} \begin{pmatrix} h-\bar{h} \\ s-\bar{s} \\ \gamma(L-\bar{L}) \end{pmatrix}} \geq t.$$

The manually given threshold t and the parameter γ are the same for all pixels. The scale value γ gives the possibility to vary the importance of the brightness difference. In case of fully saturated colors present in the background, this value may be set to zero, and is usually in $[0, 1]$. To handle cases, where no variance in color is present at a certain pixel position over the image sequence of the background, e.g. if a pixel value is dense or zero in all images, we add the identity to the covariance matrix. This means, we assume a minimum variance at each pixel position, which is necessary for the Mahalanobis-distance thresholding.

A typically segmentation result is visible in Figure (4.4) left. To eliminate noise we apply an opening operator and take the largest connected region afterwards. Small holes are filled by a closening operator, which results in the final segmentation as in Figure (4.4) right.

Distance Transform

The contour of the segmented person is found by a contour following algorithm. The extracted image contour is then distance transformed.

The distance transform is an operator usually applied to binary images. The result of the transform is an image that has at each image position the distance to the contour as the pixel value. There are several different sorts of distance transforms depending upon

Figure 4.6: Distance transformed image of the person contour.

which distance metric is being used to determine the distance between pixels. We use here the *Borgefors* metric, which assigns a horizontal and vertical displacement of 1 pixel the value 3 and a diagonal the value 4. This way integer operations can be performed. We implemented the two-pass algorithm with linear time complexity as described in [118].

An example of a distance transformed image of the segmented person contour is shown in Figure (4.6). The contour of the segmented person is shown in white in the left image.

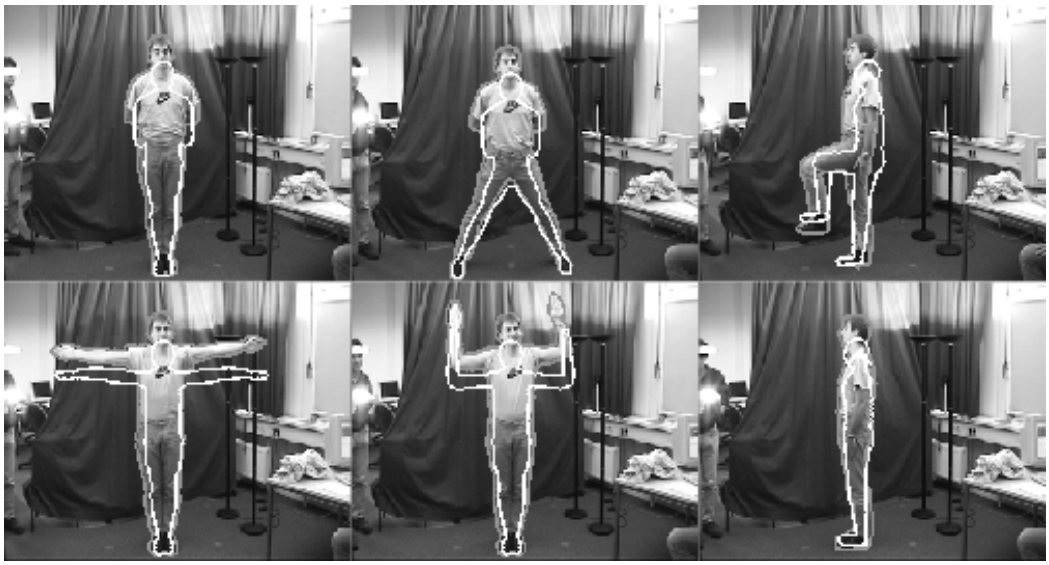


Figure 4.7: Optimization starting configuration. Initial model contour in white.

4.2.2 Objective Function

The resulting model parameters after the fitting are the scale values of each body segment, one for each direction of the local coordinate system. Let $\boldsymbol{\sigma}_i = (\sigma_{x,i}, \sigma_{y,i}, \sigma_{z,i})^T$ be the scale values for the i-th body segment. The optimization problem can then be defined as follows: Find the scale values $\boldsymbol{\theta} = (\boldsymbol{\sigma}_1^T, .., \boldsymbol{\sigma}_N^T)^T$ of the N body segments, such that the model contour looks most similar to the segmented contour of the person. Here an

optimization with numerical derivatives is used, because the objective function is far more complex than for pose estimation of articulated objects with known model size as in the previous chapter. Also real-time processing is not an issue, because fitting the template model to a person has to be done only once and may be done offline before capturing motion sequences of the person.

The objective function consists of three residuals (errors), which model the similarity of model and image silhouette.

$$f(\boldsymbol{\theta}) = \sum_i \left(\boldsymbol{e}_{P,i}(\boldsymbol{\theta})\right)^2 + \left(\boldsymbol{e}_{M,i}(\boldsymbol{\theta})\right)^2 + \left(\boldsymbol{e}_{C,i}(\boldsymbol{\theta})\right)^2. \tag{4.1}$$

The three residuals are now described in more detail.

Person to Model Distance

The contour C_M of the model is calculated with respect to the current scale parameters $\boldsymbol{\sigma}_1, .., \boldsymbol{\sigma}_N$. Afterwards for each pixel on the person's contour C_P, which does not change during the optimization, the corresponding value in the distance transformed image D_M of the model contour is taken as the error for that pixel. Afterwards each error value is normalized by the length of the person's contour. This error $\boldsymbol{e}_P \in \mathbb{R}^{|C_P|}$ has for each pixel on the person's segmented contour a value, which is the distance to the nearest pixel on the model's contour. Each entry of $\boldsymbol{e}^P$ is defined as

$$\boldsymbol{e}_{P,i} = \frac{1}{\sqrt{|C_P|}} D_M(\boldsymbol{p_i}),$$

where $\boldsymbol{p_i}, i = 1..|C^P|$ is the i-th contour point.

Model to Person Distance

Ideally the opposite distances should be also taken into account, which is the distance from each pixel on the model contour to the nearest one on the person's contour. Because the model contour changes during the optimization, only the average error of all contour pixels is taken, such that

$$\boldsymbol{e}_M \in \mathbb{R} = \frac{1}{\sqrt{|C_M|}} \sqrt{\sum_{\boldsymbol{p} \in C_M} (D_P(\boldsymbol{p}))^2}.$$

Constraints

To ensure that the estimated parameters are in a valid range, barrier functions are used. For each parameter an additional error is calculated that represents the distance to the desired interval. E.g. each scale value has to be in $[s_{\min}, s_{\max}]$. In the experiments we set $s_{\min} = 0.4$ and $s_{\max} = 1.9$. The actual choice of the barrier function is not important, used here are logarithmic function as in section 2.5. This part of the error vector is $\boldsymbol{e}_B \in \mathbb{R}^{3N}$.

Justification

We now justify briefly, why the above composition of the objective function is appropriate for our minimization problem. In the optimization process the objective function is minimized, which reads in detail:

$$f(\boldsymbol{\theta}) = |\boldsymbol{e}_P|^2 + |e_M|^2 + |\boldsymbol{e}_B|^2 \tag{4.2}$$

$$= \frac{\sum\limits_{\boldsymbol{p}\in C_P} D_M(\boldsymbol{p})^2}{|C_P|} + \frac{\sum\limits_{\boldsymbol{p}\in C_M} D_P(\boldsymbol{x})^2}{|C_M|} + |\boldsymbol{e}_B|^2 \tag{4.3}$$

Equation (4.3) shows that in the objective function the person to model contour distance and the model to person contour distance have the same influence on the optimization in spite of the higher dimensionality of $\boldsymbol{e}^P$.

The LM method is only appropriate for overdetermined problems see section 2.3.4. The number of residuals is of much higher dimensionality than the parameter vector $\boldsymbol{p}$. However this does not guarantee an overdetermination. Only if the change of each parameter does at least change one error value it is possible to reach the desired solution. Or to be more exact, only if the Jacobian of f has full rank. If only a single view is taken to estimate all scale values, there are some scale values that have no effect on the value of the error function. Or in other words, in a single view some scale values will have no visible effect on the model contour, e.g. scaling in depth parallel to the optical axis.

4.2.3 Multiple Poses

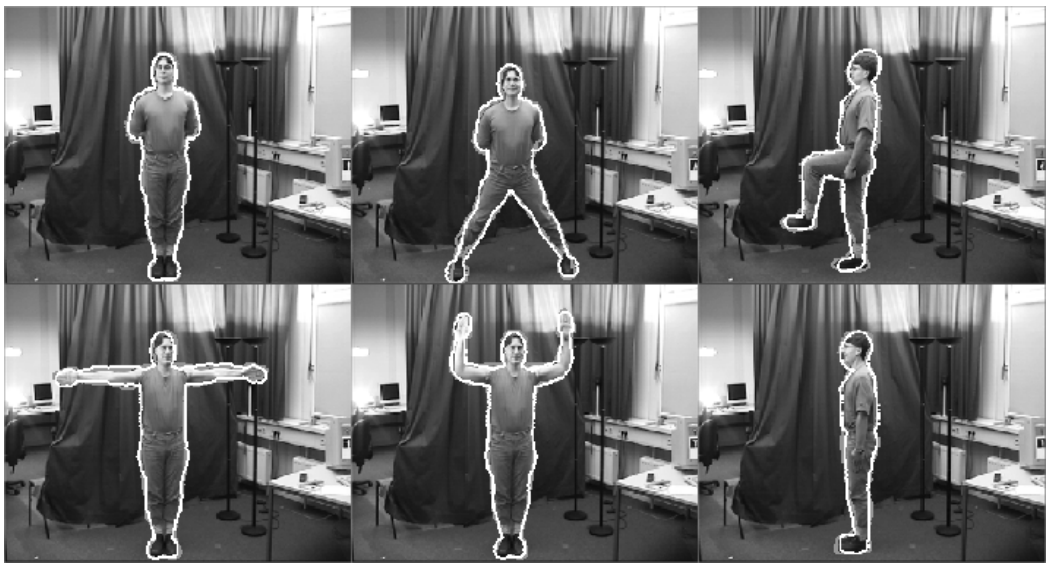


Figure 4.8: Fitting result. Model contour in white.

So far we explained how the optimization can take place for a single image of the person using the person's segmented contour and the rendered model with its contour.

The optimization minimizes the differences of appearance for both contours. However it is not possible to estimate all scale values for each body segment from a single view and pose. Therefore we extend the optimization to multiple poses.

For each of the 6 images the person has to position itself in a special configuration, called a pose in the latter. The 6 different poses are shown in Figure (4.5). The different poses allow estimation of specific scale parameters. For example the overall height is covered in the first pose. The length of the legs is specified in the second pose and the third pose gives the relation between lower and upper leg etc. Important is, that the optimization is not done for single views separately, but all parameters are estimated for all poses simultaneously.

Multiple views increase the number of error values by the dimension of $\boldsymbol{e}_P$ for each view. Additionally there is one error value e_M calculated on the model's contour.

One can imagine that it is rather difficult for a person to perfectly match all the poses. For example, it is very difficult to stand on one foot and bend the knee in exactly 90 degrees.

If the captured pose of the person is not exactly as desired, a simple scaling of all body segments can never match the contour perfectly, like the bended knee in image 3 (upper right) of Figure (4.7). Therefore we estimate for each pose additional parameters, namely the joint values, which were seen to vary significantly. These were for example the angle of the knee in image 3, or the angle of the elbow flexion and the shoulder abduct as visible in the middle image of the lower row of Figure (4.7).

In each pose the person has to stand at the same marked position on the floor. Usually people tend to move a little aside from that position, therefore the translation on the ground is also optimized within a constrained interval. Because it is not possible to stand quite straight without moving for a untrained person, the global rotation is also estimated for each pose separately. The overall optimization parameters are

$$\boldsymbol{\theta} = (\boldsymbol{\sigma_1}, .., \boldsymbol{\sigma}_N, \boldsymbol{v_1}, ..\boldsymbol{v_L})$$

where $\boldsymbol{v_j}$ is the vector with joint angles for the j-th pose and $\boldsymbol{\sigma_i}$ are the desired scaling parameters. For each pose, $\boldsymbol{v_j}$ consists of parameters for some important joints and values for the global translation and rotation.

4.2.4 Fitting Algorithm

The captured images of the person striking the 6 poses are used offline to fit the template body model. The difference between model and person contour is minimized. As the correspondences between both are given only implicitly due to the nearest neighbor approach, the fitting does only reach the global optimum, if the starting point is near enough to it. Problematic are especially the outstretched arms of the 4th pose, see lower left of Figure (4.7). If the person is significantly smaller or larger than the template model, the nearest neighbor approach will fit both the upper and lower contour part of the person's arm to only one contour part of the model. Therefore it is necessary to do the optimization in

	Height			Arm length			Length lower leg			mean
Pers.	Pers.	Mod.	Err.	Pers.	Mod.	Err.	Pers.	Mod.	Err.	Err.
0	1.95	1.98	1.5%	1.90	1.86	2.1%	0.62	0.66	5.7%	3.1%
1	1.78	1.84	3.4%	1.75	1.68	3.7%	0.58	0.62	7.3%	4.8%
2	1.68	1.69	0.8%	1.64	1.63	0.9%	0.51	0.54	7.7%	3.1%
3	1.81	1.86	2.8%	1.83	1.84	0.6%	0.59	0.65	9.5%	4.3%
4	1.92	1.97	2.7%	1.96	1.95	0.6%	0.62	0.69	11.3%	4.8%
5	1.88	1.92	1.9%	1.81	1.83	0.9%	0.61	0.66	8.5%	3.8%
6	1.89	1.86	1.6%	1.95	1.91	2.2%	0.63	0.65	3.6%	2.5%
mean.			2.1%			1.6%			7.6%	3.8%

Table 4.1: Quantitative fitting results for the 7 persons. Values in meter.

two steps. In the first step only the length of the legs, and the upper body are estimated. In the second step all parameters for all poses are estimated simultaneously. In both steps we use the LM algorithm from the MINPACK package [25], which calculates the Jacobian numerically. Additional symmetry constraints guarantee that the left and right arms and legs are of the same size. All together there are up to 100 parameters depending on the used model. For highly complex models, which have the full MPEG4-motion capability, some body segments are coupled, e.g. for the spine segments of the upper body only three scale values are estimated.

4.2.5 Experimental Results on Model Fitting

The fitting was tested for 7 persons, which had to strike all the 6 poses. To make the task easier for the people, the current camera image augmented with the desired model pose was displayed in front of them, such that they could directly see how good they strike the pose. A qualitative result for the fitting is visible in Figure (4.8), where the template was fitted to a person, which was rather different to the template model. The overall fitting process takes about 5 min for acquiring the images as the person has to strike 6 different poses and about 10 minutes for the optimization depending on the hardware. To measure the accuracy of the model, the overall height, the arm length and the length of the lower leg of the models were compared to manual measurements of the real persons. The results are shown in table (4.1).

The achieved average error of 2.1% for the height and 1.6% for the arm length is within the expected range and accurate enough for further use of the model. The error of 7.6% for the lower leg is higher, because on the one hand measuring the ground truth was more difficult and inaccurate. On the other hand the image information for the estimation of the lower leg scale values is much less than for the height and arm length. Therefore, small segmentation errors in the area of the lower leg have a larger effect on the estimation. To decrease the error further the camera calibration could be more accurate and higher resolution images could be taken. The image used here for the experiments were PAL resolution.

4.3 Physiological Constraints of Human Motion

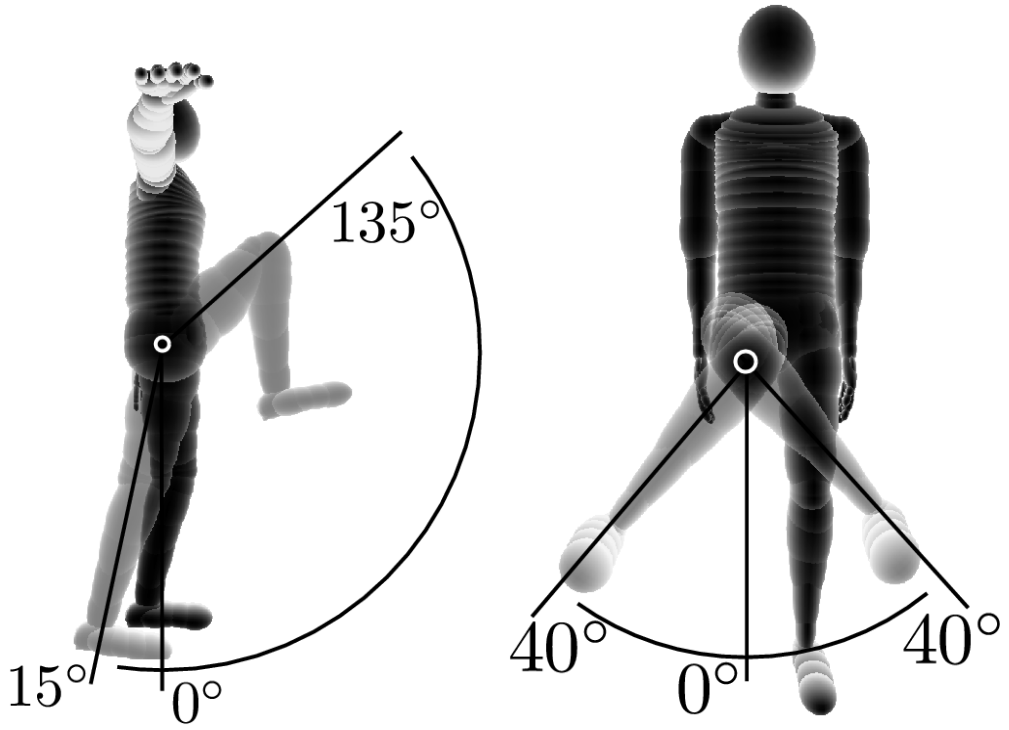


Figure 4.9: Motion limits of the hip joint. Flexion and Twisting. Rotation axis pointing outwards the paper plane.

The internal physiological constraints are the motion limitations of healthy humans. From medicine the range of possible motions is known, e.g. how far the outstretched arm can be lifted. Such knowledge can be used to constrain the estimation of joint angles.

This topic is very complex by itself, e.g. the PhD-thesis of L. Herda [61] is devoted to this problem. While introducing constraints in the estimation as mentioned in section 3.7 is easy to implement, the resulting limitations can only approximate the real motion limits, as explained below. Therefore in [61, 60] the limitations of more complex joints, e.g. the shoulder joint is modeled with quaternions and implicit surfaces.

The approximation by consecutively rotating axes is for most joints of the human body close to the real limitations. For single axis joints, like the elbow or knee, modeling the limitations with min-max-values is correct. Joints, whose motion is within small angles, like the vertebra in the spine, are also modeled correctly with min-max-values.

For more complex joints like the hip or shoulder there is a difference like illustrated in Figure 4.9 and 4.10 (angular values according to [115]). The motion shown in Figure 4.9 shows the flexion and twisting of the hip joint. The order of rotating axes in the MPEG4 specification is

1. flexion
2. abduction
3. twisting.

Therefore the motions shown in 4.10 are both abduction, on the left for zero flexion and on the right for 90 degrees flexion. As visible, the abduction interval depends on the flexion angle.

By modeling the motion limits with fixed intervals, there is a difference of 50 degrees of possible abduction angles depending on the flexion angle. A simple solution to this problem would be to change the interval depending on the flexion angle, for example with a linear function. However, it is not clear, if the real human abduct motion is linear dependent on the flexion; in fact it is difficult to obtain information about motion limits for the whole range of possible angles. In [61] the range of possible shoulder configurations is measured with a marker-based motion capture system. Unfortunately no analysis of the difference to min-max modeling with linear dependencies is done.

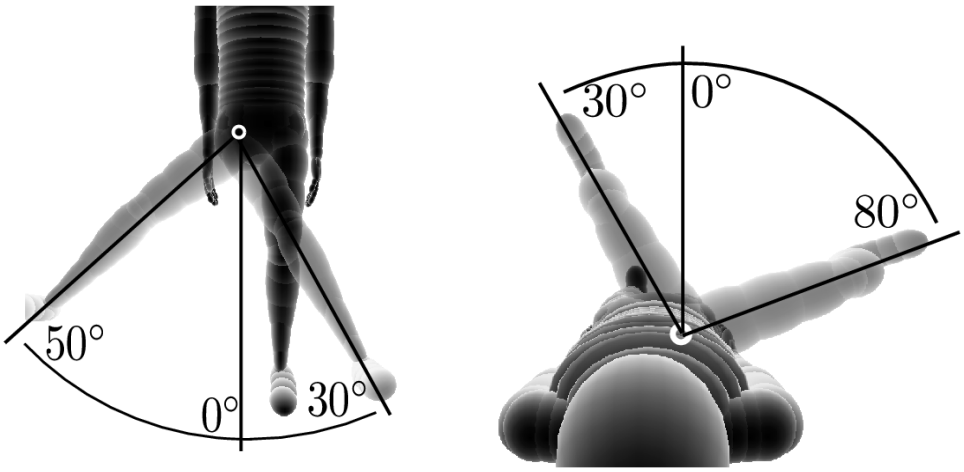


Figure 4.10: Motion limits of the hip joint. Abduction with different flexion angles. Rotation axis pointing outwards the paper plane.

Another problem arises, because motion capture systems usually capture only a subset of the real joint set of a human. For example, the shoulder complex of a real human consists of 5 joints. In common motion capture the shoulder complex is usually modeled with a single joint only.

If real motion is captured with only a single shoulder joint, the known limits from anthropometric studies will greatly limit the accuracy of captured motion, because the single joint has to represent motion of the whole shoulder complex. Therefore the application of motion limits are only applicable, if the motion model is close enough to the real human body, otherwise anthropometric data can not be applied and motion limits have to be

evaluated by other means. In this work motion limits are only applied to single axis joints like the elbow and knee, due to he above reasons, which were validated in the experiments as described in the following sections of this chapter.

4.4 Implementation Details

In this section implementation details are explained, that explain how the estimation is optimized while allowing for easy exchange of general MPEG4 models at the same time.

4.4.1 Rendering and Estimation Layer

The motion of a point $\boldsymbol{p}$ on the model is described by the motion function $\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})$. For fast calculation of the Jacobian it is necessary to take all partial derivatives at zero ($\boldsymbol{\theta}_t = \mathbf{0}$) (refer to section 3.2.2). To achieve this, the model point coordinates and all axes coordinates have to be in the same coordinate system.

The implementation has two layers of model representation to allow fast and efficient calculation of necessary derivatives. The two layers are shown in Figure 4.11. The first layer is the scene graph with a hierarchy of transformations and geometries as commonly used in computer graphics. The software library providing the scene graph is OpenSG [108], which is based on OpenGL. As visible in the Figure, each body part geometry has one additional transform, which is used to scale and position the geometry independent from the other parts. As used for model fitting as described in the beginning of this chapter.

The second layer is the axes and angle description of rotations, which is essential for pose estimation. The angle of each axis represents a MPEG4 body animation parameter. Both layers represent a model pose, while the first one is better suited for rendering, the second one is better for generating the required pose estimation data.

To build the Jacobian of the objective function, the point positions and the axes are required in the same coordinate system as the observed data. After estimation of relative joint angles for a specific frame the axes representation is updated by propagating the changes through each chain. The internal representation of rotation axes is by a quaternion and a translation, because computations with quaternions are fast and available in most software libraries like BIAS or OpenSG.

During the update of MPEG4 joint axes, a transform for each body part is stored, that gives the motion of the body part relative to the initial pose, where all joint angles are zero. To get the absolute coordinate of a point, e.g. on the upper arm, which is given in object coordinates relative to the next parent joint axis, only the stored transform needs to be applied. Therefore it is necessary to know for each model point (if given in relative body part coordinates) to which body part it belongs. This caching of segment transformations speeds up the computation, if the amount of correspondences is large, e.g. estimation from stereo data with thousands of points.

During optimization with Gauss-Newton method the rendering layer stays unchanged. Only the axes and angle values are updated after each iteration. When the minimizing

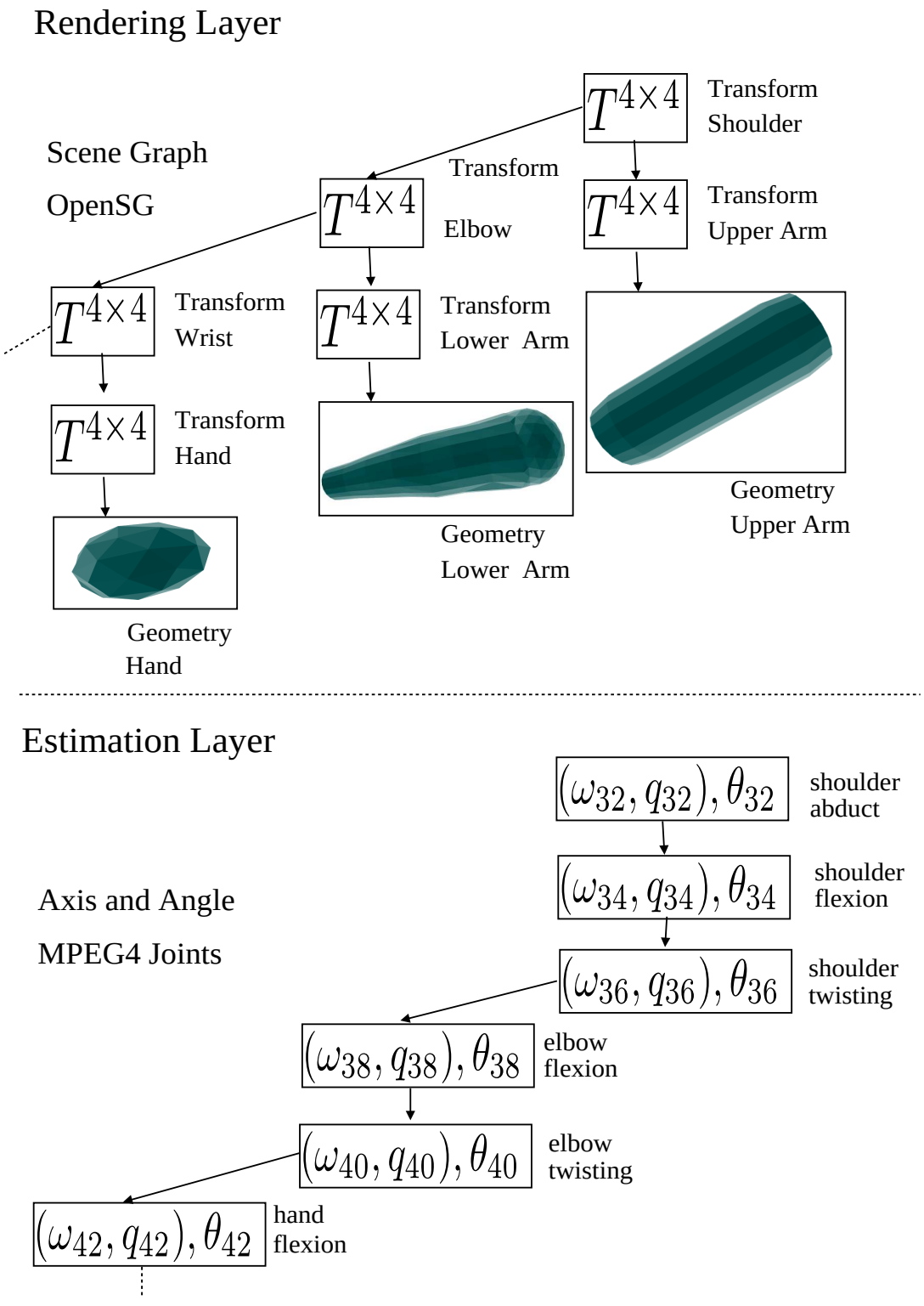


Figure 4.11: The two layers representing the body pose. Top: Rendering Layer Bottom: Estimation Layer

pose for the current correspondences is found, the scene graph is updated.

4.4.2 Initialization

To initialize the two layers a specific body pose is defined. In the MPEG4 specification a pose is suggested, where the person is standing straight, arms down next to the body, thumbs to the front (hand palm facing the body). The global orientation of the body is height in y-direction and face towards the z-direction as in Figure 4.12. In this pose all joint angles are zero, and other values are relative to this pose.

The body models used in this work are stored in VRML files. The transformations in the file define the initial rendering layer and should describe a model pose like the MPEG4 initial pose. The position (one point on the axes) of the initial MPEG4 joints is taken to be at the same position as for the rendering layer. The axes orientations however are specified manually at initial position, e.g. the direction vector $\boldsymbol{\omega}$ of the shoulder flexion axis is $(1, 0, 0)^T$ at initialization, where all joint angles are zero, the abduct direction vector is $(0, 0, 1)^T$, twisting $(0, 1, 0)^T$ and so on. These manually defined directions are the same for each body model and are stored in a separate file.

To be able to calculate a point on the axis from the rendering layer transformations, it is necessary to know which VRML/OpenSG transform corresponds to which MPEG4 joint. These mapping has to specified for each body model and allows it to use any kind of body model given in a format importable with OpenSG. The complete estimation and animation relies only on the MPEG4 joints and is therefore independent of the body model.

Additionally it allows to reanimate models with the same motion data easily, even if the model differ in size and shape from the model used for capturing.

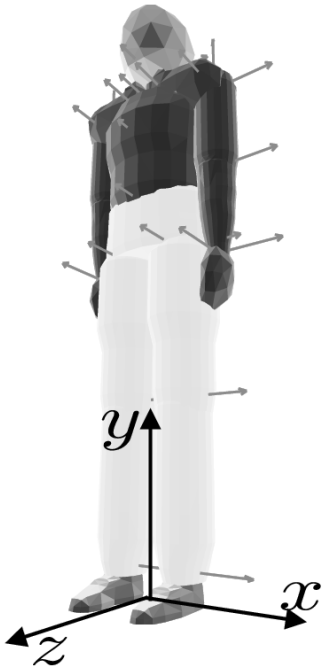


Figure 4.12: The initial pose, where all angles are zero.

4.5 Estimation from Silhouettes and Optical Flow

In this section estimation of motion for a specific person is described. The features extracted from images are silhouette information and point features tracked with the KLT [12, 125] corner tracker. The different features result in different types of optimization equations, as described in more detail in chapter 3.

In contrast to the model fitting from the previous section, here the model for the specific person is expected to be known, which reduces the motion capture to the estimation of joint angles. While the model fitting from above takes about 10 minutes, here frames are processed in less than a second. The background is not assumed to be static and known, but can be cluttered and non-static, because no explicit segmentation of fore-and background is necessary. Additionally, a smaller amount of views are sufficient in comparison to optical marker based systems, as the underlying motion and body model is directly incorporated in the image processing step. Even from a single view motion can be captured, which involves full body motion.

Because it is assumed that the person's pose is known at an initial state, it is possible to add information about the person's clothing and appearance to the model, for example by adding texture to the model. Then, it is possible to search for specific features of the texture, e.g. corners, in the next image frame. This is similar to point tracking with the KLT tracker. However to rely on this kind of feature alone, will introduce a drift (an error increase) over the sequence, as explained in later sections in more detail.

Another very common feature is silhouette information. Previous approaches to capture motion from video sequences with unknown non-static background relied so far on grey value gradients as the main feature as in [38]. However, grey value gradients are ambiguous, especially if the clothing exhibits strong gradients or the background is heavily cluttered. In the approach described here, the search for maximal grey value gradients is combined with color histogram changes, which include information of the person's clothing.

4.5.1 Overview of the Algorithm

The steps of the algorithm are briefly described here and are explained in detail in the following sections. The prerequisites for the algorithm are a 3D model, which is fitted in size to the observed person and the initial pose has to be known. The steps of the algorithm to estimate pose from 2D image data are as follows:

1. Prediction of body pose with motion model based on the previous pose, prediction of position of corner features
2. Rendering of the model in the predicted pose
3. Building of correspondences:
   - Choose corner feature for tracking with KLT
   - Build 2D-3D-point-correspondences from tracked corners by intersecting the corresponding viewing ray with the model.
   - On the contour of the rendered model find corresponding image points by searching for the highest grey value gradient and maximum change in color histogram along the normal.
   - Add 3D-point-2D-line correspondences for the silhouette
4. Pose estimation based on 3D-point-2D-point correspondences from feature tracker and 3D-point-2D-line correspondences from the silhouette information with Nonlinear Least Squares solved with Gauss-Newton as described in chapter 3.
5. Update of color histograms for each body part

To get the initial pose, the user has to position the model manually. Because the depth is difficult to measure from a single view, markers on the floor give the user helpful information, where to position the model. Small errors in the manual positioning are not crucial, because the silhouette correspondences are correcting small errors.

4.5.2 Tracking of Image Features

For the estimation described in section 3.4.3 it is necessary to have correspondences between 2D image points and 3D model points. These can be calculated by tracking 2D features

from one image to the next. Because it is assumed, that the initial pose of the person is known as in the second image of Figure 4.16, it is possible to get the relation between the image of the real person and the 3D model point by intersection of the feature's viewing ray and the 3D model surface using the known projection matrix. Then the same feature point has to be found within the next image and gives the necessary 2D image 3D model point correspondence. This feature point is found by tracking corners with the KLT tracker.

KLT Tracking

The abbreviation KLT is short for the authors *Kanade, Lucas and Tomasi*. Related literature is [125, 116, 12]. Given here is a short overview of important aspects.

The basis for tracking is the *Image Brightness Constancy*, meaning that 3D scene points project with the same brightness to image points, regardless of the point of view. Or in other words, if the camera or the object of interest moves, its projected 3D points have the same image values. This assumption does of course not reflect reality, but it approximates reality sufficiently, if the motion of the camera or object is small and lighting changes only insignificantly. The additional assumption is, that all 3D scene points visible in the first image are also visible in the next images, neglecting occlusion and image borders.

The derivation of the Kanade-Lucas-Tomasi-Tracking equation is a further application of the Gauss-Newton method (see chapter 2) as described in section 3.3.6.

The KLT tracker is chosen here to estimate the pure translation for each region, though it is also possible to estimate 2D-afffine transformations of the image region or articulated transformations. The reason for simple displacement chosen here is, that estimation time was more important then accurate estimation and because BIAS [41] provided an optimized implementation of the KLT tracker.

Corner Detection

Features tracked with the KLT method are regions in the image around specific points. According to [125] regions are reliably to track, if the coefficient matrix $J^T J$ for the region is well-conditioned, as necessary for inversion and if the structure inside the region is above the image noise. Both can be ensured by analyzing the eigenvalues within the region. If both eigenvalues are above a certain threshold, the image region can be accepted to have a 'rich' enough structure to be reliably tracked. The smaller eigenvalue is called the *cornerness* in the following.

The geometric interpretation of the eigenvalues is as follows. Two small values indicate a homogenous image region, one small and one large value indicate a straight line, whose displacement is only measurable in the direction orthogonal to it. Two large values indicate something similar to a corner or cross like shape.

For efficient tracking of the human body, there are additional requirements to be met:

- Features should not cluster in specific image regions, which is ensured by applying a minimum distance between corners.

- Homogenous distribution over the body. On each body part there should be a minimum number of features, with a minimum cornerness. High cornerness is not the main attribute for choosing a feature. A smaller amount of features increases computation time, but an insufficient number will not give enough information for valid pose estimation.

- A higher density of corners is preferable on the arms and legs, less on the upper body, because correspondences on the arms and legs determine the pose of the upper body as well, but not vice versa.

- Features have to be assigned to body parts, not to the background. Therefore features are not chosen, if their assignment is ambiguous, e.g. if they are very close to the border of the body part's image region.

- The complete set of features should not be reselected in each frame, because features, which could be found in the next image, should be tracked further. Therefore new features are only selected for those features, which could not be tracked into the next image.

- The algorithm to choose the position and amount of features should be efficient and fast itself.

The above requirements are implemented by an algorithm, which needs the following variables for each body part:

- a minimum cornerness (usually the same for all parts)

- a minimum and maximum number of features

- a minimum and initial maximum distance between features

The algorithm for one body part is then as follows:

1. Calculate a cornerness image for the current frame, where each pixel value is the smaller eigenvalue of the feature's coefficient matrix.

2. Render the model in the current pose, with unique colors for each body part (see also section 4.6.2).

3. Dilate the rendered image, to eliminate border pixels.

4. Build a set of image positions for the body part region.

5. If features were successfully tracked from the previous frame, add these to the new feature set.

6. Eliminate features, whose position is not in the region set.

7. From the region set, add the the best cornerness points to the new feature set with respect to the initial max distance. If the minimum amount of points is not found, reduce the distance and repeat until enough are found or the minimum distance is reached.

The algorithm takes less than 10ms on a 3Ghz Pentium IV with exception of cornerness image calculation.

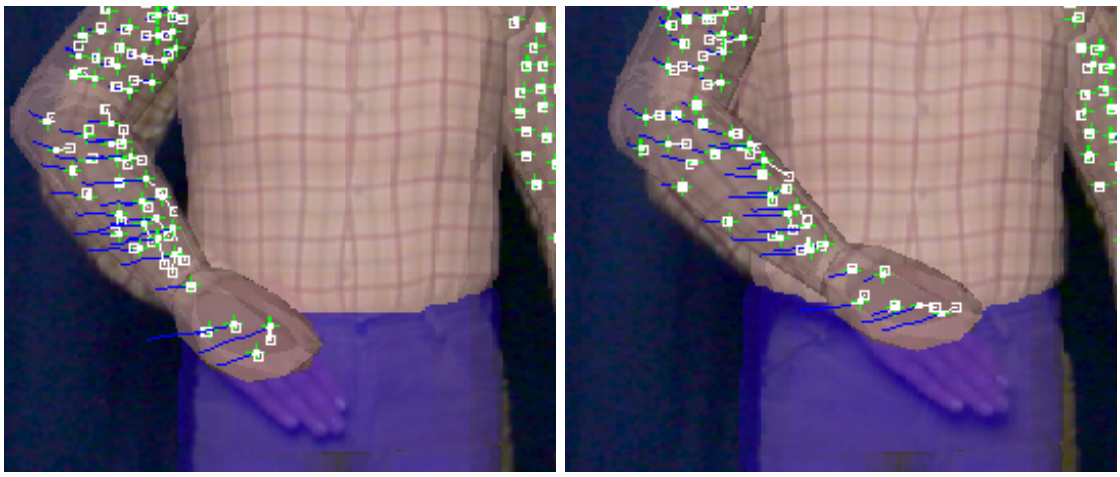


Figure 4.13: Tracking of point features (Cross marks). Boxes indicate a tracked corner. The motion of a corner over the last frames is shown as a black line.

Tracking of Corners

Tracking point features allows us to capture motion, which wouldn't be possible from the silhouette alone, e.g. an arm moving in front of the body like in Figure 4.13. By matching feature points between images and assuming the correspondence of model and image point is given by projection of the model point, the estimation will drift away from the correct estimate, because small errors in the estimation will accumulate over time, also known as drift. This happens because a pose estimate with a small error can lead to correspondences between model points and image background, such that feature points on the background are tracked. If these wrong correspondences are used for pose estimation, the error in the estimate will increase even further.

To reduce the probability of image to background correspondences, we use an erosion operator on the rendered image of the model and predict the motion from frame to frame using a constant velocity motion model.

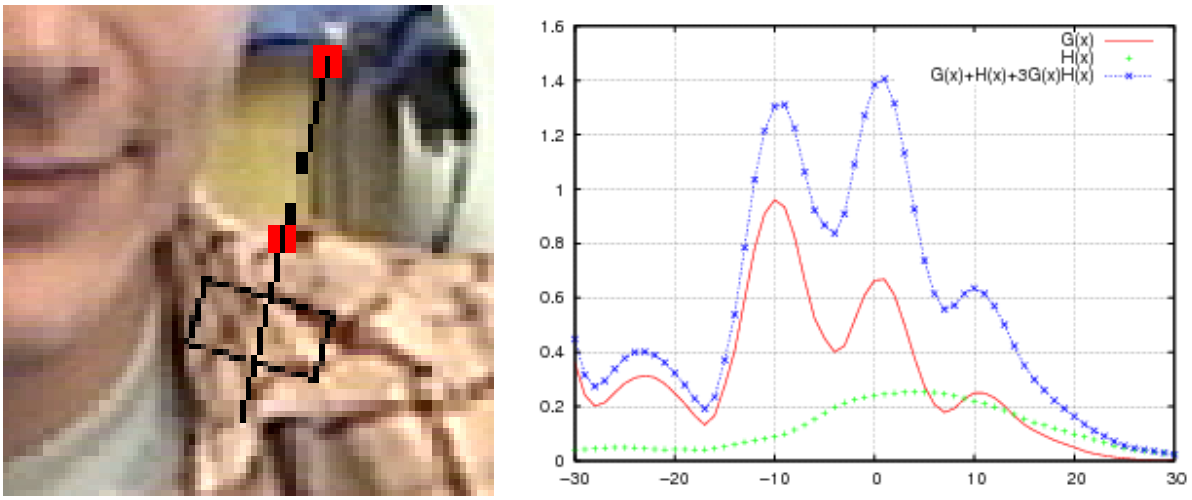


Figure 4.14: Left: Correspondence search along the normal. Right: The gradient (G(x)) and histogram (H(x)) values along the normal. Correct correspondence at zero.

4.5.3 Correspondences by Silhouette

Grey Value Gradient

To compensate the drift we add silhouette information to our estimation. This is achieved by calculating additional 2D-3D correspondences for the model silhouette and the silhouette of the real person. In contrast to [111] or visual hull approaches [30, 97, 119, 73] we don't utilize explicit segmentation of the images in fore- and background, but use the predicted model silhouette to search for corresponding points on the real silhouette.

Previous work like [72] already took this approach by searching for a maximum grey value gradient in the image in the vicinity of the model silhouette. However we experienced that the grey value gradient alone gives often erroneous correspondences, especially if the background is heavily cluttered and the person wears textured clothes. Therefore we also take color information into account.

HSL Color Histogram

As the initial pose is known, it is possible to calculate a color histogram for each body segment. We use the HSL color space to get more brightness invariance as in [129]. This reference histogram is then compared with a histogram calculated over a small window on the searched normal.

In Figure 4.14 the normal is shown and the rectangular window, which is used for histogram and gradient calculation. The assumption is, that the histogram difference changes most rapidly at that position on the normal, where the correct correspondence is. This is at the border between person and background.

We combine the grey value gradient within the region and the color histogram change

to a single measurement value. The type of combination function is chosen by analyzing the development of gradient and histogram values over 15 normals in different images.

The actual values of the combination were then evaluated experimentally trying different values and counting the number of correct correspondences manually for about 100 silhouette points in 4 different images. The final combination values reduced the number of wrong correspondences for difficult images from approx. 30-40% to approx. 3-15%. Please note that the combination values depend on other parameter settings like window size, histogram difference function etc.

A rather difficult case is shown in Figure 4.14(b), which shows a plot of the maximum search along the normal of Figure 4.14 left. The grey value gradient $G(x)$ is shown as a solid line, the gradient of the histogram differences $H(x)$ as points and the combination with lines and points. As visible, the grey value gradient alone would give a wrong correspondence, while the combination yields the correct maximum at zero. The correspondences found in this way could be integrated into the estimation the same way as the correspondences from feature tracking. However, for most silhouette parts a 2D-3D point correspondence isn't correct, because of the aperture problem. For parallel lines it isn't possible to measure the displacement in the direction of the lines. Therefore we use a formulation that only minimizes the distance between the tangent at the model silhouette and the target silhouette point, resulting in a 3D-point 2D-line correspondence as visible in Figure 4.15. The objective function for minimization is (as explained in section 3.3.5)

$$\min_{\boldsymbol{\theta}} \sum_i \left| \tilde{\boldsymbol{n}}_{\boldsymbol{\omega},\boldsymbol{i}} \left(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega},\boldsymbol{i}} \right) \right|^2 \qquad (4.4)$$

with correspondences $((\tilde{\boldsymbol{n}}_{\boldsymbol{\omega},\boldsymbol{i}}, \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega},\boldsymbol{i}}) \ , \ \boldsymbol{p}_i)$, where $\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i})$ is the function describing the motion of the 3D point $\boldsymbol{p_i}$, the projection of the moving point is $\boldsymbol{p}'_i$, the vector $\tilde{\boldsymbol{n}}_{\boldsymbol{\omega},\boldsymbol{i}}$ is the normal on the tangent line and $\tilde{\boldsymbol{q}}'_{\boldsymbol{\omega},\boldsymbol{i}}$ is the corresponding image point on the image silhouette. In this formulation a motion of the point parallel to the model silhouette line will not change the error. We calculate the normal vector as the projected 3D-face normal of the triangle, which belongs to the point $\boldsymbol{p}_i$.

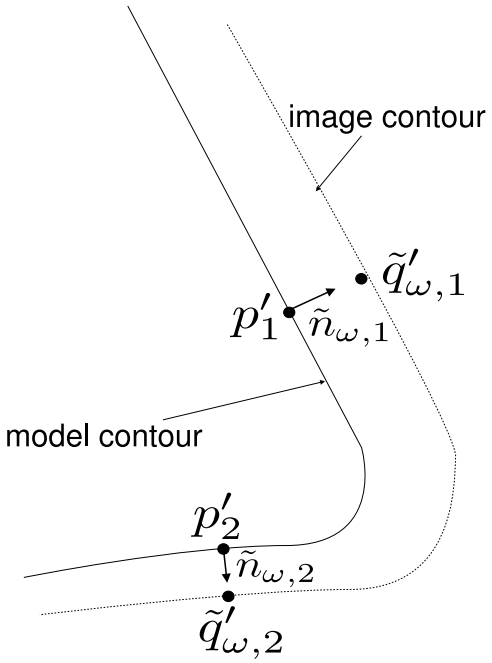


Figure 4.15: Two 3D-point-2D-line correspondences between model and image silhouette.

4.5.4 Experimental Results

Correspondences from point tracking and from silhouette difference are combined within the optimization by joining both correspondence sets together. Because we estimate pose

with different correspondences, weights are added in the Least Squares steps. That way it is possible to ensure a similar influence of 3D-2D point correspondences and 3D-point 2D-line correspondences.

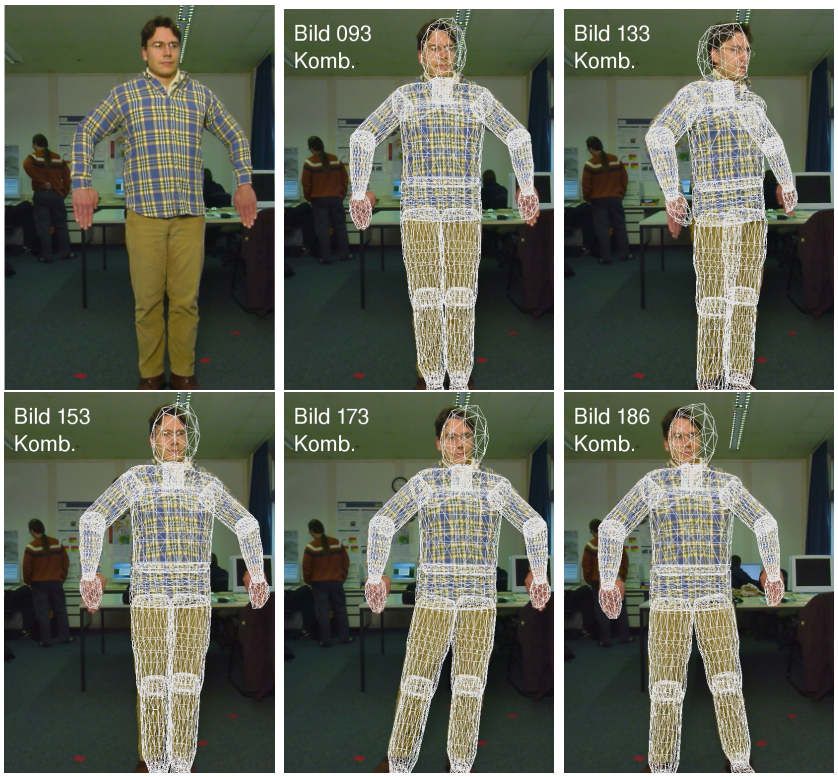


Figure 4.16: Original image and estimated model pose with 19 DOF. The pose is estimated from silhouette correspondences and tracked corners.

In the following sequences 19 DOF were estimated. Five for the global position and rotation (leaning forwards or backwards is not estimated), one for abduction of the whole shoulder complex, three for each shoulder, one for the elbow and two for each leg (twisting and abduction). Additionally the estimation was damped by a regularization term, such that a large change of joint angles in one iteration is unlikely, if only a few correspondences affect the joint. This way no correspondences for a segment lead to no change for that joint.

Figure 4.16 shows results for a simple motion, that consists of a rotation of the upper body and stepping aside afterwards. The person is wearing a checkered shirt that exhibits lots of disturbing grey value gradients. The estimated body pose is shown in white as superimposed on the real camera image. As visible, the motion could be captured successfully from a single camera view in spite of the unknown cluttered background and the inhomogeneous clothing. Results for a more complex motion for a different person are shown in Figure 4.17. The person is wearing a T-shirt and the background is non-static

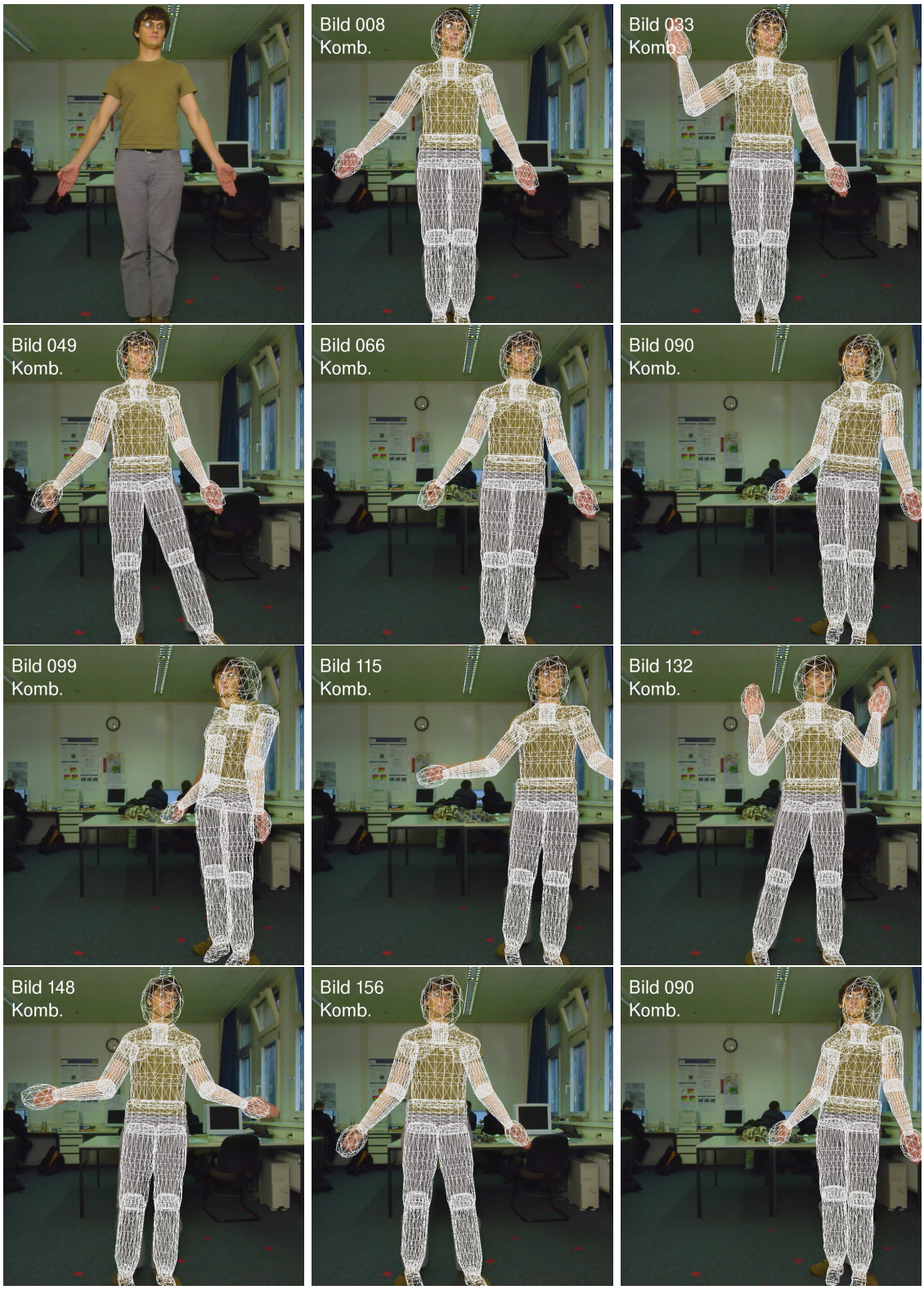


Figure 4.17: Original image and estimated sequence with 19 DOF.

and cluttered again. Motion between frames is quite large, because capturing was done with 7 fps, while the person was moving at normal speed. Even though the shoulder and the upper arm are completely hidden during some frames, the motion could be captured correctly. Processing of one frame took about one second for PAL image resolution on a Pentium IV 2 Ghz.

4.6 Estimation from Depth Data

In this section methods are described to track the motion of a person, who is observed by two cameras. The cameras have to be set up, such that stereo algorithms can be applied. In the experiments carried out here the cameras were positioned with nearly parallel optical axes and with a distance of about 20cm. Then, a stereo algorithm provides a depth image, which can be used together with internal camera parameters to estimate a 3D point cloud. These observed points are taken as input data to estimate the pose of a 3D model from the person. A Nonlinear Least Squares method minimizes the difference between 3D-model surface and observed point cloud with the Gauss-Newton optimization as explained in the previous chapter. The approach presented here can be summarized as ICP (Iterative Closest Point) [11] motion estimation for articulated objects.

To apply the estimation algorithm described in the previous chapter, necessary 3D-point-3D-point correspondences have to be established, as described after some comments about the used stereo algorithm. Results at the end of this section show, that the described optimizations for efficient correspondence calculation enables us to estimate body pose with 12 DOF from 1000 data points in 250ms on a Pentium4 3Ghz. Further results are given for more complex motions with 24 DOF using all available stereo data.

4.6.1 Stereo

The motion estimation here is based on dense depth information which could be estimated directly from correspondences between images. Traditionally, pair-wise rectified stereo images were analyzed exploiting geometrical constraints along the epipolar lines. More recently, generalized approaches were introduced that can handle multiple images and higher order constraints. See [114] for an overview. Achieving real-time performance on standard hardware has become reality with the availability of free programmable graphics hardware (GPU) and the additional benefit of keeping the CPU free for other tasks like our pose estimation [130]. The results presented in this section are calculated from depth images generated by a dynamic programming based disparity estimator with a pyramidal scheme for dense correspondence matching along the epipolar lines [44].

4.6.2 Building of 3D-3D Correspondences

To apply the pose estimation algorithm described in the previous chapter, correspondences between observed 3D points and 3D model points are necessary. As the observed points are calculated from depth maps, these correspondences are not known. Similar to [112, 16, 17, 34] an Iterative Closest Point (ICP) approach is taken. For each observed point, the nearest point on the model is assumed to be the corresponding one. With these correspondences the body pose of the model is calculated.

The following steps are repeated multiple times for each frame of a sequence. The larger the relative motion is, the more correspondences will be incorrect. The principle idea of ICP is, that an estimated relative motion decreases the difference between model

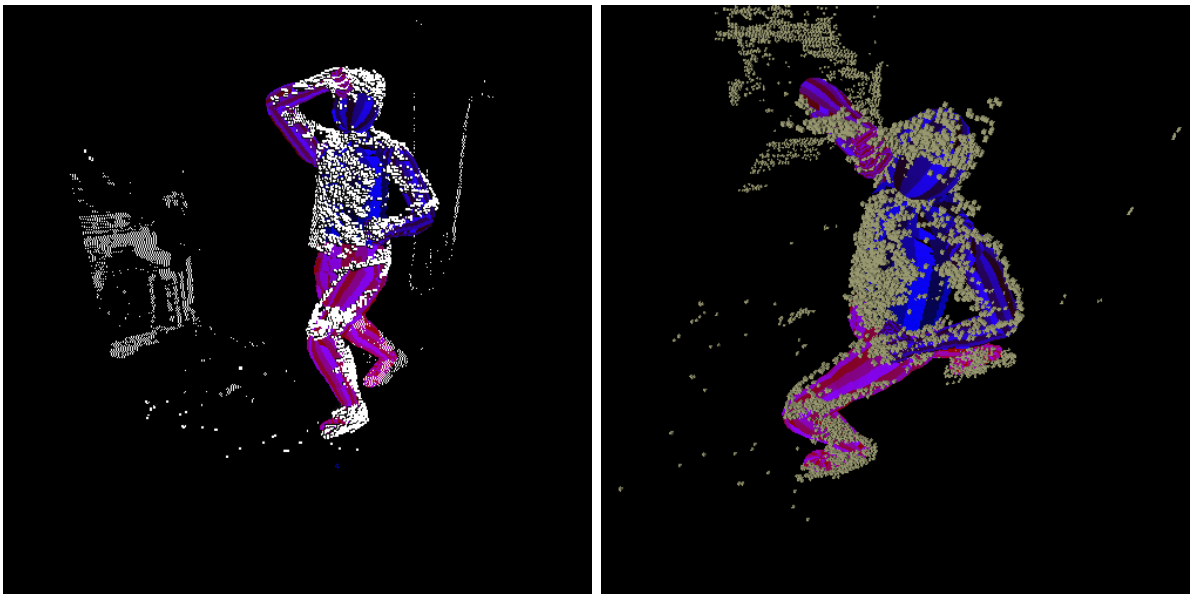


Figure 4.18: Two views on the input data. Approx 10000 depth points calculated with a stereo algorithm are shown as white boxes.

and observation, such that more correspondences are correct, if built again for the smaller difference in the next iteration.

The steps in one ICP iteration are (further details in the next paragraphs):

a Render model in current pose with a unique color for each triangle (OpenSG).

b Find visible model points by analyzing the color of rendered image pixels.

c Build correspondences from the depth map by finding nearest model point.

d Estimate relative motion from 3D-point-3D-point correspondences by Nonlinear Least Squares as explained in the previous chapter.

Between frames it is common to predict the pose for the next image by applying some kind of motion model. In this work the displacement between the last two frames is taken for prediction. Assumed is a constant velocity for each joint angle between frames.

In the following the different steps are explained in detail.

a) Rendering the Model

It is assumed, that the pose was calculated correctly in the last frame and that the displacement is small between frames. Therefore, the rendering of the model with the last calculated pose gives approximately the visible body parts. To get a correctly synthesized view, the internal parameters of the calibrated camera have to be applied to the virtual camera. The rendering library OpenSG did not support cameras, whose principal point is

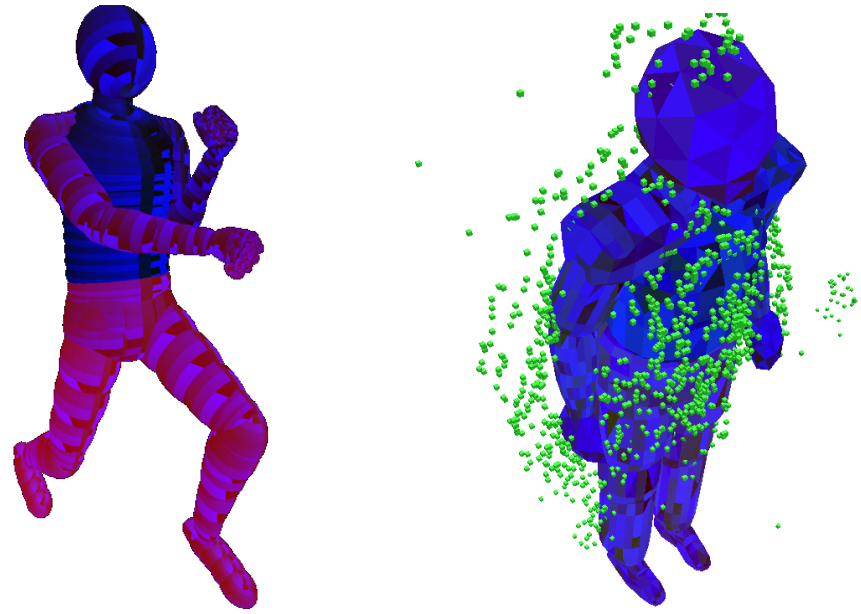


Figure 4.19: Left: Rendering of the body model with unique colors for each triangle Right:The 3D point cloud computed from the depth image. White boxes represent the observed points. The model is shown for reference and to illustrate the difference, which is minimized.

not in the image center, OpenSG was therefore extended by an OffCenterCamera, whose principal point is given in normalized device coordinates (-1 to 1).

b) Find visible Model Points

The rendered image can then be analyzed to get the visible 3D points of the model. A possible solution is to calculate for each image pixel, which belongs to the model, the viewing ray and intersect it with the 3D model. However, this is very time consuming.

Therefore it is assumed, that the corners of the model's triangles represent the surface sufficiently well. Then, a lookup table of model points can be established, that allows fast computation of the visible subset of model points.

Each triangle of the model is rendered in a unique color, such that the red value indexes the body part (corresponds to a BAP id) and the green and blue values index the vertex in the body part's geometry. The resulting image looks like that shown in the left of Figure 4.19. This approach is similar to that of [19]. For each pixel it is now sufficient to make an array access to get the three vertices, that build the triangle.

The set of visible model points are ordered into an associative array, which uses a binary search tree, where the key value is the height value of the point. The height is appropriate, because the extension of the visible point cloud is usually largest in height. A principal component analysis (PCA) could be conducted to find the coordinate with largest variance. However, for the experiments in this work the height coordinate has the largest variance,

because the person was standing in all experiments.

c) Build Correspondences by Nearest Neighbor

The observations are depth images calculated from disparity maps. Together with the internal camera parameters a 3D point cloud of observed points can be computed. For these observed points it is now necessary to find the closest visible point model point. An example for the observed point set is shown in the right of Figure 4.19 together with the body model. The green boxes represent the observed points.

The computation time for one frame depends largely on the amount of observed points, therefore the depth image is randomly subsampled to decrease the number of

correspondences and reduce computation time. To calculate the 3D observed points from the known focal length and principal point of the camera efficiently, the camera coordinate system is assumed to be aligned with the world coordinate system and the body model is positioned initially, such that it is close to the observed point set.

For each observed point the closest visible model point is found by searching in the associative array for the nearest point with regard to height. Starting from the point with the smallest height difference $\boldsymbol{p}_1$, the search is alternated in both directions. and is stopped, if the distance in height to a point is larger than the Euclidean distance d_E to the nearest point $\boldsymbol{p}_N$ found so far.

Figure 4.20 illustrates this. The last tested points are $\boldsymbol{p}_{top}$ and $\boldsymbol{p}_{bot}$.

d) Estimation of Relative Motion

To fit the body model to the observed point set, a segmentation of the person from the background is necessary. In [16] this is done by using skin color. We assume here only that there are no scene objects (or only negligible parts) within a certain distance to the person. The segmentation is then done implicitly by using a robust estimator in the pose estimation step based on reweighted least-squares (see section 2.4). We use the Tukey weight function, which results in a zero weight for correspondences, whose error value is larger than a predefined minimum distance, and are therefore not included in the estimation. The corresponding observed point belongs probably to the background scene. The estimation of body pose for one set of correspondences involves multiple iterations within the dampened Gauss-Newton optimization. The estimation is performed completely within the estimation layer. The scene graph is not updated and rendering is not done to save computation time. There is an upper bound on the number of iterations to have an upper bound on the estimation time. Iterations are also stopped if the parameter change drops below a certain threshold. If the new pose for the current set of correspondences is estimated, a new set is build by repeating the steps beginning with step **a)**.

All steps can be repeated until convergence. Because the change between image frames is small and to have a bound on computation time, we set a fixed number for the repetition of all steps, usually within 2 to 5.

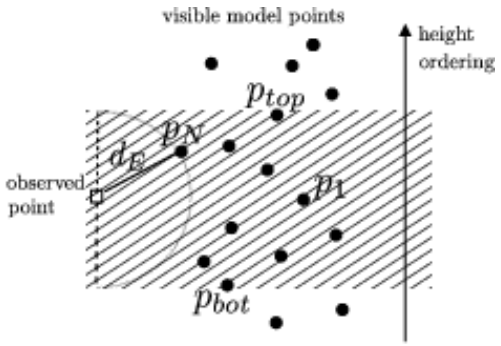


Figure 4.20: Search for the nearest model point is bound by height distance.

Figure 4.21: First frame (left). Frame 60 and frame 176 of the synthetic sequence.

4.6.3 Experimental Results for Synthetic Data

In order to analyze the accuracy and robustness of the algorithm an experiment is carried out with artificial synthetic depth data. The motion is then tracked using data, which is disturbed with different noise levels.

Commercially available stereo cameras, like the *Stereo-On-A-Chip* camera from *Videre Design*[126] or the *BumbleBee2* camera from *PointGrey Res.* [107] can provide depth data in real-time. It can be expected that these data is less accurate than depth points calculated from more sophisticated stereo algorithms like [44]. The experiments conducted here with different noise levels analyze the behavior of the developed motion tracking method with respect to less accurate real-time stereo-algorithms.

The motion involves four DOF, the flexion of the elbow and the flexion, abduct and twisting of the shoulder. Three images out of the 176 frame sequence are shown in Figure 4.21. The depth data is calculated using openGL by converting the the z-buffer values to real distance measurements after rendering the model. Figure 4.22 shows the development of the elbow flexion and the shoulder abduct angle over the sequence.

At first the motion is estimated without changing the depth data. Because the depth data is generated from the model, which is used for estimation, it can be expected that the resulting error of the estimation is very small. Figure 4.23 shows the difference of the elbow

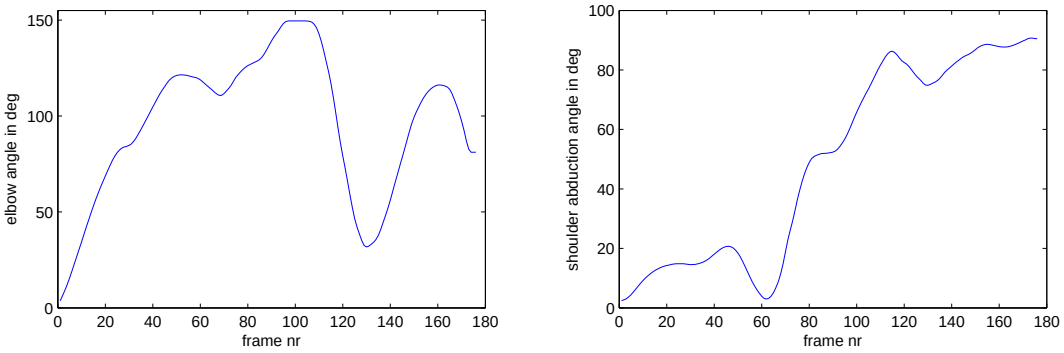


Figure 4.22: The development of the elbow angle (left) and the shoulder abduct angle (right) over the sequence. Considered as ground truth in the following.

flexion and joint abduct angles to the ground truth during the sequence. For each frame new correspondences were established 10 times (ICP iterations). In each ICP iteration the Nonlinear Least Squares problem was solved by iteration of the dampened Gauss-Newton method at most 20 times or until convergence.

The error is not zero, because the correspondences between model and data are established by finding for each data point the nearest vertex of all triangles. More accurate would be to search for the nearest point on the model surface. However, approximation of the model surface by the corner points of all triangles allows much faster computation.

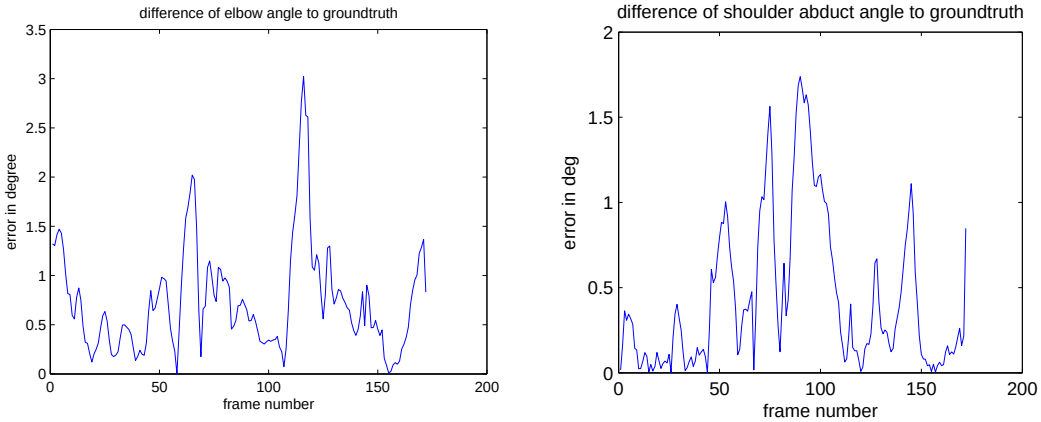


Figure 4.23: The difference of the estimated elbow angle (left) and the shoulder abduct angle (right) to the ground truth.

In order to analyze the influence of the number of ICP iterations with respect to differently dense input sequences, the estimation was repeated for differently dense input sequences with a different amount of ICP iterations. Figure 4.25 left shows the average error over the sequence (average of shoulder abduct and elbow flexion) for 1,3 and 5 ICP

iterations and for differently dense sequences (each frame, each second, each third and each 4th frame). As can be expected, the average error decreases with the number of ICP iterations and increases, if the average angular change from frame to frame increases. Interesting to note here is, that an increase in ICP iterations from 3 to 5 decreases the error not much further. If real-time estimation is desired, the results suggest, that 2 or 3 ICP iterations are most suitable.

The average angular change (elbow flexion and shoulder abduct) from frame to frame for different dense sequences is

- for each frame: 3.17 degree
- for each second frame: 6.35 degree
- for each third frame: 9.5 degree
- for each fourth frame: 12.762 degree

The results show, that the motion is correctly estimated, even if the input sequence exhibits large changes from frame to frame (12.72 degree on the average).

Then, Gaussian noise with different standard deviations is added to the original depth data. Figure 4.24 shows two views on the depth data and the model in starting pose. The z-buffer image was sub-sampled in order to generate approximately 10000 depth points. Approximately 750 data points have the closest model point on the right arm. Therefore, the estimation relies on approx. 750 correspondences each frame.

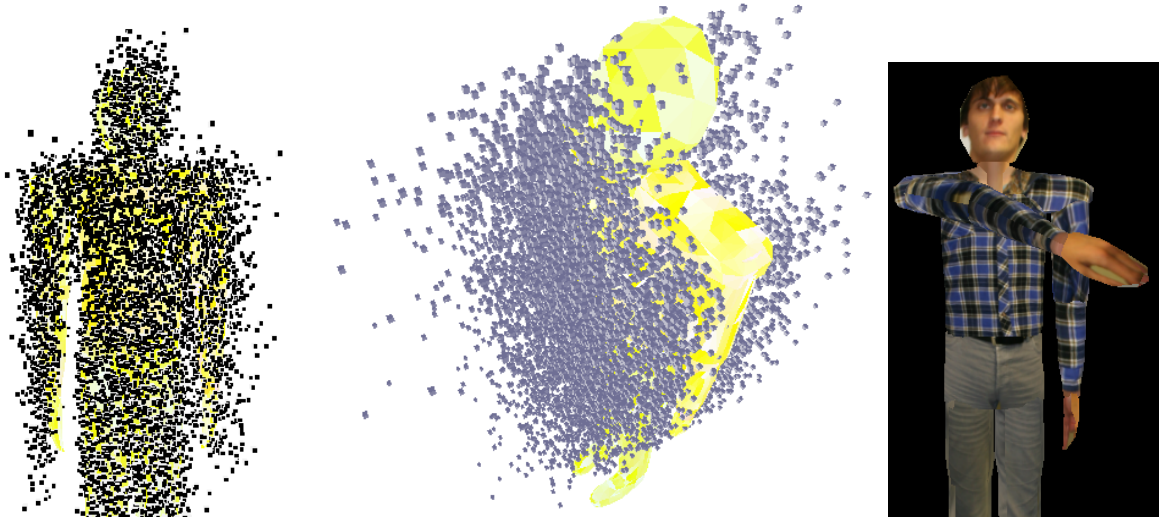


Figure 4.24: Two views on the depth point cloud with Gaussian noise (deviation 15cm). Right: At this noise level, the tracking method loses track after a few frames and gets stuck in the pose shown.

In a second experiment, the original data is disturbed by Gaussian noise in depth with a standard deviation of 5cm. The estimation was again executed for different ICP iterations and different average angular change between frames. For each parameter setting the estimation was repeated 30 times in order to calculate the deviation of the estimation results as well. The mean error for the elbow flexion and the shoulder abduct over the sequence and over 30 trials is shown in Figure 4.25 right. The results are similar to the ones before without noise. The error decrease from 3 to 5 ICP iterations is even less, suggesting, that 2 or 3 is a good trade-off between accuracy and speed. Especially in real-time online estimation, faster per frame computation allows to process more images in the same time, such that the angular change between frames gets smaller.

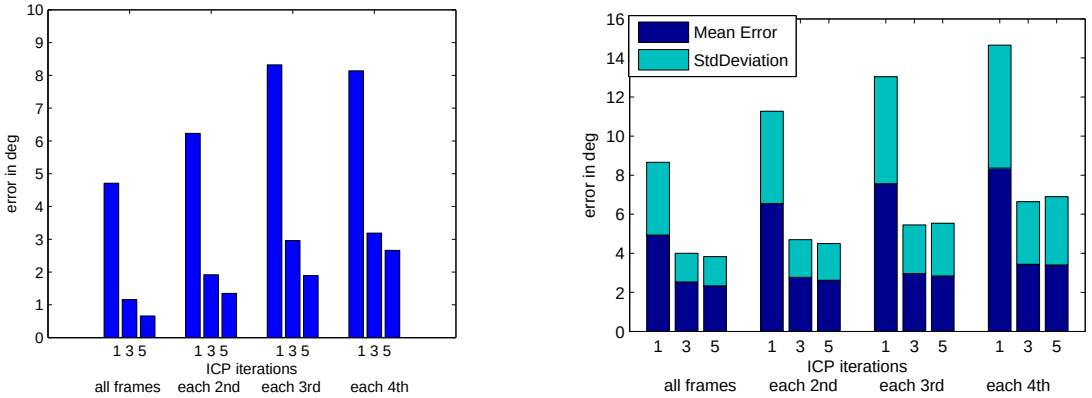


Figure 4.25: The average error over the sequence for the elbow and shoulder abduct angle (left). The original depth data is perturbed with noise (5cm deviation) (right). Shown is the error and the deviation over 30 trials for different values of ICP iterations. The error increases, when the average angular change increases from frame to frame (frames are skipped for estimation).

A further estimation was carried out with an increase in noise levels to 10cm deviation in depth and 1cm perpendicular to it. Results for 30 trials for each parameter setting are shown in Figure 4.26. The average error and the error deviation increases further. Interesting to note here is, that higher number of ICP iterations lead to an increase in the estimation error for the dense sequence with each frame and each 2nd frame. This is due to over-fitting of the data. The estimation is beginning to fit the noise rather than the real underlying motion. The effect of over-fitting is well known in machine learning [14]. For the less dense sequences, the number of ICP iterations is not high enough to show the effect of over-fitting.

When increasing the noise levels to a deviation of 15cm in depth and 2cm in the other directions, the method loses track after a few frames and gets stuck in a pose as shown in the right of Figure 4.24. From this pose the tracking is not able to recover, because too many correspondences are false due to the nearest neighbor approach.

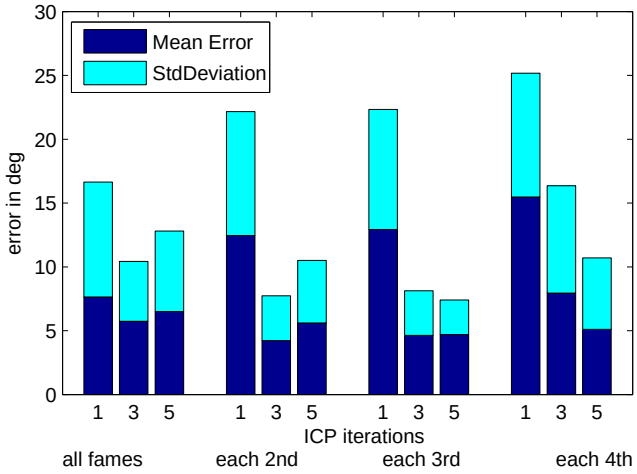


Figure 4.26: The average error over the sequence for the elbow and shoulder abduct angle (left). The original depth data is perturbed with noise (standard deviation of 10cm in depth and 1cm in other directions) (right). Shown is the error and the deviation over 30 trials for different values of ICP iterations. The error increases, when the average angular change from frame to frame increases (frames are skipped for estimation). The error increase from 3 to 5 ICP iterations is due to over-fitting.

4.6.4 Experimental Results for Real Data

Experiments for a real sequence of depth images are carried out to show the efficiency of the implementation. In the recording a person moves the arms at first in a waving manner and later crosses the arms in front of the chest. The arms close to the body is a very difficult pose to estimate correctly. Shape-from-Silhouette based approaches will have difficulties here. In Figure (4.27) three images from the sequence are shown. The top row shows the depth images with lighter values indicating closer points. The middle row shows the original images overlayed with the estimated model pose in grey. The bottom row shows the same model pose as seen from another position.

	Demirdjan	Bray	Our
DOF	ca. 18	30	28
correspondences	unknown	45	1000
model complexity	6 simple cylinders	complex hand model linear blend skinned	18 seperately fitted body parts
speed	6-10fps Pentium4 2Ghz	0.22 fps fps Sunfire 1.2Ghz	4 fps Pentium4 3 Ghz

Table 4.2: Comparison with related work

The model's position is estimated below the real person throughout the sequence, because the model was fitted offline beforehand to the person showing a bare upper body. Therefore the shape of the model does not fit the recorded person very accurate. However, the motion could be estimated correctly , which shows the robustness of the estimation with respect to inaccurate model geometry.

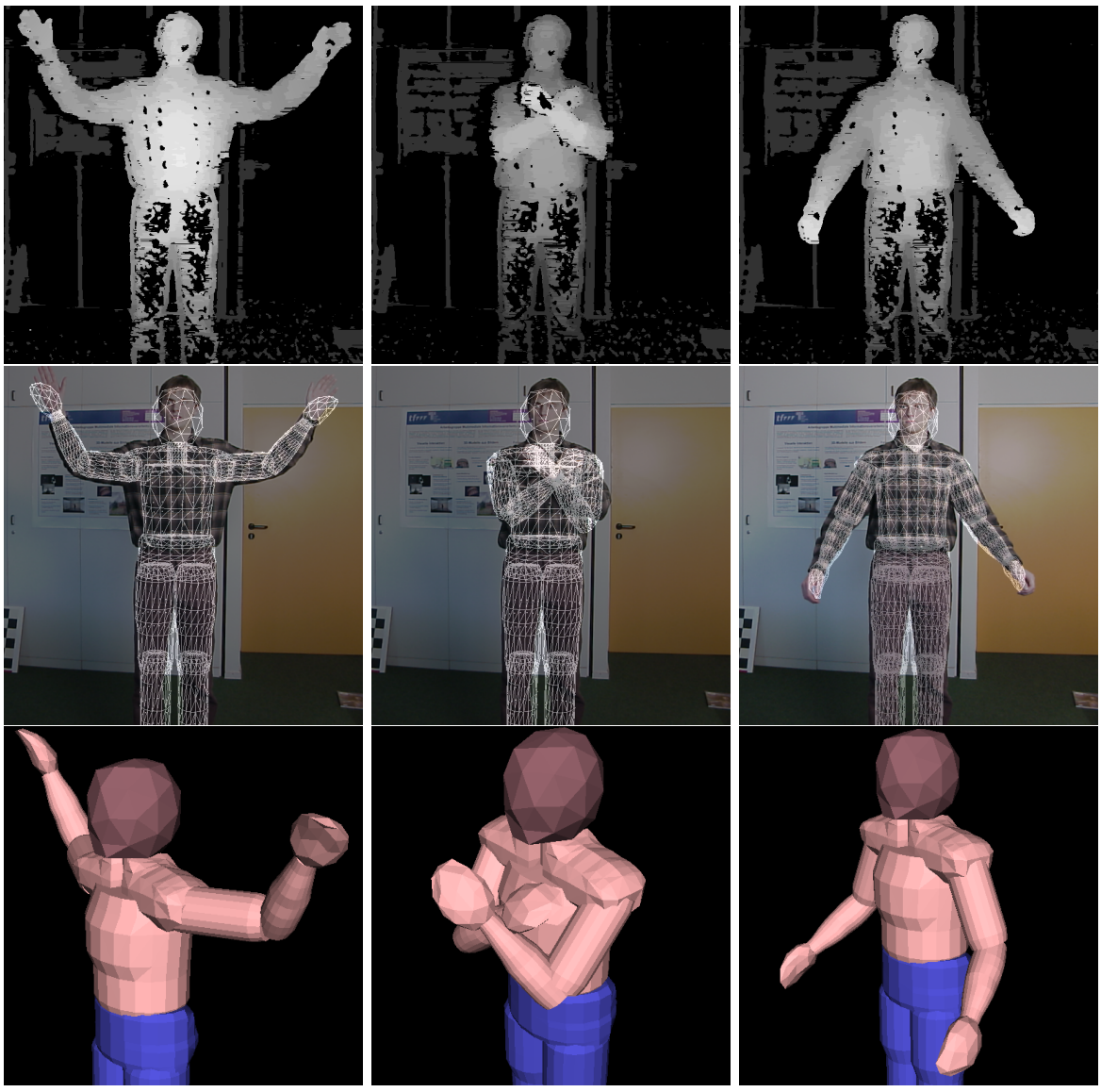


Figure 4.27: Top Row:Depth images, middle row: original image overlayed with estimated model pose, bottom row: model view from the side

For this sequence 14 DOF were estimated: The global transform, the three shoulder angles and the elbow flexion. Important for the accuracy and speed are the number of iterations and the number of data points (the subsampling rate). The fastest result were achieved by taking approx. 800 data points and 2 ICP iterations per frame. The processing

of one frame took 200ms on a 3Ghz Pentium 4 on the average. For a bowing sequence with synthetic data, that involved motion of the arms, legs and head, 26 DOF were estimated. The estimation time increased only slightly to 250ms per frame with 1000 data points.

Table (4.2) shows the effectiveness of the approach. We are able to use far more correspondences for the estimation than Bray[16, 73]. This is probably due to the analytically derived Jacobian and the optimized correspondence search. Processing more data points stabilizes the estimation significantly, if noisy depth data has to be used. In [16] the 3D data points are calculated with a structured light sensor, which probably results in more accurate 3D estimates. The approach of Demirdjan is faster than our, however the estimation method is inferior [34] to our direct approach and uses a simpler model and probably less correspondences.

Sequence with Full Body Motion

Further experiments are carried out with 24 DOF motion. The estimation time per frame depends mainly on the number of used data points. All points from the depth image are taken. The estimation time with 100.000 points was about 15 seconds per frame. Two depth images from the sequence are shown in 4.28. The floating point depth image was scaled and shifted, such that the depth range of the person is clearly visible.

The 24 DOF are in detail: 4 for each arm (3 shoulder and 1 elbow), 6 for the global transform and 5 for each leg (3 hip, 1 knee and 1 ankle). The initial pose was defined manually. The resulting poses are shown in Figure 4.29, where the model is superimposed on the original image. The model in white is overlayed on the original color image. Because the background is very dark, the overlayed model is only visible clearly, where it does not fit the image perfectly, such showing the accuracy of the estimation. The remaining inaccuracies are due to false or noisy depth data from the stereo algorithm and due to inaccuracies in the model, which does not perfectly fit the person. Especially deformations from clothing or muscle bulges are not covered by the algorithm.

The original sequence was recorded with 25fps, after 130 frames tracking of one arm is lost. When the tracking is lost, i.e. the estimated pose is too far away from the correct one, the motion tracking can not recover, because the assumption, that correspondences between model and depth points can be established by nearest neighbor, doesn't hold. The lowest amount of correspondences for valid tracking within the 130 frames was approx. 3000 with 3 ICP iterations and 5 iterations within the Gauss-Newton optimization.

Figure 4.30 shows the same results from another camera view, which has not been used for estimation. The remaining inaccuracies are due to errors in the calibration, however it shows that the motion in depth (parallel to the optical axis of the camera) is also correctly estimated.

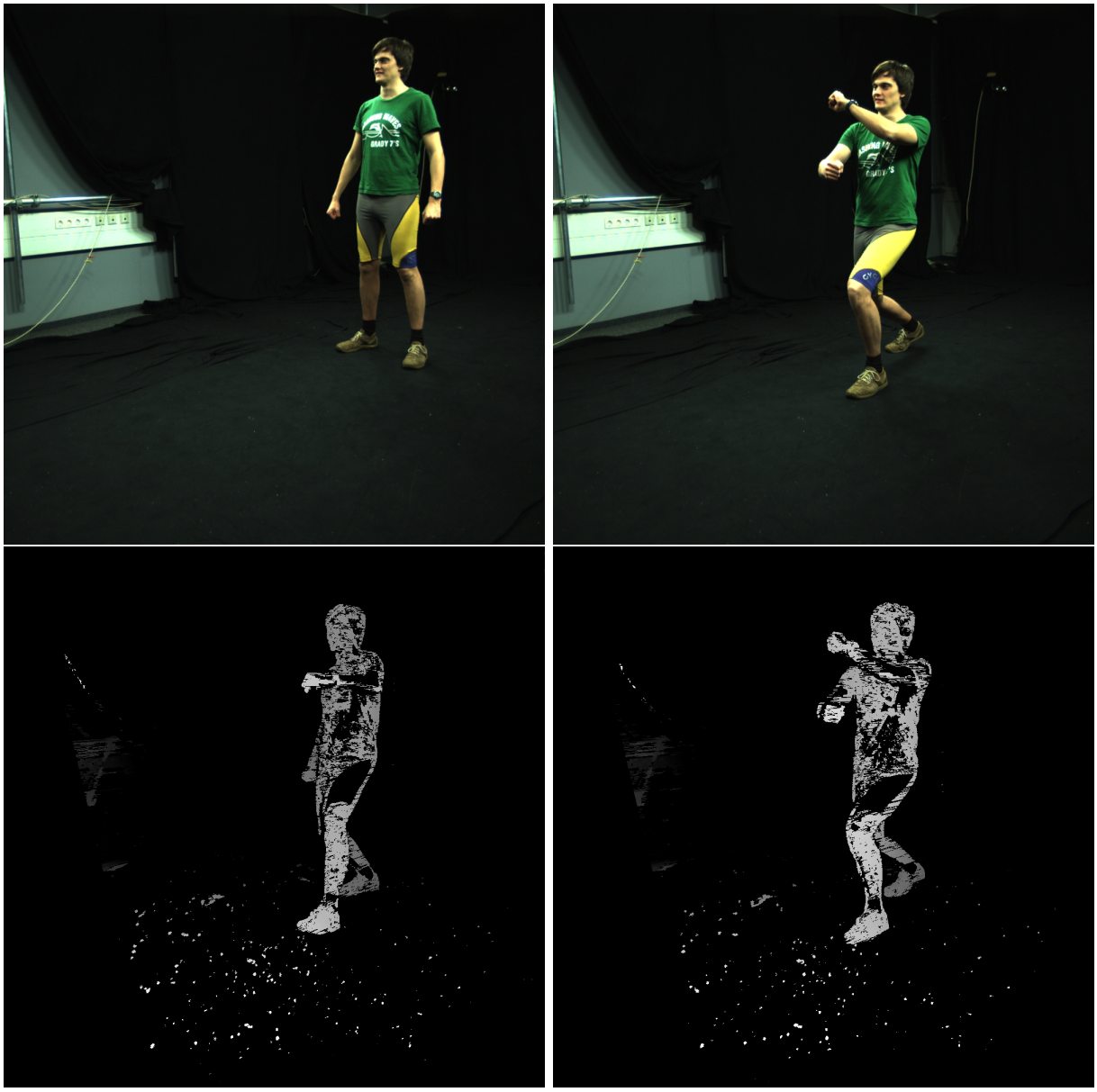


Figure 4.28: Original undistorted image and calculated depth image below it. The depth image was edited for visualization. There are more 3D points in the background, which are not shown for better visibility.

Figure 4.29: Estimated motion with 24 DOF. Model superimposed on the original image. Frames 3,20,35, 50,60,75, 90,110 and 130

Figure 4.30: The same estimated motion seen from another camera view, which has not been used for the estimation.

Computation Time The complexity and computation for the important parts of the algorithm are shown in Table 4.3. The parts are described in detail in section 4.6.2. They are

1. Building of visible model points. The model is positioned in the current pose, which is predicted from the pose of the last frame. Then, the rendered image is analyzed to find the visible model points. The computational cost of this part depends on the model complexity (number of points and triangles) and of the image size.

2. Correspondence Search. For all observed points the nearest model point is sought. This search is sped up by height ordering, such that the factor c_1 is in 0.2 to 1.0 depending on the relation between height and width of the visible model points. For a standing person this is approx. 0.2.

3. The pose estimation requires multiple iterations, such that the Gauss-Newton method converges to the next local minimum. In each iteration inversion of the Hessian matrix is necessary as well as recalculating the new position of model points.

Additional the computation time on a 2.2Ghz Intel Core2 Duo (1 CPU used) is given to show the relation between the parts. Measurements are carried out for two different body models, one with 86000 triangles and the other with 10000 triangles. Additionally a different amount of correspondences were tried. The results show, that the correspondence search scales linear with the number of observed points. The pose estimation however scales nonlinear, because the time doesn't depend on the matrix inversion alone, but a significant amount of time is necessary to recalculate the new positions of all model points used in the

	Calculation of visible model points	Correspondence Search	Pose Estimation with Gauss-Newton
Computational cost N visible model points M observed depth points $P = 24$ estimated DOF	depends on image size and model complexity (nr of triangles)	$O(c_1 MN)$ $c_1 \in [0.2, .., 1.0]$ dependent on the current pose	$O(MP^3)$ for matrix inversion
Computation Time	image size 502x502		
$N \approx 3000$	86000 triangles		5 iterations
$M \approx 6800$	14 ms	510 ms	960 ms
$M \approx 3400$	14 ms	250 ms	360 ms
$M \approx 850$	14 ms	65 ms	84 ms
$M \approx 340$	14 ms	26 ms	38 ms
Computation Time	image size 502x502		
$N \approx 1000$	10000 triangles		5 iterations
$M \approx 3500$	7 ms	84 ms	370 ms
$M \approx 1700$	7 ms	43 ms	160 ms
$M \approx 860$	7 ms	20 ms	84 ms
$M \approx 560$	7 ms	14 ms	56 ms

Table 4.3: Computational cost and computation time for one ICP iteration of the algorithm.

estimation. The amount of model points is not equal to the amount of correspondences, because multiple observed points can have the same corresponding model point.

Error Measurements An analysis of the estimation can not be carried out, because no ground truth for the motion angle is available. This is a general problem in motion capture approaches and quantitative comparisons are often not done. In [111] the estimation was compared with values from a marker-based system. However marker-based systems are not error-free, their accuracy depends on the correct placement of markers, the correct detection, camera calibration and a well fitting body model. A marker-based tracking system was not available. To give at least an impression how accurate the estimation is, the development of the error function (Equation 3.42) during optimization is shown in Figure 4.31. The error is calculated as the root mean square of the difference between model points and corresponding observed points. The graph shows one error value per frame for the estimation with 3 ICP iterations and 1 ICP iteration, which is less accurate. The difference between both is shown in the bottom of Figure 4.31. After frame 135 tracking is lost, which is visible as an increase in the RMS error. However, the relative increase in the RMS error is rather small. The RMS error alone is therefore not suited to decide automatically, if the tracking is lost. The error reflects the inaccuracies in the depth data and in the 3D model of the person.

Figure 4.32 shows all error values for all iterations from frame 83 to frame 93. Because there are up to 5 iterations within Gauss-Newton and 3 ICP iterations, there are 15 values for each frame.

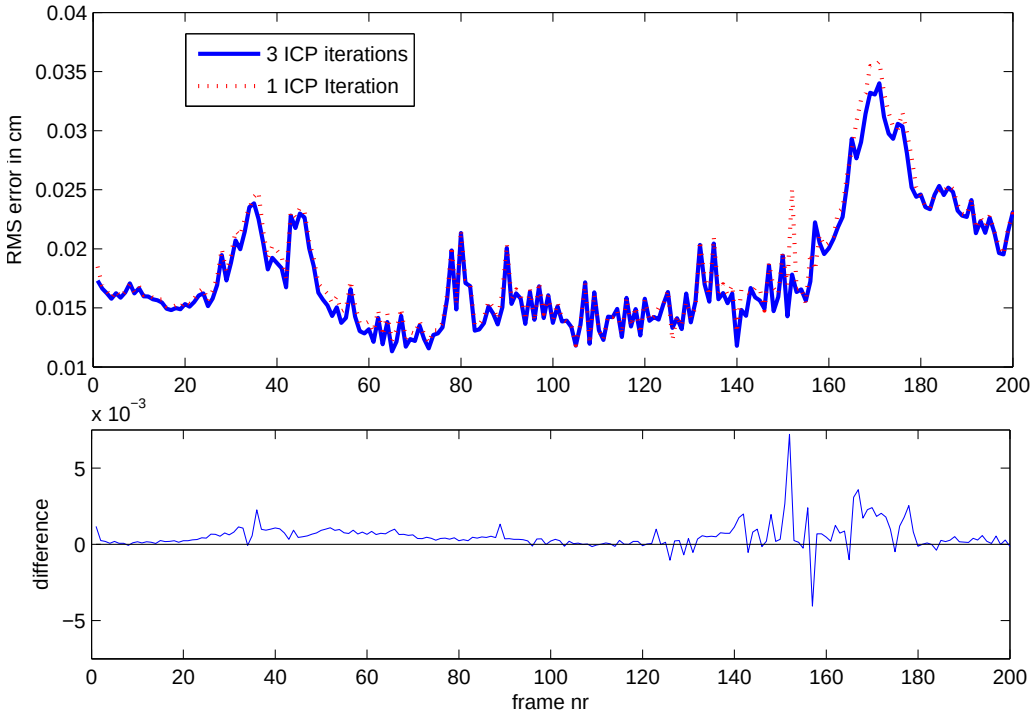


Figure 4.31: The Root Mean Square (RMS) error for the estimated sequence. Three ICP iterations per frame and 300 correspondences are used. The plot in the bottom shows the difference of the RMS error for 3 and 1 ICP iteration. Values above zero indicate, that the error decreased from 1 to 3 iterations. Tracking was lost after frame 135, which is visible as an increase in the RMS error.

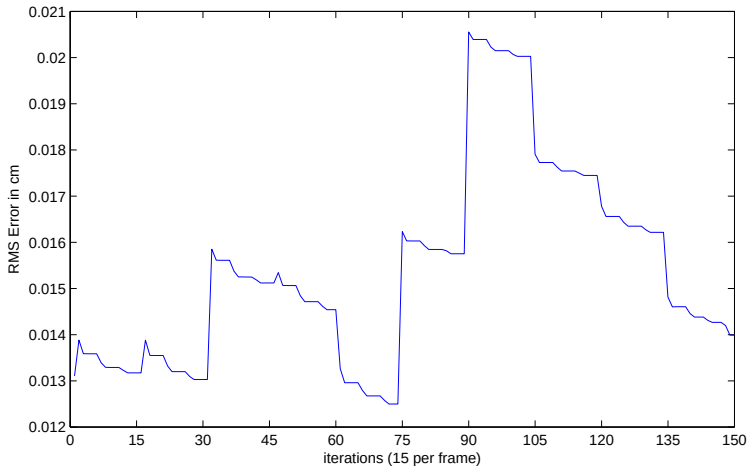


Figure 4.32: For each frame 3 ICP iterations and for each ICP iteration 5 Gauss-Newton iterations are visible in the plot.

4.7 Combination of Depth and Silhouette

In the previous section the motion of a human is captured by utilizing single cues, like depth data (3D points) from stereo imaging and 2D data as silhouettes and tracked corners. In this section 2D and 3D data are combined to track the motion of a human.

The integration of different vision cues into parameter estimation of a model is a complex and difficult topic. Different cues, like tracked edges or points, give different information about the model parameters. Additionally the measurement noise of different cues can vary dramatically. In [48] both aspects are addressed by modeling different image cues with predefined regions for each cue. For example, tracked edges have a region that is elongated along the edge and less elongated perpendicular to it. The regions define a hard limit on possible displacements. These regions are then propagated through the estimation by affine arithmetic. The resulting region in parameter space is then approximated by a Gaussian distribution. The Gaussians from each cue are combined by a *Maximum Likelihood* estimator, whose result is integrated in a classical Euler integration procedure. The defined image regions are supposed to set hard limits on the possible displacements, however the limits are softened, due to Gaussian approximation of the resulting parameter region. Therefore the approach becomes similar to a covariance based approach, where each image cue has an associated covariance matrix.

For example, edges would have a covariance largely extended along the edge and only slightly perpendicular to it. The approach taken in this work is different. The silhouette information is integrated by changing the objective function, such that the distance of the projected 3D-point to the 2D line is minimized. In the image plane this a point to line distance. An equivalent objective function is obtained by minimizing a point to point correspondence, where the covariance for one point is infinitely extended in the direction along the edge. In this thesis the information obtained from each cue is directly modeled in the objective function, instead of extending the observations with regions or covariances.

The different measurement noise of different cues is integrated in the estimation in this thesis by weighting each correspondence with a scalar. Weighting with a covariance matrix would be possible as well. However for the different cues in this work the measurement noise is not exactly known and therefore covariance matrices are assumed to be diagonal and extended the same in all directions. Additionally it is assumed, that the measurement noise is the same for all measurements of one cue, resulting in one single scalar weight for each cue.

The combination of cues is a weighted addition of all correspondences, such that the complete objective function f_{comb} for minimization is:

$$
\begin{aligned}
f_{comb} = \quad & \textstyle\sum_i w_i \quad \left|\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{p}}_{\boldsymbol{i}}\right|^2 + \\
& \textstyle\sum_j w_j \quad \left|\boldsymbol{m'}(\boldsymbol{p_j}, \boldsymbol{\theta}) - \tilde{\boldsymbol{p}}_{\boldsymbol{j}}{}'\right|^2 + && (4.5) \\
& \textstyle\sum_k w_k \quad \left|\tilde{\boldsymbol{n}}_{\boldsymbol{\omega}} \ \left(\boldsymbol{m'}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{q}}'_{\boldsymbol{\omega}}\right)\right|^2 && (4.6) \\
& && (4.7)
\end{aligned}
$$

where the sum over i are 3D-point-3D-point correspondences, the second sum over j are 3D-

point-2D-point correspondences from point tracking and the third part over k are 3D-point-2D-line correspondences from silhouette differences. Each correspondence may have its own weight, however in the experiments here, the weights are equal for all correspondences from one cue.

The weights reflect the measurement noise and the different scale of measurements, e.g. the measurement unit of 3D point positions from stereo images is meter, while the 2D measurement unit is pixels.

4.7.1 Arm Tracking with a PMD-Camera

Figure 4.33: Setup used for the arm tracking. PMD camera on the top with IR-LEDs next to it.

A *Photonic Mixer Device*(PMD) is able to measure the distance to some object. Similar to laser range scanners it is based on the time-of-flight of light. In contrast to the rather expensive laser range scanners, e.g. SICK [122], which usually give only one line of distances at a time. A PMD device gives distance values for a complete volume at a time. The construction and working principle is similar to conventional cameras. Their advantages are the lower production cost and ability to measure a complete volume of distances at a time. The time of flight is measured by phase differences between modulated send out light and received light. To become more invariant to scene lighting and less disturbing, infrared light is used. An installed IR-filter (non-IR light is blocked out) on the objective of the PMD-camera, reduces the effect of scene lighting. More details can be found in [94].

The specification of the PMD-camera used in this experiments are:

- Resolution: 64×48 pixels
- Max range: 7.5m
- depth resolution: 6mm
- frame rate: up to 50fps
- wavelength: 850nm

In Figure 4.33 the setup used in the experiments here is shown. On the top is the PMD camera with Infrared-LEDs next to it. On the bottom are conventional cameras installed, which are called standard camera in the following.

The PMD-camera provides three kinds of images: a floating point image with distance values, a grey value infrared image and an image with coefficients that are used for distance calculation, which usually looks very similar to the infrared image. The infrared grey value image is shown in Figure 4.34. It is scaled from the original size 64×48 up to 320×240 for better visibility.

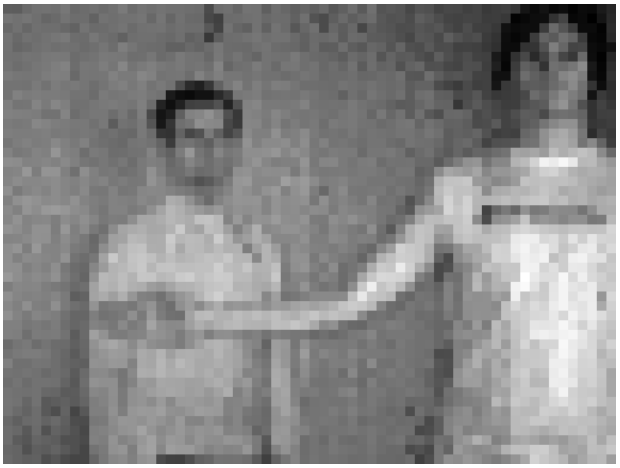


Figure 4.34: Infrared grey value image. Original size: 64×48 pixels

The PMD-depth image is best visualized by a view on the resulting 3D-scene points as in Figure 4.35, which shows four views on the same points from different angles. The depth image has been altered to eliminate values with depths between fore- and background. Pixels that have a standard deviation of 0.15 m in a $3\times$ around them are eliminated and not shown. The depth points for another unaltered depth image is shown in Figure 4.36. As visible, there are many points in between the fore- and background, though no scene objects are actually there. Consider the scene volume, whose light is gathered by one image pixel. If there is a depth discontinuity in that volume, the resulting depth value is somewhere in between the fore- and background, because light is gathered from the fore- and the background. While this is a desired effect for closed surfaces, these depth points are not advantageous for motion tracking. In addition to these values between fore-and background there are outliers, which are caused by reflected scene light. There are more outliers, if the scene lighting contains more infrared light. To reduce the influence of in-between-points and outliers, a variance filter is run on the depth image, that calculates the variance within a 3×3 window and sets all pixels with a deviation larger than a threshold to zero. Sensible values are in between 0.1m and 0.25m.

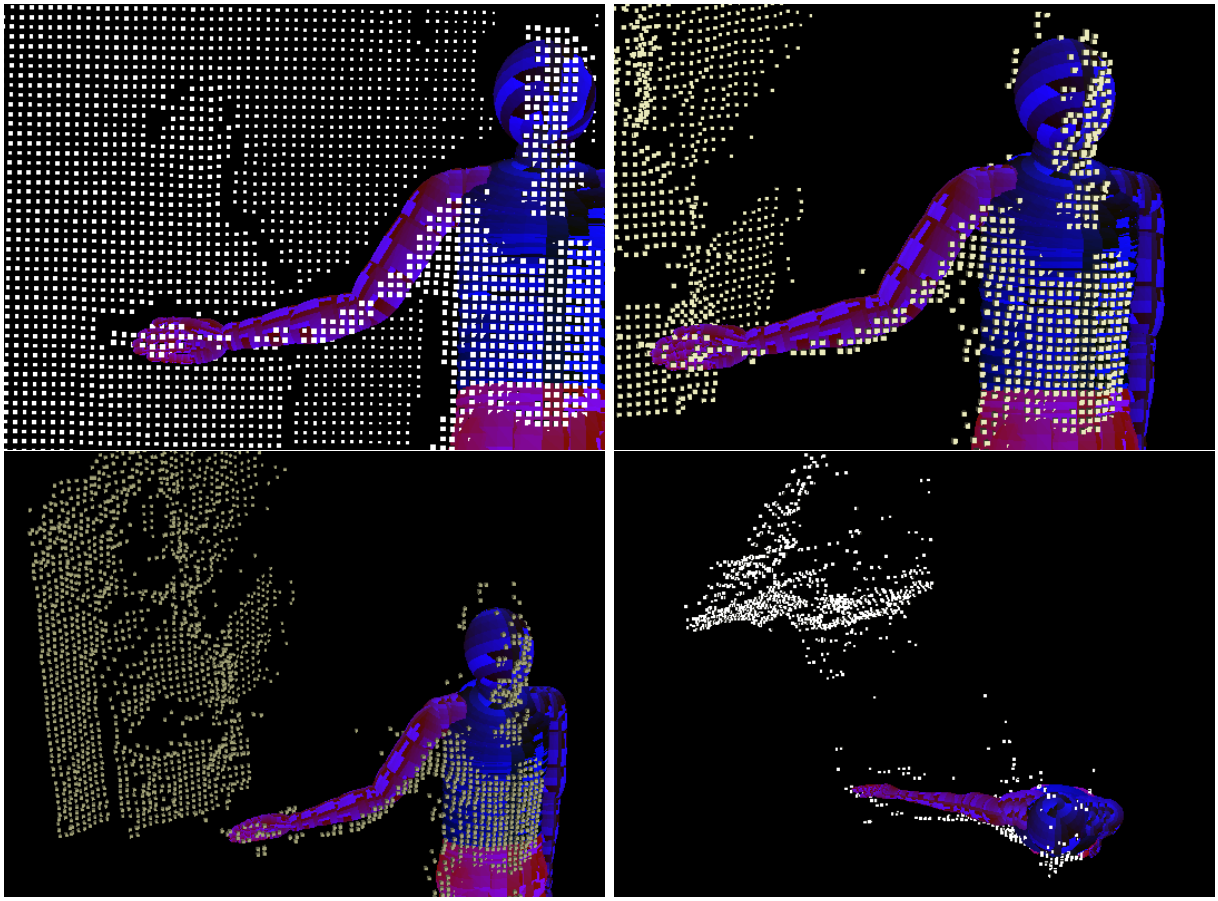


Figure 4.35: Four views on the depth points, variance filtered

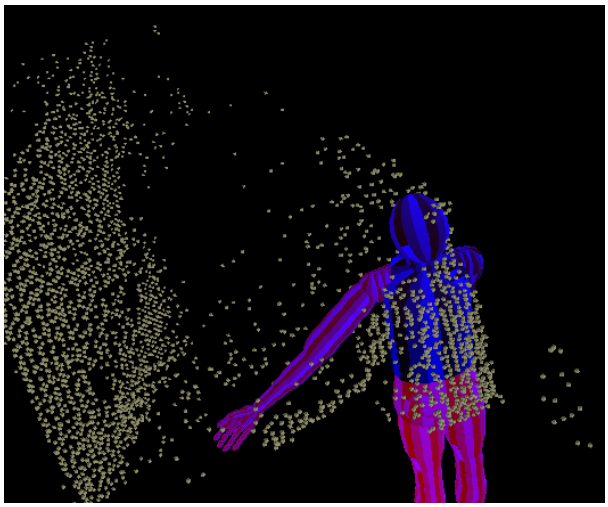


Figure 4.36: Depth points from unaltered PMD-image.

PMD-Camera Calibration

The cameras below the PMD-device captures standard RGB images with 1024×768 pixel resolution. To capture the motion of a person the exact transform (rigid motion) between the cameras is necessary.

Usually camera calibration is not a difficult issue, but in this case the standard calibration routines turned out to yield inaccurate results. The standard camera was calibrated using a planar checkerboard calibration pattern and the algorithm of [137] as available in openCV [65]. The calibration routine estimates for a sequence of checkerboard images the internal values of the camera including the radial distortion. Additionally the external parameters are estimated for each image in the sequence.

Estimation of the internal parameters of the PMD-camera needed additional assumptions. Because the resolution of the camera is rather low and the infrared images lack sharpness and contrast, the corners of the calibration pattern are detected inaccurately. Therefore it is assumed, that the principal point of the PMD-camera is in the center and that there is no radial distortion. This assumption is valid, because the field of view of the PMD-camera is small with approx. 20 degrees in the vertical direction. With these assumptions the focal length could be accurately estimated.

Because the image acquisition of both cameras is synchronized, the rigid transform between both is the same for each image from the checkerboard sequence. Averaging over all images yields the final transform. By analyzing the variance in the translation over the image sequence and by plotting epipolar lines between the images, it turned out that the transform could not be accurately estimated. The translation differed heavily from image to image. Therefore the camera calibration algorithm as described in section 3.3.4 was used to estimate the rotation only. Translation between the images was measured manually.

The calibration result is shown in Figure 4.37. Both images show the projection of depth points from the PMD-camera into the standard-camera image. The top image shows a projection with no transformation between the cameras and the bottom shows the projection with the calibrated transform. As visible the white depth points are in better alignment with the person as in the top image, but lack still some accuracy in the region of the arm. In spite of these inaccuracies, the motion of one arm could be successfully and accurately tracked by a combination of silhouette and depth information as shown in the next section.

4.7.2 Experimental Results for Combined Tracking

Because the field of view of the PMD with 20 degrees is rather small, the covered scene part in an acceptable with respect to depth outlier was about 3m. In this distance the motion of one arm is completely visible. The motion in the following sequence is estimated from 3D-point-3D-point correspondences and 3D-2D-line correspondences established from silhouette information. The motion of shoulder and elbow as well as global translation and rotation were estimated, all together 10 DOF.

The motion of the right arm could be successfully tracked over the whole length of

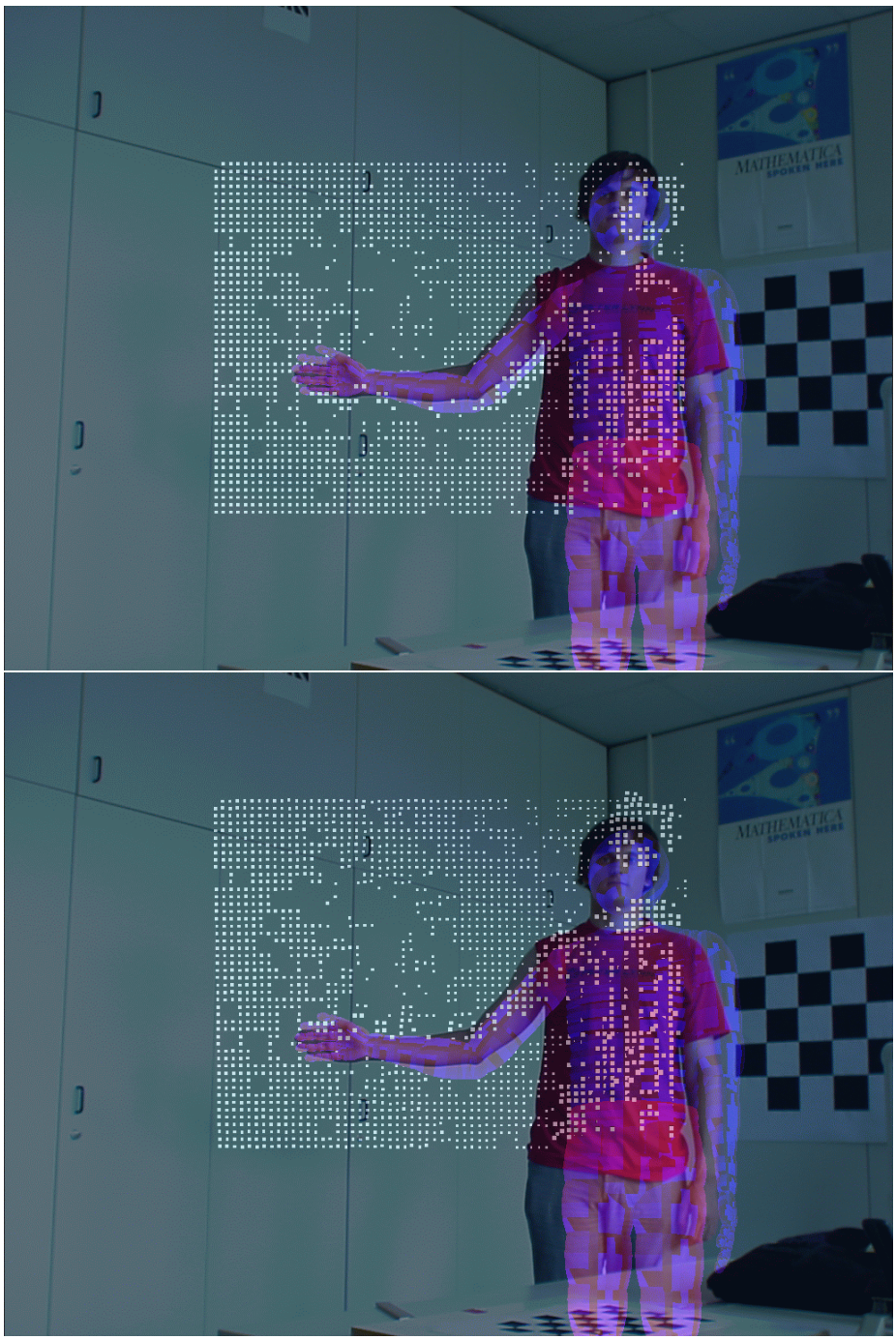


Figure 4.37: Calibration results: Depth points from PMD-camera projected into standard-camera image. Top: Same external parameters. Bottom: Applied calibration result.

Figure 4.38: Silhouette correspondences are accurate though the background is very dynamic. Each row shows the iterations for one frame. Frames 40-43

different sequences. Though the background is non-static and cluttered, because a person was walking around in the background, silhouette correspondences are accurate as visible in Figure 4.38 due to the combination of grey value gradient and color histograms as described in section 4.5.3. An example sequence of 670 frames which was recorded with 7 fps is shown in Figures 4.39 and 4.40. Depicted is the standard camera image superimposed with the rendered model in the estimated pose. The accuracy of the estimation is limited by the accuracy of the calibration and by the fitting quality of the model, which does not reveal the person's shape exactly in the shoulder region.

The estimation from depth values alone is less accurate as visible in Figure 4.41. While estimating the pose in the 670 frame sequence the tracking from depth data alone, tracking was lost for 50 frames, but recovered afterwards.

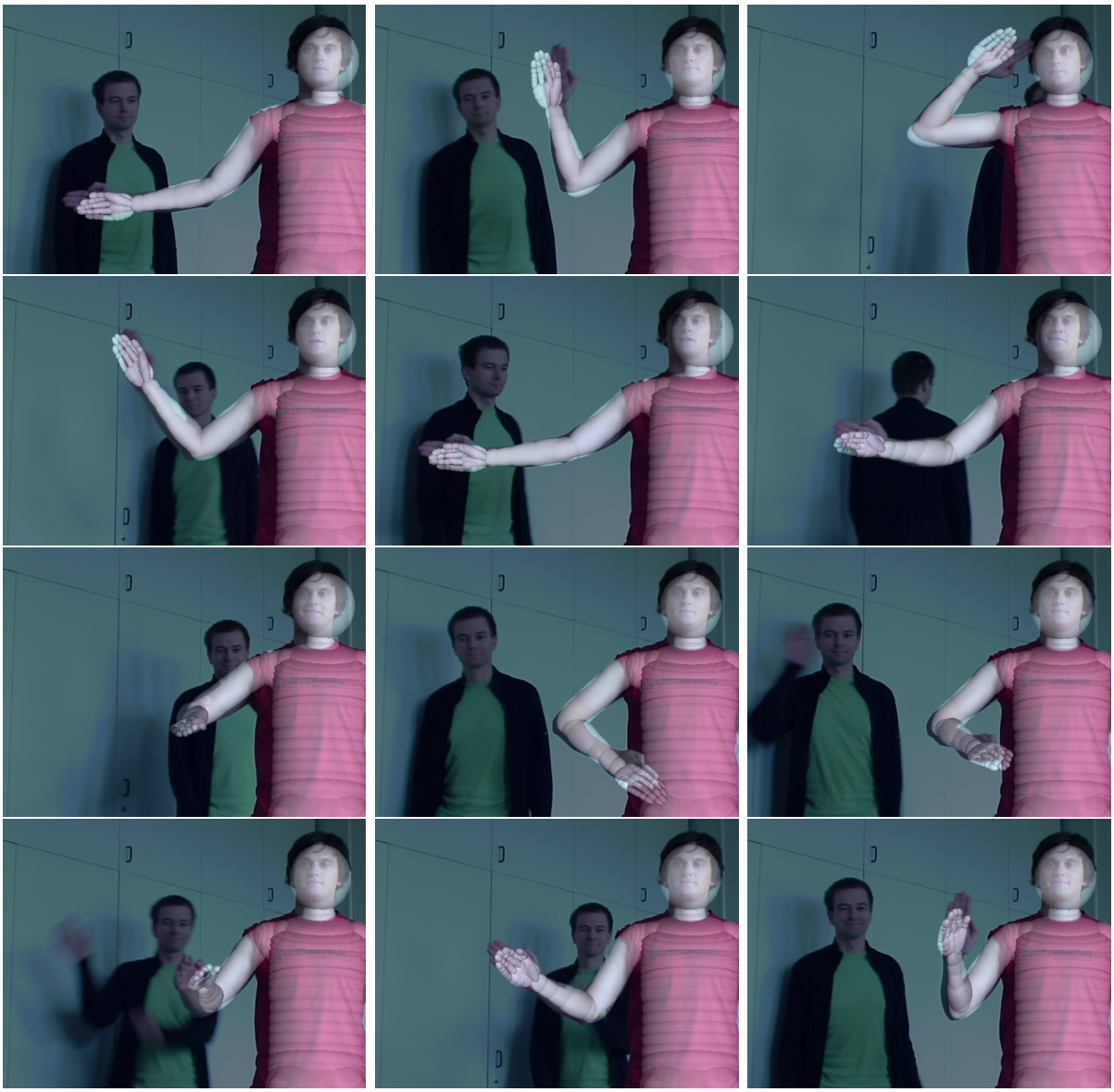


Figure 4.39: Result sequence with dynamic background (21 images out of 300).

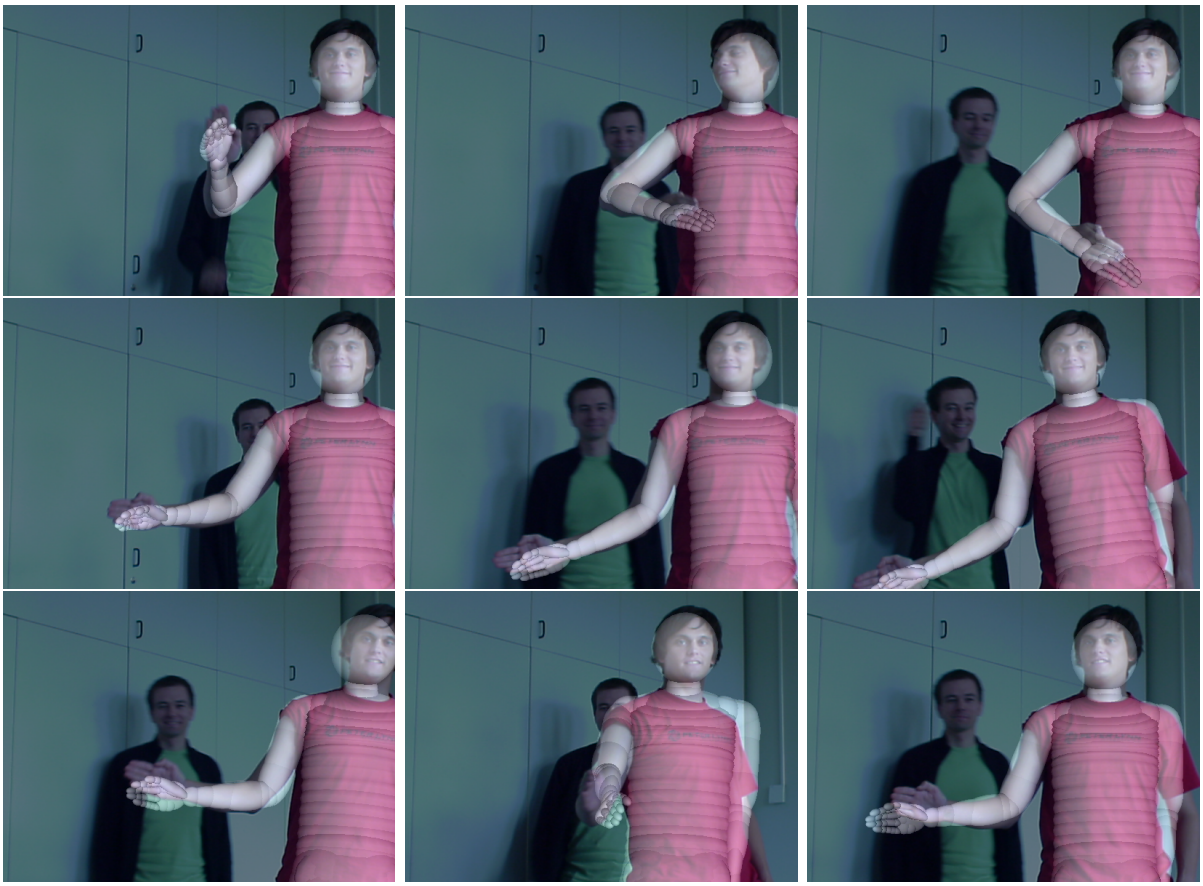


Figure 4.40: Example result sequence with dynamic background. Frames 136-225

Figure 4.41: Tracking with depth points alone is less accurate and was lost completely for 50 frames. Shown here are frames 21, 34 and 42.

Chapter 5

Conclusions

At first a summary of the described methods and algorithms is given, which is then followed by a discussion of the contributions.

Accurate marker-less systems are usually based on visual hull reconstruction methods, which require a segmentation of person and background in each camera image. They rely on accurate segmentation of the person in multi-view images. The approach made in this thesis was different. It tried to give answers to the question, how motion capture can be fast and reliable in spite of cluttered and dynamic background.

The pose and motion estimation of the observed person is carried out in an optimization framework for articulated objects. The motion function is formulated with kinematic chains consisting of rotations around arbitrary axes in 3D space. This formulation leads to a Nonlinear Least Squares problem, which is solved with gradient-based methods. Though pose estimation is often solved by gradient methods, the gradient calculation is usually numeric. With the formulation in this thesis the necessary derivatives can be derived analytically. This speeds up processing and increases accuracy. Different gradient based methods were compared to solve the Nonlinear Least Squares problem, namely dampened Gauss-Newton (GN), dampened Newton-Rhapson, standard Gradient (GD) descent and Stochastic Meta Descent(SMD).

The pose estimation requires correspondences between known model of the person and observed data. To obtain this model, a new method has been developed, which fits a template model to a specific person from 6 posture images taken by a single camera.

Different correspondence types are integrated in the pose estimation. They are 3D-point-3D-point, 3D-point-2D-point and 3D-point-2D-line correspondences. The model entity is a 3D-point and the data can be a point or a line in 2D or 3D.

Stereo image processing is independent of dynamic or cluttered background and provides a surface-like description with 3D points of the observed person. The 3D point data was acquired in the experiments by a stereo algorithm from a two camera setup. The experiments showed, that complex motion with 24 degrees of freedom (DOF) is trackable even from a single stereo view until body parts get totally occluded. The correspondence search takes a large amount of time of the whole processing. Therefore optimizations were implemented to find correspondences efficiently, like rendering the model with unique colors

for each triangle, which checks at the same time for occluded body parts.

Further methods were developed to estimate pose from a single camera view with cluttered dynamic background. Similar to other work on 2D-3D pose estimation, correspondences between model and image silhouette of the person were established by analyzing the gray value gradient near the predicted model silhouette. However, images with cluttered background and textured clothes lead to many false correspondences, which results in lost tracking even after a few frames. The solution presented makes use of color histograms on the person's body parts. The combination of gray value gradient and color histogram results in a sufficient number of correct 3D-point-2D-line correspondences.

To be able to track motion, without visible silhouettes, e.g. an arm in front of the body, image regions on the person are tracked with a point tracking method resulting in 3D-point-2D-point correspondences. The results showed, that even from a single view motions can be estimated, which include arm and leg motion. However changes in depth are difficult to estimate, as can be expected from a single view.

The combination of 3D depth data and 2D image data was tested in the experiments with a PMD camera (Photonic Mixer Device), which measures the depth to scene points by the time of flight of light. The resulting depth image is similar to those of a stereo algorithm, but the resolution is less and exhibits more outliers. An additional standard camera was used to establish correspondences between model and image silhouettes. The motion of one arm could be estimated correctly, though another person was walking around in the background. The estimation with one cue alone, e.g. stereo data or silhouette data, was not reliable enough.

5.1 Discussion of the Contribution

The work is now concluded with respect to the two contributions, the pose estimation method and the algorithms developed in order to find reliable correspondences (tracking):

Pose Estimation To answer the question asked in the introduction (section 1.1.2): Is there a common or optimal way to estimate pose of articulated objects ?

A general answer can not be given. However, the results of this work suggest, that previous methods, which are partially enhanced and unified in this work, are very promising. They apply to pose estimation of rigid and articulated objects, that have the following restrictions:

- They are model-based, i.e.a body model for the person is known, which fits the person in size and specifies the motion capabilities.
- The optimization function relies on correspondences between observed data and body model.
- They combine one or more of these correspondences: 3D-point-3D-point, 3D-point-3D-line, 3D-point-2D-point or 3D-point-2D-line

- They have no functional description of the shape of model parts, i.e. each body part can be treated as a rigid object or has a linear blend skinned shape.

Within these restrictions, there is no obvious reason why the optimization function or the optimization method should use approximative methods. One might argue, that approximative methods are faster. However, in the literature no direct comparisons are made between approximative and direct methods. Concerning computation time, we showed in section 4.6.4, that the direct approach can compete with approximative methods. Approximations to the motion function include the constraint of connected body parts indirectly, e.g. by regularization [40] or by projection [33]. The disadvantage of approximative hierarchical methods is illustrated in [87].

Further we conclude, that the Gauss-Newton optimization method is better suited than Gradient-Descent methods. Although the computational complexity of Gauss-Newton is higher, it converges faster, especially in flat regions of the error surface (section 3.8). Additionally, it only requires a single dampening parameter to be adjusted for optimal convergence speed, which is in practice more convenient, than one or two parameters for each DOF as the SMD-Gradient Descent (see section 3.8.1).

The Newton methods allow to integrate second order motion derivatives as analyzed in the pose estimation chapter for a simple arm configuration with two joints. Though the setup is quite simple, the results show, that the Newton-Rhapson method, which includes the second derivatives, approximates the real error function more closely as expected. However this advantage is counterbalanced by the need of a more sensibly chosen dampening parameter. Therefore the Gauss-Newton method is favorable in practice, because it performs well, even if the dampening parameter is chosen not very sensibly.

Tracking The contribution of this thesis with respect to the tracking part are the methods developed in order to find reliable correspondences. The search for correspondences should be fast and reliable in spite of cluttered and dynamic background. Additionally, it is investigated what kind of motions can be tracked from a single view.

Depth data from stereo algorithms is not disturbed by cluttered or changing backgrounds or dynamic lighting conditions. Therefore methods are developed to find correspondences by nearest neighbor efficiently from depth stereo data. A comparison with similar approaches showed (section 4.6.4), that the overall tracking is fast enough to compete with other methods, though no approximations to the motion function are made.

The computation time was measured without the time for stereo calculation, because there are real-time stereo algorithms available, which utilize either the GPU [134] or a on-camera DSP [126]. Such an algorithm was not available, instead a high-quality algorithm was used [44].

In order to simulate the effect of real-time stereo algorithms, whose results are more noisy, experiments with artificial data and controlled noise levels were executed. The results give an impression about tolerable noise levels and image capture speed.

Further experiments with complex 24 DOF motion shows ,that even from a single view, motion can be tracked, that includes full body motion. Of course, the method loses track,

when body parts become completely occluded. More images from additional views can aid in this cases. The extension to multiple stereo views is straightforward, because they provide more 3D data points.

However, instead of increasing the number of camera views, it was investigated how additional information from the same view can increase the tracking. Therefore a new method was developed, which finds correspondences between model and image silhouette without explicitly segmenting fore- and background in the image. This is achieved by combining gradient filter responses with color histograms, both are independent to small changes in the scene lighting. Concluding results show, that the combination of depth data and silhouette data can increase stability and accuracy of the tracking.

The experimental results are mainly qualitative, because ground truth data for human motion is difficult to obtain. Other approaches compared the results with values from a commercial marker-based system. This was not available. Also marker-based systems are not error-free, their accuracy depends on the correct placement of markers, their correct detection, camera calibration and a well fitting body model. The accuracy of the approach here also depends on the camera calibration and the fitting quality of the body model. Instead of correct marker placement and marker detection, here the data quality (accuracy and amount) of the stereo algorithm and of the silhouette detection methods control the accuracy. However, it has to be mentioned that the focus of this work was not on accurate estimation, but on reliable estimation in difficult environments and from a single (stereo) view.

The developed pose estimation is available for rigid objects, which includes camera calibration, in the open-source C++ library BIAS [41] and has been used in the following projects
ARTESAS [2], MATRIS [3, 21] and ORIGAMI [1, 9].

Parts of this thesis have been published in [51, 52, 53, 54, 55, 56].

5.2 Open Problems and Future Work

The tracking approach assumes, that the motion between image frames is small and that the pose of the person is approximately known for the previous frame. If the tracking is lost, due to incorrect or insufficient correspondences, these assumptions are not fulfilled and the estimation usually doesn't recover. This is a common problem in motion tracking. One solution can be the combination of a multi-hypotheses tracking similar to particle filter methods with the correspondence based estimation of this thesis. Particle filter approaches are usually correspondence-less and suffer from the curse of dimensionality, if the number of DOF to estimate is large. However, they exhibit the ability to track multiple hypothesis. A combination of the correspondence based pose estimation as in this thesis with a particle

filter tracking can combine both advantages without sharing the disadvantages, because a much smaller number of particles is necessary. A multi hypotheses tracking based on particles is also suited for real-time issues, because parallel processing is straightforward to accomplish. All operations for all particles are the same and is therefore easy to distribute on multiple processors.

Another important issue is the necessity, that the model of the person needs to be known.

While an off-line fitting before the capturing as in this thesis is often applicable, a simultaneous fitting of the model and estimation of joint angles would save time and could increase the accuracy of captured motion. This has been already investigated in previous work [102], where a model of scalable spheres was fitted to stereo and silhouette data. However this approach isn't applicable for real-time tracking, because the number of DOF for the optimization is too high.

Appendix

A Lines in 2D and 3D

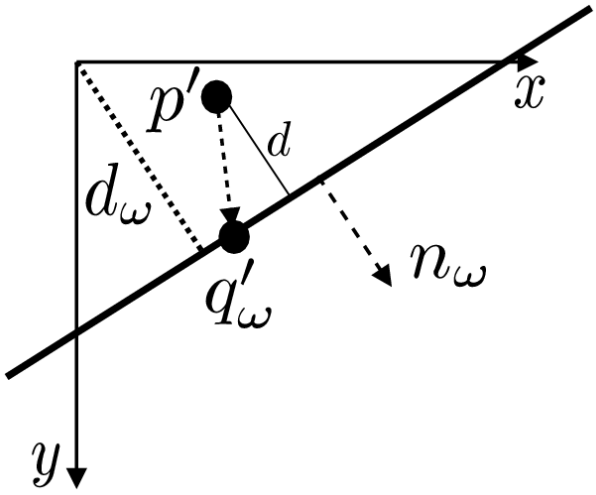


Figure 1: Description of a 2D line

A.1 Lines in 2D

A line in 2D can be described by a point $\boldsymbol{q'_\omega} \in \mathbb{R}^2$ on the line and its normal vector $\boldsymbol{n_\omega} \in \mathbb{R}^2$, which is orthogonal to the line and has unit length. For a point $\boldsymbol{p'} \in \mathbb{R}^2$ lying on the line the following condition must hold as visible in figure 1:

$$\begin{aligned} \boldsymbol{n_\omega}^T \left(\boldsymbol{q'_\omega} - \boldsymbol{p'}\right) &= 0 \quad \Leftrightarrow & (1)\\ \boldsymbol{n_\omega}^T \boldsymbol{q'_\omega} - \boldsymbol{n_\omega}^T \boldsymbol{p'} &= 0 \quad \Leftrightarrow & (2)\\ \boldsymbol{n_\omega}^T \boldsymbol{p'} - \boldsymbol{n_\omega}^T \boldsymbol{q'_\omega} &= 0 \quad \Leftrightarrow & (3)\\ \boldsymbol{n_\omega}^T \boldsymbol{p'} - d_\omega &= 0 & (4) \end{aligned}$$

Note that the sign of d_ω is important, because it is positive, if the normal points away from the origin and negative if it points towards the origin. This gives the minimal description for a 2D line by one component of the normal, e.g. n_x and d_ω.

In this work the description by a point and normal is used, because a point on the line is given in most cases, e.g. by a found correspondence. Such that a line l' is described by $l' = (\boldsymbol{n_\omega}, \boldsymbol{q'_\omega})$.

Distance of 2D point to 2D line

As visible in figure 1, the distance d of a point $\boldsymbol{p'}$ to the line given by $(\boldsymbol{n_\omega}, \boldsymbol{q'_\omega})$ is

$$\begin{aligned} d &= \left|\boldsymbol{n_\omega} \left(\boldsymbol{q'_\omega} - \boldsymbol{p'}\right)\right| \Leftrightarrow & (5)\\ d &= \left|\boldsymbol{n_\omega}\, \boldsymbol{p'} - \boldsymbol{n_\omega}\, \boldsymbol{q'_\omega}\right| & (6) \end{aligned}$$

A.2 Lines in 3D

A line in 3D can be described by a point $\boldsymbol{q}_{\boldsymbol{\omega}} \in \mathbb{R}^3$ on the line and its normalized direction vector $\boldsymbol{\omega} \in \mathbb{R}^3$, which has unit length. A point $\boldsymbol{p} \in \mathbb{R}^3$ has the following distance d to that line:

$$d = |\boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{q}_{\boldsymbol{\omega}})| \Leftrightarrow \tag{7}$$

$$d = |\boldsymbol{\omega} \times \boldsymbol{p} - \boldsymbol{\omega} \times \boldsymbol{q}_{\boldsymbol{\omega}}| \tag{8}$$

where $\boldsymbol{\omega} \times \boldsymbol{q}_{\boldsymbol{\omega}}$ is called the *momentum*. The difference vector $\boldsymbol{d} \in \mathbb{R}^3$ from the line towards the point is obtained by

$$\boldsymbol{d} = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{q}_{\boldsymbol{\omega}})) \tag{9}$$

such that the point $\boldsymbol{p}_l$ on the line closest to $\boldsymbol{p}$ is $\boldsymbol{p}_l = \boldsymbol{p} - \boldsymbol{d}$.

Cross Product Matrix

The equation (9) can also be written in matrix notation using the matrix form of the cross product. The cross product of $\boldsymbol{\omega} \times \boldsymbol{p}$ can be written as:

$$\boldsymbol{\omega} \times \boldsymbol{p} = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix} \boldsymbol{p} \tag{10}$$

B Cameras and Projective Spaces

B.1 Projective Space

Given here is at first the relation of points in a projective space $\mathbb{P}^n$ to real points in Euclidean space $\mathbb{R}^n$ and afterwards the formal definition and the advantages of projective space.

Let $\boldsymbol{p} = (p_x, p_y, p_z)^T \in \mathbb{R}^3$ be a 3D point, then the corresponding element $\bar{\boldsymbol{p}}$ in $\mathbb{P}^3$ is obtained by adding another coordinate equal to one.

$$\boldsymbol{p} \in \mathbb{R}^3 = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \quad \mapsto \quad \bar{\boldsymbol{p}} \in \mathbb{P}^3 = \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} \tag{11}$$

The nature of the projective space is such that all elements $\lambda\bar{\boldsymbol{p}} \in \mathbb{R}$ are equivalent, meaning they all describe the same point in 3D space $\mathbb{R}^3$.

Therefore, given a projective point $\bar{\boldsymbol{p}} \in \mathbb{P}^3$, the corresponding Euclidean point $\boldsymbol{p} \in \mathbb{R}^n$ is obtained by dividing the elements by w:

$$\bar{\boldsymbol{p}} \in \mathbb{P}^3 = \begin{pmatrix} \bar{p_x} \\ \bar{p_y} \\ \bar{p_z} \\ w \end{pmatrix} \quad \mapsto \quad \boldsymbol{p} \in \mathbb{R}^3 = \frac{1}{w} \begin{pmatrix} \bar{p_x} \\ \bar{p_y} \\ \bar{p_z} \end{pmatrix} \tag{12}$$

If w equals one, the augmented 3D coordinate is also called a homogenous coordinate.

Definition projective space

Let F be field, e.g. $\mathbb{R}$, and V a vector space over F, e.g. $\mathbb{R}^3$. The subspace $V\backslash\{0\}$ builds together with an equivalence relation the projective space $\mathbb{P}$, where is such that

$$\forall \mathbf{u}, \mathbf{v} \in V\backslash\{\mathbf{0}\} : \mathbf{u} \sim \mathbf{v} \Leftrightarrow \exists \lambda \in \mathbb{R} : \mathbf{v} = \lambda \mathbf{u}. \tag{13}$$

The dimension of P is $\dim(P) = \dim(F) - 1$.

Properties of projective space

Projective spaces in general and the $\mathbb{P}^2$ and $\mathbb{P}^3$ in particular have the following advantages:

- Affine transformations in $\mathbb{R}^n$ become linear in $\mathbb{P}^n$. These property is probably the best known and is often considered as *'using homogenous coordinates'*.

  An affine transform is an invertible linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$ described by a matrix $A^{n\times n}$ followed by a translation $\boldsymbol{t} \in \mathbb{R}^n$. In projective space there is a linear transformation matrix $\bar{A}^{(n+1)\times(n+1)}$ describing the same transformation:

$$\bar{A}^{(n+1)\times(n+1)} = \begin{bmatrix} A^{n\times n} & \boldsymbol{t}^{3\times 1} \\ \mathbf{0}^{1\times 3} & 1 \end{bmatrix} \tag{14}$$

- An affine space is embedded, which allows to distinguish between points and directions. Points have the last coordinate equal to 1, while directions have the last coordinate equal to 0. These points and directions can be directly interpreted as points and directions in the Euclidean space by omitting the last coordinate. Therefore the addition of a point and a direction results in a point, the addition of two directions results in a direction and the addition of two points is not defined.

- The $\mathbb{P}^2$ in particular has a strong similarity to the canonical camera. A canonical camera is in alignment with the world coordinate system and has its principal point at zero and a focal length of 1 (see next section for a description of the general camera model).

  All points in the projective space $\mathbb{P}^2$ can be interpreted as 3D rays in Euclidean space of the canonical camera. Each ray represents the set of Euclidean 3D points that project onto the same image point. The last coordinate can therefore be interpreted like the distance to the optical center of the canonical camera, such that a point on the image plane has its last coordinate equal to one. These points are the Euclidean 2D points of $\mathbb{R}^2$, which lie in the image plane.

- Projective transformations reduce the dimensionality [45]. In this work the perspective projection is of special interest, as described in the next section. While the

general projection by a pinhole camera can be described in three subparts, it becomes a single matrix multiplication in projective space.

Using the $\mathbb{P}^3$ it is possible to project not only points, but also pure directions, which has to be interpreted like projecting a point at infinity. The projection of these so called ideal points is independent of the position of the camera, only the cameras orientation is important.

For further properties of projective space and geometric relations the interested reader is referred to the appendix of [46] for a brief summary or to [59, 47] for a more detailed and elaborate description.

Rigid Body Motions in Projective Space

As mentioned in the previous section affine transforms in $\mathbb{R}^n$ become linear in $\mathbb{P}^n$. In this work rigid body motions in $\mathbb{R}^3$ are of special interest. A rigid body motion (see section 3.1.2) can be described by a rotation $R \in \mathbb{R}^{3\times3}$ and a translation $\boldsymbol{t} \in \mathbb{R}^3$, such that in $RSet^3$ the rigid motion of a point $\boldsymbol{p}_1$ to a new position $\boldsymbol{p}_2$ is:

$$\hat{\boldsymbol{p}} = R\boldsymbol{p} + t \tag{15}$$

The same transform in $\mathbb{P}^3$ with projective points $\bar{\boldsymbol{p}}_\mathbf{1}$ and $\bar{\boldsymbol{p}}_2$ reads:

$$\bar{\boldsymbol{p}}_2 = \begin{bmatrix} R & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} \bar{\boldsymbol{p}}_\mathbf{1} \tag{16}$$

and $\bar{\boldsymbol{p}}_\mathbf{2}$ is obtained by division with the fourth coordinate as explained in section B.1.

B.2 Perspective Projection

A camera maps a 3D scene onto the 2D image plane. This mapping can be the perspective projection, which is usually modeled with the pinhole camera model. Because real cameras are not perfectly pinhole cameras, possible distortions produced by the lens have to be accounted for as well.

The general camera projection of a 3D-world scene point into 2D image plane can be divided in four subparts:

a The transformation from 3Dscene coordinates to 3D camera coordinates

b Reduction to 2D image plane coordinates (pinhole projection)

c Conversion to 2D pixel coordinates

d Application of distortion models to reflect real world lenses

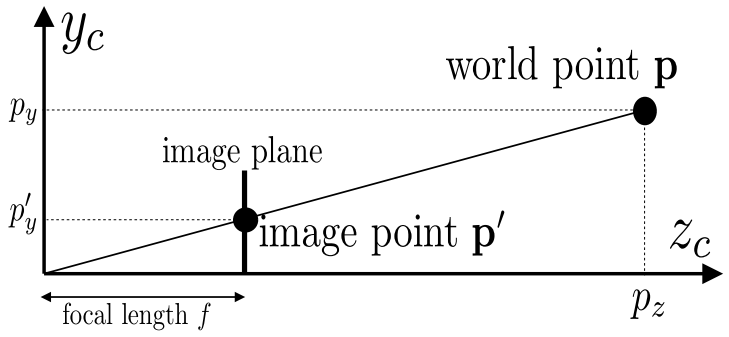


Figure 2: Perspective projection with the pinhole model

a) Transform to camera coordinates

The position and orientation of the camera within the world can be described by a rotation matrix R and a vector t. To apply the pinhole projection equation (next step) world scene points have to be described in the camera coordinate system, see figure 3. This achieved by converting the world scene point coordinates $\boldsymbol{p}_w \in \mathbb{R}^3$ to camera coordinates $\boldsymbol{p}_c \in \mathbb{R}^3$. Let $\bar{\boldsymbol{p}}_w \in \mathbb{P}^3$ and $\bar{\boldsymbol{p}}_c \in \mathbb{P}^3$ be the corresponding points in projective space, then

$$\bar{\boldsymbol{p}}_c = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \bar{\boldsymbol{p}}_w \tag{17}$$

The three parameters of the rotation R and the translation $\boldsymbol{t}$ are the extrinsic parameters of the camera.

b) Pinhole Projection

The pinhole camera model is commonly used to describe the projection properties of real cameras. Figure 2 shows the projective projection of a point $\mathbf{p} = (p_x, p_y, p_z)^T$, which is projected into the image plane at $\mathbf{p}' = (p'_x, p'_y)^T$. By use of homogeneous coordinates this can be written as:

$$\overline{\mathbf{p}}' = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \overline{\mathbf{p}} \tag{18}$$

where f is the focal length.

c) Conversion to pixel coordinates

The third subpart is the conversion to pixel coordinates, which reflect the position on the real physical image sensor. Let $\boldsymbol{p}'_{\boldsymbol{c}}$ be the projected point of the last step and p'_I be the position in the image. Using descriptions in projective space the conversion can be written

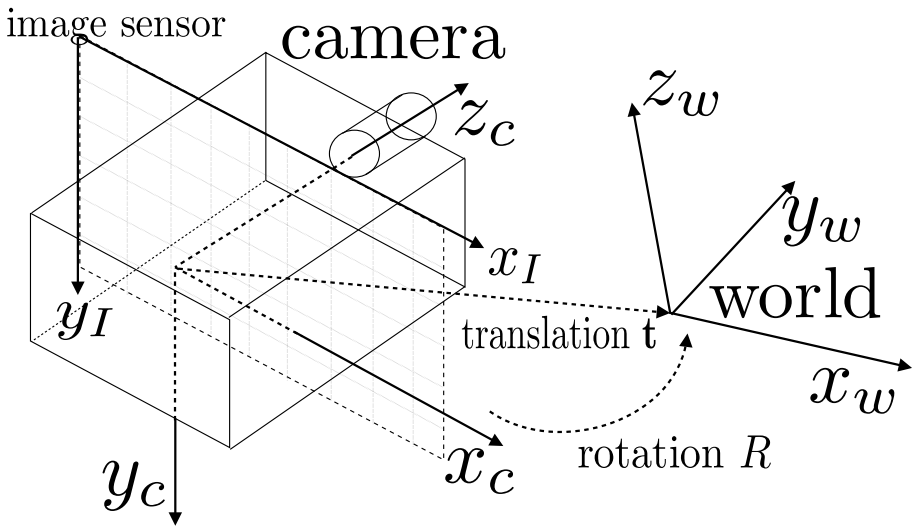


Figure 3: Schematic illustration of relevant coordinate systems. The camera coordinate system $(x_c, y_c, z_c)^T$ is rotated and translated with respect to the world. The image sensor is drawn to show the image coordinate system $(x_I, y_I)^T$.

as:

$$\bar{\boldsymbol{p}}'_{\boldsymbol{I}} = \begin{bmatrix} k_x & 0 & c_x \\ 0 & k_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \bar{\boldsymbol{p}}'_{\mathbf{c}} \tag{19}$$

where k_x, k_y reflect the scale of the image sensor, which can be different in x- and y-direction. The principal point $(c_x, c_y)^T$ is the intersection of the optical axis and the image sensor. The scale values k_x, k_y together with the focal length f are the pixel scale of the camera $s_x = fk_x$ and $s_y = fk_y$. The principal point and the pixel scale are known as the intrinsic or internal camera parameters.

The transformation matrix above is also referred to as the K matrix [59].

Projection Matrix It is common to describe the above three parts of the general projection with a projection matrix $P \in \mathbb{R}^{3\times4}$. Let $\bar{\boldsymbol{p}}_w \in \mathbb{P}^3$ be the 3D-world scene point and $\bar{\boldsymbol{p}}'_I \in \mathbb{P}^2$ be the position in the image after projection, then

$$\overline{\mathbf{p}}'_{\mathbf{I}} = P\,\overline{\mathbf{p}}_{\mathbf{w}} = \begin{bmatrix} k_x & 0 & c_x \\ 0 & k_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \overline{\mathbf{p}}_{\mathbf{w}} \tag{20}$$

d) Lens Distortion

Real cameras are not perfect pinhole cameras. To collect a sufficient amount of light lenses are used. The smaller the focal length of the lens, the larger the field of view. Lenses with

a high field of view induce an image distortion. The distortion can be modeled by a radial model as follows (similar to [137]).

Let $(p'_{I,x}, p'_{I,y})^T$ be the (distortion free) image position of the projected 3D scene point. Then the real image position $(p'_x, p'_y)^T$ with distortion can be written as:

$$p'_x = p'_{I,x} + x[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{21}$$
$$p'_y = p'_{I,y} + y[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{22}$$

with $x = (p'_{I,x} - c_x)$ and $y = (p'_{I,y} - c_y)$ is the image vector from the principal point.

C Absolute Pose Estimation

In this section a method is presented that finds the initial projection matrix from given 3D-point-2D-point correspondences. This initial matrix can be refined by using the optimization method given in chapter 3.3.2.

To estimate pose under projection the initial pose of the object is important, because the optimization may converge into local minima that are not the desired pose. For projection with pinhole cameras, the projection Equation does not incorporate whether the object is in front or behind the camera. Therefore, a valid pose with a local minima is usually found also if the object is positioned behind the camera.

In addition to ambiguities due to local minima the iterative process of pose estimation needs less iterations, if the initial pose is near to the correct final pose.

Presented here is a linear method that calculates an initial pose, whose orientation is close to the real one and whose position is in front of the camera. The method can be found in more detail in the work of Perwass [101], where a description in Geometric Algebra is used. For completeness a description in Euclidean space is given here. An implementation of this method together with camera pose calibration can be found in the class *BIAS::CamPoseCalib* in the open source C++ library BIAS[41].

The steps of the algorithm are as follows:

a Find three image points with largest possible distance to each other and embed them in 3D space according to projection parameters.

b Construct a orthonormal basis from two vectors between the image points, and a orthonormal basis from the three corresponding 3D points.

c Take the transform between both bases as the initial pose by moving the object into the image plane.

d Move the object away from the center, such that the projected vector between two 3D points has the same length as the vector between the corresponding image points.

C.1 a) Find Three Image Points with Maximal Distance to Each Other

Let $\tilde{\boldsymbol{p}}'_{\mathbf{1}}$ be the point which is farthest away from the image center. The second point $\tilde{\boldsymbol{p}}'_{\mathbf{2}}$ is found by taking the point farthest away from $\tilde{\boldsymbol{p}}'_{\mathbf{1}}$. The third point is found by taking the point, whose perpendicular distance to the vector $\tilde{\boldsymbol{p}}'_{\mathbf{2}} - \tilde{\boldsymbol{p}}'_{\mathbf{1}}$ is largest:

$$\tilde{\boldsymbol{p}}'_{\mathbf{3}} = \arg \max_{\tilde{p}'_i} |(\tilde{\boldsymbol{p}}'_{\mathbf{2}} - \tilde{\boldsymbol{p}}'_{\mathbf{1}}) \times (\tilde{\boldsymbol{p}}'_{\boldsymbol{i}} - \tilde{\boldsymbol{p}}'_{\mathbf{1}})| \tag{23}$$

Embed the image points into 3D space at the plane with distance equal to the focal length s_{xy} in pixel. With $c(c_x, c_y)^T$ denoting the principal point, the embedded points are $\tilde{\boldsymbol{p}}_{\boldsymbol{j}} = (\tilde{p}'_{jx} - c_x, \tilde{p}'_{jy} - c_y, s_{xy})^T, j \in \{1, 2, 3\}$. The corresponding 3D points are denoted by $\boldsymbol{p}_{\boldsymbol{j}}, j \in 1, 2, 3$.

C.2 b) Construct Orthonormal Bases

Let $(\boldsymbol{i}_{\mathbf{1}}, \boldsymbol{i}_{\mathbf{2}}, \boldsymbol{i}_{\mathbf{3}})$ be the orthonormal basis for the image points. Then

$$\begin{aligned} \tilde{\boldsymbol{i}}_{\mathbf{1}} &= \tilde{\boldsymbol{p}}_{\mathbf{2}} - \tilde{\boldsymbol{p}}_{\mathbf{1}} \\ \tilde{\boldsymbol{i}}_{\mathbf{3}} &= \tilde{\boldsymbol{i}}_{\mathbf{1}} \times (\tilde{\boldsymbol{p}}_{\mathbf{3}} - \tilde{\boldsymbol{p}}_{\mathbf{1}}) \\ \tilde{\boldsymbol{i}}_{\mathbf{2}} &= \tilde{\boldsymbol{i}}_{\mathbf{3}} \times \tilde{\boldsymbol{i}}_{\mathbf{1}} \end{aligned}$$

and normalization leads to $(\boldsymbol{i}_{\mathbf{1}}, \boldsymbol{i}_{\mathbf{2}}, \boldsymbol{i}_{\mathbf{3}})$. The orthonormal basis $(\boldsymbol{o}_{\mathbf{1}}, \boldsymbol{o}_{\mathbf{2}}, \boldsymbol{o}_{\mathbf{3}})$ for the corresponding 3D object points is calculated in the same way.

C.3 c) Move object into Image Plane

Let $T_i^{4\times4}$ be the transformation whose rotational part is $(\boldsymbol{i}_{\mathbf{1}}, \boldsymbol{i}_{\mathbf{2}}, \boldsymbol{i}_{\mathbf{3}})$ and whose translation is $\tilde{\boldsymbol{p}}_{\mathbf{1}}$. Let $T_o^{4\times4}$ be the transformation of the object basis with rotational part $(\boldsymbol{o}_{\mathbf{1}}, \boldsymbol{o}_{\mathbf{2}}, \boldsymbol{o}_{\mathbf{3}})$ and translational part $\boldsymbol{p}_{\mathbf{1}}$. Then the transformation $T_{oi}^{4\times4}$, that moves the object into the image plane with an already good approximated orientation, is:

$$T_{oi} = T_i \, T_o^{-1} \tag{24}$$

C.4 d) Move Object Away

The orientation is now already calculated. To get an good approximation of the position also, it is necessary to move the object away from the image plane. This is achieved by moving the object along the viewing ray to the first image point $\tilde{\boldsymbol{p}}'_{\mathbf{1}}$, such that two object points project already correctly onto their corresponding image points.

The scaled vector $\tilde{\boldsymbol{p}}_{\mathbf{1}}$ is added to the translational part of T_{oi}. The scale value s is:

$$s = \frac{|T_{oi}(\boldsymbol{p}_{\mathbf{2}}) - T_{oi}(\boldsymbol{p}_{\mathbf{1}})|}{|\tilde{\boldsymbol{p}}_{\mathbf{2}} - \tilde{\boldsymbol{p}}_{\mathbf{1}}|} \tag{25}$$

The use of T_{oi} as a function denotes applying the transform to the homogenized vectors. The final transform T_f is obtained by adding $s\tilde{\boldsymbol{p}_1}$ to the translation part of T_{oi}.

The transform T_f gives the initial pose of the object, which can be used for the Nonlinear Least Squares estimation. If the position of a camera is to be evaluated its initial pose is T_f^{-1}.

D Second Derivatives of Articulated Movement

The Hessian matrix for the nonlinear Least Squares problem requires the second derivatives of $\boldsymbol{m}$, which form the Hessian with $H_{ij} = \frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}$. We take first a closer look at the general form of the first derivative:

$$\frac{\partial m}{\partial \theta_i} = \frac{\partial \left(R_x(\theta_\alpha) \circ \cdot\cdot \circ R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_{i-1})\right)(\boldsymbol{z})}{\partial \boldsymbol{z}}\; R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{p}) \tag{26}$$

where $\boldsymbol{p} = R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_{i+1}) \circ \cdot\cdot \circ R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_p)(\boldsymbol{p})$.
Of specific interest is now the partial of $\boldsymbol{z}$:

$$\frac{\partial \left(R_{x,\theta_\alpha} \circ \cdot\cdot \circ R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z})}{\partial \boldsymbol{z}} \tag{27}$$

Please note that this partial is a matrix itself as $R_{\boldsymbol{\omega},\boldsymbol{q},\theta_p}(\boldsymbol{p}) : \mathbb{R}^3 \to \mathbb{R}^3$. Applying the chain rule to the outermost rotation gives:

$$= \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\; \frac{\partial \left(R_{y,\theta_\beta} \circ \cdot\cdot \circ R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z})}{\partial \boldsymbol{z}} \tag{28}$$

Further application of the chain rule leads to:

$$= \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\; \frac{\partial R_{y,\theta_\beta}(\boldsymbol{z}_\beta)}{\partial \boldsymbol{z}_\beta}\; \cdots \frac{\partial \left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}} \tag{29}$$

As mentioned above, the actual values of $\boldsymbol{z_i}$ do not matter as they vanish in the derivative as we will see now from equation (3.1):

$$\begin{aligned}
\left[\frac{\partial R_{\boldsymbol{\omega},\boldsymbol{q},\theta}(\boldsymbol{p})}{\partial \boldsymbol{p}}\right]_{ij}^{3\times 3} &= \frac{\partial (R_{\boldsymbol{\omega},\boldsymbol{q},\theta}(\boldsymbol{p}))_i}{\partial \boldsymbol{p_j}} \\
&= \frac{\partial \left(\boldsymbol{p} + \sin\theta(\boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{q})) + (1 - \cos\theta)(\boldsymbol{q} - \boldsymbol{p}^p)\right)}{\partial \boldsymbol{p}} \\
&= \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{p}} + \frac{\partial \sin\theta(\boldsymbol{\omega} \times \boldsymbol{p})}{\partial \boldsymbol{p}} - \frac{\partial (1 - \cos\theta)\boldsymbol{p}^p}{\partial \boldsymbol{p}} \\
&= I^{3\times 3} + \sin\theta \frac{\partial(\boldsymbol{\omega} \times \boldsymbol{p})}{\partial \boldsymbol{p}} - (1 - \cos\theta)\frac{\partial(\boldsymbol{p} - (\boldsymbol{p}^{\,T}\boldsymbol{\omega})\boldsymbol{\omega})}{\partial \boldsymbol{p}} \\
&= I^{3\times 3} + \sin\theta[\omega\times]^{3\times 3} - (1 - \cos\theta)I^{3\times 3} + (1 - \cos\theta)\omega\omega^T \\
&= \sin\theta[\omega\times]^{3\times 3} + \cos\theta\; I^{3\times 3} + \omega\omega^T - \cos\theta\; \omega\omega^T
\end{aligned} \tag{30}$$

where $[\omega\times]^{3\times3}$ is the cross product matrix:

$$[\omega\times]^{3\times3} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{31}$$

As visible in equation (30), the partial of equation (27) is independent of z and becomes the identity if $\boldsymbol{\theta}$ equals zero.

Now we can derive the second derivative using equation (29). Assume at first $j < i$:

$$\begin{aligned} \frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j} &= \frac{\partial \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\ \frac{\partial R_{y,\theta_\beta}(\boldsymbol{z}_\beta)}{\partial \boldsymbol{z}_\beta}\ \cdots \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}} R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{p})}{\partial\theta_j} \\ &= \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\ \cdot\cdot \frac{\partial \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_j}\right)(\boldsymbol{z_j})}{\partial \boldsymbol{z_j}}}{\partial\theta_j} \cdot\cdot \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}} R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{p}) \end{aligned} \tag{32}$$

with $\boldsymbol{p} = R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_{i+1}) \circ \cdot\cdot \circ R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_p)(\boldsymbol{p})$. This simplifies for $\boldsymbol{\theta} = \mathbf{0}$ to:

$$\left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}\right|_{\boldsymbol{\theta}=\mathbf{0}} = \left.\frac{\partial \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_j}\right)(\boldsymbol{z_j})}{\partial \boldsymbol{z_j}}}{\partial\theta_j}\right|_{\theta_j=0} (\boldsymbol{\omega}_i \times (\boldsymbol{p} - \boldsymbol{q}_i)) \tag{33}$$

Using the the derivative $\frac{\partial}{\partial\theta}$ of equation (30) and inserting zero gives:

$$j < i: \quad \left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}\right|_{\boldsymbol{\theta}=\mathbf{0}} = [\boldsymbol{\omega_j}\times]^{3\times3}\ (\boldsymbol{\omega}_i \times (\boldsymbol{p} - \boldsymbol{q}_i)) \tag{34}$$

The case where $j = i$ is easily derived from equation (3.2):

$$\begin{aligned} j = i: \quad \frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j} &= \frac{\partial \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\ \frac{\partial R_{y,\theta_\beta}(\boldsymbol{z}_\beta)}{\partial \boldsymbol{z}_\beta}\ \cdots \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}} R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{p})}{\partial\theta_i} \\ &= \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\ \cdot\cdot \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}}\ \frac{\partial R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{p})}{\partial\theta_i} \\ \left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_i}\right|_{\boldsymbol{\theta}=\mathbf{0}} &= \left.\frac{\partial(R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i))(\boldsymbol{p})}{\partial\theta_i}\right|_{\theta_i=0} \\ &= \left.\frac{\partial\left(\cos\theta_i(\boldsymbol{\omega}\times(\boldsymbol{p}-\boldsymbol{q})) + \sin\theta_i(\boldsymbol{q}-\boldsymbol{p}^p)\right)}{\partial\theta_i}\right|_{\theta_i=0} \\ &= \left.(-\sin\theta_i(\boldsymbol{\omega}\times(\boldsymbol{p}-\boldsymbol{q})) + \cos\theta_i(\boldsymbol{q}-\boldsymbol{p}^p))\right|_{\theta_i=0} \\ &= \boldsymbol{q} - \boldsymbol{p}^p \end{aligned} \tag{35}$$

The result $\boldsymbol{q} - \boldsymbol{p}^p$ is equal to equation (34) with $i = j$.

The last case is $j > i$:

$$\begin{aligned}\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j} &= \frac{\partial \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\ \frac{\partial R_{y,\theta_\beta}(\boldsymbol{z}_\beta)}{\partial \boldsymbol{z}_\beta}\ \cdots \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}} R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{p})}{\partial\theta_j} \\ &= \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\ \cdot\cdot \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}}\ \frac{\partial R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{p})}{\partial\theta_j} \\ &= \frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\ \cdot\cdot \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}}\ \frac{\partial R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{z_i})}{\partial \boldsymbol{z_i}}\ \frac{\partial \boldsymbol{p}}{\partial\theta_j}\end{aligned} \tag{36}$$

To simplify this equation we insert zero:

$$\begin{aligned}\left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}\right|_{\boldsymbol{\theta=0}} &= \left(\frac{\partial R_{x,\theta_\alpha}(\boldsymbol{z}_\alpha)}{\partial \boldsymbol{z}_\alpha}\ \cdot\cdot \frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q},\theta_{i-1}}\right)(\boldsymbol{z_{i-1}})}{\partial \boldsymbol{z_{i-1}}}\right)\Bigg|_{\boldsymbol{\theta=0}} \left.\frac{\partial R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i)(\boldsymbol{z_i})}{\partial \boldsymbol{z_i}}\right|_{\theta_i=0} \left.\frac{\partial \boldsymbol{p}}{\partial\theta_j}\right|_{\boldsymbol{\theta=0}} \\ &= I^{3\times3}\ \left.\frac{\partial\left(\cos\theta_i(\boldsymbol{\omega}\times(\boldsymbol{z_i}-\boldsymbol{q}))+\sin\theta_i(\boldsymbol{q}-{\boldsymbol{z_i}}^p)\right)}{\partial \boldsymbol{z_i}}\right|_{\theta_i=0} \left.\frac{\partial \boldsymbol{p}}{\partial\theta_j}\right|_{\boldsymbol{\theta=0}}\end{aligned} \tag{37}$$

Using the derivation in (30) we get:

$$\begin{aligned}\left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}\right|_{\boldsymbol{\theta=0}} &= \left(\cos\theta_i\,[\boldsymbol{\omega_i}\times]^{3\times3} + \sin\theta_i(I^{3\times3} + \boldsymbol{\omega_i}{\boldsymbol{\omega_i}}^T)\right)\big|_{\theta_i=0} \left.\frac{\partial \boldsymbol{p}}{\partial\theta_j}\right|_{\boldsymbol{\theta=0}} \\ &= [\boldsymbol{\omega_i}\times]^{3\times3}\ \left.\frac{\partial \boldsymbol{p}}{\partial\theta_j}\right|_{\boldsymbol{\theta=0}} \\ &= [\boldsymbol{\omega_i}\times]^{3\times3}\ \left.\frac{\partial R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_{i+1})\circ\cdot\cdot R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_j)\circ\cdot\cdot\circ R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_p)(\boldsymbol{p})}{\partial\theta_j}\right|_{\boldsymbol{\theta=0}}\end{aligned} \tag{38}$$

The derivative on the right side is already known from equation (26) and can be splitted in the same way:

$$\left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}\right|_{\boldsymbol{\theta=0}} = [\boldsymbol{\omega_i}\times]^{3\times3}\ \left.\frac{\partial\left(R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_{i+1})\circ\cdot\cdot\circ R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_{j-1})\right)(\boldsymbol{z})}{\partial \boldsymbol{z}}\ R'_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_j)(\boldsymbol{p_j})\right|_{\boldsymbol{\theta=0}} \tag{39}$$

where $\boldsymbol{p_j} = R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_{j+1})\circ\cdot\cdot\circ R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_p)(\boldsymbol{p})$. Inserting zero we get

$$j > i: \quad \left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}\right|_{\boldsymbol{\theta=0}} = [\boldsymbol{\omega_i}\times]^{3\times3}\ (\boldsymbol{\omega}_j\times(\boldsymbol{p}-\boldsymbol{q}_j)) \tag{40}$$

Summary

Summarizing the results of equations (34),(35) and (40) we get finally for $i, j \in \{\alpha, \beta, \gamma, 1, .., p\}$:

$$\begin{aligned} j \le i: \quad & \left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}\right|_{\boldsymbol{\theta=0}} = [\boldsymbol{\omega_j}\times]^{3\times3}\ (\boldsymbol{\omega}_i\times(\boldsymbol{p}-\boldsymbol{q}_i)) \\ j > i: \quad & \left.\frac{\partial^2 m}{\partial\theta_i\,\partial\theta_j}\right|_{\boldsymbol{\theta=0}} = [\boldsymbol{\omega_i}\times]^{3\times3}\ (\boldsymbol{\omega}_j\times(\boldsymbol{p}-\boldsymbol{q}_j)) \end{aligned} \tag{41}$$

The axis that is nearer in the chain to the point $\boldsymbol{p}$ gives the first derivative, while the axis further away gives the second derivative.

The remaining second derivatives for the translational part of m are:

$$\frac{\partial^2 m}{\partial \theta_{x,y,z}\, \partial \theta_i} = \mathbf{0} \tag{42}$$

and as visible from equation (26):

$$\frac{\partial^2 m}{\partial \theta_i\, \partial \theta_{x,y,z}} = \mathbf{0} \tag{43}$$

with $i \in \{\alpha, \beta, \gamma, 1, .., p\}$.

Bibliography

[1] ORIGAMI - EU project on Mixed Reality. www-dsp.elet.polimi.it/origami, 2004.

[2] ARTESAS, Advanced Augmented Reality Technologies for Industrial Service Applications. www.artesas.de, 2004-2006.

[3] MATRIS, Markerless real-time Tracking for Augmented Reality Image. www.ist-matris.org, 2004-2007.

[4] ISO/IEC 14496. Part 2: Generic coding of audio-visual objects, 1999.

[5] T. Gentils A. Hilton and D. Beresford. Popup-People: Capturing 3D Articulated Models of Individual People. In *In IEE Colloquim on Computer Vision for Virtual Human Modelling*, pages 1–6, 1998.

[6] J. K. Aggarwal and Q. Cai. Human Motion Analysis: A Review. *Journal of Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[7] H. Araujo, R. Carceroni, and C. Brown. A Fully Projective Formulation to Improve the Accuracy of Lowe's Pose Estimation Algorithm. *Journal of Computer Vision and Image Understanding*, 70(2), 1998.

[8] S. Baugher and A. Rosenfeld. *Computer Vision and Image Processing, Eds. L. Shapiro and A. Rosenfeld*, chapter Corner Detection and Localization in a pyramid, pages 103–122. ACADEMICPRESS Inc., San Diego, USA, 1992.

[9] G. Bazzoni, E. Bianchi, O. Grau, A. Knox, R. Koch, F. Lavagetto, A. Parkinson, F. Pedersini, A. Sarti, G. Thomas, and S. Tubaro. The ORIGAMI Project – Advanced Tools and Techniques for High-End Mixing and Interaction between Real and Virtual Content. In *IEEE Proceedings of 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT)*, 2002.

[10] M. Bergtholdt, J. H. Kappes, and C. Schnörr. Learning of Graphical Models and Efficient Inference for Object Class Recognition. In *Proc. of Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, pages 273–283, 2006.

[11] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 92.

[12] S. Birchfeld. KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker. www.ces.clemson.edu/~stb/klt, 1998.

[13] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[14] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] A. 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, ISBN-13: 978-3540762171, 2000.

[16] M. Bray, E. Koller-Meier, P. Mueller, L. Van Gool, and N. N. Schraudolph. 3D Hand Tracking by Rapid Stochastic Gradient Descent Using a Skinning Model. In *1st European Conference on Visual Media Production*, pages 59–68. IEE, March 2004.

[17] C. Bregler and J. Malik. Tracking People with Twists and Exponential Maps . In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 8–15, 1998.

[18] Christoph Bregler, Jitendra Malik, and Katherine Pullen. Twist Based Acquisition and Tracking of Animal and Human Kinematics. *Int. J. of Computer Vision*, 56(3):179–194, 2004.

[19] Roman Calow, Bernd Michaelis, and Ayoub Al-Hamadi. Solutions for Model-Based Analysis of Human Gait. In B. Michaelis, editor, *Proc. of DAGM-Symposium*, pages 540–547, Magdeburg, 2003.

[20] Rikk Carey, Gavin Bell, and Chris Marrin. ISO/IEC 14772-1:1997 Virtual Reality Modeling Language (VRML97) , 1997.

[21] J. Chandaria, G. Thomas, B. Bartczak, K. Koeser, R. Koch, M. Becker, G. Bleser, D. Stricker, C. Wohlleber, M. Felsberg, F. Gustafsson, J. Hol, T.B. Schön, J. Skoglund, P.J. Slycke, and S. Smeitz. Real-Time Camera Tracking in the MATRIS project. In *Proc. of International Broadcasting Convention(IBC)*, pages 321–328, Amsterdam, NL, 2006.

[22] Fangxiang Cheng, W.J. Christmas, and J. Kittler. Recognising Human Running Behaviour in Sports Video Sequences. *International Conference on Pattern Recognition*, 2:1017– 1020, 2002.

[23] G. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette for articulated objects and its use for human body kinematics estimation and motion capture. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, Madison, Wisconsin, USA, 2003.

[24] Edwin K.P. Chong and Stanislaw H. Zak. *An Introduction to Optimization, Second Edition*. Wiley, 2001.

[25] Wayne R. Cowell. MINPACK: Numerical Library for Function Minimization and Least-Squares Solutions. www.netlib.org/minpack.

[26] J. J. Craig. *Introduction to Robotics - Mechanics and Control*. Pearson, Prentice Hall, third edition, 2005.

[27] Open Source 3D Graphics Creation. Blender: www.blender.org.

[28] Carolina Cruz-Neira, D.J. Sandin, and T.A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *Proc. of SIGGRAPH*, pages 135–142. ACM SIGGRPAH / Addison Wesley, 1993.

[29] P. David, D. Dementhon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous Pose and Correspondence Determination. *Int. J. Comput. Vision*, 59(3):259–284, 2004.

[30] E. de Aguiar, C. Theobalt, M. Magnor, and H.-P. Seidel. Reconstructing Human Shape and Motion from Multi-View Video. In *2nd European Conference on Visual Media Production (CVMP)*, London, UK, 2005.

[31] P. de Groen. An Introduction to Total Least Squares. *Nieuw Archief voor Wiskunde*, 4(14):237–257, 1996.

[32] D.F. DeMenthon and L.S. Davis. Model-Based Object Pose in 25 Lines of Code. *International Journal of Computer Vision*, 15:335–343, 1995.

[33] D. Demirdjian. Combining Geometric and View-Based Approaches for Articulated Pose Estimation. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 183–194, Prague, Czech Republic, May 2004.

[34] D. Demirdjian, T. Ko, and T. Darrell. Constraining Human Body Tracking. In *Proc. of Internatinal Conference on Computer Vision (ICCV)*, Nice, France, October 2003.

[35] D. Demirjian. EStereo: Real-Time C++ Disparity Estimation. https://sourceforge.net/projects/estereo.

[36] J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2126–2133, 2000.

[37] A. Doucet, S. Godsill, and C. Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 10:197–209, 2000.

[38] Tom Drummond and Roberto Cipolla. Real-time Tracking of Highly Articulated Structures in the Presence of Noisy Measurements. In *Proc. of Internatinal Conference on Computer Vision (ICCV)*, pages 315–320, Vancouver, Canada, 2001.

[39] H.-L. Eng, K.-A. Toh, A.H. Kam, J. Wang, and W.-Y. Yau. An automatic drowning detection surveillance system for challenging outdoor pool environments. *Proc. of Internatinal Conference on Computer Vision (ICCV)*, page 532, 2003.

[40] Lars Mündermann et al. Validation Of A Markerless Motion Capture System For The Calculation Of Lower Extremity Kinematics. In *Proc. American Society of Biomechanics*, Cleveland, USA, 2005.

[41] J.-F. Evers-Senne, J.-M. Frahm, D. Grest, K. Köser, B. Streckel, J. Woetzel, and J.-F. Woelk. Basic Image AlgorithmS (BIAS) open-source-library, C++. www.mip.informatik.uni-kiel.de/Software/software.html, 2006.

[42] Eyetronics. www.eyetronics.com.

[43] L. Falkenhagen. Depth Estimation from Stereoscopic Image Pairs assuming Piecewise Continuous Surfaces . In *Proc. of European Workshop on combined Real and Synthetic Image Processing for Broadcast and Video Production*, Hamburg, Germany, November 1994.

[44] L. Falkenhagen. Hierarchical block-based disparity estimation considering neighbourhood constraints. In *Proc. of International Workshop on SNHC and 3D Imaging*, 1997.

[45] O. Faugeras. *Three-Dimensional Computer Vision, a Geometric Viewpoint*. Mit Press, Cambridge, 1993.

[46] J.-F. Frahm. *Camera Self-Calibration with Known Camera Orientation*. PhD thesis, University Kiel, MIP, Prof. Koch, 2005.

[47] J. Gallier. *Geometric Methods and Applications for Computer Science and Engineering*. Springer Verlag, New York Inc., 2001.

[48] S. Goldenstein, C. Vogler, and D. Metaxas. Statistical Cue Integration in DAG Deformable Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):801–803, 2003.

[49] J. J. Gonzalez, I. S. Lim, P. Fua, and D. Thalmann. Robust Tracking and Segmentation of Human Motion in an Image Sequence. In *ICASSP*, volume 3, pages 29–32, Hong Kong, 2003.

[50] S. Grange, E. Casanova, T. Fong, and C. Baur. Vision-based Sensor Fusion for Human-Computer Interaction. In *Proc. of Int. Conf. on Intelligent Robots and System,*, volume 2, pages 1120 – 1125, 2002.

[51] D. Grest, D. Herzog, and R. Koch. Human Model Fitting from Monocular Posture Images. In *Proc. of Conf. on Vision, Modeling and Visualization (VMV)*, Nov. 2005.

[52] D. Grest, Dennis Herzog, and R. Koch. Monocular Body Pose Estimation by Color Histograms and Point Tracking. In *Proc. of Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, Berlin, Sept. 2006. Springer, LNCS.

[53] D. Grest and R. Koch. Multi-Camera Person Tracking in a Cluttered Interaction Environment. In *Proc. of International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 480–488, Paris, France, 2005. Springer Berlin / Heidelberg.

[54] D. Grest, J. Woetzel, and R. Koch. Nonlinear Body Pose Estimation from Depth Images. In *Proc. of Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, volume 3663, pages 285–292, Vienna, Sept. 2005. Springer Berlin / Heidelberg.

[55] Daniel Grest, Jan-Michael Frahm, and Reinhard Koch. A Color Similarity Measure for Robust Shadow Removal in Real Time. In *Proc. of Vision, Modeling and Visualization (VMV)*, pages 253–260, Munich, Germany, Nov. 2003.

[56] Daniel Grest and Reinhard Koch. Realtime Multi-Camera Person Tracking for Immersive Environments. In *Proc. of International Workshop on Multimedia Signal Processing*, pages 387– 390, Siena, Italy, Oct. 2004. IEEE.

[57] MPEG Moving Pictures Experts Group. Mpeg home page. www.chiariglione.org/mpeg, 2006.

[58] I. Haritaoglu, D. Harwood, and L. S. David. W4: Real-Time Surveillance of People and Their Activities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):809–830, 2000.

[59] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[60] L. Herda, R. Urtasun, and P. Fua. Hierarchical implicit surface joint limits for human body tracking. *Journal of Computer Vision and Image Understanding*, 99(2):189–209, 2005.

[61] Lorna Herda. *Using Biomechanical Constraints to Improve Video-Based Motion Capture*. PhD thesis, EPFL, Lausanne, Switzerland, 2003.

[62] D. Hestenes. Reforming the mathematical language of physics. *American Journal of Physics*, pages 104–121, 2003.

[63] T. Horprasert, I. Haritaoglu, and D. Harwood. Real-Time 3D Motion Capture. In *Proc. of Perceptual User Interfaces*, pages 87–90, Nov. 1998.

[64] Weiming Hu, Tieniu Tan, Liang Wang, and S. Maybank. A Survey on Visual Surveillance of Object Motion and Behaviors. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 34(3):334–352, 2004.

[65] Intel. openCV: Open Source Computer Vision Library. www.intel.com/technology/computing/opencv/index.htm, 2006.

[66] M. Isard and A. Blake. CONDENSATION – Conditional Density Propagation for Visual Tracking. *International Journal of Compputer Vision*, 29:5–28, 1998.

[67] R. Stiefelhagen J. Ziegler, K. Nickel. Tracking of the Articulated Upper Body on Multi-View Stereo Image Sequences. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 774–781, New York, June 2006. IEEE Computer Society.

[68] B. Jähne. *Digital Image Processing* . Springer Verlag 6th Edition, 1997.

[69] O. C. Jenkins, G. Gonzalez, and M. M. Loper. Tracking Human Motion and Actions for Interactive Robots. In *HRI '07: Proceeding of the ACM/IEEE International Conference on Human-Robot Interaction*, pages 365–372, New York, NY, USA, 2007. ACM Press.

[70] A. Jepson and M.J. Black. Mixture Models for Optical Flow Computation. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 760–761, 1993.

[71] J. Matas K. Zimmermann, T. Svoboda. Multiview 3D Tracking with an Incrementally Constructed 3D Model. In K. Daniilidis M. Pollefeys, editor, *Proc. of Third International Symposium on 3D Data Processing, Visualization and Transmission 2006*, pages 488 – 495. IEEE Computer Society Press, 2006.

[72] I. Kakadiaris and D. Metaxas. Model-Based Estimation of 3D Human Motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 22(12):1453 – 1459, 2000.

[73] Roland Kehl, Matthieu Bray, and Luc J. Van Gool. Full Body Tracking from Multiple Views Using Stochastic Sampling. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 129–136, 2005.

[74] S. Kiesler and P.J. Hinds, editors. *Robot-Human Interaction, Special issue of Journal on Human Computer Interaction*, volume 19. Lawrence Erlbaum Associates, 2004.

[75] R. Koch. Dynamic 3-D Scene Analysis through Synthesis Feedback Control. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(6):556–568, 1993.

[76] R. Koch. *Automatische Oberflächenmodellierung starrer dreidimensionaler Objekte aus stereoskopischen Rundum-Ansichten*. PhD thesis, Universität Hannover, Germany, 1996.

[77] R. Koch and J-F. Evers-Senne. Chapter: View Synthesis and Rendering Methods. In *3D Video Communications*. Wiley, 2005.

[78] M. Lee, I. Cohen, and S. Jung. Particle Filter with Analytical Inference for Human Body Tracking. In *Proc. of the IEEE Workshop on Motion and Video Computing*, page 159, 2002.

[79] Vincent Lepetit and Pascal Fua. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–104, 2005.

[80] Matt Liverman. *The Animator's Motion Capture Guide: Organizing, Managing,Editing*. Charles River Media, ISBN-13: 978-1584502913, 2004.

[81] David G. Lowe. Fitting Parameterized Three-Dimensional Models to Images. In *Trans. on Pattern Analysis and Machine Intelligence*, pages 13(5):441–450, 1991.

[82] Jason Luck, Dan Small, and Charles Q. Little. Real-Time Tracking of Articulated Human Models Using a 3D Shape-from-Silhouette Method. In *RobVis '01: Proceedings of the International Workshop on Robot Vision*, pages 19–26, London, UK, 2001. Springer-Verlag.

[83] D. MacKay. *Learning in Graphical Models, M.Jordan (ed.)*, chapter Introduction to Monte Carlo Methods, pages 175–204. MIT Press, 1999.

[84] N. Magnenat-Thalmann, R. Laperriere, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface '88*, pages 26–33, Toronto, Ont., Canada, Canada, 1988. Canadian Information Processing Society.

[85] MAXON. Cinema 4D 3D Modeling and Animation Software. www.maxon.net.

[86] Autodesk Maya. www.autodesk.com/maya, 2006.

[87] M. Meredith and S. Maddock. Inverse skinning. In *In Proc. of Conf. on Visual Media Production (CVMP)*, pages 163–172, 2006.

[88] MetaMotion. Electro mechanical motion capture products. www.metamotion.com/motion-capture/electro-mechanical-motion-capture.htm.

[89] B. Michoud, E. Guillou, H. Briceno, and S. Bouakaz. Real-time marker-free motion capture from multiple cameras. In *Proc. of Internatinal Conference on Computer Vision (ICCV)*, pages 1–7, Brazil, 2007.

[90] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. *F.J. Perales, E.R. Hancock (Eds.), AMDO*, chapter Human Body model acquisition and motion capture using voxel data. Springer, 2002.

[91] T. Moeslund, A. Hilton, and V. Krueger. A Survey of Advances in Vision-based Human Motion Capture and Analysis. *Journal of Computer Vision and Image Understanding*, 104(2-3):90–127, 2006.

[92] T. B. Moeslund and E. Granum. A Survey of Computer Vision-Based Human Motion Capture. *Journal of Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[93] A. Mohr and M. Gleicher. Building Efficient, Accurate Character Skins from Examples. In *ACM SIGGRAPH*, pages 562–568, 2003.

[94] T. Möller, H. Kraft, J. Frey, M. Albrecht, and R. Lange. Robust 3D Measurement with PMD Sensors. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 2005.

[95] Sales Motion Capture Distribution and Consulting. www.metamotion.com.

[96] MPEG-4 Specification. Wikipedia. WebPage: en.wikipedia.org/wiki/Mpeg4, 2006.

[97] L. Mündermann. Chapter "Markerless Motion Capture for Biomechanical Applications " in Dagstuhl Seminar Proceedings 06241: Human Motion - Understanding, Modeling, Capture and Animation. 13th Workshop. Springer, Eds. B. Rosenhahn and D. Metaxas, 2006.

[98] R.M. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[99] V.I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[100] P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-Based Probabilistic Tracking. In A. Heyden et al., editor, *Proc. of European Conference on Computer Vision (ECCV)*, LNCS 2350, pages 661–675, 2002.

[101] C. Perwass. Geometric Algebra with Applications in Engineering. Habilitation at Cognitive Systems Group, University Kiel, Prof. Sommer, 2007.

[102] R. Plänkers and P. Fua. Model-Based Silhouette Extraction for Accurate People Tracking. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 325–339. Springer-Verlag, 2002.

[103] R. Plänkers and P. Fua. Articulated Soft Objects for Video-based Human Motion Capture. In *Proc. of Internatinal Conference on Computer Vision (ICCV)*, pages 394–401, Vancouver, Canada, 2001.

[104] E. Polat, M. Yeasin, and R. Sharma. Robust tracking of human body parts for collaborative human computer interaction. *Computer Vision and Image Understanding*, 89:44–69, January 2003.

[105] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C online*. Cambridge University Press, 1986. www.nr.com.

[106] Chee Kwang Quah, Andre Gagalowicz, Richard Roussel, and Hock Soon Sah. 3D Modeling of Humans with Skeletons from Uncalibrated Wide Baseline Views. In *Proc. of International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 379–389, Paris, 2005. Springer.

[107] Point Grey Reesearch. Web Page of Point Grey Research: Digital Camera Technology. www.ptgrey.com, 2007.

[108] Dirk Reiners, Gerit Voss, Marcus Roth, and al. OpenSceneGraph library (OpenSG). www.opensg.org.

[109] B. Rosenhahn. *Pose Estimation Revisited*. PhD thesis, Cognitive Systems, University Kiel, Prof. Sommer, 2003.

[110] B. Rosenhahn, L. He, and R. Klette. Automatic Human Model Generation. In *Proc. of International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 41–48, 2005.

[111] B. Rosenhahn, U. Kersting, D. Smith, J. Gurney, T. Brox, and R. Klette. A System for Marker-Less Human Motion Estimation . In W. Kropatsch, editor, *Proc. of Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, pages 230–237, Wien, Austria, Sept. 2005. Springer.

[112] B. Rosenhahn and G. Sommer. Adaptive Pose Estimation for Different Corresponding Entities. In *Proc. of Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, pages 265–273. Springer-Verlag, 2002.

[113] B. Rosenhahn and G. Sommer. Pose Estimation of Free-form Objects. . In *Proc. of European Conference on Computer Vision (ECCV)*, pages 414–427, 2004.

[114] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings of IEEE Workshop on Stereo and Multi-Baseline Vision*, Kauai, HI, December 2001.

[115] M. Schünke. *Funktionelle Anatomie - Topographie und Funktion des Bewegungssystems*. Thieme, Stuttgart, 2000. ISBN: 978-3131185716.

[116] J. Shi and C. Tomasi. Good features to track. In *Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, June 1994. IEEE.

[117] N.T. Siebel and S. Maybank. Fusion of Multiple Tracking Algorithms for Robust People Tracking. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 373–, Copenhagen, 2002.

[118] Pierre Soille. *Morphological Image Analysis, Principles and Applications, 2nd Edition*, pages 47–48. Springer Verlag, 2002.

[119] A. Sundaresan and R. Chellappa. *Computer Vision for Interactive and Intelligent Environments, Eds. C. Jaynes and R. Collins*, chapter Markerless Motion Capture using Multiple Cameras. IEEE Press, 2006.

[120] Vicon Motion Systems and Peak Performance Inc. www.vicon.com, 2006.

[121] Ascension Technologies. www.ascension-tech.com/products/motionstarwireless.php, 2006.

[122] SICK Sensor Technologies. www.sickusa.com, 2006.

[123] C. Theobalt. Chapter "Video-based Capturing and Rendering of People" in Dagstuhl Seminar Proceedings 06241: Human Motion - Understanding, Modeling, Capture and Animation. 13th Workshop. Springer, Eds. B. Rosenhahn and D. Metaxas, 2006.

[124] C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, and H.-P. Seidel. Seeing people in different light: Joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics*, 13:663–674, 2007.

[125] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, Carnegie Mellon University, Pittsburg, PA, 1991.

[126] Videre. Videre Design: Stereo On A Chip. www.videredesign.com, 2007.

[127] B. Wettegren, L.B. Christensen, B. Rosenhahn, O. Granert, and N. Krüger. Image Uncertainty and Pose Estimation in 3D Euclidian Space. *Proceedings DSAGM*, pages 76–84, 2005.

[128] Jochen Wingbermühle, Claus-E. Liedtke, and Juri Solodenko. Automated Acquisition of Lifelike 3D Human Models from Multiple Posture Data. In *Proc. of International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 400–409, 2001.

[129] F. Woelk, I. Schiller, and R. Koch. An Airborne Bayesian Color Tracking System. In *In Proc. of Intelligent Vehicles Symposium*, pages 67 – 72, Las Vegas, USA, June 2005.

[130] J. Woetzel and R. Koch. Real-time Multi-stereo Depth Estimation on GPU with Approximative Discontinuity Handling. In *CVMP 2004, London, UK*, pages 245–254, March 2004.

[131] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. PFinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[132] S. J. Wright. On the convergence of the Newton/log-barrier method. *Mathematical Programming*, 90(1):71–100, 2001.

[133] X3D - Web 3D Consortium. www.web3d.org/x3d, 2006.

[134] Ruigang Yang, Greg Welch, and Gary Bishop. Real-time consensus-based scene reconstruction using commodity graphics hardware. pages 207–214, Beijing, China, October 2002.

[135] Z. Zhang. Parameter Estimation Techniques:A Tutorial with Application to Conic Fitting. *Image and Vision Computing Journal*, 15(1):59–76, 1997.

[136] Z. Zhang. Parameter estimation techniques:a tutorial with application to conic fitting. http://www-sop.inria.fr/robotvis/personnel/zzhang/Publis/Tutorial-Estim/Main.html, 1997.

[137] Z. Zhang. Flexible Camera Calibration By Viewing a Plane From Unknown Orientations. In *Proc. of Internatinal Conference on Computer Vision (ICCV)*, pages 666–673, Corfu, Greece, 1999.

[138] V. B. Zordan, A. Majkowska, B. Chiu, and M. Fast. Dynamic response for motion capture animation. In *SIGGRAPH '05: ACM SIGGRAPH 2005*, pages 697–701, New York, NY, USA, 2005. ACM Press.