Marker-less 3D Pose Estimation

Jeffrey Resodikromo

MSc Thesis

August 2012

ICA - 3253481

Noldus
Information Technology

Universiteit Utrecht

Dept. of Information and Computing Sciences
Utrecht University
Utrecht, the Netherlands

University Supervisor:
dr. R.T. Tan
Company Supervisor:
dr. ir. N.P. van der Aa

Contents

Abstract

This thesis presents the research done on a marker-less pose estimation method. The method by Gall *et al.* has been implemented, analysed and improved for use by Noldus IT. The method consists of a combination of two optimization approaches, namely a local and a global optimization. For the local optimization a Weighted Least Squares problem is solved to minimize the distance between the model and the silhouettes. The global optimization uses a combination of a particle filter and simulated annealing, called the Interacting Simulated Annealing, to estimate the pose where the local optimization fails. Given the mesh model, camera calibration data and the camera video sequences an accurate 3D pose is estimated. A full analysis is performed on the method and based on the experimental results, improvements were made to increase the quality of the estimation. A successful dynamic threshold was added to create a seamless connection between the two optimization approaches. Adding skeleton correspondences to the local optimization taking the inner structure of the model into account, increases the quality of the estimation. Our implementation provides an efficient and accurate pose estimation for use in practical applications in a controlled environment and gives the basis for extensions.

Acknowledgements

With this thesis I completed my master at the Utrecht University. This report describes my master thesis on behalf of Noldus Information Technology B.V., located in Wageningen, the Netherlands. Since this thesis was accomplished as a collaboration between Noldus IT and Utrecht University, both the practical use of marker-less based pose estimation as scientific research is made on this topic.

Notations

To avoid ambiguities in the notation of the mathematical definitions, we used to following conventions. Normal lower-case Roman letters denote scalar variables, such as s. Vectors are denoted by a bold lower-case Roman letter, for example $\mathbf{v}$, and matrices are denoted by a bold upper-case Roman letter, as in $\mathbf{M}$. There is one special case for which the previous notation does not apply. To keep the thesis readable, the skew-symmetric matrix of a vector, as defined in 3.5, is not given by an upper-case bold Roman letter, but by a special notation of a vector. This special notation is given by a lower-case bold Roman letter with a hat above it, so the skew-symmetric matrix for a vector $\mathbf{v}$ is denoted by $\hat{\mathbf{v}}$. A set is represented by an upper-case calligraphy letter, $\mathcal{C}$. An element of a set is given by the type of the element and a subscript, for example for a set of vectors $\mathcal{V}$, an element of this set is $\mathbf{v}_i \in \mathcal{V}$.

Chapter 1

Introduction

The research done in this thesis is based on the method proposed by Gall *et al.* [1] to examine the use of marker-less 3D pose estimation for general purposes for Noldus IT. Marker-less pose estimation is a widely studied topic with applications in many areas, such as computer graphics, medical science, sport science, game development, activity recognition and animation. While marker based systems are already widely used, for example in sports, film making, video games and medical applications, marker-less based systems are still an open issue. Figure 1.1 shows some examples of marker-based applications. A marker-less system has to be robust, accurate and has a low computational cost. Besides these requirements, the system should provide a general solution in all cases, without imposing prior assumptions on the motion, person and background. The only provided data for the marker-less system is given by the cameras. Active nor passive markers are used and only vision based methods are used to determine the motion.

(a) (b) (c)

Figure 1.1: Few examples of marker-based pose estimation. a) motion of a person transfered to a 3D skeleton model, b) scene from the Lord of the Rings, c) tracking the motion of a walking person.

1.1 Motivation

Estimating the pose of a person is important in studies where actions of a person have to be tracked. A pose is defined by the configuration of all limbs of the human body in a given scene. From the pose, the action and gestures of the person can be determined. One of the research projects carried out by Noldus IT is to research methods which are capable of automatically analyse behaviour of humans in the Restaurant of the Future [1], located in Wageningen, the Netherlands. In this restaurant, research is carried out to study the behaviour of people during their lunch or dinner. This restaurant also acts as a sort of canteen, in which people have to select the food of their choice, pay at a counter and consume their bought meal at a table, see Figure 1.2 for two pictures taken in the Restaurant of the Future. The research is aimed at studying the behaviour of people in a restaurant. Some of these behaviours include the food selection behaviour in different environments, the route of persons in the restaurant, social interaction between humans during lunch or diner and eating behaviour for different types of food.

To study the behaviours, the restaurant provides the researcher with cameras placed throughout the entire restaurant. From a central point all videos are recorded and it is possible to move the cameras, in the sense of rotation, tilt and zoom. Using these videos, humans can determine the actions of persons in the videos, for example if a person looks at a meal, but does not choose it or the time taken to finish a meal. In its current state, the annotation of actions is all done manually. This can be a tedious job if this has to be done for multiple cameras over a long sequence. To provide an automatic system, capable of doing the annotation automatically, the pose of the person becomes an important tool. From the pose, different kinds of behaviours and actions can be extracted.

For the research carried out at the restaurant, it is vital that the persons who are investigated are not interrupted by any means and could behave in a normal way. For this purpose, the whole restaurant looks like a normal restaurant and no invasive techniques

[1] http://www.restaurantvandetoekomst.wur.nl/NL/

(a) (b)

Figure 1.2: Impression of the Restaurant of the Future. a) shows the restaurant, b) shows the food shelves.

should be used. Therefore, a motion capture system based on markers is not suitable, as each person should be equipped with markers at the entrance of the restaurant. This would cause delays at the entrance and the investigated persons can have an uncomfortable feeling due to the markers. Furthermore, skin-tight garments are required to ensure a rigid movement of the markers. A non-invasive way is to use a marker-less based system.

The Restaurant of the Future is only one application for a marker-less based system. Since Noldus IT provides software to study the behaviour of people in many applications, they are also interested in using this system for general purposes in which non-invasive techniques are recommended. Provided the pose of a human, other techniques such as action recognition or gesture recognition can be applied to see what the person is doing. One of the research topics at Noldus IT is counting the number of bites a person takes from his or her meal. This can be extracted from the pose of the person. Other applications of pose estimation are determining the mood of a person. From the gestures and the pose it can be determined whether the person is angry, sad or happy. It is even possible to determine whether the person is tired or energetic. Another application is to locate different body parts, like the hands which are used in for example gesture recognition. The pose estimation provides the pose of the entire body and as such it gives the location of the hands. This thesis describes the research, on behalf of Noldus Information Technology B.V., of a method to provide a marker-less pose estimation, without any restrictions on the appearance, which can be used for a wide variety of applications. The main focus will be lying on general applicability in a controlled environment.

1.2 Approach

Estimating the pose of a person through videos can be done in several ways. One of the techniques is the model-based pose estimation techniques in which a bone hierarchy represents the skeleton and the mesh is formed by shapes or a set of vertices. The skeleton and mesh are deformed such that fit the pose of a person seen in an image. We will present, analyse and extend the method presented by Gall *et al.* [1], which is a model-based pose estimation technique. The method presented in this paper estimates the pose from multi-view video recordings, by fitting a 3D model of the observed person to the images from the video sequences. It accurately finds the skeleton pose and from the skeleton pose the mesh deformation is calculated. This is done by a three step algorithm, in which the first two steps estimate motion of the skeleton and the shape. The last step enhances the shape of the model to match the fine details of the observed person in the images. An example of fine details are the loose fitting of clothes, such as a robe or a skirt and flowing garment motions. For our applications we are not really interested in the fine garment motions, but only in the skeleton pose. As such, we will omit the last step.

This method on it self can not be used on videos with multiple persons, but only on a single person. Despite the fact that multiple persons appear in the Restaurant of the Future, we will focus our research on single persons only. If we are able to robustly

and accurately find the pose of a single person, extensions can be made to apply it to multiple persons as in Liu *et al.* [2]. Since this method runs fully automatic without requiring manual interventions, it is suitable in cases, where large sequences have to be processed. This method claims to work on low quality sequences, is less sensitive to silhouette noise compared with other approaches and gives accurate results even under some occlusions. For sequences taken in a real environment, the background subtraction will give silhouette noise, objects could occlude the observed person and the quality will not be high. Before this method can be tested on this kind of sequences, an analysis has to be made on this method first. For this purpose, we will use the Multimodal Motion Capture Indoor Dataset (MPI08) to analyse the method. Given the results of this analysis, improvements on the method are made and analysed.

1.3 Contributions

This thesis aims to develop a system capable of estimating the pose. A full analysis on the method is given, based on the method presented by Gall *et al.* [1]. We implemented the method, with the exception of the surface estimation, in C++ and make use of the OpenCV library, which provides us with optimized computer vision based algorithms. Specifically, the contributions of this thesis consist of:

- The development of a system that could automatically estimate the pose of an observed person from multiple views.
- Analysis and evaluation of the local optimization, global optimization and a combination of these two.
- An extension on the limb evaluation, to provide a correct connection between the local optimization and the initialization of the global optimization by using a dynamic threshold based on the size of each limbs.
- Another extension in which an additional correspondence measure is used for the local optimization to increase the quality of the estimation. This correspondence uses the skeleton of the model, such that the inner structure of the model is taken into account when the pose is estimated.
- A list of improvements and future research that can be made to this method, which includes increasing the quality of the estimation, but also to make the method more applicable for Noldus IT.

1.4 Layout

This thesis document has the following structure. We will first state work previously done by others in Chapter 2. Next, we will provide some background information on the basics of kinematics in Chapter 3. In Chapter 4, a method to determine the whole

pose of a person is explained in detail. Experiments are carried out on this method and a detailed analysis of the results is given in Chapter 5. An analysis is given on the efficiency of the method in Chapter 6 Finally, in Chapter 7, conclusions are provided along with future work.

Chapter 2

Related work

Over the past few years, the amount of literature on motion capture has grown rapidly and it is still an active field. Technological advances make hardware for creating a motion capture lab cheaper, but at the same time, the specifications of the hardware itself have increased. This made applications for using motion capturing very interesting. As such, the interest in this field has grown very rapidly.

Commercial systems, such as OptiTrack[1] and PhaseSpace[2], make use of sensors to estimate the locations of joints and their orientations. From this data, the motion of the person can be determined. This kind of tracking is straightforward and doesn't require much processing to estimate poses. Marker-less motion capture on the other hand, is one of the most challenging problems in computer vision since the pose is extracted from videos only, without the use of markers.

2.1 Overview

Most methods can be classified into two classes, the generative and the discriminative approaches [3]. The generative, or also called the model-based approaches, require an a priori model of the tracked object or human body. These approaches can be split up into two methods: the global optimization and the local optimization approach, each with its own advantages. Where the local optimization is capable of estimating the pose fast, it is prone to errors as it only seeks locally for solutions. An error is not detected by the optimization and will propagate through the sequence. On the other hand, global optimization methods are able to estimate the pose correctly and are able to recover from errors. However, since it searches globally, it requires a lot more computation time to estimate the pose.

The discriminative approaches do not require an a priori model. Instead, a direct relationship is created between the pose and the image observations. Discriminative approaches can be split up further into two main classes, the learning-based and the example-based approaches. Learning-based approaches learn a function which maps

[1] http://www.naturalpoint.com/optitrack/

[2] http://www.phasespace.com/impulse_motion_capture.html

from image space to pose space using training data. Example-based approaches do not learn the mapping; instead a large database containing different poses is used. Given an image, a similarity check is performed on each pose in the database and an interpolation between the candidate poses gives the resulting pose.

Since we require that the pose estimation works in general cases, learning-based approaches are less appropriate. As these approaches estimate poses using a dataset, a pose not contained in the dataset can not be estimated correctly. Another issue is that people differ in size, gender and appearance. To be able to capture motion for each individual, the dataset has to contain a lot of different poses for different persons. Therefore, we will only provide an overview on methods based on generative methods.

Bregler *et al.* [4] introduced a mathematical technique to represent joints, called twists. By using the fact that body segments do not move independently and the use of the exponential map formulation, they created a robust method to estimate the pose using a local optimization approach. Using twists and kinematic chains gives an efficient way of estimating a pose. However, the local optimization is prone to errors and so it gets stuck in local optima. Vlasic *et al.* [5] also use a local optimization approach. In their approach, they fit the skeleton inside the visual hull. A manual intervention, like any other local optimization, is required in some frames to correct the skeleton. They provide a tool to readjust poses at real-time, and iterate a second time over the frames to propagate the user constraints backwards. Local optimizations are not usefull for our purpose, since they require manual intervention and can't run fully automatic.

Global optimization approaches use stochastic tracking techniques like particle filter, which do not require manual intervention and do not get stuck in local optima, at the drawback of having a high computation time. This gives them the ability to handle fast and abrupt pose changes. Gall *et al.* [1] introduce the Interacting Simulated Annealing approach which is based on a particle filter and simulated annealing. Corazza *et al.* [6] make use of a customized version of adapted fast simulated annealing to match the silhouettes to the visual hull of the 3D model. Daubney and Xie [7] use a probabilistic graph to represent the body of a human. Where particle filters sometimes fall in the wrong optimum causing tracking errors, their method permits greater uncertainty on the location and rotation of the root node. Given the hypothesis of the root node, the posterior over the other joints are estimated conditioned on this value. While global optimizations are capable of running fully automatic, the required computation time is very high. We seek for a fast method and so these global optimizations are not practical.

While the pose is estimated, the mesh of the model does not fit the observed person well, due to loose clothing. Therefore, several methods infuse another stage in which the mesh is deformed, after the skeleton has been estimated. These techniques are capable of capturing the time-geometric variation in shape, for example loose clothes, hair or skirts. It can be achieved by either directly deforming the mesh [8] or by using a combination of skeleton estimation with mesh deformation [5, 9, 10]. This gives a smooth and accurate mesh deformation, which could help in the estimation process.

Standard models are not very useful for a model-based pose estimation, as people differ a lot from each other. Size, clothing and gender differ from person to person.

Models created from each individual would give more accurate results. Several methods exist to create a model from a person. A full body laser scanner gives a model with the most accurate results. Creating a model for people wearing loose clothing is more difficult, but can be achieved by using a database of registered body scans [11]. Balan and Black [12] use a learned algorithm based on the SCAPE model [13] to find the human body shape using skin segmentation. From this shape the gender of the person is estimated. These methods all require 3D scans of the persons, created in a controlled environment, which in some cases is not suitable. Using multiple cameras together with near-perfect background subtraction, the model can be acquired by Shape-from-Silhouette [14, 15]. This does not require a strict controlled environment. Another way is to use voxel reconstruction and create a model from the obtained voxels. These methods which do not require a strict controlled environment are practical for purposes in which no controlled environment is guaranteed or available.

Local and global optimization approaches are not practical for use, however, a combination of these two is. Our work is related to the method of Gall *et al.* [1]. While local optimization methods get stuck in local optima and global optimization methods have a high computational cost, the method by Gall *et al.* combines these two methods to provide a fast method, which is still capable of acquiring accurate results. They also add a surface deformation approach after skeleton estimation. In our work we won't use the surface deformation, since for our application the perfect deformation of the mesh is not necessary. This method is fully automatic, in the sense that given a 3D mesh model, the calibration data and the videos, the method can estimate the pose in each frame, without requiring any manual intervention. It is capable of giving accurate results even on low quality sequences and is less sensitive to silhouette noise than visual hull based approaches. Also, no restrictions on the pose or the observed person are required. So for general purposes this method will be suitable.

Chapter 3

Kinematics

We describe a 3D model in a 3D world by a 3D object using a set of vertices and relations between the vertices. These relations can be lines, triangles, quads, surfaces, etc. A 3D object can represent a real object, but also an object that does not exist. Like a real object, these 3D objects can be moved around. This can be accomplished by a translation and/or a rotation of the object, which is called a motion. Two different motions exist, rigid and non-rigid motion. A rigid motion of an object is a motion which preserves the distance between the vertices at any time. In other words, the object is not deformed by the rigid motion. A non-rigid motion, on the other hand, can move the object around in 3D space but can also deform an object, such as stretching or skew. Like a real person, the model of a person can not be deformed in any way and as such, only rigid motions apply to these models. Since the only kind of models used in this thesis are 3D models of real persons, we will only make use of rigid motions which are explained in Section 3.1. Since a person only has joints along which a limb can rotate, an axis-angle approach can be used to move the model to a different pose, which is explained in Section 3.2. Section 3.3 will describe the notation of the model of a human used in this thesis.

3.1 Rigid motion

The movement of object parts is analysed by attaching a frame to each part and determining how the frames move with respect to each other. The position of a frame which moves around in space, the so called *moving frame* M, is always with respect to another frame, which is called the *fixed frame* F. The rigid motion of M with respect to F, describes the relative position of M, as shown in Figure 3.1. In this thesis all frames are right-handed, meaning a frame defined by $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^3$ must satisfy the constraint $\mathbf{z} = \mathbf{x} \times \mathbf{y}$.

A rigid motion $\mathbf{M}$, can be decomposed as a rotation $\mathbf{R}$ followed by a translation $\mathbf{T}$. In the Euclidean space, the rotation and translation are described by a matrix. Multiplying either matrix with a vertex will give the new coordinates of that vertex. A rotation in a 3D environment is a rotation around the x-, y- and/or z-axis and each of

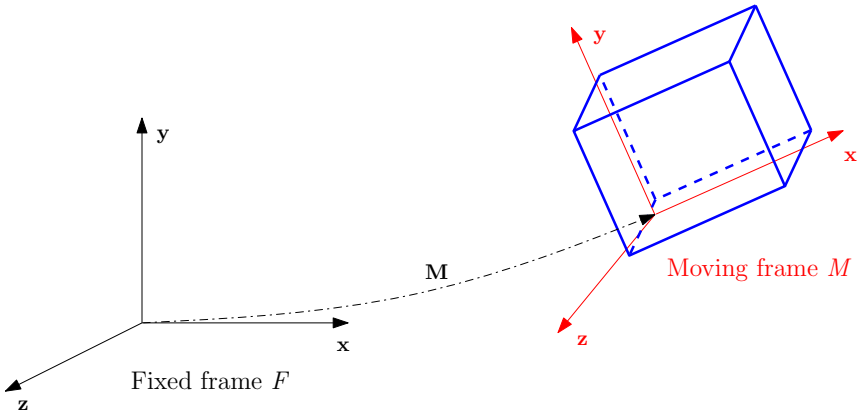


Figure 3.1: Rigid motion of an object attached to a moving frame with respect to a fixed frame.

them is described by their own matrix:

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{3.1a}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{3.1b}$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.1c}$$

A rotation around two or more axes is accomplished by taking the product of the rotation matrices, resulting in a composed rotation matrix $\mathbf{R}$.

A translation is nothing more than adding a vector $\mathbf{t}$ to the coordinates of the vertices of the 3D model. In matrix form this is

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.2}$$

where $\mathbf{I}$ is a 3×3 identity matrix. Matrix $\mathbf{T}$ is in *homogeneous coordinates* [16], therefore the coordinates of all vertices of the model must be converted to their corresponding homogeneous coordinates before a translation matrix can be applied to the vertices.

Adding an extra dimension, which value is set to one, will be sufficient. Thus if we have a 3D vertex $\mathbf{V} = (v_x, v_y, v_z)$, we write it as $\mathbf{V} = (v_x, v_y, v_z, 1)$, which is in homogeneous coordinates.

A rotation and a translation can also be combined into one matrix, creating a rigid motion

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}. \tag{3.3}$$

As you can see, this matrix is a transformation in homogeneous coordinates. To transform a vertex, the vertex must be in homogeneous coordinates to compute its new location. The resulting vertex will still be in homogeneous coordinates. Projecting this vertex back to non-homogeneous coordinates is done by dividing each coordinate of the vertex by the last (fourth) value. Then the last element is removed from the vertex, such that a vertex with only 3 coordinates remains. The motion is a linear transformation, keeping the fourth element always at value 1. In this case it is sufficient to only remove the last element from the vertex to project it back to non-homogeneous coordinates.

3.2 Twist

A rigid motion is a composition of a rotation and a translation as explained in the previous section. Another way of describing a rigid motion is the *twist* representation [17]. It is based on the fact that a rigid motion can be modelled as a rotation around an axis and a translation along this axis. In the next two sections, it will be explained how a rigid motion can be described by a twist. Before explaining a twist, the definition of a skew-symmetric matrix has to be explained first.

Skew-symmetric matrix Since the cross product is a linear operator, it can also be defined by

$$\mathbf{a} \times \mathbf{b} = \hat{\mathbf{a}}\mathbf{b}. \tag{3.4}$$

where $\hat{\mathbf{a}}$ is called the *skew-symmetric matrix* of $\mathbf{a}$, defined as

$$\hat{\mathbf{a}} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{3.5}$$

3.2.1 Exponential coordinates for rotation

In this section the exponential coordinates of a rotation matrix will be given. Consider a point $\mathbf{p}$ attached to an object with rotation axis $\boldsymbol{\omega}$, as shown in Figure 3.2. The velocity of $\mathbf{p}$ can be defined by the perpendicular vector on $\boldsymbol{\omega}$ and on $\mathbf{p}$. When the object is rotating with constant unit velocity around axis $\boldsymbol{\omega}$, the velocity of $\mathbf{p}$ may be written as

$$\mathbf{p}'(s) = \boldsymbol{\omega} \times \mathbf{p}(s) = \hat{\boldsymbol{\omega}}\mathbf{p}(s). \tag{3.6}$$

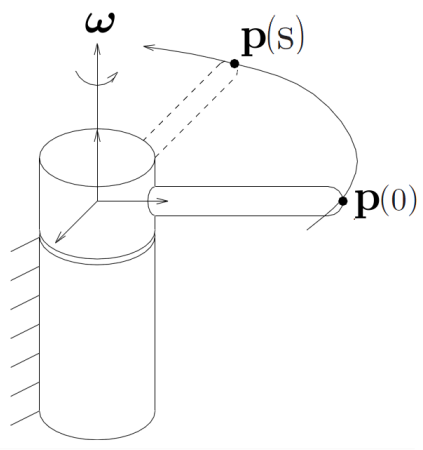


Figure 3.2: Trajectory of a point on a rotating object.

This equation is a time invariant linear differential equation, since at any time, the velocity is always the same.

The solution to Equation (3.6) is given by

$$\mathbf{p}(s) = e^{\hat{\boldsymbol{\omega}} s}\mathbf{p}(0), \tag{3.7}$$

where $\mathbf{p}(0)$ is the initial location of point $\mathbf{p}$ and where the exponential of $\hat{\boldsymbol{\omega}}s$ can be written in a Taylor series

$$e^{\hat{\boldsymbol{\omega}} s} = I + \hat{\boldsymbol{\omega}}s + \frac{s^2}{2!}\hat{\boldsymbol{\omega}}^2 + \frac{s^3}{3!}\hat{\boldsymbol{\omega}}^3 + \ldots. \tag{3.8}$$

Equation (3.8) is an infinite series. Hence it is not very efficient to use it for computation. Luckily, this equation can also be written in closed-form

$$e^{\hat{\boldsymbol{\omega}} s} = I + \hat{\boldsymbol{\omega}}\sin s + \hat{\boldsymbol{\omega}}^2(1 - \cos s), \tag{3.9}$$

which is referred to as the Rodrigues' rotation formula [18]. It should be noted that $\boldsymbol{\omega}$ has unit length. Otherwise, when $||\boldsymbol{\omega}|| \neq 1$ the equation becomes

$$e^{\hat{\boldsymbol{\omega}} s} = I + \frac{\hat{\boldsymbol{\omega}}}{||\boldsymbol{\omega}||}\sin(||\boldsymbol{\omega}||s) + \frac{\hat{\boldsymbol{\omega}}^2}{||\boldsymbol{\omega}||^2}(1 - \cos(||\boldsymbol{\omega}||s)), \tag{3.10}$$

or $\boldsymbol{\omega}$ can be normalized first before using Equation (3.9).

From Equation (3.7) it follows, that if we rotate $\mathbf{p}$ around $\boldsymbol{\omega}$ for θ units, the net rotation is given by

$$R(\boldsymbol{\omega}, \theta) = e^{\hat{\boldsymbol{\omega}}\theta}. \tag{3.11}$$

The proof that every rotation matrix can be represented as the exponential function of $\hat{\boldsymbol{\omega}}$ and θ can be viewed in Murray *et al.* [17].

3.2.2 Exponential coordinates for rigid motion

In the previous section we have discussed the exponential representation of a rotation. But a rigid motion is a composition of a rotation and a translation. In this section we will extend the previous section by adding the translation.

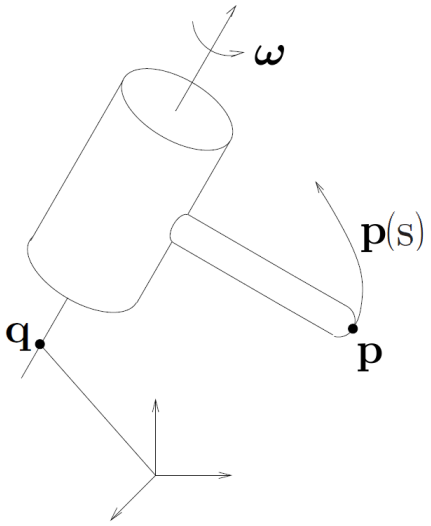


Figure 3.3: Trajectory of a point on a rotating object, rotated around a translated axis.

Once again, take $\boldsymbol{\omega}$ as the unit vector, a point $\mathbf{p}$ on the object and $\mathbf{q}$ as a point on the axis as shown in Figure 3.3. The velocity of a point $\mathbf{p}$(s) attached to the object is

$$\mathbf{p}'(s) = \boldsymbol{\omega} \times (\mathbf{p}(s) - \mathbf{q}) = \hat{\boldsymbol{\omega}}(\mathbf{p}(s) - \mathbf{q}). \tag{3.12}$$

This equation is almost similar to Equation 3.6, except that the axis is not going through the origin. Equation 3.12 can be rewritten using homogeneous coordinates and $\mathbf{v} = -\boldsymbol{\omega} \times \mathbf{q}$ in

$$\begin{bmatrix} \mathbf{p}' \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{\omega}} & \mathbf{v} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \hat{\boldsymbol{\xi}} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}. \tag{3.13}$$

Taking $\mathbf{p}(0)$ as the initial location of point $\mathbf{p}$, gives the solution

$$\mathbf{p}(s) = e^{\hat{\boldsymbol{\xi}} s}\mathbf{p}(0) \tag{3.14}$$

where the points are in homogeneous coordinates. $e^{\hat{\boldsymbol{\xi}} s}$ has the same form as Equation (3.8) and is also an infinite series

$$e^{\hat{\boldsymbol{\xi}} s} = I + \hat{\boldsymbol{\xi}} s + \frac{s^2}{2!}\hat{\boldsymbol{\xi}}^2 + \frac{s^3}{3!}\hat{\boldsymbol{\xi}}^3 + \ldots. \tag{3.15}$$

Likewise, it can be written in a closed-form and using $s = \theta$, see Proposition 2.8 in Murray *et al* [17]

$$e^{\hat{\boldsymbol{\xi}}\theta} = \begin{bmatrix} e^{\hat{\boldsymbol{\omega}}\theta} & (I - e^{\hat{\boldsymbol{\omega}}\theta})(\boldsymbol{\omega} \times \mathbf{v}) + \boldsymbol{\omega}\boldsymbol{\omega}^T\mathbf{v}\theta \\ 0 & 1 \end{bmatrix}, \tag{3.16}$$

where $\hat{\boldsymbol{\omega}}\theta$ can be calculated using Equation 3.9. This equation only applies if $||\boldsymbol{\omega}|| = 1$. If this this is not the case, then the length of $\boldsymbol{\omega}$ can be normalized by an appropriate scaling of θ. There is one case when this equation doesn't hold, namely when $||\boldsymbol{\omega}|| = 0$. An $\boldsymbol{\omega}$ of length zero means there is no rotation. In this case the equation becomes

$$e^{\hat{\boldsymbol{\xi}}\theta} = \begin{bmatrix} I & \mathbf{v}\theta \\ 0 & 1 \end{bmatrix}, \tag{3.17}$$

and this twist will only consist of a translation, see Proposition 2.8 in Murray *et al* [17].

Finally, the transformation $e^{\hat{\boldsymbol{\xi}}\theta}$ can be used to map points from their initial position to their final location by multipling this transformation with a point $\mathbf{p}$, which is in fact the same as multiplying its corresponding rigid motion with $\mathbf{p}$

$$\mathbf{M}(\hat{\boldsymbol{\xi}}, \theta)\mathbf{p} = e^{\hat{\boldsymbol{\xi}}\theta}\mathbf{p}. \tag{3.18}$$

Again, the formal proof that every rigid motion can be expressed as the exponential of some twist and a θ is given by Murray *et al.* [17].

Now we know how the exponential coordinates for a rigid motion are defined. To summarize, $\theta\hat{\boldsymbol{\xi}}$ is the twist defining the motion of a joint, where θ is the angular change of the limb with respect to the previous limb, and $\hat{\boldsymbol{\xi}}$ is the skew-symmetric matrix of $\boldsymbol{\xi}$, which defines the parameters of the twist

$$\boldsymbol{\xi} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \text{ and } \hat{\boldsymbol{\xi}} = \begin{bmatrix} 0 & -\omega_z & \omega_y & v_1 \\ \omega_z & 0 & -\omega_x & v_2 \\ -\omega_y & \omega_x & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.19}$$

with $\boldsymbol{\omega}$ the unit vector pointing into the rotation axis and $\mathbf{v}$ the cross product of the location of the axis and $\boldsymbol{\omega}$.

3.3 Model

A limb of a person can be modelled as a rigid body: a solid object which can not deform in any way. Under any transformation the distance between two points should remain the same. In the previous section we have seen that any rigid body motion can be modelled as a twist. So a joint belonging to a limb can be modeled by a twist $\theta\hat{\boldsymbol{\xi}}$. This twist can only rotate around one axis, therefore we are only able to model revolute joints (joints with one degree of freedom). If we want to model a joint with two or three degrees of freedom, e.g. shoulders or hips, multiple revolute joints are concatenated to simulate the effect of joints with more than one degree of freedom.

3.3.1 Kinematic chain

The position of a 3D vertex $\mathbf{V}_i$, influenced by limb k, depends on the angle and the twist of joint j belonging to k. Subsequent, the position and orientation of limb k, depends on the angle and twists of all its predecessors. Consequently, all limbs on the chain from the root to limb k $(0, 1, \ldots, k)$ influence the transformation of $\mathbf{V}_i$. This chain is called a *kinematic chain*, containing $k + 1$ limbs, interconnected by k joints.

The transformation of $\mathbf{V}_i$, depends on the initial configuration of the skeleton and the joints on the kinematic chain from the root to limb k. Assume we have the rigid motions $\mathbf{M}_0$ and $\mathbf{M}_i$ with $i = 1, \ldots, k$ for the root and each joint on the kinematic chain. Computing the transformation of $\mathbf{V}_i$ is done by applying $\mathbf{M}_0$ first, then $\mathbf{M}_1$, $\mathbf{M}_2$, etc. Which gives this equation

$$T_{\mathbf{x}}\mathbf{V}_i = (\mathbf{M}_0(\mathbf{M}_1)\ldots(\mathbf{M}_k\mathbf{V}_i)))), \tag{3.20}$$

Due to the associative property of matrix multiplication, this is the same as

$$T_{\mathbf{x}}\mathbf{V}_i = (\mathbf{M}_0\mathbf{M}_1\ldots\mathbf{M}_k)\mathbf{V}_i. \tag{3.21}$$

In the previous section it is explained that a rigid motion can be described by a twist and from Equation (3.18) we have seen that the exponent of the twist is the same as a rigid motion. So using this fact and Equation (3.21), the coordinates of the transformated point becomes

$$T_{\mathbf{x}}\mathbf{V}_i = \prod_{j=0}^{n_{k_i}} e^{\theta_{o_{k_i}(j)}\hat{\boldsymbol{\xi}}_{o_{k_i}(j)}}\mathbf{V}_i, \tag{3.22}$$

where $o_{k_i}(j)$ stands for the order of joints in the kinematic chain, k_i is the limb associated with vertex $\mathbf{v}_i$ and n_{k_i} are all joints influencing the position and rotation of limb k_i. Since the skeleton of a body does not contain any loops, all influencing limbs lie on the kinematic chain from root to limb k_i.

3.3.2 3D Pose

A 3D model of a person contains several kinematic chains and is actually a tree, as shown in Figure 3.4, where all kinematic chains have the same root. Changing the state of one or more kinematic chains, will put the model in a different pose.

For each joint j in the skeleton, the rotation axis and its location are known (from the 3D model and not subject to change) and so $\hat{\boldsymbol{\xi}}_j$ is known. Therefore, the unknowns, defining the state of a kinematic chain, are the global twist $\theta_0\hat{\boldsymbol{\xi}}_{\mathbf{0}}$ of the skeleton and the angles of all joints on the chain. If we want to define the state of the entire model, the state of all kinematic chains must be combined. This gives us a vector

$$\mathbf{x} = (\theta_0\hat{\boldsymbol{\xi}}, \boldsymbol{\theta}) = (\theta_0\mathbf{v}_0, \theta_0\boldsymbol{\omega}_0, \boldsymbol{\theta}), \tag{3.23}$$

where $\boldsymbol{\theta} = (\theta_1, ..., \theta_N)$ are the joint angles of the entire model. In the next chapter, this vector pose $\mathbf{x}$ will be important, since we want to find $\mathbf{x}$ that moves the 3D model in a pose that matches the pose of the person in the video.

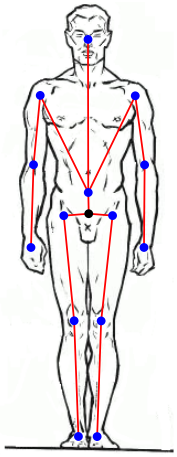


Figure 3.4: The pose of a model.

Chapter 4

Pose estimation

In this chapter the entire method used to determine the pose of a human from videos is explained in full detail. First an overview of the whole method is given in Section 4.1. In the next Section 4.2 an approach is given to find a good pose. However, it is not always capable of finding a good pose. A second approach is given in Section 4.4 which always finds a good pose, but at the cost of computation time. Combining these two approaches will give us a method capable of finding the pose in the entire video sequence. To be able to find out if the first approach fails to find a good pose, an evaluation function is defined which is discussed in Section 4.3. If the evaluation function detects a wrong pose, the second approach is triggered.

4.1 Overview of method

The general idea of the pose estimation is that the pose of the person in each frame of a video sequence has to be found, given only the images of the video sequence from static calibrated cameras, the calibration data of the cameras and the 3D mesh model. It should be noted that frames used in this chapter are not the same as the frame used in the previous chapter. In this chapter, a frame refers to an image from the video-sequence. Since we know that the pose of the person between two subsequent frames does not change instantly and the poses don't differ to much from each other, we can use the estimated pose of the previous frame as an initial pose for the current frame. It will give us a good indication for the next pose. This applies for each frame, except for the first frame which doesn't have a previous frame. Section 4.5 will explain how the initial pose for the first frame can be found.

To find the next estimated pose, the configuration vector $\mathbf{x} = (\theta_0\hat{\boldsymbol{\xi}}_{\mathbf{0}}, \boldsymbol{\theta})$, transforming the previous estimated pose to the new estimated pose must be computed. For this purpose we make use of a combination of two methods:

1. Local Optimization of the skeleton
2. Global Optimization of the skeleton

These two methods will estimate the skeleton pose of the person in the video. From this estimated skeleton pose, we can also determine the mesh deformation by using *quaternion blending* [19].

4.1.1 Input and output

Both the local optimization and global optimization require input to estimate the pose. First of all, a 3D mesh model, consisting of the skeleton and the mesh, of the person is required. The mesh constitutes the skin of the model, and consists of triangles with no topology or connectivity. The skeleton is a hierarchical tree, where each node represents a joint and the edges can be interpreted as the limbs of the skeleton. The rigging of the skeleton to the mesh is done by *automatic rigging* [20]. This rigged 3D mesh model will be optimized by both methods and will propagate through the entire pipeline of the method. Secondly, the images from each camera view are required and we also need the binary foreground silhouettes of each image. For simplification, we will use the term silhouettes to denote the binary foreground silhouettes. These silhouette images will be used by both the local and the global optimization. Additionally, the local optimization uses a second image, namely the image from the video-sequence itself for estimating the pose. Lastly, for each camera we needs its calibration data [16], such that can project the 3D model to the image plane of a camera.

Projected surface model The method also makes use of the 2D projection of the 3D model. This is created by taking all the patches from the model and projecting them to the image. A projection of a patch is done by multiplying the camera matrix [16], composed of the intrinsic and extrinsic camera calibration parameters, of a camera with each vertex of the patch. The multiplication gives a 2D point in homogeneous coordinates, so the x and y coordinates have to be divided by the third dimension to get the correct 2D coordinates. Projecting all three vertices of the patch to the screen and filling this triangle will give a projected patch. Doing this for all patches, gives the projected surface. However, not all patches have to be projected, as some patches are not in view of the camera, they are at the back of the model and are occluded by patches facing the camera. To determine if a patch is in view of the camera *back-face culling* is used [21]. It makes use of the normal of the patch. If it is pointing towards the camera, it is a visible patch, otherwise it is not needed to project the patch to screen. This will cut the number of patches that will be projected by half. Usually, back-face culling can be done by a rendering API, such as OpenGL or Direct3D.

4.1.2 Pipeline

At each frame of the video sequence, the pose of the previous frame is used as input for the local optimization, which will optimize the limbs to the closest local minimum, meaning it will find the closest best solution. Searching for the closest local minimum can be done fast, however this is also error prone, since it only seeks locally and so it is not capable of getting out of a local minimum. As a result, when an error occurs

in one frame (due to the local optimum), the error will propagate over the rest of the video sequence, giving a wrong pose estimation over time. Hence, we need to recover from this error in an early stage. For this purpose, the global optimization is used to re-estimate limbs which are misplaced. The global optimization searches the entire search space to find the global optimum. Since it searches the entire search space, it is a very costly algorithm. In order to overcome this disadvantage, the global optimization is only initiated if the local optimization gets stuck inside a local optimum, i.e. a limb is misplaced. Hence a limb evaluation is used to detect a wrong estimation. This limb evaluation will evaluate each individual limb and label it as a wrong estimate if it is misplaced.

Through the entire pipeline of the method, a pose vector is used which moves the initial model to the pose in a frame. In the succeeding frame, this vector will be optimized, such that another pose vector is gained which moves the initial model to the pose in this succeeding frame. Hence the pose vector is an absolute motion from the initial pose to a pose which is updated in each frame.

A schematic overview of the entire pipeline is given in Figure 4.1 and in the following sections each optimization technique is explained in more detail.

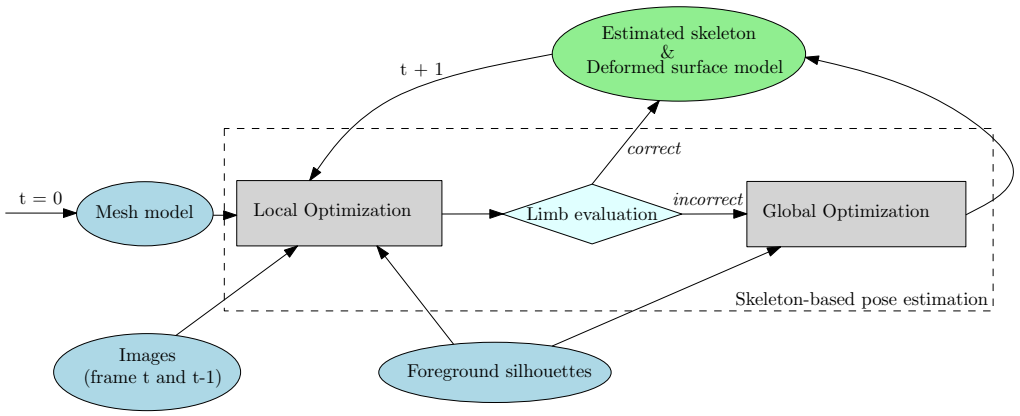


Figure 4.1: Pipeline of the method.

4.2 Local optimization

The local optimization makes use of the Weighted Least Squares (WLS) problem [22] to find the nearest optimum, which can be either the global or a local optimum. The WLS problem solves for a set of given equations by finding the solution that minimizes the sum of squared errors of the equations. In the case of finding the best pose, the vector $\mathbf{x} = (\theta_0\hat{\boldsymbol{\xi}}_\mathbf{0}, \boldsymbol{\theta})$ moving the model into a different pose, has to be found. The set

of equations are created from correspondences between the 3D model and the images in two ways. The first one is created by finding closest points between the projected vertices of the model and the contour of the silhouette. The second finds a matching between SIFT features [23] from two consecutive frames, and moves the model according to this matching. The WLS finds a solution that moves the model into a pose with the smallest total correspondence error.

4.2.1 Point correspondences

In order to solve the WLS problem, the equations for the WLS have to be constructed first. The equations are created from correspondences $(\mathbf{V}_i, \mathbf{p}_i)$ $(i = 0, ..., I)$ between the 3D model and the images, which will give the error between the fitted model and the person in the frames. To create a more accurate result, two different correspondences are used, namely the contour correspondences and the texture correspondences. In both cases the projected 3D model is needed.

Assume we have a 3D vertex $\mathbf{V}_i$ from the 3D model and its related 2D point $\mathbf{v}_i$ from the projected surface. The contour correspondences are the closest points between the contour of the projected surface and the contour of the image silhouette. So for each projected vertex of the model, $\mathbf{v}_i$, lying on the contour of the projected surface, the closest 2D point $\mathbf{p}_i$ on the contour of the silhouette image in the current frame has to be found. Using a closest point algorithm, all correspondences between the contours can be found.

For the texture correspondences, SIFT features between two consecutive frames taken from the same camera are matched. In this case, $\mathbf{p}_i$ is the location of the SIFT feature in the current frame, matched with the same SIFT feature in the previous closest to the projected vertex $\mathbf{v}_i$ in the same previous frame. Note that in both cases there are correspondences between two 2D points, $(\mathbf{v}_i, \mathbf{p}_i)$. However, since every point $\mathbf{v}_i$ is a projection of a vertex $\mathbf{V}_i$, there is also a 3D-2D relation, $(\mathbf{V}_i, \mathbf{p}_i)$.

4.2.2 Correspondence error

In order to determine the error between the 2D and 3D point of a correspondence, all 2D points have to be changed into 3D entities. Meaning that projection rays, originating from the camera origin through the 2D points $\mathbf{p}_i$, need to be defined. From these projection rays, we can calculate the error between the projection ray and the 3D model point, which represents the error of a correspondence.

Plücker coordinates To represent a ray we are going to use Plücker coordinates, which describe lines in space [18]. The parametric representation of a line in space is given by $\mathbf{y}(t) = \mathbf{p} + t\mathbf{n}$, where $\mathbf{p}$ is a point on the line and $\mathbf{n}$ is the direction vector of the line. From this parametric representation the Plücker coordinates of the line are constructed, which is $(\mathbf{n}, \mathbf{m})$ consisting of the direction vector $\mathbf{n}$ of the line and the vector $\mathbf{m}$, where $\mathbf{m} = \mathbf{p} \times \mathbf{n}$.

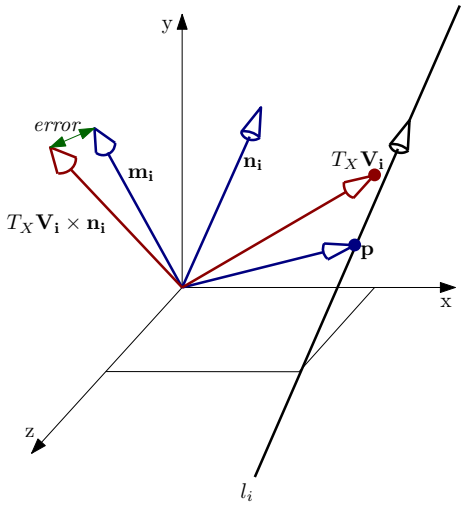


Figure 4.2: Error of a correspondence.

The correspondences consist of a static 2D point from an image and a 3D point from the model, which can be transformed. Recall from Equation 3.22 that $T_{\mathbf{x}}\mathbf{V}_i$ is the transformed point of $\mathbf{V}_i$. Now we can rewrite a correspondence as a pair $(T_{\mathbf{x}}\mathbf{V}_i, \mathbf{p}_i)$. The goal is to find a transformation for $T_{\mathbf{x}}\mathbf{V}_i$ that minimizes the distance between this transformed 3D point and the ray l_i going through $\mathbf{p}_i$ on the image plane of camera $\mathcal{C}$. The parametric equation of this ray is defined by $\mathbf{y}_i(t) = \mathbf{p}_i + t(\mathbf{p}_i - \mathbf{c}_{origin})$, where c_{origin} is the origin of $\mathcal{C}$ and the Plücker coordinates of this ray are: $(\mathbf{n}_i, \mathbf{m}_i)$, where $\mathbf{n}_i = \mathbf{p}_i - \mathbf{c}_{origin}$ and $\mathbf{m}_i = \mathbf{p}_i \times (\mathbf{p}_i - \mathbf{c}_{origin})$. The error is expressed as the difference between the vector perpendicular to $(T_{\mathbf{x}}\mathbf{V}_i, \mathbf{p}_i)$ and $\mathbf{n}_i$, and the vector $\mathbf{m}_i$, as shown in Figure 4.2. In equation form this becomes

$$||\Pi(T_{\mathbf{x}}\mathbf{V}_i) \times \mathbf{n}_i - \mathbf{m}_i||_2. \tag{4.1}$$

Due to the fact that the transformed point is in homogeneous coordinates and the Plücker coordinates are not, we have a projection Π from homogeneous to non-homogeneous coordinates. The projection is just neglecting the homogeneous component, since the fourth element always has a value of 1. Since we seek for an alignment between the contours of the silhouette image and the projected surface, the error should be as close to zero as possible, preferably zero. An error of zero means we have a perfect match, the transformed point lies on the ray, while a value close to zero means a good match.

4.2.3 Weighted Least Squares

From the created correspondences between the images from all camera views and the 3D model, a system of equations can be constructed. We want to find the vector $\mathbf{x}$, that transforms the entire model in such a way, that the sum of squared errors over all correspondences is minimized. Using Equations (3.22) and (4.1) and having $\mathcal{I}$ correspondences, the error to be minimized can be expressed by the following equation

$$\underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2} \sum_{i}^{\mathcal{I}} w_i ||\Pi(T_{\mathbf{x}}\mathbf{V}_i) \times \mathbf{n}_i - \mathbf{m}_i||_2^2. \tag{4.2}$$

w_i is a parameter which specifies the weight for each correspondence. Later in this section it will be explained what value this weight has.

Equation (4.2) is a weighted least squares problem. However, the equations to be solved are not linear, because of the exponential form of the transformation matrices in each equation. A least squares problem has no closed-form solution, meaning the formula can not be evaluated in a finite number of standard operations. On the other hand, a linear least squares problem has a closed-form solution. So we make use of the linear least squares problem to solve our problem. To do so, we iteratively solve an approximated linear model, constructed from the least squares model. At each iteration, the equations will be refined by the solution found in the previous iteration, such that it will converge to a minimum. Also, since in each iteration we move the model according to the result in the previous iteration, the correspondences are not correct any more. So a new set of correspondences is created in each iteration.

Convergence

As the weighted least squares problem converges to a solution, the difference between two consecutive iterations will differ less. Therefore, as a convergence criterion we will use

$$\frac{|S_{t-1} - S_t|}{S_{t-1}} < 0.01, \tag{4.3}$$

where S_t means the sum of squared errors over all correspondences at iteration t, S_{t-1} is the sum of squared errors at iteration $t-1$ and the constant 0.01 is the *convergence precision*. Meaning we take the square of Equation 4.1 for each correspondence and then take the sum over all correspondences. This makes the algorithm stop after the difference in the sum of squared errors between two iterations is less than 0.1%. It is not guaranteed, however, that this convergence criterion will stop the algorithm in a finite number of iterations. Since the solution is approximated in every iteration, it is possible that the amount of difference between two iterations is always greater than 0.1%. To prevent the algorithm from running an infinite number of iterations when it cannot converge correctly, we will stop after 30 iterations. Running the algorithm for more iterations will not necessarily lead to a better solution as is seen from experiments.

In case of running 30 iterations, the optimization is not stable and the best local solution can not be found correctly. However, the solution is not necessarily a bad solution, since we already search close to the local optimum. In Section 4.3 the limbs are evaluated, and in case of a bad solution, the global optimization will be used to find a better solution.

Linearisation

To linearise the model, the transformation matrices will be linearised by using the first order Taylor approximation: $\exp(\theta\hat{\boldsymbol{\xi}}) \approx \mathbf{I} + \theta\hat{\boldsymbol{\xi}} = \mathbf{I} + \theta_0\hat{\boldsymbol{\xi}}_0 + \theta_1\hat{\boldsymbol{\xi}}_1 + ... + \theta_j\hat{\boldsymbol{\xi}}_j$, where $\mathbf{I}$ is the identity matrix. Inserting this Taylor approximation into Equation (4.2) results in

$$\underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2} \sum_{i}^{\mathcal{I}} w_i ||\Pi((\mathbf{I} + \theta_0\hat{\boldsymbol{\xi}}_0 + \sum_{j=1}^{n_{k_i}} \theta_{o_{k_i}(j)}\hat{\boldsymbol{\xi}}_{o_{k_i}(j)})\mathbf{V}_i) \times \mathbf{n}_i - \mathbf{m}_i||_2^2. \tag{4.4}$$

Stabilising

As the problem has a lot of different possible solutions, the optimization is not stable. It will not converge properly to a minimum without a proper guidance. Therefore, strong deviations from the predicted angles are penalized, such that we narrow down the search space. In order to do so, the linear system is extended by adding an additional equation

$$\beta\theta_j = \beta(\hat{\theta}_j - \tilde{\theta}_j), \tag{4.5}$$

for each joint j, where $\tilde{\theta}_j$ is the total relative joint angle found in the previous iterations for the current frame, initially set to 0. $\hat{\theta}_j$ is the predicted angle, obtained by a linear 3rd order auto-regression, i.e. $\hat{\theta}_j = (\theta_{j,t-1} + \theta_{j,t-2} + \theta_{j,t-3})/3$, where t is the current frame and $t-1, t-2, t-3$ are the previous three frames. β is the *stabilising factor*, set proportionally to the number of correspondences. We use $\beta = 0.5 * |(T_{\mathbf{x}}\mathbf{V}_i, \mathbf{x}_i)_i|/\#\text{joints}$, such that these equations have some impact on the solution, but are not the main contribution to the solution.

Weighting factor

The weighting factor w_i is used to give different weights to the two different correspondences. Initially, it is possible that there is a large distance between the projected surface and the silhouette. As texture correspondences can handle larger displacements than the contour correspondences, the texture correspondences get a higher weight in the beginning. Initially, the weights w_i^C for the contour correspondences will be set to 1 and the weights for the texture correspondences will be $\sum_i w_i^T = \alpha \sum_i w_i^C$, where $\alpha = 2.0$. After the first iteration, the solution already converges to a local minimum, such that the weights for texture correspondences can be decreased to $\alpha = 0.1$. This will let the contour correspondences influence the transformation more in the next iterations, as the contour correspondences can estimate the pose more accurately.

Approximated solution

The WLS will give an approximated solution, in the form of vector $\mathbf{x}$, containing the estimated global twist and the estimated joint angles. From the estimated angles and the known twists, we can reconstruct the rigid body motion for each joint, using Equation 3.16. The estimated global twist contains the first six parameters of the vector pose $\mathbf{x}$. As you can see, for the global twist no angle θ is approximated and without it we can not create a rigid body motion. However, recall from Equation (3.23), the angle is already encapsulated in the six parameters of the global twist. So θ must be extracted first before creating the rigid body motion. θ is the length of the rotation axis $\boldsymbol{\omega}$, and dividing the entire twist with this length, will extract θ from the twist. Now the global rigid motion can be reconstructed from the global twist and the extracted θ, using the same equation. The found rigid motions are relative motions, so in order to move the mesh model into the right pose, we apply these rigid motions to the previous pose. Because of the fact that every limb is seen as a rigid structure, quaternion blending can be used between every two iterations. This will approximate the smooth surface deformation of the mesh.

4.2.4 Solving the Weighted Least Squares problem

To be able to solve the WLS problem in Equation (4.4), we need to rewrite the system of equations to the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is the design matrix, $\mathbf{x}$ is the vector containing the unknown parameters and $\mathbf{b}$ is the vector of the observations. Combining all I correspondences into the system of linear equations and rewriting the equations will give

$$\mathbf{W}\begin{bmatrix}\mathbf{D}_{00} & \mathbf{D}_{0j}\\ \mathbf{D}_{10} & \mathbf{D}_{1j}\\ \vdots & \vdots\\ \mathbf{D}_{i0} & \mathbf{D}_{ij}\end{bmatrix}\begin{bmatrix}\mathbf{v}_0\\ \boldsymbol{\omega}_0\\ \theta_1\\ \vdots\\ \theta_k\end{bmatrix} = \mathbf{W}\begin{bmatrix}\mathbf{m}_0 - \Pi(\mathbf{V}_0)\times\mathbf{n}_0\\ \mathbf{m}_1 - \Pi(\mathbf{V}_1)\times\mathbf{n}_1\\ \vdots\\ \mathbf{m}_i - \Pi(\mathbf{V}_i)\times\mathbf{n}_i\end{bmatrix}, \tag{4.6}$$

where $\mathbf{D}_{i0}$ is the design matrix part for the global twist, $\mathbf{D}_{ij}$ is the design matrix part for joints $1 \dots j$, $\mathbf{m}_i$ and $\mathbf{n}_i$ are the Plücker coordinates and $\mathbf{V}_i$ is the 3D vertex of correspondence i. See Appendix A on how to rewrite the system of linear equations to this proper form.

After rewriting the system of equations, we can use the Householder algorithm to find the unknowns.

4.3 Limb evaluation

When the local optimization has converged to a minimum, it is not guaranteed that the solution is good. Due to the minimization over a summation in Equation (4.4), it can be the case that while we have a low error, some limbs are still misaligned. A correct limb with a low error and a misaligned limb with a high error, give the same average error

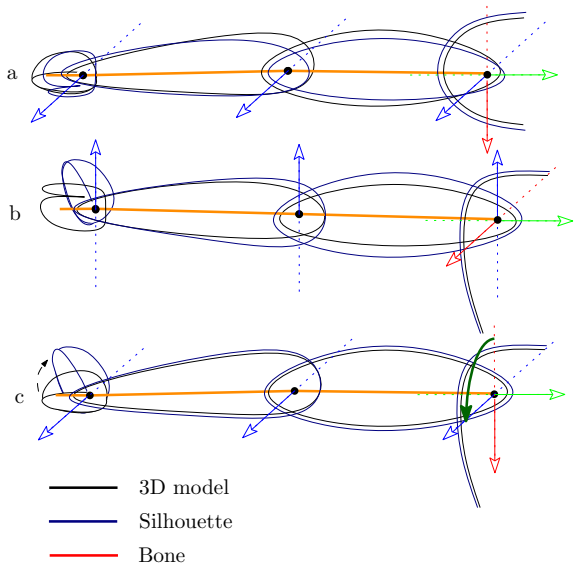


Figure 4.3: Rotation axis of the hand is not correct in a) and b). a). view from top, b). view from front, c). corrected pose by rotation along green axis in the shoulder seen from front-view

as two correct limbs with an equal average error. While the first case is obviously worse than the second, both have the same average error and both cases can occur. Hence, limbs with a high error should be detected and corrected. For that purpose, each limb is evaluated and assigned as misaligned if the outcome is higher than a predefined upper bound, which is set to 500, corresponding to an error of approximately 22 millimetre. The evaluation is done by calculating the energy in much the same way as in Equation (4.2). But this time the average error is calculated over the correspondences $\mathcal{I}_k \subset \mathcal{I}$ belonging to the same limb k

$$E_k(\mathbf{x}) = \frac{1}{I_k} \sum_{i}^{I_k} ||\Pi(T_{\mathbf{x}}\mathbf{V}_i) \times \mathbf{n}_i - \mathbf{m}_i||_2^2, \tag{4.7}$$

and we calculate this error for each limb. If a misaligned limb is detected, then all subsequent limbs will also be labelled as misaligned. Because a change in a limb will cause all subsequent limbs to be changed as well, due to the kinematic chain property, stating that the position and orientation of a limb is also determined by the configuration of all preceding limbs. For example a re-estimation of a misaligned shoulder, will cause the position and orientation of the arm and hand to be changed. Meaning, these limbs have to be re-estimated as well. Also, all preceding limbs up to a joint with three degrees of freedom are labelled as misaligned. When, for example, the wrist (which has only one

DoF) is misaligned, then this can be caused by a wrong estimation of the shoulder. The whole arm can be rotated in a way, such that the wrist can only rotate in a direction which will never result in the correct pose, as shown in Figure 4.3.

After evaluation, the global optimization is initiated if at least one misaligned limb has been detected.

4.4 Global optimization

Usually, one wants to find the minimum of an objective function, given some constraints. In most cases, the search space of the objective function contains a global optimum and a lot of local optima with a variety in energy, distributed over the search space. If we use a local optimization technique, one ends up almost always in one of the local optima, depending on the starting location. Looking beyond the local optima to find the global optimum requires a different approach: we have to search through the entire search space. Several techniques exist that are capable of searching the entire search space, such as swarm-based optimization algorithms [24], evolutionary algorithms [25] and particle filtering. All these techniques fall in the category global optimization. The approach used in this thesis is Interacting Simulated Annealing (ISA) [9], which is based on a particle filter that uses Boltzmann-Gibbs measures, similar to the annealing scheme used in simulated annealing.

4.4.1 Interacting Simulated Annealing

ISA is a method based on simulated annealing and the particle filter. Before going into detail on ISA, simulated annealing and the particle filter will be briefly explained.

Simulated Annealing

Simulated annealing is an approach that searches for a good approximation of the global optimum. However, it is not guaranteed to find the most optimal solution. Like other local optimization techniques, simulated annealing takes a starting solution and moves to a better solution among its current neighbours. The drawback of always moving to a better solution is that it gets trapped in a local optimum very quickly. Simulated annealing prevents this by accepting worse solutions with a certain probability if no better solution is available. Accepting worse solutions makes the simulated annealing robuster to local optima as it explores the search space better. The probability of accepting a worse solution depends on the energy of the current solution, this worse solution and on a parameter, commonly known as the temperature. A higher temperature gives a higher probability to accept the worse solution. To prevent the method from always accepting a worse solution, resulting in a method that will not converge, the temperature is decreased over time. As the temperature decreases, the probability of acceptance is also decreased. Decreasing the temperature is done according to an annealing scheme. This way, the approach is first moving towards a region containing good solutions, then it moves even further into a region containing the best solution. The disadvantage of

simulated annealing is that it can not search in multiple regions of interest, as is the case in finding the best pose, where the parameters of a good solution are not always close to the global optimum depending on the used image features.

Particle Filter

Particle filters, also known as interacting particle systems, use a set of particles, where each particle represents a solution. By using a set of particles, the focus of the search can be spread among different regions of interest, by distributing particles among these regions. Particle filters consist of the following three steps. In the first step each particle is weighted by an energy function. From these weights it can be determined where in the search space there is a region with good solutions, such that we can narrow the search space. In the second step, particles with a good solution are kept, but particles representing a bad solution are removed from the particle set and are replaced by particles with a solution close to other good solutions. The last step diffuses all particles, such that the method adapts its search to the new regions of interest. These three steps are iterated and eventually, the set of particles will converge towards the global optimum.

ISA

ISA combines the annealing property of simulated annealing with the set of particles from the particle filter. All particles are distributed over the search space and selected in the same way as in particle filter. The main difference is the weighting of the particles, which is done according to the annealing scheme. The annealing scheme will make the distribution of particles converge faster than when no annealing scheme is used.

4.4.2 Algorithm

From the limb evaluation we know the correct and misaligned limbs. Since the correct limbs are not subject to change, we can fix these limbs. As a result, the search space can be decreased by removing those fixed limbs from the search space. The projection of the parameter space of the skeleton onto a lower dimensional search space

$$P(\mathbf{x}) \rightarrow \tilde{\mathbf{x}} \in \mathbb{R}^m, \tag{4.8}$$

makes sure that only the m misaligned limbs can be changed in the optimization process.

The optimization consists of three steps: weighting, selection and mutation, which are iterated several times until a predefined number of iterations has been reached, as shown in Figure 4.4. During this optimization, each particle $\mathbf{r}_i \in \mathcal{R}$ represent a solution, which is the vector pose $\mathbf{x}_i$. During the run of the method, the set of particles $\mathcal{R}$ is constantly updated by removing, adding and changing the particles. The notation $\mathcal{R}_k$ represents the state of the particle set and $\mathbf{r}_{k_i} \in \mathcal{R}_k$ is a particle at iteration k.

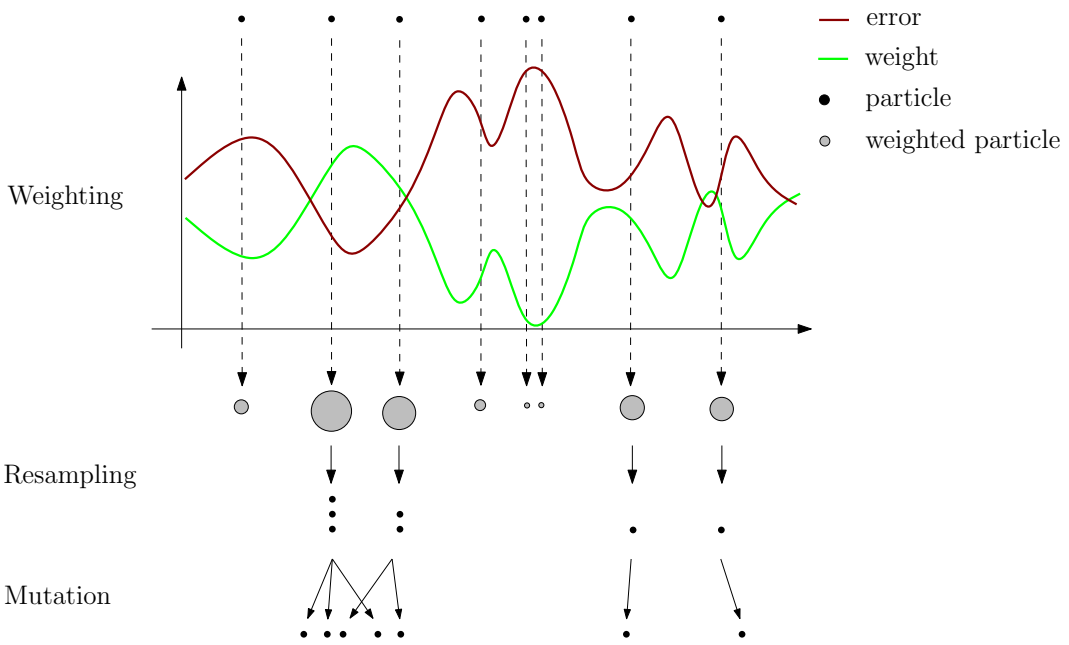


Figure 4.4: Steps taken in the ISA method. After weighting the particles (gray circles), the particles are resampled and mutated (black circles).

Weighting From the set of particles $\mathcal{R}_k$, we weight each particle $\mathbf{r}_{k_i} \in \mathcal{R}_k$ by the Boltzmann-Gibbs measure

$$\pi_i = \exp(-\beta_k E(\mathbf{r}_{k_i})), \tag{4.9}$$

at iteration k, where $\beta_k = (k+1)^b$ with $b = 0.7$ is the annealing scheme producing the temperature and $E(\mathbf{r}_{k_i})$ is the energy of particle $\mathbf{r}_{k_i}$. The weights are normalized, such that $\sum^i \pi_i = 1$. These normalized weights of the particles are the probabilities that a particle is chosen in the selection step. This function makes sure that particles with a high error get a low weight, and particles with a low error get a high weight. Due to the annealing scheme β, the difference in weight between particles with a low error and a high error becomes larger when k increases. This property is important for the selection step, as in later iterations we want to converge to a single solution and we do not want to keep multiple optimums.

Selection In this step, each particle is selected with a probability of $\pi_i / \max_l \pi_l$, ensuring that at least the particle with the highest weight is selected. Other particles with a high weight (low error) have a high probability of being selected and particles with a low weight (high error) might be removed from the particle set.

After this selection, from the n particles, only m particles are accepted. To guarantee that at every iteration we have a sufficient number of particles, we add $n - m$ particles to the selected particles. This is done by stratified resampling [26] using the normalized weights. The resampling replaces the removed particles by selecting $n - m$ new particles from the old set, where once again particles with a higher weight have a high probability of being chosen. After resampling, the set of particles can contain several particles representing the same solution.

Mutation To make sure that we keep searching the search space, the particles are spread out according to a Gaussian kernel K_k, whose covariance matrix is proportional to the sampling covariance matrix

$$\Sigma_k = \frac{\alpha_\Sigma}{n-1}(\rho\mathbf{I} + \sum_{i=1}^{n}(\mathbf{r}_{k_i} - \boldsymbol{\mu}_k)(\mathbf{r}_{k_i} - \boldsymbol{\mu}_k)^T), \qquad \boldsymbol{\mu}_k = \frac{1}{n}\sum_{i=1}^{n}\mathbf{r}_{k_i}, \tag{4.10}$$

scaled by $\alpha_\Sigma = 0.4$ and where $\rho = 0.0001$ is a small positive constant to make sure that the variance (values along the diagonal of the matrix) does not become zero.

At iteration $k = 0$, no initial particle set exists yet. In [9] the initial particle set is created from a linear interpolation between the predicted pose and the estimated pose from the local optimization. Linear interpolation between two poses means that for each joint we create an angle list of n linear interpolated values between the angle from the predicted pose and the angle from the estimated pose. Then n unique particles are created by randomly taking (and removing from the angle list) an interpolated value for each joint. However, as the limb evaluation already states, the misaligned limbs from the local optimization are incorrect. Using this pose would already lead to incorrect results, so for a better estimation, instead of using the estimated pose from the local optimization, the estimated pose from the previous frame would be a better initial pose. Nevertheless, the limbs, global position and orientation which are correctly estimated by the local optimization can still be used. So we make a combination between the result from the local optimization and the estimated pose from the previous pose. To do so, the estimated pose from the local optimization is used and all misaligned limbs are substituted by the estimated limbs from the previous frame. Next, the particles are diffused by a zero-mean Gaussian distribution with a covariance matrix proportional to the covariance matrix created over the last poses, as in Gall *et al.* [27]. They create this covariance matrix according to a regression which is implemented by Gaussian processes. This makes sure that the focus of the search lies in the neighbourhood of the predicted pose and the estimated pose from the previous frame, since we know that these two poses will probably not differ a lot from the real pose. After the last iteration, the mean of the particle set

$$\mathbf{r}_k = \sum_{i=1}^{n}\pi_i\mathbf{r}_{k_i}, \tag{4.11}$$

represents the single estimate for the pose in frame k. Where the mean rotation for the global rigid motion is calculated according to [28].

4.4.3 Energy function

In Equation (4.9), we defined a weighting function for a particle which uses an energy function to determine the weight of the particle. The energy resulting from the energy function corresponds to the error of the pose $\tilde{\mathbf{x}}_{k_i}$ represented by particle $\mathbf{r}_{k_i}$ and is computed as follows:

$$E(\mathbf{x}_{k_i}) = E_S(\mathbf{x}_{k_i}) + \gamma E_R(\tilde{\mathbf{x}}_{k_i}). \tag{4.12}$$

The first term of this equation measures the silhouette error between the silhouette and the projected surface model.

The silhouette error for pose $\mathbf{x}_{k_i}$ for view $\mathcal{C}$ calculates the pixel-wise differences between the silhouette image $S_\mathcal{C}$ and the projected surface model $M_\mathcal{C}(\mathbf{x}_{k_i})$, generated by projecting the surface of the transformed model according to the pose of the particle. So the error of a pose for view $\mathcal{C}$ is given by

$$\begin{aligned} E_S^\mathcal{C}(\mathbf{x}_{k_i}) =& \frac{1}{|M_\mathcal{C}^0|} \sum_{\mathbf{q} \in M_\mathcal{C}^0} |M_\mathcal{C}(\mathbf{x}_{k_i})(\mathbf{q}) - S_\mathcal{C}(\mathbf{q})| \\ &+ \frac{1}{|S_\mathcal{C}^0|} \sum_{\mathbf{r} \in S_\mathcal{C}^0} |S_\mathcal{C}(\mathbf{r}) - M_\mathcal{C}(\mathbf{x}_{k_i})(\mathbf{r})|, \end{aligned} \tag{4.13}$$

where $M_\mathcal{C}(\mathbf{x}_{k_i})(\mathbf{q})$ and $S_\mathcal{C}(\mathbf{q})$ are the pixel values for a pixel $\mathbf{q}$ and the sets of pixels located inside the silhouette and projected surface model are $M_\mathcal{C}^0$ and $S_\mathcal{C}^0$ respectively. Note that both images are binary images, and so all pixel values are $0, 1$. To penalize pixels that are far way from the silhouette in the other image, a Chamfer distance transform is applied to the silhouette images. In the ideal case, the Chamfer distance transform is also applied to the projected surface models. However, this would be very costly as this has to be done for every particle. To compensate for this, each pixel inside the projected surface model is set to 0, as is the case for the Chamfer distance transform, and a constant penalty is set for each pixel outside the projected surface model. A value of 8 is a penalty suggested by [27]. The silhouette energy term $E_S(\mathbf{x}_{k_i})$ is finally defined as the average of $E_S^\mathcal{C}(\mathbf{x}_{k_i})$ over all views.

The second term of Equation (4.12) is a penalty for strong deviations from the predicted pose $\hat{\mathbf{x}}_{k_i}$ in the lower dimensional parameter space

$$E_R(\tilde{\mathbf{x}}_{k_i}) = ||\tilde{\mathbf{x}}_{k_i} - P(\hat{\mathbf{x}}_{k_i})||_2^2. \tag{4.14}$$

In Equation (4.12), γ is the weighting factor of this penalty and is set to 0.01, giving some smoothness to the prediction and ensuring that poses very different from the previous pose are unlikely to occur.

4.5 Initial Pose

In the entire method, we assume there is a previous frame and so a previous pose is available. This previous pose will act as a guidance for the best solution of the current frame. When the method is initialized on the first frame, no previous frame is available. Consequently, no previous pose is available and without the initial pose, the method will not work.

Finding the initial pose is a hard task. Assume a 3D mesh model is given, including its rigged skeleton. The skeleton can adopt a lot of possible poses. If the skeleton consists of j joints, the total degree of freedom is $j + 6$, where we have 6 degrees of freedom for the global translation and rotation of the model. Each joint can rotate 360 degrees, the global rotation can rotate 360 degrees in the $x-, y-$ and $z-$direction and the translation is in 3 dimensions in the entire 3D space. As one can see, it would be a very difficult task to find the pose that exactly fits the silhouettes. However, we can find an initial pose with the help of ISA and the local optimization.

First of all, we have to bound the search region for the translation as the pose can lie anywhere in the 3D space. For this purpose, we put a bounding box around each biggest object in the foreground silhouette of the first frame. A ray is shot from each camera through the centerpoint of the bounding box and the intersection point of these rays is the initial position of our model. This will be the ideal case, however, the rays will not exactly intersect at the same position. So an approximation of the intersection point has to be found, which minimizes the sum of squared distances to all rays. The distance from a point $\mathbf{c}$ to a line $\mathbf{y} = \mathbf{p} + t\mathbf{n}$ can be given in equation form:

$$d = ||\mathbf{c} - \mathbf{a}||^2 - \frac{||(\mathbf{c} - \mathbf{a}) \cdot \mathbf{d}||^2}{||\mathbf{d}||^2}. \tag{4.15}$$

Since we want to find a point $\mathbf{c}$ that minimizes the sum of squared distances of $\mathbf{c}$ to all lines, the derivative of the summation should be equal to zero.

$$0 = \sum_{i=0}^{m} ||\mathbf{c} - \mathbf{a}_i - \mathbf{d}_i \frac{(\mathbf{c} - \mathbf{a}_i) \cdot \mathbf{d}_i}{||\mathbf{d}_i||^2}. \tag{4.16}$$

The solution can be found by solving the least squares for this problem.

After solving the least squares, an approximated initial point is found. Next the global rotation and the pose have to be found. For this we make two assumptions. First of all, the global orientation of the model has the same $x-$ and $z-$orientation of the silhouettes. In other words, we only have to find the correct rotation around the y-axis. Secondly, the pose of the initial model is approximately the same as the pose of the person in the first frame. Both assumptions reduce the search space and allow us to find a good initial position and orientation in a reasonable time. The number of particles that is used is 35 * number of joints. All particles are randomly spread in a 50×50 bounding box around the approximated initial position, the global rotation around the $y-$axis is set to a random value between $(-\pi, pi)$ and the amount of rotation for the joint angles of the skeleton are kept at 0. We iterate 20 times, to ensure we have a good

convergence. After convergence, the pose is refined by the local optimization, in which the global position and orientation as well as all of the joint angles are estimated. The final result is the initial pose for the first frame.

Chapter 5

Experiments

In this chapter several experiments are carried out on the method. First we will describe the used dataset, how the validation of the results is carried out and the settings of the method. Then an experiment is done on both the local optimization and the global optimization to evaluate the performance of both optimizations individually. Next the whole method with settings from Gall *et al.* [1] is tested on the dataset, changes in the method are tested. Finally, an efficiency analysis of the method is given.

5.1 Dataset description

To test our method, we need a proper dataset. The first dataset that is going to be used is the dataset recorded for Gall *et al.* [1], which from now on we will call the Motion Capture Dataset (MCD) [1]. Since this dataset is also used by Gall *et al.*, which is the basis of our method, we will also use this dataset in our experiments, such that we can evaluate differences. This dataset provides five video sequences, from which one of the sequences is a video of a walking dog. All sequences are recorded with the following settings: 40Hz at a resolution of 1004×1004 and each sequence is captured from 8 uniquely placed cameras, preventing ambiguity in the videos. The sequences consist of several actions like dancing, performing a handstand, dancing with a skirt and performing a wheel. Each action in each sequence is performed by a single person. The recording is done in a special created environment with a green background. This allows for a perfect background subtraction and as such, the provided foreground silhouettes do not contain any gaps or noise. This allows the experiments to fully focus on the method, without interference of noise or gaps in the silhouettes. No ground truth from a marker-based motion capture system is provided, but the results from Gall *et al.* [1] are used as ground truth. The 3D mesh models have been acquired by a static full body laser scan and are in approximately the same pose as the first frame of each video sequence. A skeleton, which has 36 degrees of freedom including the six degrees of freedom for the global position and orientation of the model, is rigged to the mesh model.

[1] http://www.vision.ee.ethz.ch/ gallju/projects/skelsurf/skelsurf.html

The second dataset is the Multimodal Motion Capture Indoor Dataset (MPI08) dataset [2] [29, 30] and is acquired in the same environment as the previous dataset. This dataset provides more different sequences than the MCD dataset and has sequences with basic movements like a person walking. A sequence with a walking person is a more realistic scenario in our case than for example a person doing the handstand. The dataset is recorded with the same number of cameras with the same resolution, but with a lower frame rate of 25Hz as the MCD dataset. The 3D mesh models used in this dataset are also acquired by a static full body laser scan, but are of a lower quality compared to the first dataset. The skeleton rigged to the model has 28 degrees of freedom. This smaller number of joints imposes restrictions on the pose in comparison with a higher degree of freedom. For example, in the model with 28 degrees of freedom, there is only one revolute joint which represents the pose. In the model with a higher number of joints, the wrist is modelled by two revolute joints. So in the first case, when the wrist has to rotate in a different direction than its currents rotation axis, the whole arm has to be rotated, while in the second case this isn't necessarily needed due to the two revolute joints. A visual difference between two models can be seen in Figure 5.1.

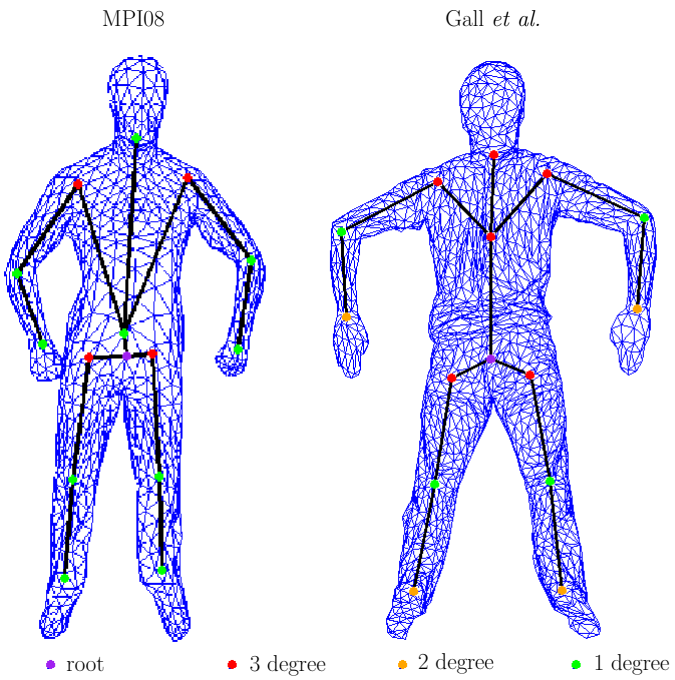


Figure 5.1: Differences in model between the MPI08 and MCD datasets.

A note should be taken on the MCD dataset. The provided skeleton for the 3D

[2] http://www.tnt.uni-hannover.de/project/MPI08_Database/

mesh model is not accurate in the weighting. Usually, when a skeleton is rigged to a mesh model, each vertex will become influenced by one or more joints and the amount of influence of each joint is determined by a weight. The sum of all weights over the influences for each vertex should sum up to one, such that a proper blending is possible. However, in this dataset the sum of weights do not always sum up to one. As a result, when blending is done on the mesh model, artefacts can occur. To ensure this will not happen, we will take the difference between the sum of weights and one, and add or subtract this difference from the joint with the greatest influence, such that the total sum of weights is one.

We will use three sequences in our experiments. The MCD dataset provides results which we can use as our ground truth, but since it doesn't contain any sequence with basic movements like walking, we use a walking sequence from the MPI08 dataset. For the walking sequence we took the first sequence of the ab series of sequences. The person in this video does not wear textured clothing and so it could be a challenging sequence for the method where SIFT features are calculated. Since we still want to have a validation on the results of our results, we took two sequences from the MCD dataset, namely the handstand and the dance sequence. The dance sequence comes as close to a walking sequence than any other of the MCD sequences. Therefore, we choose to use this one along with the handstand sequence, as the handstand sequence provides some challenges for the method, like a total rotation of the body and a crouching position.

Sequence	Joints	Vertices	Frames	Cameras
Walking [29]	22	1191	0 - 405	8
Dance [1]	32	1502	165 - 738	8
Handstand [1]	30	2501	50 - 450	8

Table 5.1: Used sequences for evaluation.

Initial pose

For our method, an initial pose is required. Both datasets provide a model which has approximately the same pose as the starting pose at each first frame. Since we can make this assumption, we use the method described in Section 4.5 to calculate the initial pose of each sequence and use it for our experiments.

5.2 Validation measure

To evaluate our results, we need a measure to define the correctness of our results. For this purpose we will use the results from Gall *et al.* [1] as our ground truth to measure the errors for the MCD dataset. For each frame we take the difference in joint locations and calculate the average error and deviation over the entire sequence. Evaluating the joint locations is not enough in the case of the last joint on each kinematic chain. The location at this joint is determined by the orientations of all preceding joints.

Meaning, the orientation of the last joint on each kinematic chain has no effect on its own location, but it has effect on vertices influenced by this joint. There is one side not on the evaluation of the orientation. Though it helps to evaluate the joints, two poses can have different limb orientations, while their location is still correct, i.e. there is ambiguity in poses. For example a straight arm can be found by multiple poses. Any rotation around the axis parallel to the straight arm gives a correct arm pose, but with a different orientation. If it is required for the application to measure the correctness of the hands, the orientation of the joint should also be taken into account. However, in our application we are not interested in the hand, but only in the location of the joint, since we want to estimate the pose. Yet, for completeness we will still give an evaluation on the correctness of the hand, by looking at orientation of the last joint on each kinematic chain. As a last measure, we will count the number of times the global optimization is initialized.

In the MPI08 dataset, no ground truth is available. For the experiments carried out on this dataset, no concrete evaluation can be given. Therefore, we will use our own vision to determine whether a pose is visually correct or incorrect.

5.3 Settings

This method has several settings that can be adjusted and most of these settings are correlated. For the local optimization, the evaluation function and the global optimization the following parameters can be adjusted:

- Local optimization
  - Stabilising factor (weight on the predicted pose)
  - Convergence precision (difference in error between two iterations)
  - Maximum number of iterations (number of iterations to solve the least squares)
- Limb evaluation
  - Error threshold (threshold value for detecting wrong limbs)
- Global optimization
  - Number of particles
  - Number of iterations
  - Initial variance (initial variance to diffuse particles in search space in the first iteration)
  - Dynamic variance scheme (factor for variance scheme to diffuse particles between iterations)
  - Annealing scheme (speed at which the system converges)
  - Prediction weighting factor

Changing the parameters of each of these parts, has an influence on the other parts. For example a higher error threshold will allow bigger errors in the pose. When the pose has an error above this threshold, it will be very different from the real pose. Finding the correct pose requires the method to search in a wider area and as such, the initial variance has to be increased. Changing parameters in one part also has an influence on other parameters in the same part. For example setting the initial variance to a higher value also means that the number of particles has to be increased as well. A bigger initial variance spreads the set of particles out to a greater area, and if the number of particles is not increased, the area will not be well covered by the particles.

Before testing the entire method, we will test how the local optimization and the global optimization perform separately. This will give a good indication on how well both parts of the method perform individually. By separating the method, we could also investigate the settings for each part, without being influenced by the other optimization. From the results we can see which of the parts of the method fail at which motions. For both cases we will test the handstand sequence from the MCD dataset and we will use a walking sequence from the MPI08 dataset. To view the full results of the experiments in video format the reader is referred to [3].

5.4 Local optimization

In this experiment, only the local optimization is tested, with the focus on the core of the local optimization, namely the contour correspondences. On their own, the contour correspondences should be sufficient to estimate the pose. The texture correspondences are left out in this experiment as this is another experiment and texture correspondences only provide an additional quality. Also, texture correspondences are mainly useful if the observed person wears textured clothing. By disabling the texture correspondences, bias of the textured clothing is prevented, since the observed person in the walking sequence wears uniform coloured clothing and the persons in the handstand and dance sequence do wear textured clothing.

The local optimization should provide accurate results unless the optimization gets stuck in a local optimum. Whenever this happens, the error will propagate over the sequence in most cases and eventually will result in a total wrong estimation of the pose. This will probably happen to the legs and arms and less often to the torso, as a bigger body part could recover from an error more easily than smaller body parts. Since only the local optimization is tested the result of the local optimization of the previous pose is always used as the initial pose for the next frame.

First we will test the local optimization on several settings and we will compare the results. The best settings are then combined to test the local optimization again. From these results we can give an evaluation on the robustness and accuracy of the local optimization.

[3] A video can be viewed at http://www.staff.science.uu.nl/ tan00109/student/2012_m_jeffrey/

5.4.1 Settings experiments

Three different settings can be changed: convergence criterion, stabilising factor and the maximum number of iterations. We will deal with each parameter individually and the results of the experiments are shown in Table 5.2. In all experiments all other settings than the setting that is tested are set to a high/low value to ensure that these settings do not interfere with the tested setting. For the convergence criterion a value of 0.0001 is low enough and for the maximum number of iterations, 50 would be sufficient. As for the stabilising factor, we do not want the pose to be strongly estimated towards the prediction, but we also do not want a factor which causes an unstable optimization. Therefore, we choose a value of 0.5 as this was the value used in the method from Gall *et al.*.

For the dance sequence we evaluated the results up to frame 550. After that great errors occur to the legs, which leads to incorrect results for all settings. In all cases, the local optimization does not recover from this error.

Convergence criterion We will first test different settings for the convergence criterion on the dance sequence. This sequence is long and provides some parts in which limbs are close to other limbs and at some parts movements are fast. Therefore, a good convergence is needed to keep up with the fast movements and to prevent limbs getting attached to other parts of the observed body. Four experiments are done with a value of 0.01, 0.001, 0.0001 and 0.00001.

As expected, it can be seen from Table 5.2 that the precision increases as the threshold for the convergence criterion is set to a lower value. But after decreasing the value from 0.0001 to 0.00001, the error suddenly increases. A further inspection leads the cause to the wrong estimation of the left arm. For a setting of 0.00001, the local optimization has more time to converge to the optimum. In this case however, it actually leads to a bigger error in the upcoming frames, since movement is fast and close to the body, the limb converges further to another body part. After a sudden movement of the observed arm in another direction, the limb gets lost. While for the setting of 0.0001, a solution close to the optimum is also good. In this case, it happens less that the solution moves the limb further to the body. When the observed arm moves back again, the limb is somewhat more closer to the observed arm and does not get stuck to other body parts, giving a smaller average error. Observing all body parts except for the left arm, we can see that the average error for each joint has not decreased, between a convergence setting of 0.0001 and 0.00001. Also, for the handstand sequence we notice an increased error when the convergence criterion is lowered. Looking at the results, we see that for the setting 0.0001 and 0.00001 after frame 405, the left leg is estimated incorrectly, in contrast to the correct estimation for setting 0.001. This is caused by the same problem as for the dance sequence.

Maximum iterations The maximum number of iterations, is correlated to the convergence criterion. A lower convergence criterion indicates a lower acceptance of the difference between two iterations. The maximum number of iterations has an effect on

	Convergence	0.01	0.001	0.0001	0.00001
Dance (start-550)	Avg	58.47	49.68	33.66	40.59
	Dev	50.42	34.47	20.56	23.82
Handstand	Avg	132.61	49.27	51.46	53.13
	Dev	77.85	30.13	43.58	44.62
	Stabilising	0.1	0.5	1.0	2.0
Dance (start-550)	Avg	49.68	33.66	29.04	36.46
	Dev	34.91	20.56	17.11	19.65
Handstand	Avg	50.86	51.46	45.06	54.49
	Dev	26.42	43.58	20.01	44.63

Table 5.2: Joint location difference of different settings for the local optimization.

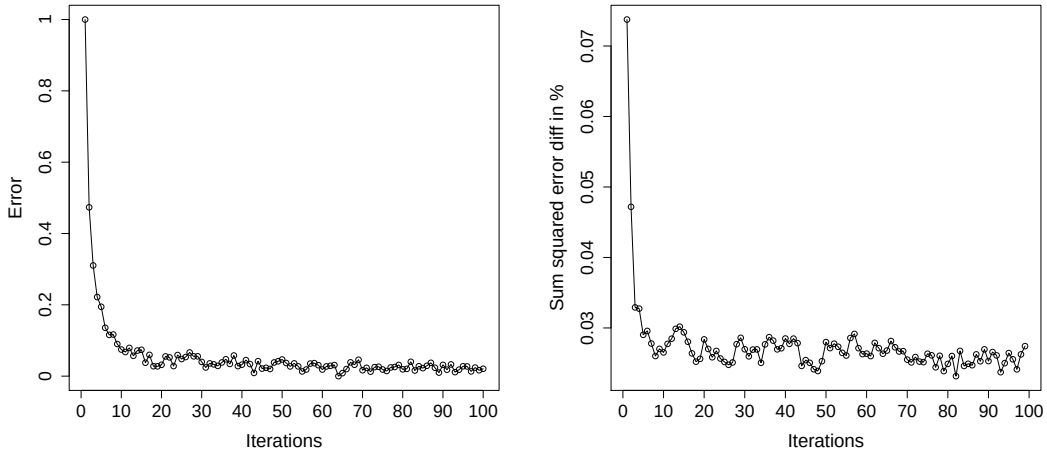
this, as it could prematurely stop the optimization while, according to the convergence criterion, it hasn't converged yet.

The graph shown in Figure 5.2(a) gives the normalized average sum of squared errors over the entire sequence for each iteration. These results are obtained from the walking sequence. It allows us to see that after the first few iterations the system already converges towards a solution. Then it starts to slowly converge and eventually it oscillates around the local optimum. In Figure 5.2(b) we could see that after the first ten iterations the difference in sum of squared errors between two iterations does not decrease much, while the sum of squared error itself still does decrease slowly. As a convergence criterion a value of 0.0001 is used, which is reached in 88 frames out of 405 frames. In all other 317 frames the criterion is not reached. The average number of iterations needed to reach the criterion is 88.8 iterations with a standard deviation of 23.7. This means, that despite the vast number of frames that do not reach the convergence criterion, the convergence criterion is still useful. It has such a low value that the chance of having a difference in sum of squared error between two iterations below this threshold, while the optimization has not converged to a local optimum, is very small. Having the criterion prevents the optimization from searching for a better solution while it is already at a local optimum.

From both graphs we can conclude that the optimization converges to a solution in the first few iterations and then it starts to slowly converge to a local optimum. After 30 iterations, the amount of error that decreases is very small. Therefore, we will use 30 iterations as an upper-bound to bound the computation time.

Stabilisation factor The stabilisation factor of the least squares depends on a constant, the number of correspondences and the number of joints. We change the constant factor and use four different settings, namely 0.1, 0.5, 1.0 and 2.0. While the lowest value would probably give an unstable system, the highest values would strongly steer the system towards the predicted pose.

For the stabilisation factor we can see that the setting with a stabilisation of 1.0 gives the best result in both cases. A low stabilisation factor gives large errors since in



(a) (b)

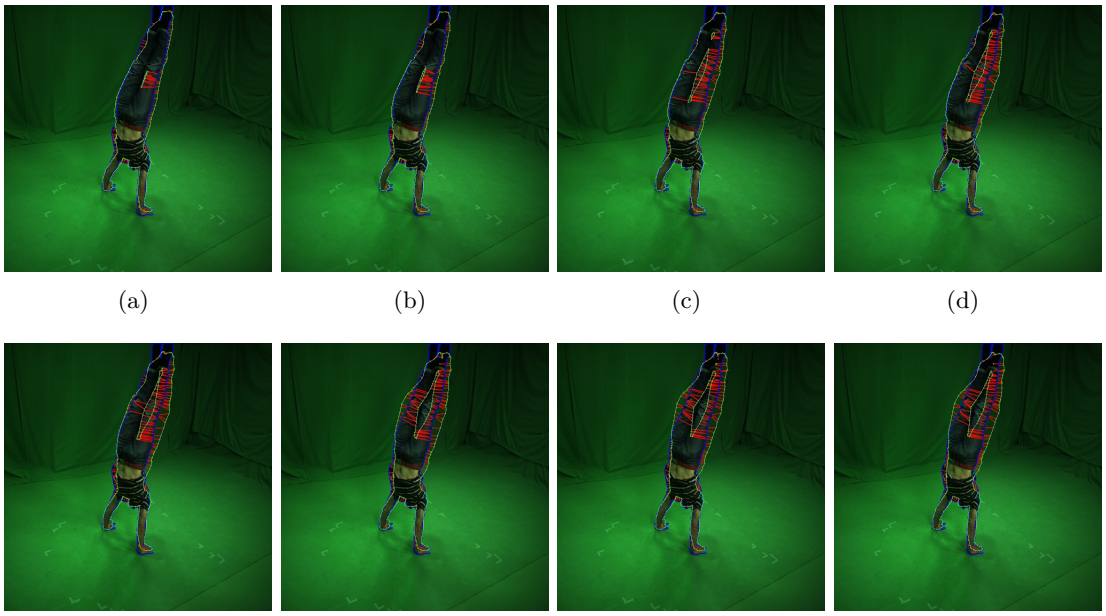
Figure 5.2: 5.2(b) is difference in sum of squared error with previous iteration. 5.2(a) is normalized sum of squared error.

cases with occlusions, the prediction could not steer the optimization towards a correct result. A setting of 2.0 also leads to high errors as this is caused by an over-prediction. Abrupt changes are not covered by the prediction as the prediction assumes that the movement is still in line with the movements in the previous frames. So the prediction steers the pose forwards, while the movement is in the opposite direction. This causes problems when the movement is close to other parts of the body. When the movement of a body part towards another body part changes abruptly in direction, the prediction will still steer the estimation towards the body part, while the observed part is moving away. Consequently, wrong correspondences are created between the wrong body parts and so one body part gets stuck towards the other body part.

From these results we can conclude that a minor prediction of 1.0 is a good guidance in case of occlusions. However, a high setting leads to a over-prediction, causing the optimization to converge to the predicted pose. This is a problem when movements close to the body are fast and change abruptly.

5.4.2 Local optimization experiment

The local optimization is prone to errors as we have already found out in the previous experiments. In this section we will give a full evaluation of the local optimization on all sequences. This allows us to see at which motions the local optimization fails or has a hard time estimating the correct pose. For all sequences we used the following settings: a convergence criterion of 0.0001 corresponding to an allowed error difference of 0.1%, a stabilisation factor of 1.0 and a maximum of 30 iterations.



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

(e) (f) (g) (h)

Figure 5.3: Gap in projected model.

Handstand sequence

When we look at the handstand sequence in the first frame, the difference between the ground truth and our result is rather small, as can be seen in Figure 5.4. It is at the first event, when the observed person is starting to perform the handstand that the errors of some limbs increase. As the person is standing straight again, the pose is estimated correctly, with a small error. During the handstand, minor errors occur to both legs. However, at some parts during the handstand the legs get split up towards the contour of the legs of the observed legs, increasing the error of both legs. This error is caused by the a gap in the projected surface of the model between the two legs, while there is no gap between the two legs in the silhouettes. The gap produces a contour, meaning that at the gap, vertices are mapped towards the closest contour points of the silhouette, which is in this case at the outer part of the legs. This error is shown in Figure 5.3. The optimization recovers from this, when the observed legs are closer to each other, giving less space for the estimation to create a gap between the legs.

The last error occurs when the person is changing his position to stand on his feet again. The first part is tracked correctly, but after frame 395 the errors increase for the legs as well as the arms. At this point, the correspondences between the model and the silhouette are not providing enough information to estimate the correct pose. This is due to the complex motion, but also due to the person going into a crouching position. When in a crouching position, the torso, arms and legs are tight together, and so the contour is really small. From the silhouettes the contours from the arms, legs and torso

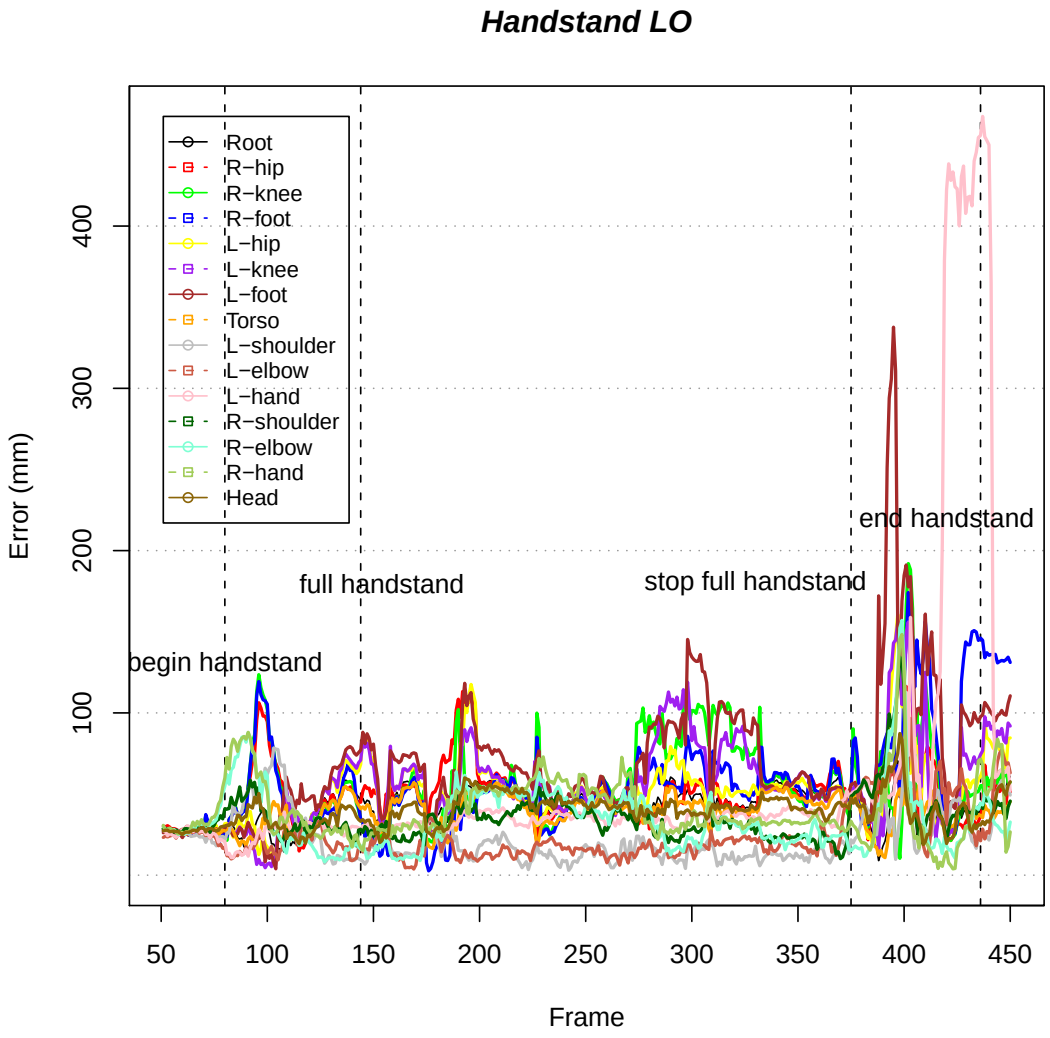
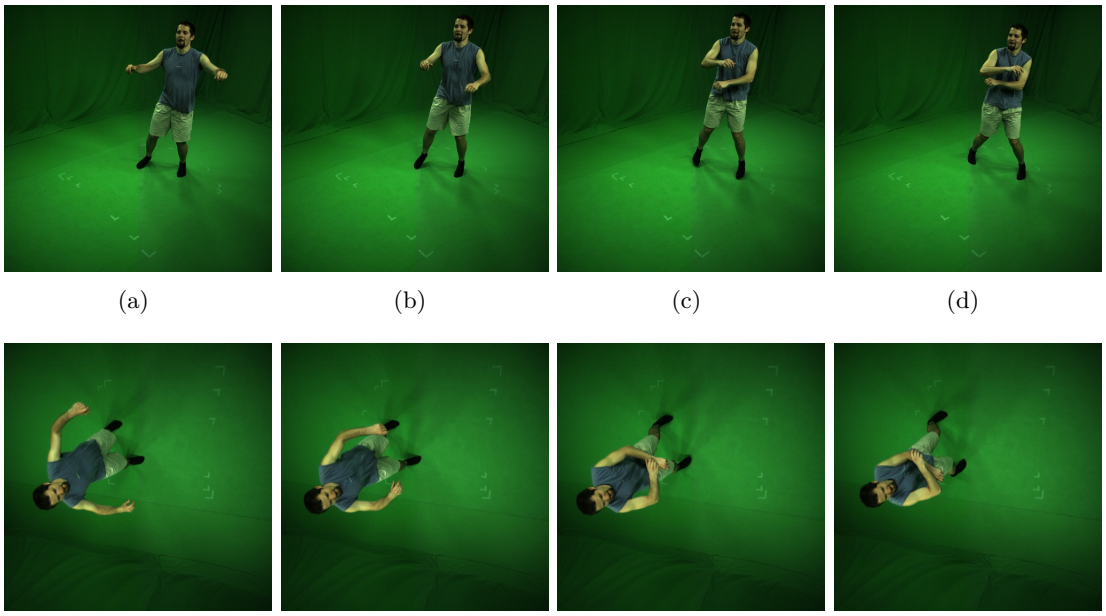


Figure 5.4: Local optimization error of the handstand sequence.



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

(e) (f) (g) (h)

Figure 5.5: Hard part of the dance sequence with overlapping arms (frame 294, 297, 301 and 306).

are hard to distinguish. This leads to a small number of correspondences from which several are wrong, resulting in an inaccurate result. In this particular case, the local optimization has a hard time finding the correct pose for the left leg and eventually it is lost. At the end of the sequence the whole upper body is estimated correctly, the left leg is entirely misplaced and the right leg is, due to the missing left leg, placed at the middle of the two observed legs.

Dance sequence

The first part of the sequence is tracked very well, with errors below 50 mm as shown in Figure 5.6. At frame 300, the observed person moves both arms towards his body, as shown in Figure 5.5. In the next frames, both arms are moved out again, resulting in a wrong estimate of the left arm. Movement at this point is fast and wrong correspondences are formed between the left arm and the torso, resulting in a arm which sticks to the torso. Same happens for the right arm, but for this arm, movement is slightly slower and less wrong correspondences are created. Eventually, the lower left arm resides in torso. At two points in the sequence the error is at a reasonable value for the left arm again, since the observed arm is moving close to the estimated arm. However, after a few frames it is lost again.

For both legs an error occurs when the lower part of the observed leg is moved past the other observed leg. Both get stuck in the other leg, due to occlusions and fast

movement. The prediction at this point does not have enough influence to guide the legs towards the correct pose, while wrong correspondences are created. Eventually the lower parts of the legs get estimated inside the upper parts of the legs.

At some points during the sequence, the orientation of the wrists is wrong though they are at the correct location. This mainly happens when the arms are next to the body, causing the optimization to create correspondences between the hand and torso. Since the hands are small and they only consist of a few vertices, a small number of correspondences is created. Therefore, a few correspondences have a great impact on the estimation of the hands. Whenever they are wrong, it could lead to a slightly wrong estimation which is propagated over the next frames.

5.4.3 Evaluation

From the experiments it is clearly seen that most of the occurred errors are not recovered by the local optimization and propagate through the sequence, as we have stated in the theory. This mainly happens for the arms and the legs, when they are close to other parts of the body, or are moving past other parts of the body. Causing the arms or legs to move in the wrong direction, as we have seen in both sequences, caused by wrong contour correspondences and fast movements. Errors also occur when the movements are hard to see from the silhouettes, meaning they are partially occluded, there is self-occlusion or body parts are occluded in several views. Although a weak a-priori pose is used to estimate the pose, it is still hard to determine the real motion for the occluded body parts.

The local optimization is sensitive to the correspondences of the hands and feet of the observed person, due to their small size and the small number of correspondences for these parts. As a result, a small error in their correspondences can have a great impact on their estimation. In most cases, the optimization propagates this error, resulting in a wrong pose. However, there are cases in which the pose is corrected again, but this is not usual. The fact that in these cases the method recovers from these errors, is because of the movement of the hands or feet and is not inherent to the local optimization. They move in such a way that the observed body part comes close to the wrong estimated part and so some correct correspondences can be created, resulting in a correct estimation. This doesn't occur in one frame, but happens over a few frames before it is totally corrected.

One noticeable remark is that the global position and orientation of the model is almost always correct and it seems to follow the person correctly. From these observations we can see that bigger parts of the body are better estimated than the smaller parts. This is inherent to the calculation of the correspondences and the frame rate of the dataset. A fast movement of a small part gives wrong correspondences, while the same amount of movement for a bigger part is less likely to give wrong correspondences. An example is shown in Figure 5.7. A higher frame rate of the videos could solve the problem of fast movements for smaller body parts, as this decreases the speed of movements between two frames.

We can conclude that the local optimization is capable of tracking the person, but

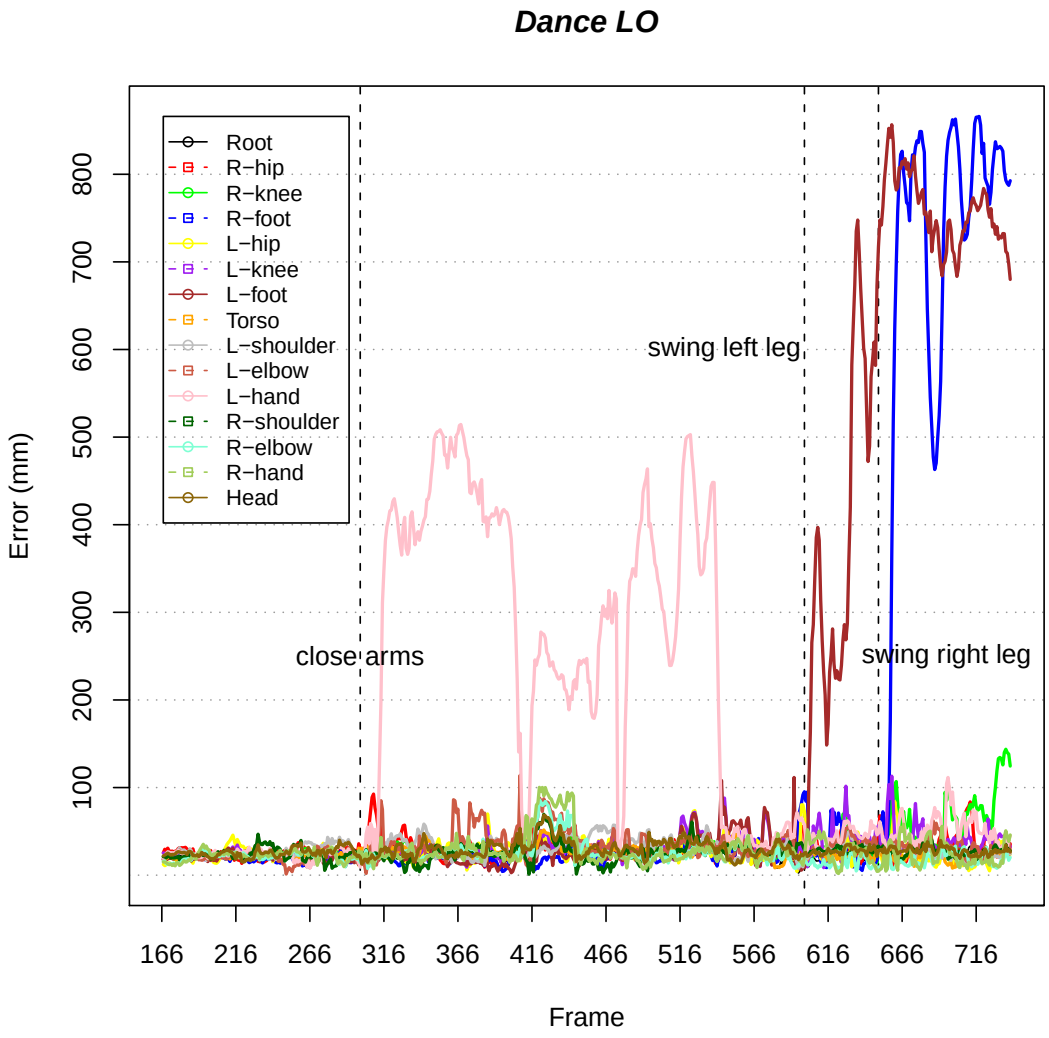


Figure 5.6: Local optimization error of the dance sequence.

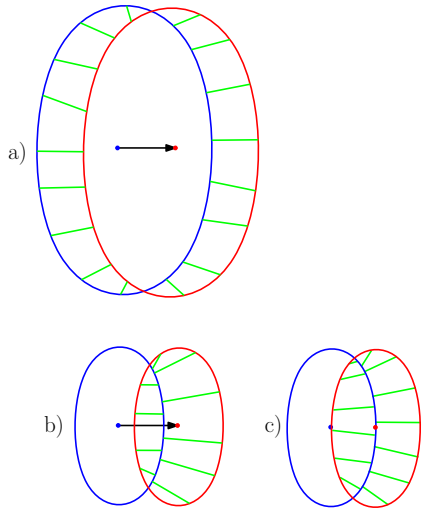


Figure 5.7: Parts a) and b) of different size move with same speed. a) will result in correct overlapping, but the result of b) is c), which is a wrong estimate

can not recover from errors. Errors occur more often for smaller parts of the body, while bigger parts are less prone to error. The number of errors is highly dependent on the visibility of the body parts, i.e. usable contours have to be extracted from the silhouettes. The number of errors also depends on the frame rate and the speed of the moving body parts.

5.5 Global optimization

In this experiment the global optimization is used to estimate the entire pose, the global position and orientation of the model. The global optimization should estimate the pose in each frame correctly, otherwise it is not a correct global optimization, though errors could occur due to self-occlusion. Whenever this happens, the global optimization should be able to recover from this when the visibility on the occluded parts is clear again.

The global optimization uses a set of particles to estimate the pose. At each frame an initial set is required. In these experiments it is constructed by a linear interpolation between the predicted pose and the estimated pose from the previous frame. Since the whole pose has to be estimated, the computation time is very large. While time and resources are limited, we have to make a proper selection on the experiments.

5.5.1 Settings experiments

Several different settings can be changed: the number of particles and iterations, the initial variance, the dynamic variance scheme, the annealing scheme and the prediction weighting factor. In the following experiments we will deal with each of these settings. Due to limited time and resources, we are not able to perform the global optimization several times with different settings on the entire sequences. Therefore, where needed, we selected several parts of sequences in which we thought the motions were harder to estimate. The initial pose from these partial sequences are gained by running the entire global optimization on the original sequence up till the frame where the partial sequence starts. The resulting pose at this point is used as the initial pose for the partial sequence. The settings for this run are: 25 particles times the number of joints with no maximum, 15 iterations, a dynamic factor of 0.4, annealing scheme of 0.7 and the prediction weight is set to 0.01. These settings were taken from [27, 1], with the exception of the number of particles, which is set to 25 to ensure that the optimization contains enough particles. The results gained from runs with these settings provide a sufficient initial pose for our partial sequences.

For the dance sequence, we start at the begin frame and end at frame 500. In this part, a hard motion is present in which the person is moving his arms close to its body. The local optimization fails at this point to estimate the arms correctly. A second part of the sequence that is used starts at frame 600 and ends at the last frame of the sequence, where in between these two frames the legs are moving in a complex way.

For the handstand sequence we start from the beginning and end at frame 200. Between those two frames the observed person is turning himself upside down. A second part which we will be using starts at frame 360 and ends at the end of the entire sequence. Between frame 200 and frame 360 no hard motion is present, as the observed person is only standing upside down and only moves half a meter.

Number of particles and iterations

The number of particles represents the population in search space, and the number of iterations determines the time spent to search for the global optimum. Increasing both values will give a better estimated pose as the search space is better populated and more time is spent to search for the global optimum, but both increments are at the cost of computation time. It is claimed in [1, 27] that 15 iterations and (20 * #joints) with a maximum of 300 particles are sufficient to estimate the pose correctly. From these claims we will experiment with a different number of particles with 15 iterations, and a different number of iterations for (20 * #joints) particles. We won't impose a maximum on the number of particles to ensure that we have a sufficient number of particles.

Handstand sequence								
	Frame 50-170				Frame 350-450			
#Particles	15	20	25	35	15	20	25	35
Root	41.28	46.11	63.59	40.80	65.33	54.68	54.33	47.51
R-hip	52.20	61.08	68.16	63.74	101.07	75.90	89.13	57.55
R-knee	54.11	58.38	74.94	65.26	86.31	88.87	74.63	117.90
R-foot	77.00	56.45	81.64	67.87	78.38	92.69	76.03	89.21
L-hip	58.36	45.53	73.36	40.29	88.63	94.51	97.29	66.36
L-knee	40.50	47.13	62.78	36.26	352.66	318.78	70.28	226.21
L-foot	56.62	54.22	70.01	42.07	429.10	551.52	85.91	252.55
Torso	50.20	59.42	82.64	47.58	53.89	38.05	44.18	35.48
L-shoulder	71.11	107.46	112.70	66.84	64.93	56.058	46.23	64.33
L-elbow	65.35	112.46	92.19	42.90	116.30	109.66	53.35	57.81
L-hand	60.84	105.41	78.10	49.09	165.59	162.64	71.28	60.58
R-shoulder	73.91	37.22	38.25	44.20	72.87	74.23	60.22	62.96
R-elbow	437.46	64.66	73.55	74.39	88.92	90.36	75.32	78.03
R-hand	433.20	57.41	64.76	65.55	107.25	106.55	82.03	93.55
Head	98.52	81.13	119.61	54.33	59.49	45.01	42.70	44.75
Avg	111.38	66.27	77.09	53.41	128.72	130.63	68.19	90.32

Table 5.3: Joint location error (in mm) of different particle settings for the handstand sequence.

Dance sequence								
	Frame 165-500				Frame 600-738			
#Particles	15	20	25	35	15	20	25	35
Root	25.75	28.75	25.78	25.42	25.21	22.03	18.07	19.07
L-hip	32.94	36.02	30.17	30.14	35.09	29.18	23.03	24.70
L-knee	34.84	36.81	32.64	29.60	104.24	65.51	57.42	49.61
L-ankle	31.95	32.78	24.50	24.92	189.51	98.97	90.34	85.16
R-hip	29.21	37.91	36.90	30.58	25.82	37.50	25.16	33.23
R-knee	34.64	35.58	36.55	40.63	46.87	40.99	50.57	38.74
R-ankle	33.71	36.00	33.31	34.56	46.30	49.45	47.29	40.51
Torso	28.51	22.39	26.44	23.88	37.78	37.55	21.28	26.61
L-shoulder	31.49	30.75	32.40	32.00	26.86	20.55	27.43	15.99
L-elbow	78.37	37.95	37.26	49.63	86.77	86.97	40.01	49.38
L-hand	107.38	45.37	42.78	130.44	517.19	65.55	60.50	53.98
R-shoulder	32.03	27.92	29.70	27.21	30.14	25.60	31.14	28.62
R-elbow	42.05	41.15	49.51	34.34	23.49	22.39	25.57	22.32
R-hand	112.27	38.45	136.08	140.30	58.75	54.24	59.79	55.14
Head	27.55	28.25	28.08	25.59	25.47	19.77	30.42	27.49
Avg	45.51	34.41	40.14	45.28	85.30	45.08	40.54	38.04

Table 5.4: Joint location error (in mm) of different particle settings for the dance sequence.

In Table 5.3 and 5.4 we can see that in both sequences most errors decrease as the number of particles increases. This is intuitive as more particles will populate the search space better, giving a better result in the end. However, we also notice that not in all cases the average error decreases as the number of particles increases. By looking at each individual limb, the error does decrease for most of the limbs, however, some limbs produce a very large error which in turn increases the average error.

One problem with the global optimization is the distribution of the optima in search space. We illustrate this problem by some examples shown in Figure 5.8. In the most optimal case, the function has a bell formed shape, where the global optimum is at the peak and several local optima surround it. Due to the annealing scheme, the ISA converges towards the global optimum provided that there is a sufficient number of particles. The second energy function has two major peaks (or more) separated from each other. Particles converge to these peaks and having the particles at different optima yields a high variance, leading to a mutation step which spreads all particles widely in search space. At the end of the global optimization, particles are distributed among multiple optima and it is not necessarily guaranteed that the mean of the distribution is at the global optimum. In the third case, the function contains a large basin with a local optimum which is far away from a small peak with the global optimum. Whenever all particles fall into the basin, the search focusses in this basin from which it is very hard to find the global optimum, especially when no particles are at the small peak. The ISA uses an annealing scheme to converge to an optimum, while the dynamic variance scheme spreads the particles. With particles residing in the basin, the dynamic variance scheme focusses its search in the basin. We will elaborate more on this in the experiment on the annealing scheme.

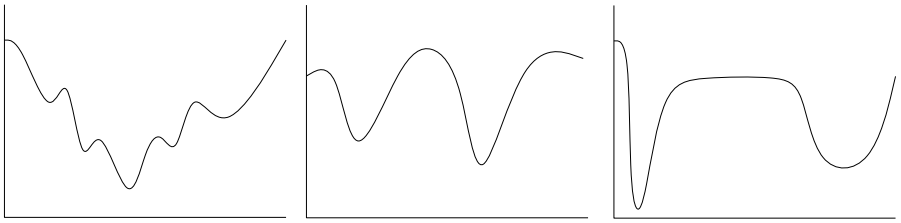


Figure 5.8: Cases of energy functions, with on y-axis the energy and x-axis the pose configuration. a) has the most optimal function, b) contains two major optima and c) contains a big basin and a peak far away from the basin.

In our particles experiment we notice that the second case occurs for the settings with a higher number of particles. Two peaks occur, one which is the best pose and a second representing a pose in which a body part is inside another body part. Despite the fact that the second pose has a higher error, particles are still kept in this optimum. Due

to the large number of particles there is a higher chance of survival for these particles. This yields a large spread in search space at each iteration. In the end, the resulting pose lies somewhere between the two poses, which is not a correct pose. This propagates over the frames and eventually there is a chance that the optimization converges towards a wrong optimum. This observation explains why in some cases the error of a limb increases when the number of particles increases.

In the previous experiment the number of particles that was sufficient to estimate the pose was 25 particles for each joint. In the following experiment the number of iterations is changed to see if an improvement can be made. A lower number of iterations would be better, as this decreases the amount of computation time. Two different settings are used and compared to the results with a setting of 20 iterations. The two settings we choose are 10 and 20 iterations, which are tested on the two partial sequences of the dance sequence. With these settings it can be seen if it is possible to speed up the method or to improve the accuracy of the optimization.

Dance sequence						
	Frame 165-500			Frame 600-738		
Iterations	10	15	20	10	15	20
Root	27.81	25.78	24.95	19.70	18.07	22.26
L-hip	33.34	30.17	30.50	22.61	23.03	25.39
L-knee	34.85	32.64	34.15	68.20	57.42	58.11
L-ankle	34.96	24.50	30.53	101.03	90.34	90.57
R-hip	37.88	36.90	28.59	31.09	25.16	35.16
R-knee	37.31	36.55	30.71	45.01	50.57	37.00
R-ankle	36.74	33.31	28.17	50.10	47.29	42.39
Torso	32.37	26.44	24.68	30.02	21.28	32.33
L-shoulder	56.32	32.40	29.44	15.74	27.43	23.65
L-elbow	231.00	37.26	33.99	58.90	40.01	59.60
L-hand	344.94	42.78	57.63	62.23	60.50	59.08
R-shoulder	45.92	29.70	28.29	33.97	31.14	28.38
R-elbow	59.10	49.51	37.56	28.41	25.57	19.25
R-hand	240.30	136.08	142.83	60.82	59.79	43.53
Head	45.99	28.08	27.31	28.91	30.42	25.36
Avg	86.59	40.14	39.29	43.78	40.54	40.14

Table 5.5: Joint location error (in mm) of different iteration settings for the dance sequence.

The results of the settings on the dance sequence are shown in Table 5.5, including the results with a setting of 15 iterations. Using 10 iterations leads to worse results, especially in the first part of the sequence, where the arms are not correctly estimated. This is caused by having several local optima and 10 iterations is not enough to make the optimization converge to one optimum. This error propagates and eventually the pose is incorrect. Having 20 iterations does lead to slightly better results, but the extra

time taken to compute the additional 5 iterations for each frame is not proportional to the accuracy gain. So we can conclude that the best number of iterations is 15, as this number has a good quality to accuracy ratio.

Initial variance

The setting of the initial variance is strongly related to the number of particles used. Increasing the variance can only be done when the number of particles is increased as well. An increase in the variance spreads the particles over a bigger search space and so an increase in the number of particles is required to ensure a good population in the bigger search space. An increased value for this setting will make the method more robust for errors, as it is more capable of recovering from a large error caused by a wrong estimation in the previous frame. Like where the estimation lies in the basin as in the right case of Figure 5.8. Consequently, the number of particles has to be increased, which in turn increases the computation time. For the initial variance the covariance matrix proportional to the predicted covariance matrix is used, as stated in Section 4.4.2. This gives a small variance in most cases.

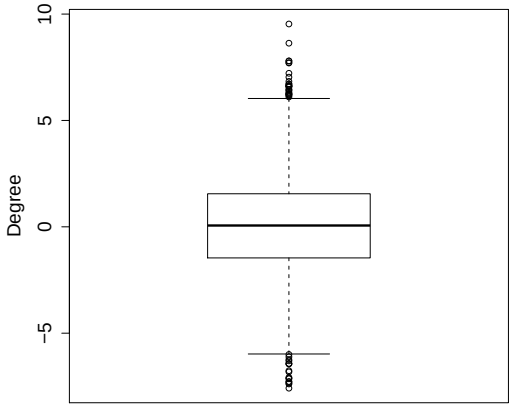
The average initial variance created from the Gaussian distribution lies between 1 and 2 degrees. While with such a low variance the particles are spread only locally, the particles are spread well in search space due to the dynamic variance scheme, which will be seen in the next experiment. At each frame, the correct pose lies close to the estimated pose of the previous frame. Consequently, the optimization does not need to search in a big search space such that the initial variance can be kept low.

Dynamic variance factor

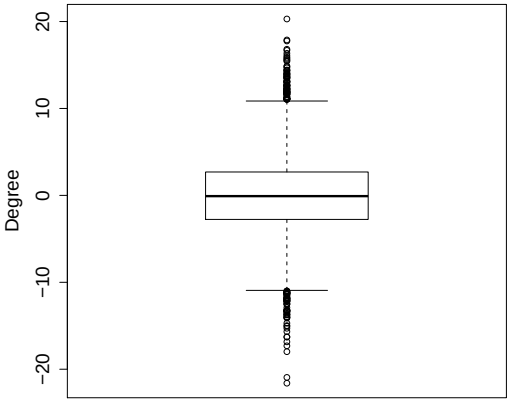
This value is the scaling factor for the dynamic variance scheme used to distribute particles between two iterations. A higher factor gives a bigger spread in search space, as it scales the covariance matrix created over the distribution of the resampled particles. So a higher dynamic variance factor is less prone to big errors, as it seeks further in search space. However, just as with the initial variance, the amount of possible spread in search space depends on the number of particles.

To demonstrate the effects of a low and high dynamic variance factor, we will use frame 55 from the handstand sequence. In this frame the left elbow is artificially rotated by 1 radian which is the same as 57.3 degrees. We will run a few tests on this setting to see how big the spread of the particles is. Figure 5.9 shows the box plots of different dynamic variance factors. The data shown in these box plots give the amount of mutation, in other words the amount of rotation, of the left elbow in each of the 15 iterations of the global optimization. It is clearly visible that a higher dynamic variance factor gives a bigger spread. Despite the low initial variance, the dynamic variance scheme is still capable of spreading all particles in a wide search space, even with a low dynamic variance factor of 0.2.

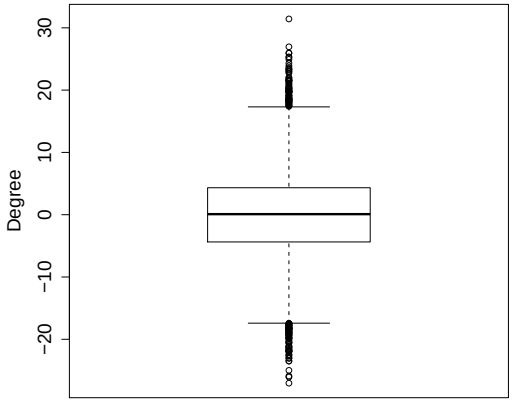
We will now artificially rotate joints of the model by a fixed number, so that the model is not in the correct pose any more. Not only one joint is rotated, but each



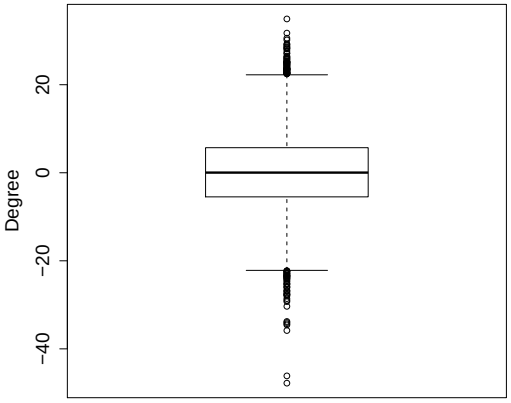
(a) Variance factor of 0.2



(b) Variance factor of 0.4



(c) Variance factor of 0.6



(d) Variance factor of 0.9

Figure 5.9: Amount of spreading in one frame for different dynamic variance factor settings.

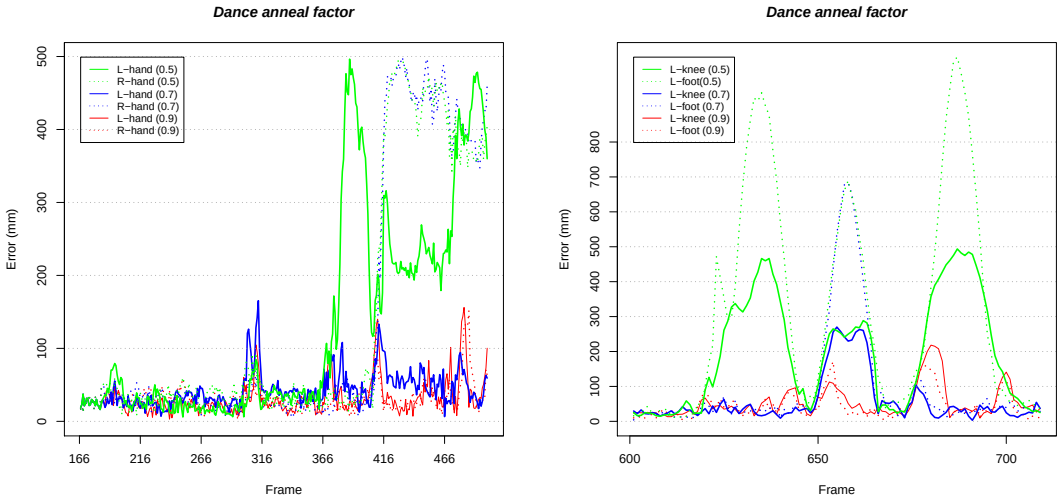
joint in the whole skeleton. When only one joint is out of place, the optimization could focus on one limb only as all other joints are correct. The number of wrong poses grows exponential in the number of misaligned joints of different kinematic chains. In case of two misaligned limbs in different kinematic chains, for example the left elbow and right elbow, the number of wrong poses grows exponentially. The optimization has to converge to an optimum in which both the left arm and the right arm have to be correct. During the optimization it occurs that in one particle the left arm is correct while the right arm is not correct, while in another particle the right arm is correct and the left arm is not. This keeps the variance high for both limbs, resulting in an optimization that keeps spreading the particles in search space. The greater the error, the harder it is to converge to the correct pose for both arms.

	All joints		Left & right elbow	
Rotation in radian	Avg	Dev	Avg	Dev
0.15 (8.59°)	39.21	12.48	39.21	12.48
0.20 (11.45°)	36.42	13.91	-	-
0.25 (14.32°)	61.54	13.04	31.44	6.12
0.30 (17.18°)	123.96	29.11	-	-
0.35 (20.05°)	-	-	34.99	6.59
0.55 (31.50°)	-	-	36.84	10.42
0.75 (42.97°)	-	-	39.80	8.10
0.80 (45.84°)	-	-	61.47	14.88
0.85 (48.70°)	-	-	45.36	11.30

Table 5.6: Joint location difference when joints are manually rotated.

We will increase the amount of rotation in each test and see up to which point the optimization is able to find the correct pose again with a dynamic variance factor of 0.4, with (25 * #joints) particles and 15 iterations. We will still use frame 55 from the handstand sequence and compare the results with the ground truth. We did two experiments, one by rotating all joints and another by rotating the left and the right elbow. The results are shown in Table 5.6. It can be seen that when the amount of rotation is above 11 degrees for all joints, the optimization fails to estimate the correct pose again. This could be a problem in the optimization when all joints are incorrect and not close to the optimal solution. Therefore, the limb evaluation function has to detect the wrong limbs in an early stage, especially when multiple limbs are wrong. Otherwise, even the global optimization will not find the correct pose. However, when only two joints are incorrect the error that is tolerable by the optimization is a lot higher than when all joints are incorrect. The fact that a rotation of 48.70 degrees gives a lower error than a rotation of 45.84 degrees is due to the fact that in the first case the right arm is estimated incorrectly and the left arm correctly, while in the second case it is the other way around.

Due to the dynamic variance scheme, the global optimization is able to find the correct pose when all limbs are misaligned, given that the pose is already close to the



(a) Part 1 - frame 166 to 500

(b) Part 2 - frame 600 to 738

Figure 5.10: Location error of different annealing factors (see legend) on the partial dance sequences.

solution. When a few limbs are misaligned, the error of the misaligned joints can be higher and the optimization is still able to find the correct post.

Annealing scheme

The annealing scheme indicates the speed at which the temperature is decreased. A fast decrease in the temperature makes the system converge faster to an optimum, but this could cause the particles to converge to fast, which could result in a wrong solution. A low setting gives a slow convergence which could result in an incorrect result if the optimization has not converged yet after the maximum number of iterations has been reached. So both the annealing scheme and the number of iterations influence each other and both have an influence on the convergence of the optimization. While the number of iterations increases the time to converge towards an optimum, the value of the annealing scheme indicates the speed of convergence. So it is expected that with a low number of iterations, the optimization is only able to converge when the annealing value is set to a high value.

Using 15 iterations, we tested a few settings for the annealing scheme on the dance sequence, as the dance sequence provides some difficult parts in which the optimization could fail. It is clearly visible in Figure 5.10 that a low annealing factor of 0.5 does not perform well. In cases in which multiple optima exist in the energy function, the optimization does not have enough time to converge to a solution and so it estimates the pose over several optima. This happens when the arms are close to the body in frame 366 and when the legs are moving close to each other between frames 600 and 700. Both

situations have one optimal peak and a large basin containing a local optimum, like in the right case in Figure 5.8. The optimal peak is the pose in which all limbs lie exactly at the right position and the basin contains a pose in which one body part lies inside another body part, for example the arm lying inside the torso. This basin is large, as all poses with the arm somewhere inside the body have an equal error. A higher annealing factor performs better. The optimization converges faster towards the correct optimum and stays there. In a few parts of the sequence an annealing factor of 0.9 gives a slightly higher error compared to an annealing factor of 0.7. Two examples are shown between frame 670 and the end of the sequence, as can be seen in Figure 5.10. At this part, the higher annealing factor makes the optimization converge too fast to a wrong optimum, giving a higher error. Although in a few cases a higher annealing factor makes the optimization converge too fast towards a local optimum, it is negligible in contrast with the accuracy gain in the overall sequence. While both an annealing factor of 0.7 and 0.5 fail to estimate the pose between frames 416 and 500 and between frames 650 and 665 with large errors, the setting with an annealing factor of 0.9 gives a lot better results.

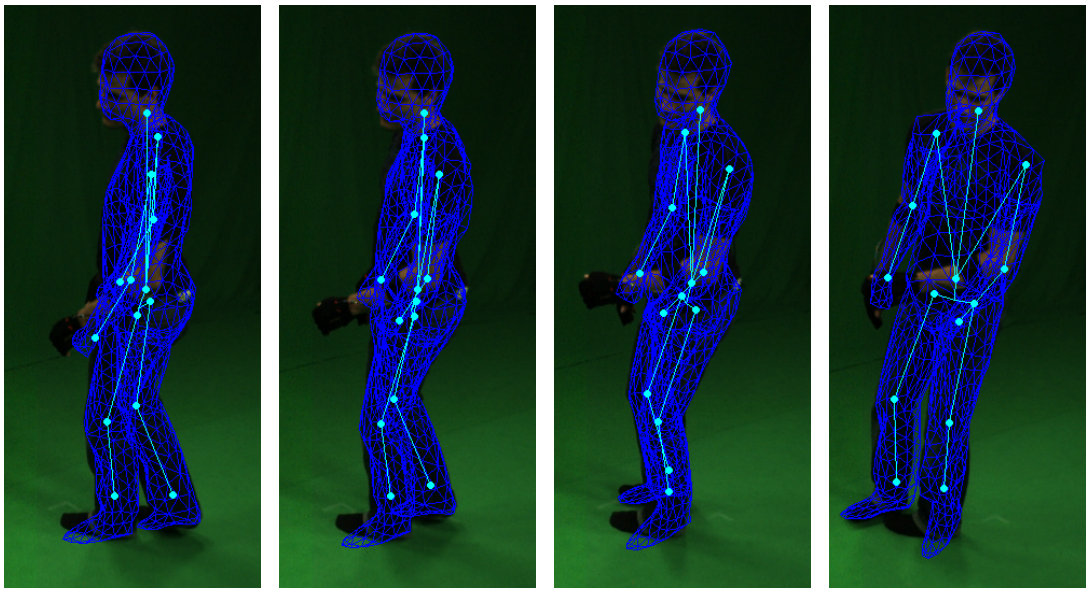
Prediction weighting factor

The weighting factor of the prediction in the global optimization ensures that large deviations from the predicted pose are not likely to occur. A higher value makes the distribution converge towards the predicted pose. This will help estimate poses in case of ambiguity or occlusions. This prediction factor makes the assumption that previous movements are approximately the same as the current motion. While in most cases this is a correct assumption, for example swinging an arm forward is a continuous movement, it could also be a dangerous assumption. A fast abrupt movement in the opposite direction, with a strong prediction to the other direction will give a wrong estimate towards the predicted pose. However, this prediction is especially useful for occlusions.

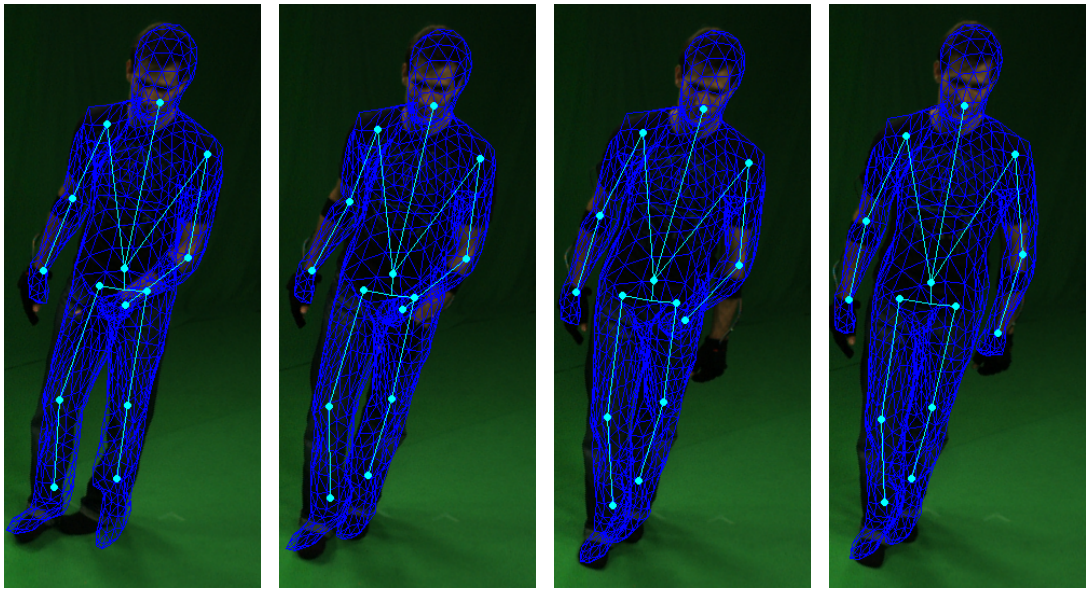
We will demonstrate the effects of the prediction factor on the walking sequence, where the arms are swinging next to the body. In one particular case, namely the turn, there is some occlusion from the left arm. We will see how the optimization performs on this sequence under different prediction settings. A high prediction should prevent the arm from getting stuck to the body, since it is predicted that the arm moves forwards.

In Figure 5.11 it is seen that with no prediction, the arm is not estimated correctly when it is partially occluded. As no prediction is used, most particles attain the pose of the left arm from the previous frame, when it is partially or totally occluded. In Figure 5.11(e) the pose is correct again, but this is because of the observed arm being at approximately the same location as the estimated arm. That is why the pose is lost again in the preceding frames, as can be seen in Figure 5.11(g). After a few frames the observed arm is correctly estimated again, because it is not occluded any more.

In a second experiment we used a prediction of 0.01, which stands for adding 1% of the prediction error to the energy of a particle. The results of the same frames as in the experiment with no prediction are shown in Figure 5.12. Although the pose of the arm is still not perfectly fitted to the observed arm, it is a lot better than when no prediction is used at all. It is capable of tracking the movement of the arm, giving better results.

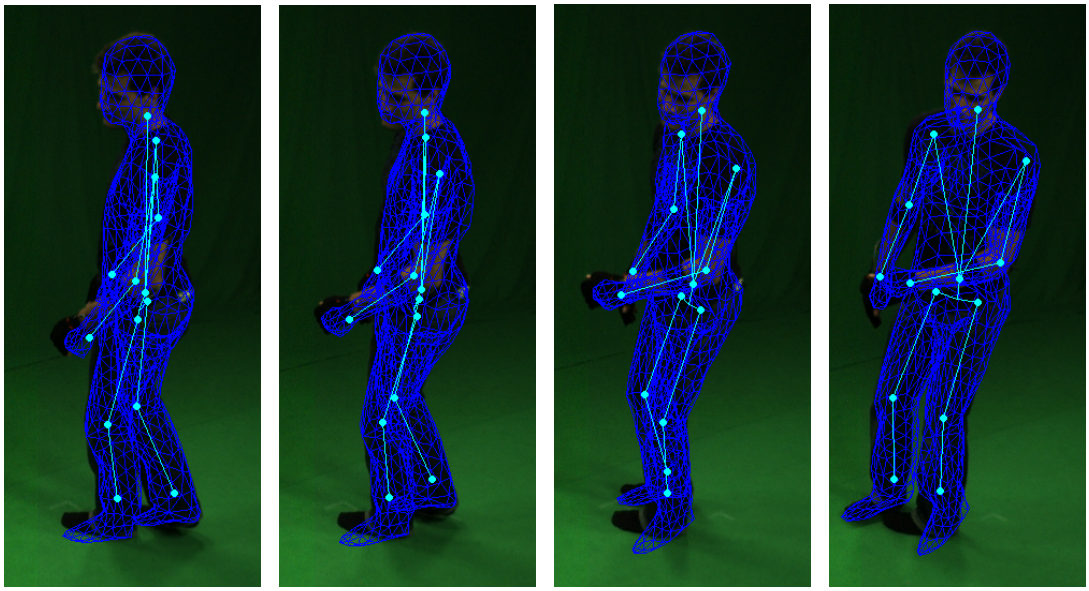


(a) Frame 178 (b) Frame 180 (c) Frame 183 (d) Frame 189

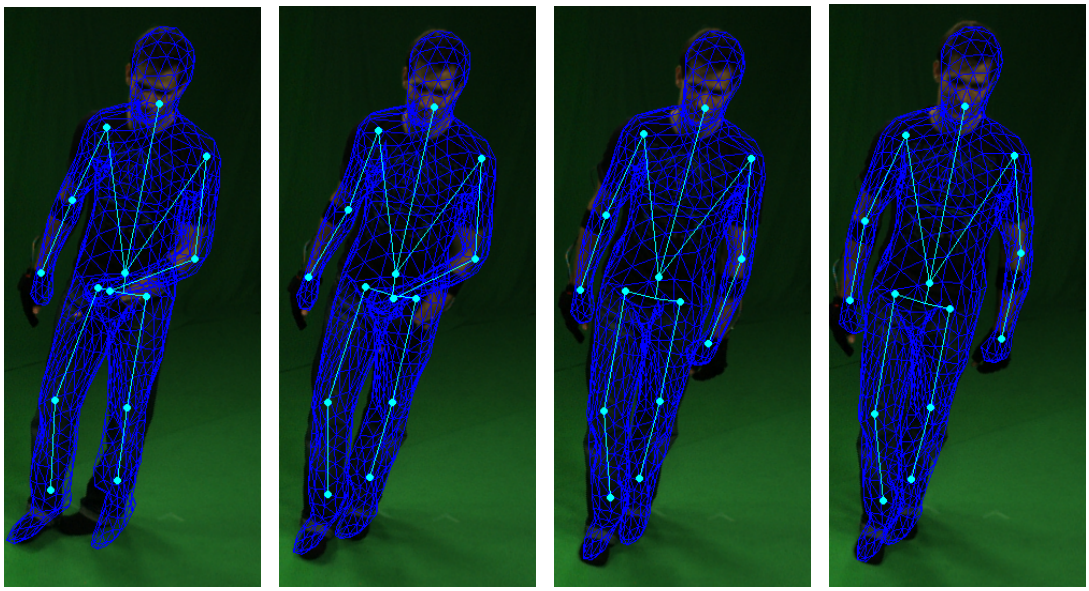


(e) Frame 195 (f) Frame 201 (g) Frame 211 (h) Frame 219

Figure 5.11: Global optimization with no prediction from camera view 4.

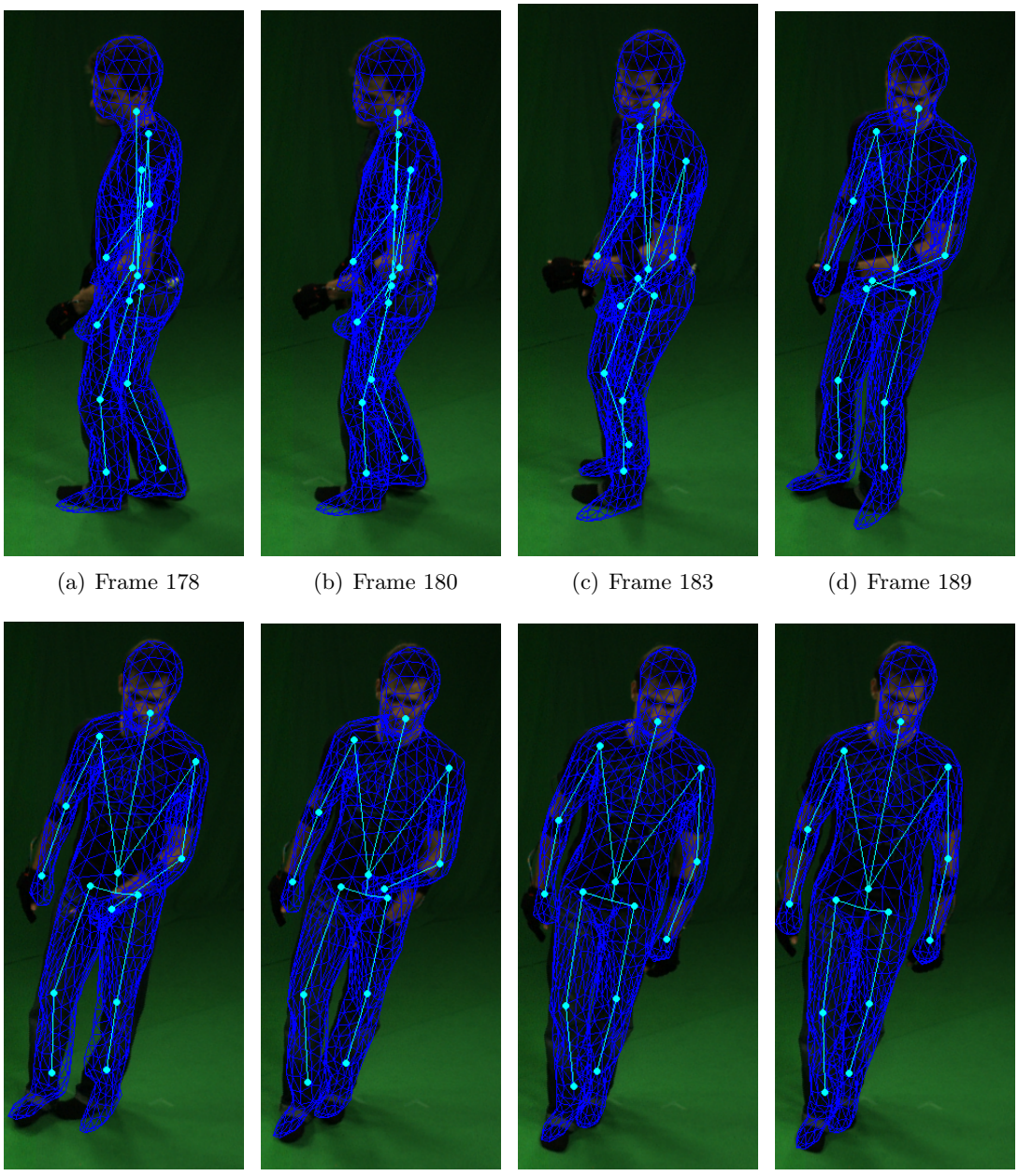


(a) Frame 178 (b) Frame 180 (c) Frame 183 (d) Frame 189



(e) Frame 195 (f) Frame 201 (g) Frame 211 (h) Frame 219

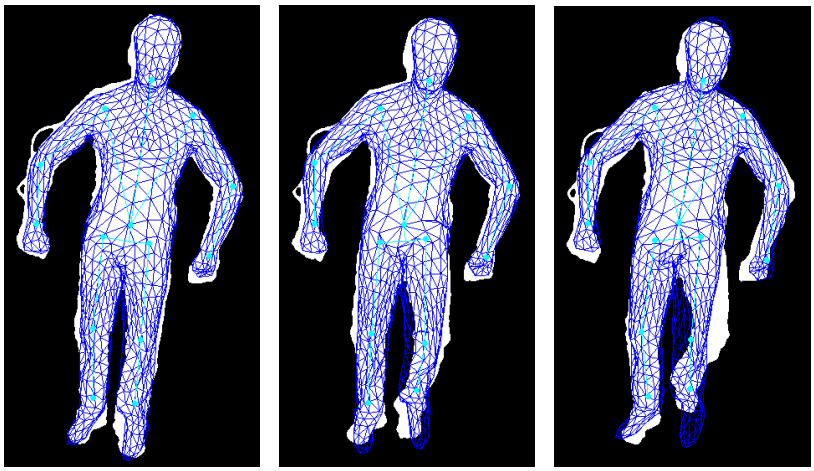
Figure 5.12: Global optimization with prediction of 0.01 from camera view 4.



(a) Frame 178 (b) Frame 180 (c) Frame 183 (d) Frame 189

(e) Frame 195 (f) Frame 201 (g) Frame 211 (h) Frame 219

Figure 5.13: Global optimization with prediction of 0.1 from camera view 4.



(a) Frame 370 - no prediction (b) Frame 374 - prediction of 0.01 (c) Frame 376 - prediction of 0.1

Figure 5.14: Error of using a prediction in the global optimization.

However, there is also a slight disadvantage of using the prediction. When a movement is long and occurs with the same speed, or when the a limb is not moving at all, the prediction assumes that the next position of the limb is according to the motion in the previous frames. With a high prediction factor this could lead to a late detection of a movement. This occurred at the end of the walking sequence where the observed person is standing still for quite a while and then he moves his left leg upwards. While using no prediction, the pose is immediately estimated correctly, while with a small prediction the moving leg is detected 4 frames after the movement has started, see Figure 5.14(b). The prediction causes the leg to stay at the same position as the last frames and only when the difference between the projected surface and the silhouette is high enough, the limb will move. This difference increases when the observed leg is moved upwards and fewer parts of the projected surface are overlapping with the silhouettes.

In a third experiment a higher prediction factor is used to see what the impact of a high prediction is on the estimation. A prediction factor of 0.1 is used and the results are shown in Figure 5.13. It can immediately be seen that, despite the use of the prediction factor, the results are just as worse as when no prediction is used. The error (a movement after a long period of standing still) created by having a prediction is worse when it is set to a higher value. Where a prediction of 0.01 starts to track the leg 4 frames after the movement has started, a higher prediction of 0.1 tracks the leg after 7 frames, shown in Figure 5.14(c). The observed leg has to move even further upwards to create a bigger gap between the projected surface and the silhouettes to overcome the error of having a large distance between the estimated pose and the predicted pose.

So while a prediction improves the quality of occluded body parts, it should not be set too high, as this would lead to wrong estimates again and extra errors caused by a movement after a period of no movements.

5.5.2 Global optimization experiment

In this experiment the global optimization with the best settings found in the previous experiments is tested. A full analysis is given on the results gained from the handstand and the dance sequence. Especially on the difficult parts of the sequences, as when using the global optimization combined with the local optimization and the local optimization fails, the global optimization has to find the correct pose again at all times. The settings which are used in this experiment are: number of particles is set to 25 * number of joints, number of iterations is 15, the initial variance is set to the variance of the Gaussian distribution over the previous poses, a dynamic variance factor of 0.4, an annealing scheme of 0.9 and a weighting factor for the prediction of 0.01.

Handstand sequence

Looking at the result of the handstand sequence over time, it can immediately be seen that the motion of the 3D model is not smooth. The model jitters a lot around its position. This is only noticeable by looking at the resulting video over time [4]. The pose looks accurate when viewing the results of each frame individually. This is typical for ISA, where the pose is sampled from a distribution. In each frame, the particle set is randomly generated from the previous pose and the predicted pose. After generation, the particles are randomly distributed. After a fixed number of iterations, the estimated pose is calculated from the set of particles. However, since the number of possible poses is infinite, and all particles are randomly generated and distributed, the estimated pose is never the same. Even when we estimate the pose for the same frame twice. So it is unlikely that for two subsequent frames the estimated poses have a smooth transition.

Despite the jitter, the estimated pose in each frame is accurate up till the point when the handstand starts, as seen in Figure 5.15. During the handstand, the average error increases has increased, but at most frames the error of all joints lies below 100 mm. It can be noticed from the handstand sequence that the right hand and left foot are at not well estimated at the start of the full handstand. Despite a correct location, the orientation of these limbs is wrong. While the motion during the handstand is very small and the pose does not change a lot, the errors are not corrected by the global optimization. This is caused by a very small deviation in the poses during the handstand, which keeps the initial variance low and the predicted pose at its current estimation.

Like in the local optimization case, it is also hard for the global optimization to estimate the correct pose when the observed person is in a crouching position, after the handstand ended. There is some ambiguity in the pose, as the silhouette at this part in the sequence is a white blob. Therefore, several different poses can have the same or approximately the same error. The error decreases slowly between frames 410 and 430 as the person is standing up on his feet again, with the exception of the right arm which takes longer to correct. At frame 440 the person is standing straight up again and the estimated locations of the joints are with an average of 43 mm off from the ground truth. In the end, only two joints are not correctly estimated, namely the left and right hand.

[4] A video can be viewed at http://www.staff.science.uu.nl/ tan00109/student/2012_m_jeffrey/

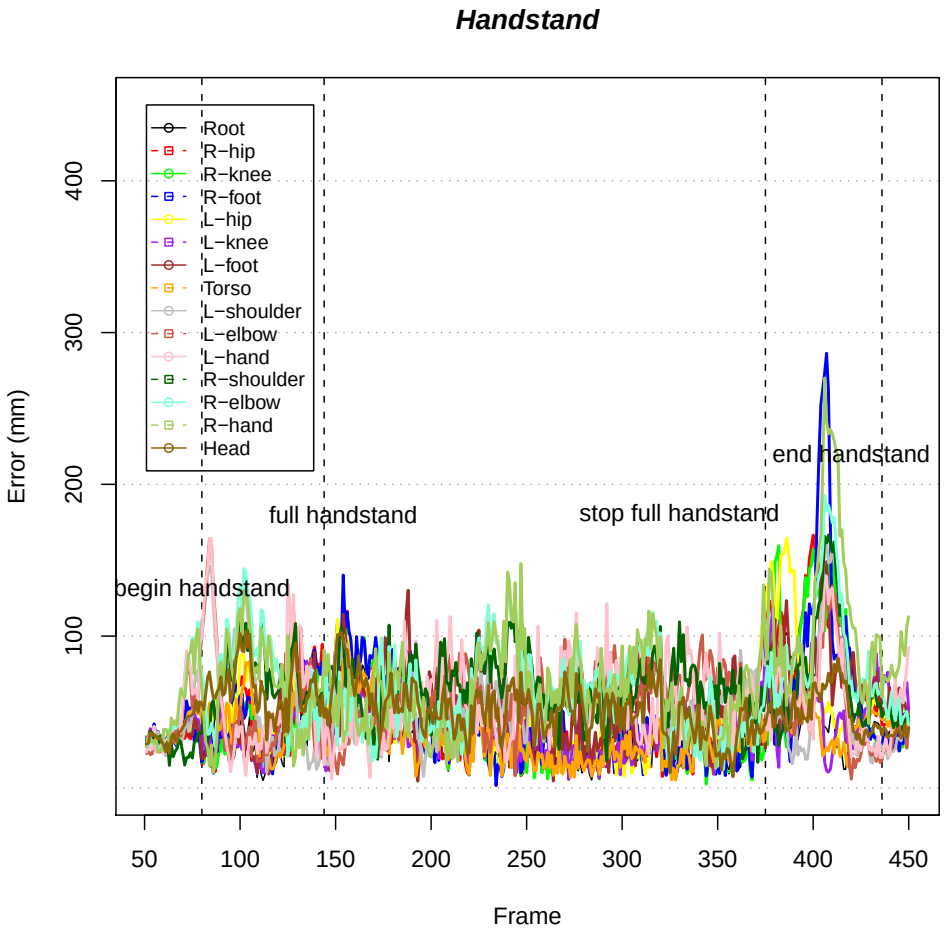


Figure 5.15: Joint location error using global optimization on the handstand sequence.

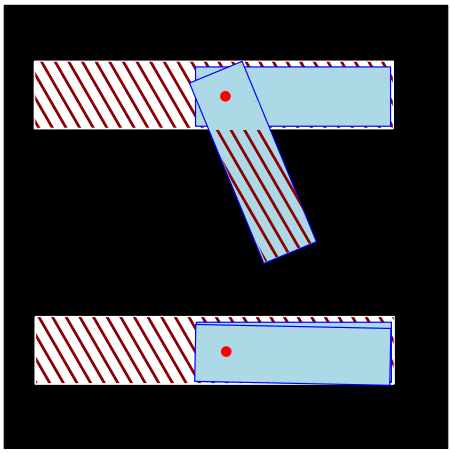


Figure 5.16: With a small variance, the global optimization converges to the nearest best solution.

We have seen that during the change in position of the person, the global optimization could find a pose, but the smaller limbs are the hardest to estimate. The main reason for this is the small addition of the error of a foot or hand to the overall error and so the difference between two same estimated poses with a only a difference in the hands or feet is very small. This mainly happens when the hands or feet are close to other parts of the body. A pose in which the hand or foot lies inside the other part of the body, only gives a slight extra error to the pose. This small error is not corrected by the global optimization and so this error will propagate through the next frames. Eventually, the arm or leg will move away from the other body part and so the hand or foot have to be optimized again as it will not lie inside a body part any more. Therefore, it searches for the best solution. This is dependent on the variance and when this is to low, it can converge to a local optimum, which is a pose in which the hand or foot lies inside the arm or leg. This error is shown in Figure 5.16

Another observation is the variation in the rotation of the feet and hands. Since these parts contribute little to the total error, a change in these parts does not affect the total error greatly. Therefore, in some cases these parts do not rotate, while the observed parts are rotating, giving only a slight difference in rotation between frames. This causes a small variation across the frames, while a bigger variation is needed to spread the particles.

Dance sequence

Just like the handstand sequence, the ISA produces jitter for the dance sequence. The results are shown in Figure 5.17 and it is seen that the global optimization performs well on this sequence. The peaks mostly resemble a bit higher errors for the hands and feet, but all below 120 mm. No great errors occur and no limbs are estimated incorrectly with the exception of the hands and feet. Just as in the handstand sequence, the global

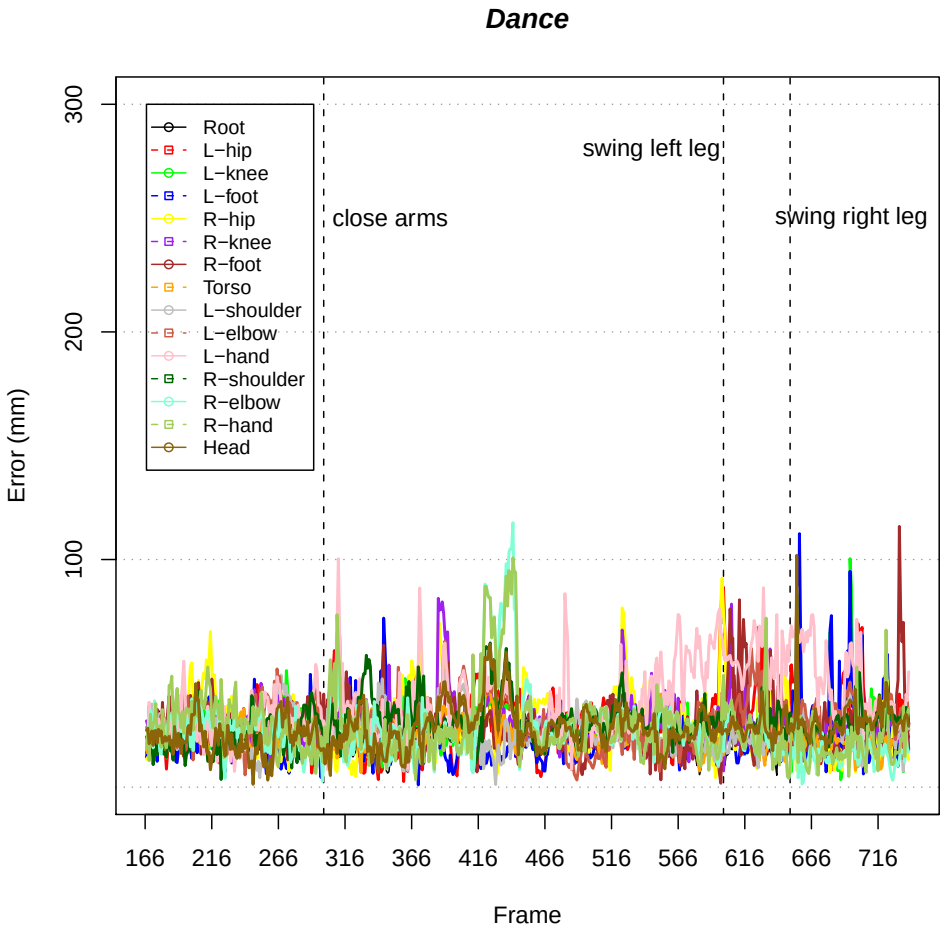


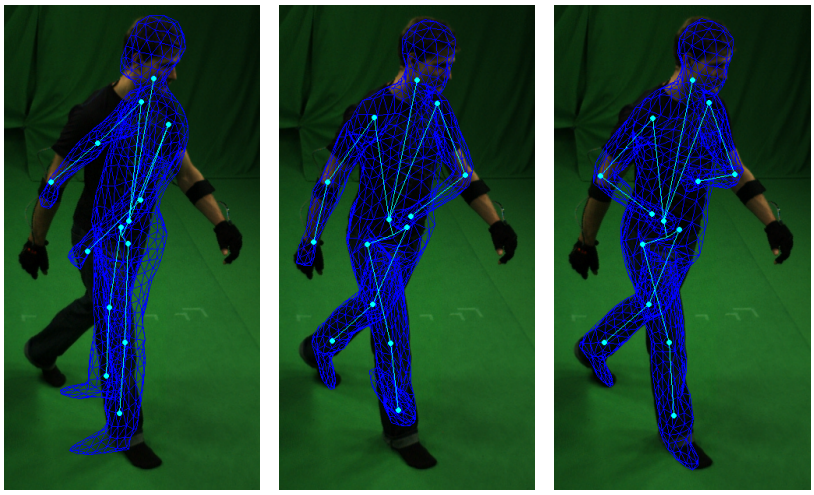
Figure 5.17: Joint location error using global optimization ont the dance sequence.

optimization sometimes can not find the correct orientation of the hands and feet.

5.5.3 Initial pose

The theory in Section 4.5 explained how the initial pose was estimated. It contains two assumptions to restrict the amount of possible poses. While it provides a quick way of finding the initial pose, the assumption that the model is already in approximately the same pose as the observed person in the first image is hard. This means that the pose in another video sequence can't be estimated when it does not contain an observed pose which is approximately the same as the model. This experiment will be carried out to see whether it is possible to find an initial pose using the global optimization, despite the huge amount of possible poses.

Before testing on the whole search space, the model will be put on the correct global



(a) Beginpose (b) 20.000 particles (c) 100.000 particles

Figure 5.18: Estimation of the initial pose using different numbers of particles.

position and in the correct global orientation. The search will only be on the joints and global rotation to see if it is already possible to find the pose in this situation. The assumption is made that the position of the person is correct and that the model and observer person are both standing up. Still, a large number of particles is required. For this experiment we used 20.000 particles. Initially, all joints in each particle are randomly rotated between $(-\pi, \pi]$ and a random orientation is for the y-axis between $(-\pi, \pi]$. The initial variance is set to zero, as we manually mutated all particles using the randomized rotations. The settings for the global optimization are: 25 iterations, a dynamic variance factor of 0.4, an annealing scheme of 0.7 and no prediction is used. We took the walking sequence and started to find the initial pose at frame 137.

The results of the run are a correct global location and global orientation, see Figure 5.18. The head, torso and legs were also found correctly. However, the arms and feet were not. They got stuck in other body parts. We have already seen in the global optimization experiments that the ISA is prone to this kind of error. To make sure that this is not the cause of the number of particles, we tried the same setting, but with 100.000 particles. Although the results of this setting are better for the feet, the arms are incorrect.

We can conclude that despite a high number of particles, it is not guaranteed that a correct initial pose can be found. This also provides the insight that despite this is a global optimization, it is not always capable of finding the global optimum, at least not for a reasonable number of particles within a reasonable time.

5.5.4 Evaluation

Several experiments have been carried out on the settings of the global optimization. We have seen that more particles lead to better estimations, but when the optima are not distributed well in search space, the optimization has trouble converging to the right optimum. When the number of particles is set to high, it can even lead to worse results, since particles are kept among several optima. Setting the number of iterations higher results in a better convergence and setting the annealing scheme lower results in a faster convergence. This solves the problem of having particles at several optima, but in other cases this leads to wrong estimates, as the optimization converges to fast towards the wrong optimum. In Figure 5.8 only the left case is the most suitable distribution of optima for the dynamic variance scheme. This scheme makes sure that the mutation of the particles is according to the variance of the resampled particles in each iteration. Since it is dynamic it is dependent on the distribution of the particles in search space. In the second case, this means that when particles are in two (or more) optima, it will yield a high variance. In this case the optimization can not converge to an optimum. In the last case, most particles reside in the basin and a few in the peak. This will yield a variance which keeps most particles in the basin, resulting in a pose which resembles the local optimum in the basin. Despite the fact that the dynamic variance scheme performs well on the first case it is still prone to errors for the other two cases. While in most situations during the video sequences, the first case occurs and so a correct pose is estimated, there are possibilities that the other two cases will also occur. In the handstand sequence, we have seen that this occurs when the person is in a crouching position. In this position, the silhouettes are mostly white with few contour lines, and so there is ambiguity in the pose.

Apart from the jitter, the global optimization tracks the pose well and errors are fixed in subsequent frames. However, sometimes it is hard to find the correct pose for the smaller limbs, namely the hands and the feet. These limbs are small compared to the rest of the body, contributing only marginally to the error of the pose. Therefore, a particle representing a pose with an incorrect hand is also likely to be chosen for the next iteration.

A variety among particles at the start of each frame is important to ensure a good spreading. This variety is created from the previous poses, and if all these poses lie close to each other, there is a small initial variance. While a small initial variance is already sufficient in most cases, as the dynamic variance scheme spreads the particles further if necessary, a large error can not be fixed by the global optimization. An example would be the hand which lies inside the arm. The correct pose can be reached by rotating the hand by 180 degrees. First of all, the variance is low such that particles are not spread wide into search space. Secondly, the prediction weighting factor prevents particles from adopting a pose which is far away from the predicted pose. We have seen that this has mostly happened to the smaller limbs.

The prediction gives a guidance for the estimation in case of occlusion or partial occlusion. However, as we have already stated, it also limits the amount of freedom in spreading. The value of the prediction weighting factor should not be set to high, as

this would result in incorrect estimations when an abrupt change in movement occurs, for example after a slow motion or no motion at all.

It can be concluded that the global optimization is capable of finding the entire correct pose, given that the estimate pose of the previous frame is close to the observed pose in the current frame. It is also capable of finding the correct pose which contains only a few incorrect limbs, with a higher error tolerance. In case of occlusions, the prediction weighting factor gives a guidance for the occluded limbs. However, when there is no approximation on the pose at all, no correct estimation can be given and so no correct initial pose can't be given when the model is not in the approximately the same pose as the observed person.

5.6 3D Pose estimation

In this section several experiments are carried out using the combination of the local and the global optimization. First an experiment is done to test the result of our implementation of the whole method with the results from the paper from Gall *et. al* [1]. All parameter settings used in our implementation are set to the best settings we have found so far for both the global and the local optimization. To following list will provide an overview of the best settings:

- Local optimization
  - Stabilising factor: 1.0
  - Convergence precision: 0.0001
  - Maximum number of iterations: 30
- Global optimization
  - Number of particles: 25
  - Number of iterations: 15
  - Initial variance: variance of Gaussian distribution
  - Dynamic variance scheme: 0.4
  - Annealing scheme: 0.9
  - Prediction weighting factor: 0.01

For the limb evaluation, no setting has been tested yet. Therefore, we will use a threshold of 500 which corresponds to an error of approximately 22 millimetres[5]. This is also the setting which is used by Gall *et al.*. The method is tested on the handstand and the dance sequence from the MCD dataset and on the walking sequence from the MPI08 dataset..

[5]By taking the distance in the 3D world between the top point of the 3D model and the lowest point and assuming the person is approximately 1.80 meters, a distance value of 500 in 3D world can be translated to approximately 22 millimetres.

The second set of experiments will be the same as the first experiment, but in this experiment the texture correspondences are included in the local optimization process. Once again the same sequences as in the first experiment are used. This experiment will show whether the texture correspondences give a better accuracy and if the increased computation time is still in line with the amount of increased accuracy.

The last set of experiments will be based on new techniques implemented into the method. These new techniques are based on observations made in the previous experiments and on the mathematical properties of the method.

5.6.1 Original results

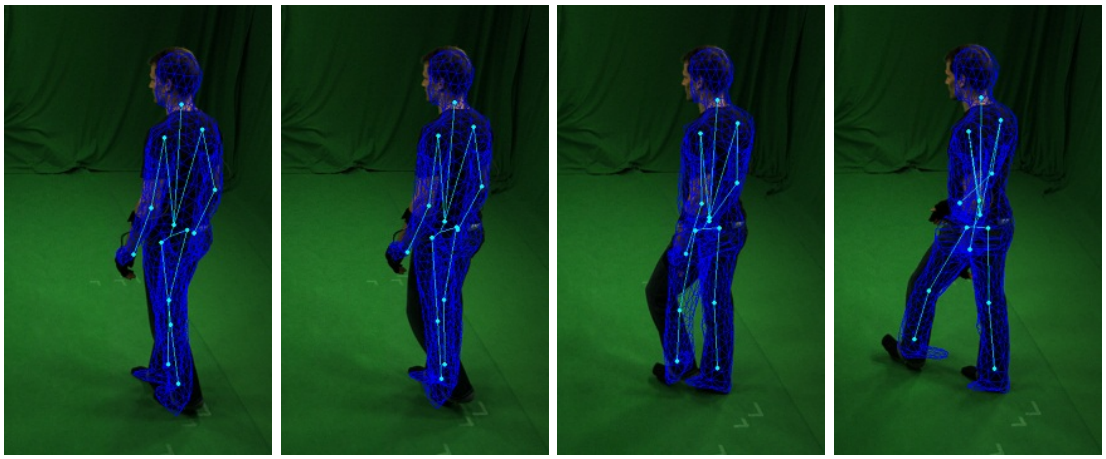
In this first set of experiments an evaluation is given on the entire method which is the same as in Gall *et al.*, with the exception of the correspondences. Only the contour correspondences are used as these are the most important correspondences. The whole local optimization relies on the contour correspondences and the texture correspondences could provide an additional quality to the results. In Subsection 5.6.2 an experiment is done on the method which includes the use of the texture correspondences.

Some slight changes are already made to the method to give a better estimation. First of all, when a joint is misaligned, all joints up to a joint of 3 degrees of freedom are labelled as misaligned, see Section 4.3. The second change is that the initial particle set is constructed from an interpolation between the previous pose and the predicted pose, see Section 4.4.2.

Walking sequence

The pose of the observed person is estimated quite well over the entire sequence. Only one big error occurs. After the first turn, the observed left leg is moved forwards, passing the observed right leg in the progress. During the passing, the whole left leg gets rotated around its axis, shown in Figure 5.19. While the observed leg in the subsequent frames bends during the movement, the left leg of the model can't bend in the same direction since the leg is rotated. Meaning it attains the pose of the observed right leg. This small error propagates over a few frames and after a while the whole leg is rotated by 180 degrees. At this point the leg can bend in the same direction as the observed left leg, giving a correct pose again. When the person moves his right leg past the left leg again, correspondences can be created which rotates the left leg back again to its proper position. This is done by the local optimization. Between the two correct poses (while the leg is rotated 180 degrees) the correct pose for the leg can't be found, as this would mean the global optimization has to find a rotation of 180 degrees. For this purpose the variance is too small to spread the particles from the local optimum towards another optimum.

One other smaller error occurs, which is the rotation of the left hand. While the joint location of the hand seems correct, its rotation is not. At the start of the sequence the hand is in the correct rotation, however, as soon as the observed person starts to walk, the left hand gets rotated by 180 degrees over the next few frames. While both the local



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

Figure 5.19: Propagation of a rotated leg over time.

and the global optimization are prone to errors in the smaller limbs, the optimization does not fix this error. Therefore, the hand stays rotated by 180 degrees through the entire sequence.

Handstand sequence

Up till the point where the person is starting to perform the handstand, the estimation is entirely done by the local optimization. After frame 86, which is when the person is halfway performing the handstand, the evaluation function marks the torso as misaligned in each frame till frame 408, and so the entire pose is estimated by global optimization. The cause of this is the shirt of the observed person not being in line with the observed person itself. As such, the geometry of the torso is not identical to the observed torso, giving correspondences with a large error as shown in Figure 5.21. By initializing the global optimization at each frame, the results of this experiment, shown in Figure 5.20, are the same as in the case of using only the global optimization.

Dance sequence

The error between our results and the ground truth is fairly low, as can be seen in Figure 5.22. The average error stays below 50 mm, with some joints having an error bigger than 50 in some frames. While the estimated arms at some points in the sequence have a bigger error compared to other parts of the body, they still track the observed person well.

At frame 408 the lower right arm is estimated incorrectly and this is not corrected by the global optimization. In the next few frames, the global optimization is again initialized on this part of the body, but no correct pose is found. In contrast, the estimation leads towards an estimated pose in which the lower arm resides in the upper

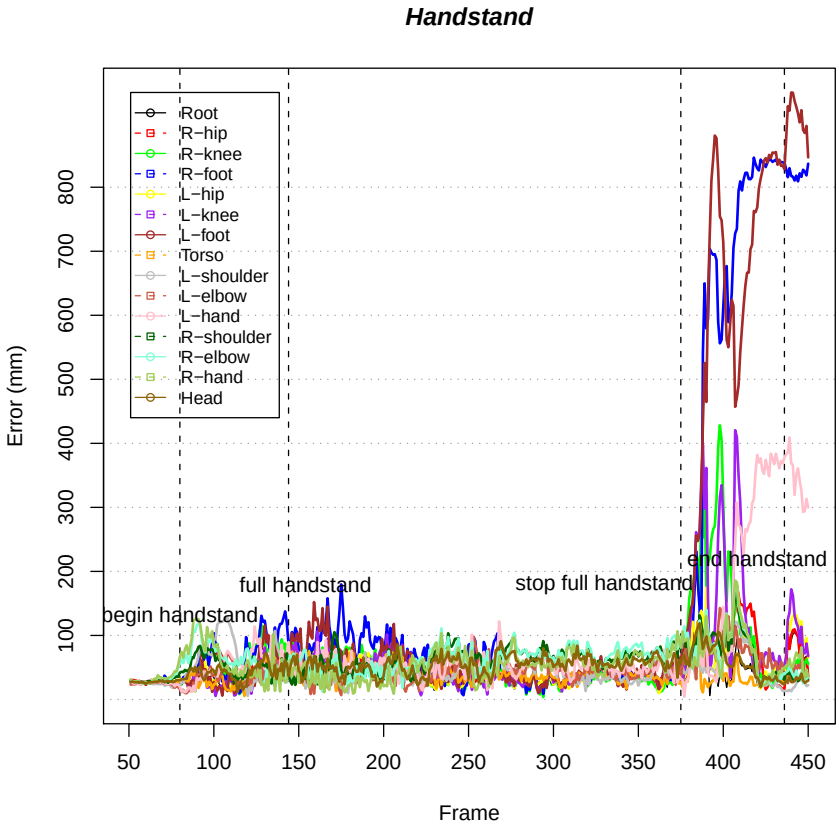
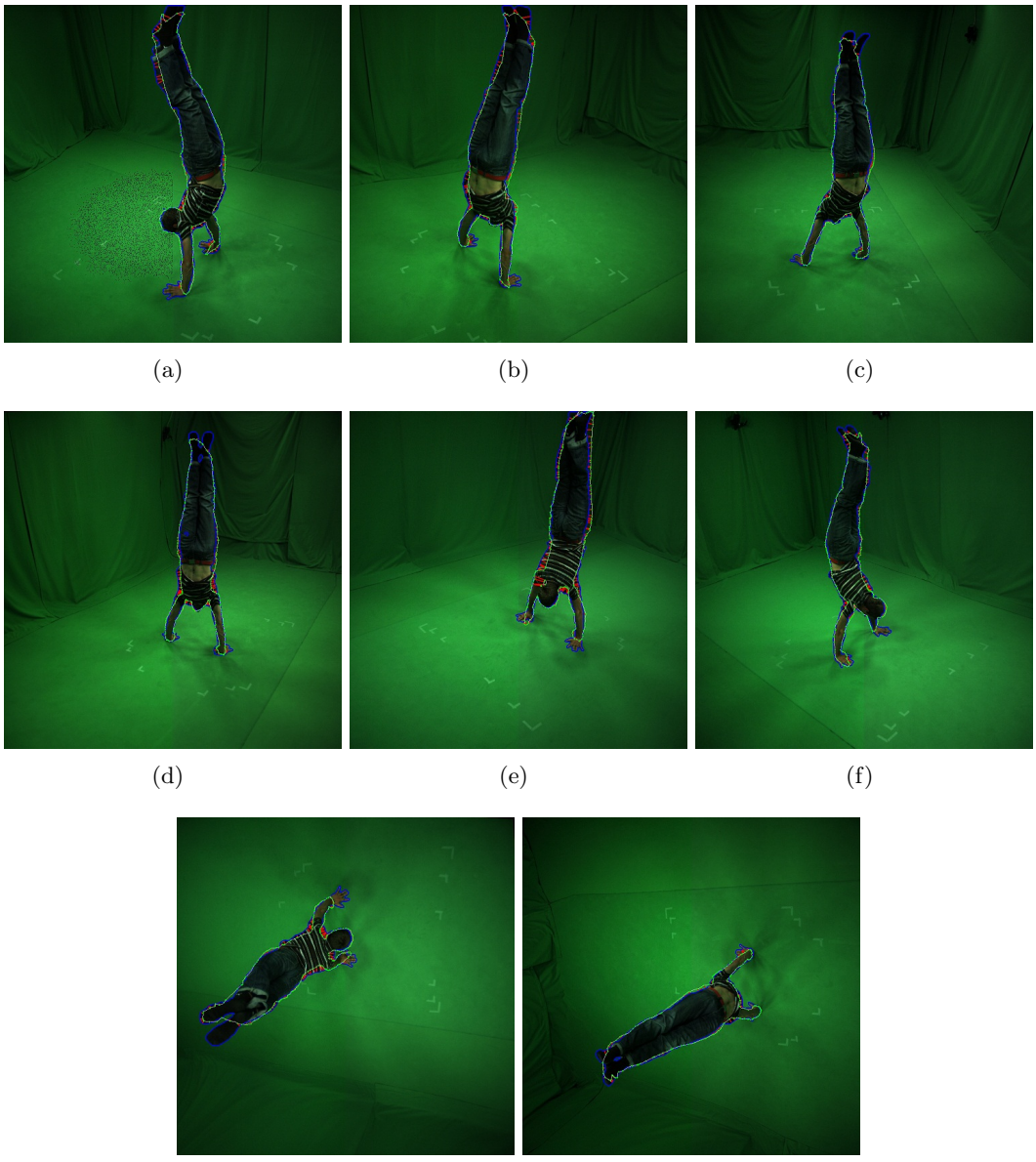


Figure 5.20: Joint location error using both optimizations on the handstand sequence.



(a) (b) (c)

(d) (e) (f)

(g) (h)

Figure 5.21: Large torso error between the model and the observed person, due to a change in the geometry of the observed torso. The blue line is the contour of the silhouette, the white line is the contour of the projected model and the red lines are the errors.

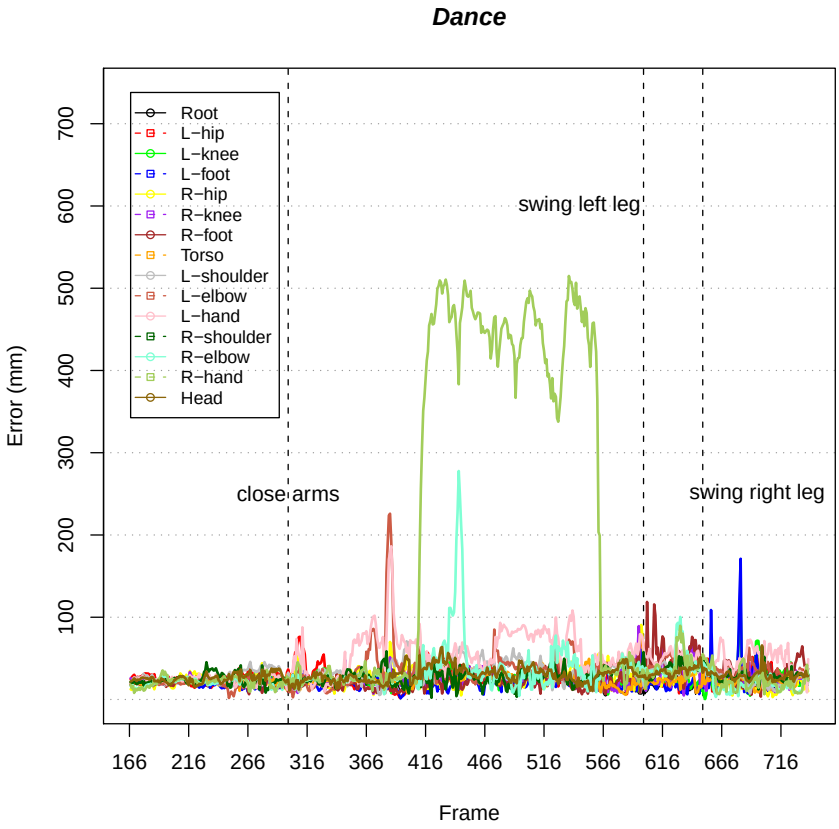


Figure 5.22: Joint location error using both optimizations on the dance sequence.

arm. Inspecting this leads to a problem in the global optimization. Whenever the variance, which distributes the particles, is too high or multiple optima lie close to each other, the particles are spread among several optima as in Figure 5.8b). The right arm is estimated correctly again at frame 563. At this point the observed arm is next to the body and as such, the estimated lower arm can move through the body, without having a change in error for the arm, as every pose with the estimated arm somewhere inside the torso gives the same error. Eventually, this leads to a correct pose again for the left arm, with the exception of the hand, which is still rotated by 180 degree.

At frame 605 the observed lower right leg is moved towards the observed upper right leg, by pulling up the lower leg. During the movement, the global optimization moves the lower leg slightly more into the upper leg, as there is almost no difference in error between the global optimum and a solution in which the lower leg is inside the upper leg. The cause of a small difference between the global optimum and the local optimum is that the observed lower leg is in such a position, that in each view another part of the body is covering the silhouette part of the lower right leg. This means that the location of the lower right leg does not contribute to the error (except of course when it lies outside the silhouette), because another body part already covers the observed lower right leg. This is only a small error and the optimization is capable of recovering from this small error, but despite the fact that the optimization recovers, the right foot gets rotated by 180 degrees, such that it lies inside the lower leg.

Evaluation

	Distance		Angle		Global Opt.		Runtime
	Avg	Dev	Avg	Dev	Entirely	Total	(hours)
Dance	35.83	57.40	35.01	53.27	61	251	13:58
Dance (+ texture)	24.38	7.98	8.43	9.22	42	200	13:30
Handstand	68.73	110.80	35.48	41.78	356	368	29:25
Handstand (+ texture)	48.31	27.99	32.55	30.86	350	375	31:12
Walking	-	-	-	-	136	258	5:51

Table 5.7: Results of the method. Distance is in mm and angle in degree.

During the tests, several cases occurred in which the method performed poorly. First, when a part of the body, for example an arm, is very close to another part of the body, the local optimization will move the arm inside the other part of the body. The wrong estimation is due to the contour correspondences, which always seeks correspondences between the two closest points of the contour of the projected model and the contour of the silhouette. The global optimization is capable of solving this in most cases, however, in some cases the particles are spread and kept among two or more optima. This results in a pose in between the optima, not necessarily being an optimum.

In all experiments we have seen that the global optimization is initialized many times on the whole body, especially in the handstand sequence. When we inspect the cause

of this, we see that the threshold is causing this. In most cases the torso is, according to the evaluation function, a wrong estimate with an error above our threshold. As a consequence, the whole upper body part has to be re-estimated, resulting in more than 50% of the joints to be misaligned. According to [1], this means the whole body has to be re-estimated. However, when we look at the error between our estimation of the joint and the ground truth after the local optimization, we see that the torso is very close to the ground truth. Actually, we can say that the torso is the most accurate joint, along with the root, shoulders and head.

When the movement between two frames is fast, the model also has to move fast. This is not a problem if the arm moves far away from other parts of the body. However, if the movement is close to another part of the body, it is likely that the local optimization will find correspondences between the arm and other parts of the body, because the silhouette part of the body is closer to the arm than the silhouette part of the arm. This error does not occur instantly, but over a few frames. Unfortunately, the error is not detected early on by the evaluation function, as this function uses the contour correspondences to detect a wrong limb. If the correspondences are already wrong, but the error is below a threshold, the error is not detected. This indicates that this method is prone to errors for movement close to the body, as these faults are not detected in time. Also, the evaluation is heavily dependent on the correspondences. When the optimization has converged with the wrong correspondences, the evaluation function will detect it as a correct pose and so the global optimization is not initiated.

5.6.2 Texture correspondences

This section will cover the method using the texture correspondences. As texture correspondences are created from SIFT features, it takes additional time to compute the pose in each frame. An evaluation is given on the error and the quality of the results by using texture correspondences, especially by looking at the number of initializations of the global optimization and at the amount of time needed to compute the pose for each frame.

Evaluation

While SIFT features must be calculated for each frame, it is for sure that the local optimization takes additional time to compute all correspondences. The results of the method with the use of texture correspondences are shown in Table 5.7. The quality of the dance sequence has improved, such that no errors occur at al. The number of initializations of the global optimization have decreased, aswell as the number of global optimizations on the entire pose. Despite the fact that the texture correspondences take time to compute, the method is able to estimate the pose faster than in the case without the texture correspondences. The quality of the estimations after the local optimization have improved, such that less initializations of the global optimization are required, resulting in a faster estimation. While the method without the texture correspondences fails to estimate the orientation of the hands and feet at some points

during the sequence, the method with the texture correspondences has no trouble finding the correct orientation of the hands and feet.

The texture correspondences also increase the quality for the handstand sequence. This is due to the fact that in the last part of the sequence the texture correspondences have increased the accuracy of most limbs. In the other parts of the sequence, the results are approximately the same. This means that while the observed person is standing upside down, the torso is misaligned at each frame, which in turn initializes the global optimization. Due to the same number of initializations of the global optimization and the additional computation time for the texture correspondences, the runtime has increased for this sequence.

5.6.3 Algorithmic changes

The last set of experiments involve several changes in the algorithm. The first experiment will cover the change in the error function of the correspondences of the local optimization. A displacement of the hand by 50% of its size is, when we look at the ratio between size and displacement, a worse displacement than when a bigger part of the body, the torso, is displaced by the same amount of displacement. Using a static threshold, therefore, is not a good measure as this threshold will be used for all limbs. We will use a dynamic threshold according to the size of the body part, i.e. the torso could hold larger errors, while only small errors are allowed for small limbs. Also, we will drop the condition of estimating the whole body pose if at least 50% of the joints is misaligned. A wrong estimation of two hands and a foot lead to the labelling of both arms and one leg. This counts to more than 50% of the total number of joints. However, only the hands and foot are misaligned, which is not the cause of a wrong estimation of the whole body, but of the arms and leg. To make sure that there is no bias from the texture correspondences, we left them out in the following experiments.

Dynamic threshold

To compute the dynamic threshold, the size of each limb has to be calculated first in a preprocessing step, which can be done once when the mesh model is loaded. Each limb is assumed to have a cylindrical shape, where the line formed by the location of the joint belonging to that limb and its child goes through the centre of the cylinder. The size of a body part is determined by the average shortest distance from each vertex influenced by that limb, to that line. In other words the radius of the cylinder is calculated. For the last joint on each kinematic chain (head, hands and feet) no predecessor exists. Therefore, the line for these body parts is formed by the location of the joint and the mean center of all vertices influenced by this joint. Now that the average distance to each limb has been calculated, the dynamic threshold can be set. This dynamic threshold is in ratio with the average distance to each limb. Several thresholds are tested, ranging from 10% to 40% of the average distance to each limb.

The results of the experiment on the handstand and the dance sequence using the dynamic threshold are shown in Table 5.8. First an evaluation on the handstand sequence

Sequence	Threshold	Distance		Angle		Global Opt.		Runtime (hours)
		Avg	Dev	Avg	Dev	Entirely	Total	
Handstand	normal	68.73	110.80	35.48	41.78	356	368	29:25
Handstand	10%	42.18	39.47	31.65	50.72	44	349	23:42
Handstand	20%	40.58	21.79	21.30	34.23	9	194	4:25
Handstand	30%	46.25	24.47	51.48	92.60	5	66	1:59
Handstand	40%	40.78	20.62	24.31	37.34	1	25	1:13
Dance	normal	35.83	57.40	35.01	53.27	61	251	13:58
Dance	10%	43.72	83.48	28.63	46.70	18	485	24:48
Dance	20%	31.24	41.32	26.09	45.01	0	221	4:21
Dance	30%	26.94	15.38	30.47	52.12	0	92	2:21
Dance	40%	36.77	79.33	26.62	46.56	0	38	1:14

Table 5.8: Results of the method using a dynamic threshold for the limb evaluation function.

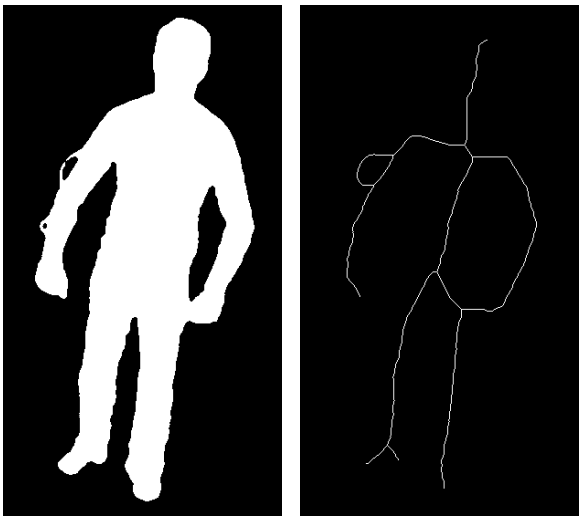
is given before an evaluation is given on the dance sequence.

For the handstand sequence, the number of initializations of the global optimization is high, as was shown in the handstand experiment in Section 5.6.1. This was caused by a false labelling of the torso. Using a dynamic threshold, the number of initializations decreases, since the torso can hold larger errors and so no false initializations of the torso occur. Because we dropped the condition of estimating the entire pose when more than 50% of the joints are misaligned, the number of initializations of the global optimization on the entire pose has decreased drastically. Due to this, the runtime decreases.

Compared to the results from using a static threshold (threshold of 500), the results of the dynamic threshold are better. The main reason for this is that less global optimizations are performed on the entire pose, even in the last part of the handstand sequence where it is difficult to estimate the crouching pose. As a result, the limbs which are correct can not be changed, and so less poses are possible. In the case of the crouching position, this means that there is less ambiguity in the poses, since some limbs are fixed already.

The results are only worse for the dance, when using a dynamic threshold of 10%, compared to the static threshold. The global optimization is initialized too quickly on the body parts, leading to wrong estimations, while the results from the local optimization are still correct. It is also initialized too many times without being required to be initialized, resulting in a higher runtime. While the errors for this sequence do not differ greatly from the results gained using a static threshold, the amount of computation time has decreased drastically when using a dynamic threshold.

The dynamic threshold is invariant to changes in environments, where the static threshold has to be set for each dataset. No concrete relation on distance can be given between different datasets. Where a distance of, for example 500, in one dataset relates to 22 millimetres in the real world, the same distance of 500 in another dataset relates to different distance in the real world. This means that for a static threshold, the threshold



(a) (b)

Figure 5.23: Thinning of a silhouette

has to be set for each dataset. The dynamic threshold doesn't need to do that, as it is dependent on the size of each limb.

It can be concluded that using a dynamic threshold is beneficial, because the number of false initializations of the global optimization decreases, resulting in a lower runtime. The dynamic threshold ratio should not be set to high, as this would put a high threshold on each limb. This means that errors are detected to late, or it might be the case that the errors are not detected at all. We have found that a dynamic threshold ratio of 30% gives a faster method while the results are still accurate.

Skeletonisation

The second change involves adding an additional type of correspondence to the local optimization. Correspondences are created between the contour of the silhouette and the projected model. In some cases the number of correspondences is low as part of the model is overlapping and so no contour is available. The inner structure of the model is not taken into account during optimization. To give another guideline for the limbs, when few correspondences are available or none at all, we will use the skeletonisation of the silhouette to create correspondences between this skeletonisation and the skeleton of the 3D model. The skeletonisation is done on the silhouette by using a thinning algorithm [31]. The thinning algorithm gives as result a thin version of the silhouette. An example is shown in Figure 5.23.

For each limb, we sample several uniformly distributed points between the joint and its predecessor. A correspondence is created between each sampled point and the closest point on the skeletonisation.

Like the texture correspondences, the skeleton correspondences should only have an

influence at the first iteration of each frame and after the first iteration it should only provide a slight influence. For the skeleton correspondences the amount of influence is similar to the influence used for the texture correspondences, see Section 4.2.3 where we called the amount of influence the weighting factor. This influence is dependent on the number of contour correspondences. In the following experiment the skeleton correspondences are added to the method while the texture correspondences are left out. In the previous experiment a dynamic threshold of 30% provided a faster and more accurate estimation. Therefore, we shall use this in the following experiment. We tested several options for the weighting factor in the first iteration and in the following iterations.

	Weight in it.		Distance		Angle		Global Opt.
Sequence	1st	2nd - ..	Avg	Dev	Avg	Dev	Total
Dance	-	-	26.94	15.38	30.47	52.12	92
Dance	1.0f	0.2f	30.53	25.78	25.59	46.47	91
Dance	1.0f	0.1f	28.58	26.84	26.68	47.86	84
Dance	0.5f	0.1f	26.01	11.78	15.62	29.90	88

Table 5.9: Results of the method using additional skeleton correspondences in the local optimization.

The results are shown in Table 5.9 and for clarification we have added the results of the sequence without the use of skeleton correspondences. It can be seen that the use of the skeleton correspondences has approximately the same location error as when these correspondences are not used. However, the rotation error has decreased when using skeleton correspondences with a small weighting factor. The cause for a better estimation is the correct estimation of the hands. While the hands were rotated inside the arms without the use of skeleton correspondences, our skeleton correspondences provided a way to prevent this, although it is not guaranteed that no body parts get stuck inside other body parts. Whenever the a body part gets stuck, wrong skeleton correspondences are created next to the creation of wrong contour correspondences, which keeps the body part inside the other. Since the number of correspondences increases with the use of the skeleton correspondences, the computation time of the local optimization has increased. An additional 30 minutes for all settings was required to estimate the pose in the sequence.

Chapter 6

Efficiency analysis

Two analysis are given, one for each optimization method. For both methods, the speed is dependent on the number of joints, size of the mesh, the number of cameras (which also stands for the number of images at one time step) and size of the images. To give a indication of the speed and efficiency of our algorithm, we will use the "big $\mathcal{O}$" notation. We will try to specify this in terms of the number of joints j, number of vertices v, number of cameras c.

A projection of a 3D model to one of the camera images is done by projecting all patches of the model to screen. Since the model consists of v vertices, the number of patches is $v - 1$. So we can project a model in $\mathcal{O}(v)$, however, this has to be done for each camera, giving $\mathcal{O}(vc)$.

Moving the 3D model according to the estimated angles means we have to transform all joints and all vertices. But before that can be done, the rigid motions of the joints are required. For each joint, we have the estimated angle and its twist and from these the rigid motion is calculated using Rodrigues' formula. This is done by three multiplications of two matrices, two scalar multiplication, a matrix subtraction and a few matrix additions. Since all matrices used have size 3×3, the total time taken for calculating the rigid motion is constant. However, the rigid motion for a joint also depends on the rigid motions of all preceding joints. In the worst case all other joints precede a joint, so all rigid motions have to be multiplied with the rigid motion, taking $\mathcal{O}(j)$ time to calculate the final rigid motion of a joint.

Next all joints and vertices have to be transformed according to these rigid motions. The transformation of a vertex is influenced by multiple joints, in worst case all joints, plus the root, influence the vertex. So it takes $\mathcal{O}(jv)$ time to calculate all transformed vertices and $\mathcal{O}(j)$ to calculate all transformed joints.

The following paragraphs will show the complexity of the local optimization and the global optimization on one frame.

Local optimization For the local optimization, we have i correspondences. From each correspondence three linear equations are created, one for each dimension and each equation has j unknowns, namely all joints. The contour correspondences are created

between the vertices on the contour of the projected model and the contour of the silhouette. Lets say the contour of the silhouette consists of s pixels. Calculating the closest points between these two point sets at most $\mathcal{O}(js)$ time. When texture correspondences are used, the SIFT features have to be calculated for each frame. Fortunately, this only has to be done once and then the features can be used in each iteration. The number of texture correspondences depends on the number of SIFT features and number of vertices. If there are f SIFT features which is the sum of all SIFT features over all c cameras, the time taken to compute the texture correspondences is $\mathcal{O}(jf)$, by using closest point.

The least squares problem is solved by using the Householder algorithm, which takes $\mathcal{O}(3ij)$ time. The results of the householder algorithm are stored in the upper-right triangle of the design matrix and calculating the final result is done by iterating over each first j rows. So iterating over this triangle takes $\mathcal{O}(j^2)$ time.

The local optimization runs at most a fixed number of times, so the number of runs is constant. Therefore, the total computation time of estimating the pose using the local optimization takes $\mathcal{O}((js + jf) + 3ij + j^2)$ time.

Global optimization The global optimization is the most time consuming part of the whole method. As a preprocessing step, the Chamfer images on the silhouettes need to be calculated once. At each step in the global optimization, the particle set consists of p particles. p depends on the number of joints that are misaligned. In worst case, all j joints are misaligned and so there will be $j * 25$ particles. The ISA consists of three steps.

The first step, the selection step, iterates over all particles and only selects the particles which have a low error. A second iteration over the particles set is performed to select additional particles to create a particles set again with p particles again.

The second step, the mutation step, spreads all particles according to a covariance matrix of the joints, created over all particles. Creating this covariance matrix takes $\mathcal{O}(pj)$ time. Mutating all particles according to this covariance matrix also takes $\mathcal{O}(pj)$ time.

The last step, the weighting step, takes the most time. The error of each particle is calculated to be able to weight the particle. The calculation of the error of one particle requires the pose represented by the particle to be projected to 2D. This 2D projection can be used to calculate the difference with the silhouette and vice versa. First of all, the initial model has to be moved according to the particle. This takes $\mathcal{O}(j^2 + jv + j)$ time. Then a 2D projection can be made, which takes $\mathcal{O}(v)$ time. Calculating the difference between the two images is done by iterating over both images simultanously, which takes $\mathcal{O}(xy)$ times, where x and y are the dimensions of the image. Both the projection of the mesh model and the calculation of the difference must be done for all c cameras. Beside this error, the prediction error has to be calculated. This is linear in the number of joints and this is neglectable compared to the computation time of the difference between two images. So the time needed to calculate the error of a particle is $\mathcal{O}((j^2 + jv + j) + c(v + xy))$, but this is only for one particle. The time taken to calculate the error of all particles therefore is $\mathcal{O}(p((j^2 + jv + j) + c(v + xy)))$.

Computation time It is seen that the required computation time depends on the quality of the model (number of joints, number of vertices), the number of cameras and the quality of the images. These factors may vary in different environments. The most computation time is spent on the global optimization. However, since the global optimization is only initiated after misalignment, it is not used in each frame and so we can not give a good approximation on the time needed to estimate the pose in an entire sequence. In Table 6.1, the time needed to estimate the pose in one frame is given. A normal consumer computer is used with an Intel Core 2 Duo processor at 2,66 GHz. These times are gained by making the program multi-threaded, as explained in the next paragraph. The used settings are the same as in Section 5.6.

first column walking sequence

	Runtime (hours)	
	Walking	Handstand
Local optimization	3,5 sec	4,5 sec
Local optimization (+ texture)	13 sec	60 sec
Local optimization (+ thinning)	9 sec	10 sec
Global optimization (half of the joints)	42 sec	112 sec
Global optimization (all joints)	86 sec	220 sec

Table 6.1:

The vast amount of difference in computation time between the handstand and walking sequence when using texture correspondences is due to the textured clothing of the observed person in the handstand sequence. This person wears very textured clothing and as such, many SIFT features are generated. This leads to a great number of correspondences in comparison to the number of correspondences in the walking sequence, resulting in a higher computation time.

Implementation details To decrease the runtime of the entire method, a multi-threaded implementation is used. At several stages in the method, a multi-threaded approach is possible. The parts with the highest computationally cost are the creation of the correspondences in the local optimization and the calculation of the error of the particles in the global optimization and so we will make these parts multi-threaded. Parallelizing these parts is done by giving each camera its own thread to create the correspondences and giving each particle its own thread to calculate its energy.

Calculating the energy of each particle takes a lot of time, due to the calculation in difference between images. However, in most images the observed person only covers a small part of the silhouette. Therefore, the images can be cropped, such that a bounding box is created which encapsulates both the projected model and the observed person. Everything outside this box is not taken into account when the error is calculated. This speeds up the calculation since only a small part of the images is used.

Chapter 7

Conclusions

7.1 Conclusions

In this work we have implemented and analysed the 3D pose estimation method by Gall *et al.* [1]. It provides a good trade-off between accuracy and speed, since it combines a local optimization and a global optimization. Since the global optimization interferes whenever the local optimization fails, this method is able to run fully automatic. Their results are very accurate in all sequences. Our implementation of their method did provide accurate results, but in some cases it fails, mainly in case of the hands and feet. The only difference is that we omitted the surface estimation step. Because the results of surface estimation contribute 10% to the mesh, used by Gall *et al.*, our expectation is that it would not have a great impact on the estimation. However, this small mesh refinement could prove to be valuable in the estimation of the smaller limbs. A further explanation on this is given in Section 7.2.

We managed to get more accurate results and speed up the method by factor 600%. This is accomplished by changing critical details of the method and by adding sophisticated techniques to the method. The changes in the method are based on theoretical incorrect details and provides a more correct method. On the other hand, the addition of new techniques gives a better estimation and leads to a decrease in overall computation time. Our introduced dynamic threshold gives more accurate results and leads to less faulty initializations of the global optimization, consequently speeding up the method. In contrast to a static threshold, the dynamic threshold is also invariant to changes in environments and so it can be used in any environment. The addition of skeleton correspondences to the local optimization, creates an additional type of correspondences which takes the skeleton into account. The use of these skeleton correspondences provides an increase in accuracy for both location and orientation of the joints.

Both the local optimization as the global optimization are based on silhouettes and so a proper background subtraction is required. Since the used datasets provide a perfect background subtraction, i.e. no noise is given and the foreground silhouettes do not contain gaps, the focus of the experiments was fully on the evaluation of the method on a perfect dataset. We have seen that the results gained from the method are very

accurate on a dataset with perfect background subtraction.

While the pose estimation works well, a good initial pose is still required. We have seen that provided an approximation of the pose in the first frame an accurate begin pose can be estimated. Without any approximation, no proper begin pose can be constructed using the global optimization.

Many settings have an influence on the estimation of the method. We have analysed different possibilities and can conclude that the method is prone to changes in settings. For all settings an analysis is given and from this evaluation a good setting was found which gave accurate results for all our test cases.

We managed to speed up the method, but speed is still a limitation. While the local optimization gives accurate results, the time required to estimate the pose in one frame is more than a few seconds. Whenever the global optimization is initialized, the time taken increases to two minutes. Despite the fact that our implementation can be optimized, the amount of time needed will still be high, such that real-time results are nearly impossible to get using this method. The main reason for this is the fact that the method is a model-based method. While model-based method provide accurate results, they require a lot of processing time, due to the use of a 3D mesh model. This indicates that this method and in general all model-based methods, are not suitable in applications in which a real-time pose estimation is required. However, in applications in which the pose can be extracted in a post-processing step, this method is a practical solution, as it efficiently provides accurate results in a full automatic manner. This is suitable for many general applications in which it is required to run without any human interference. Our method provides accurate locations of each limb, which can be used as input for other methods. An example would be face recognition. Our method already provides the location of the head, making it easier to apply face recognition. Or when gesture recognition is required, our method provides the location of the hands. Beside the pose, our method also provides the mesh. This gives additional info about the pose and person. From the mesh, the orientation of the person can be extracted, which is not possible from the skeleton alone. Another use for the mesh is the ability to detect collisions with other persons or objects, but also for self-collision.

In the particular case of the Restaurant of the Future, this method is practical in applications which do not require real-time estimation. While full occlusions were not tested in this thesis and are subject for future work, all applications in which a controlled environment is available this method will be suitable. In a controlled environment this method will work efficiently and gives accurate results. The main advantage is that it can be used for a wide variety of applications. However, for use in an uncontrolled environment (Restaurant of the Future) and where the observed person is subject to occlusions, a lot of future work has to be done.

7.2 Future work

A robust and accurate pose estimation method has been investigated and improved. Although the method still has some limitations, such as the running time and it is only

capable of tracking a single person, it is capable of finding a pose which can be used for many applications. A few improvements that can be made to the method:

- GPU implementation - Many operations are done on matrices and vectors. While our current implementation runs fully on the CPU, it is possible to speed things up by implementing these operations for the GPU.
- Multi-threading - Although our implementation already makes use of multi-threading, we only tested it on a dual-core processor. The number of parallel processes is the same as the number of cameras. Each camera view provides correspondences for the local optimization, independent from other camera views. Therefore, a processor with multiple cores provides a faster estimation.

These improvements are aimed at speeding up the method and to drop some assumptions, without having an effect on the estimation. Beside these improvements, the method itself should be evaluated further. The dataset used in this thesis is captured in a controlled environment, provides a perfect background subtraction, videos are captured in a controlled environment and only a single person is observed. While the method is capable of estimating a correct pose for this dataset, no evaluation has been given on the performance of this method on datasets which are not perfect. However, this is an important future work as it gives insight on the usability of this method in an uncontrolled environment. Despite the fact that the Restaurant of the Future is an uncontrolled environment, this thesis focussed on the speed, as it was proportionally large, accuracy and on the general usability of the method. Although it has been seen that the method is capable of handling self-occlusion, it has not been tested how well it performs to other occlusions, for example occlusions due to objects in the scene or due to an incorrect background subtraction, which would occur in a real environment like in the Restaurant of the Future.

Surface estimation In this thesis, the main goal was to find the pose of the observer person. While we succeeded to find an accurate pose, the mesh of the model does not fit the silhouettes perfectly. To gain a more accurate estimation of the surface, an additional step has to be applied after skeleton estimation. Gall *et al.* [1] did used this additional step, which we omitted. It pushes the vertices of the mesh model towards the silhouette contours of the images. Since in the next frame, the surface estimation of the previous frame is slightly used, the estimation could improve as a better fit between the silhouette and the model can be made. This would apply to hand and feet as these are very small in size. A better fit of the hands and feet to the silhouettes could provide smaller errors for these limbs, resulting in smaller rotation errors.

Multi-person pose estimation The estimation of a single person is accurate in all sequences we have seen. When estimating the pose of multiple persons at the same time, the foreground silhouettes contain multiple silhouettes of different persons. Provided that these silhouettes can be segmented, the algorithm can run on each separated

silhouette, without being influenced by silhouettes of other persons. Silhouettes of persons which are not overlapping each other, can be segmented easily. However, interacting persons or persons close to each other give overlapping silhouettes, making segmentation harder. The method presented by Gall *et al.* is extended by Liu *et al.* [2] to work on two closely interacting persons. They first segment each individual by assigning each pixel uniquely to one person. After segmentation, an adapted version of Gall *et al.* is used, which is capable of handling occlusions and is applied on each individual. For the application of estimating the pose in the Restaurant of the Future, where multiple persons are in the same area, it can be hard to estimate the poses of every individual. Especially in case of queues, where multiple persons are standing close to each other. Other cases which give troubles, are the occlusions from parts of the body. The algorithm can estimate a pose when some parts are not visible in all views, though when a body part is occluded in all camera views, there is no way in which the algorithm can find a correct pose for this body part, as no suitable image data can be provided. Some of these occlusions can be prevented by adding more cameras and by placing them at good places. But in most cases this would not help. For example, when a person is sitting at a table, the legs will be occluded from each view.

Initial pose The assumption of requiring a model, which is in approximately the same position and pose as the observed person in the first frame, can be dropped when voxel reconstruction in combination with automatic rigging is used. This combination is able to create a mesh model, including the skeleton of the observed person at any frame in the sequence. The created mesh model and skeleton can be used as the initial pose for the same frame in the sequence. This provides the capability of acquiring the initial pose at the start of the sequence. However, this would also mean that given a long sequence, parts of the sequence which are of interest, can be easily estimated, without the need of estimating the pose from the start.

The method is capable of finding a pose, however, it has been seen from the experimentations that several improvements can be made to the accuracy as well as the speed. In the following paragraphs we will show some improvements and describe how it could improve the quality of the estimation.

Collision detection While most tracking errors lead to a pose in which one body part lies inside another body part, both the global optimization and the local optimization can not recover from this effect. While the energy function of both optimizations could be adjusted such that they could cope with this kind of error, it is better to prevent it. One way would be to restrict the skeleton such that these poses can not occur. This restriction can be created by manually putting a minimum and maximum rotation on each joint. However, since the skeleton consists of several kinematic chains, it should be taken into account that the rotation of one joint has an influence on all joints on the same kinematic chain, such that the maximum and minimum rotation of these joints change. For example the minimum and maximum rotation of the lower arm is dependent on the

rotation of the upper arm. When the upper arm is next to the body, the lower arm is restricted that it can not rotate inside the torso, but when the upper arm is stretched forward, the lower arm has much more freedom.

A better solution would be to use a training set of different plausible poses. These poses are all anatomically correct and do not contain self-intersections. To use this dataset, an additional energy function is added to the global optimization which takes the plausible poses into account, like in Gall *et al.* [27]. All plausible poses are taken from a motion dataset. A pose which does not occur in the dataset gives a higher error, resulting in high errors for poses with self-intersection. An advantage of this is that any anatomically incorrect pose is possible, since all poses in the dataset are correct. The drawback of this method is that, while it is a quick way of preventing self-intersection, it is restricted to a certain skeleton and it requires a dataset of plausible poses. When a different skeleton is used, a new restriction has to be created and a new dataset is required, as the skeleton differs from the previous one.

A more generic solution would be to include a self-collision detector, which can detect poses with limbs inside other limbs. This method is invariant to skeletons and is used without any prior knowledge on the skeleton. The most common way of detecting self-collision is by creating a bounding box around each body part. The bounding boxes are checked with each other whether they overlap each other or not. Upon overlapping, the body parts encapsulated by the bounding boxes could collide with each other. Different kind of bounding boxes can be used. The best bounding box is a box that encapsulates the body part perfectly. However, the use of such a bounding box is very expensive, as it would be a complex model. Therefore, a trade-off between the complexity of the bounding box and the accuracy of the collision detection has to be made. Some different bounding boxes are: sphere, axis-aligned bounding box, oriented bounding box or a convex hull. An example of a method that detects the overlapping of the bounding boxes is Bounding Volume Hierarchies (BHV) [32]. This puts bounding boxes in a tree structure. Each leaf in this tree contains exactly one body part and each node is a bounding box around all bounding boxes of its children. At the root, the body part is checked if it is overlapping with the bounding box of ones of its children. If so, the tree is traversed into this child. This traversal continues until a leaf is reached, meaning it collides with a body part, or when no bounding box is overlapping with the body part. While the skeleton of the model only contains a few limbs, the tree created from the skeleton is small. As a such it is not much beneficial to create a tree and update it in each frame, since in each frame the configuration of the skeleton changes. Another way could be to use a feature-based minimum distance determination, described by Kuffner *et al.* [33]. The keep track of the distance between the bounding boxes and when it is below a certain threshold, the bounding boxes are colliding with each other. This method provides a good way of using a self-collision detector for the global optimization. Each particle could be checked for self-collision using this method and a self-colliding pose gets a high error.

XOR operator The limb evaluation function depends on the correspondences created by the local optimization. The local optimization itself uses these correspondences to converge to a solution and so the limb evaluation can not give a quantitative evaluation on the limbs. Because the optimization minimizes the error of the correspondences, it could occur that the error of bad correspondences is minimized. As such, the average error of the correspondences for a limb could be below the threshold, while due to the bad correspondences the pose will not be correct. It also fails when no correspondences are created at all, which could occur due to self-occlusion. In future work, an investigation can be made to improve the limb evaluation function. The global optimization uses the difference between the projected surface and the silhouette to calculate the error of a pose. This is similar to the XOR operator on two binary images, which can be used to determine whether a pose from the local optimization is correct or not. Since the projected surface will not fit the silhouettes perfectly, a threshold must be used to allow small variations. While this evaluation function will only state whether a pose is correct or not, it does not allow for a precise detection of misaligned limbs. For this purpose, a partial Hausdorff distance can be used. The partial Hausdorff distance [34] should only be calculated between all vertices of a limb and the contour of the silhouette. So, in other words this means that the maximum over the minimum distance from each vertex towards the silhouette contour should be taken. Whenever this maximum exceeds a threshold (which can be a dynamic threshold the same as introduced in this thesis) the corresponding limb is misaligned.

Distribution measure The global optimization is accurate when the search space contains a global optimum with several local optima surrounding it as in the left graph in Figure 5.8. However, in the other two cases the optimization fails as it keeps the particles spread among several optima. The resulting estimation is created by taking the average over all particles, taking the weighting of the particles into account. This means that when the particles are spread among several optima, the resulting pose lies somewhere between the optima.

Whenever this occurs, the mean would not suffice. There are several ways of measuring the distribution of the particle set. The first one, which we used in this thesis, is taking the average over all weighted particles. Another measure is to take the most optimal particle in the particle set as the resulting pose. However, this is very prone to errors, since the most optimal particle is not guaranteed to be the best solution.

Taking the average over the most optimal particle and the particles that lie close to this particle, would give a better measure of the distribution in the most optimal peak. This measure gives a solution in which the particles lie in the same optima. As a such, the resulting pose would be an optimum.

Appendix A

System of linear equations

A WLS problem is of the form

$$\mathbf{AWx} = \mathbf{b}, \tag{A.1}$$

where $\mathbf{A}$ is the design matrix, $\mathbf{W}$ is the diagonal weight matrix, $\mathbf{x}$ is the vector of unknowns, and $\mathbf{b}$ is the vector with the observations. The problem we want to solve is

$$\underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2} \sum_{i}^{I} w_i ||\Pi((\mathbf{I} + \theta_0 \hat{\boldsymbol{\xi}}_0 + \sum_{j=1}^{n_{k_i}} \theta_{o_{k_i}(j)} \hat{\boldsymbol{\xi}}_{o_{k_i}(j)}) \mathbf{v}_i) \times \mathbf{n}_i - \mathbf{m}_i||_2^2, \tag{A.2}$$

where we sum over the errors of all correspondences. However, Equation (A.2) is not of the form used in the WLS problem Equation (A.1). Therefore, this equation has to be rewritten. All Plücker moments $\mathbf{m}_i$ are the observations, since they are constant and we are looking for a transformation that minimizes the distance between the moment and the cross product of the transformed 3D point and $\mathbf{n}_i$. The vector $\mathbf{x} = (\theta_0 \hat{\xi}_{\mathbf{0}}, \theta_1, .., \theta_k)$ contains the unknowns, giving us a total of $k + 6$ unknowns (k joints and 6 DoF for the global twist). The design matrix is constructed by creating three linear equations for each correspondence, one for each dimension (x, y and z). If we have k joints and I correspondences, the size of the design matrix will be $[(k + 6) \times (3I)]$, the size of the unknown vector $\mathbf{x}$ is $[1 \times (k+6)]$ and the size of the vector containing the observations is $[1 \times (3I)]$. Since an additional equation for each joint is added to the system, constraining the estimated angle to lie close to the predicted angle, the final size of the design matrix will be $[(k + 6) \times (3I + k)]$ and the final size of $\mathbf{b}$ is $[1 \times [(3I + k)]$. The error of one correspondence is given by this equation

$$w_i ||\Pi(\mathbf{I} + \theta_0 \hat{\boldsymbol{\xi}}_0 + \sum_{j=1}^{n_{k_i}} \theta_{o_{k_i}(j)} \hat{\boldsymbol{\xi}}_{o_{k_i}(j)}) \mathbf{v}_i) \times \mathbf{n}_i - \mathbf{m}_i||. \tag{A.3}$$

We want to minimize the error, so we set the result of each equation to zero

$$w_i (\Pi(\mathbf{I} + \theta_0 \hat{\boldsymbol{\xi}}_0 + \sum_{j=1}^{n_{k_i}} \theta_{o_{k_i}(j)} \hat{\boldsymbol{\xi}}_{o_{k_i}(j)}) \mathbf{v}_i) \times \mathbf{n}_i - \mathbf{m}_i) = 0. \tag{A.4}$$

Move $w_i\mathbf{m}_i$ to the other side of the equation, and multiply $\mathbf{v}_i$ with each twist.

$$w_i(\Pi(\mathbf{I}\mathbf{v}_i + \theta_0\hat{\boldsymbol{\xi}}_0\mathbf{v}_i + \sum_{j=1}^{n_{k_i}} \theta_{o_{k_i}(j)}\hat{\boldsymbol{\xi}}_{o_{k_i}(j)}\mathbf{v}_i) \times \mathbf{n}_i) = w_i\mathbf{m}_i. \tag{A.5}$$

Move $\mathbf{n}_i$ inwards.

$$w_i(\Pi(\mathbf{I}\mathbf{v}_i) \times \mathbf{n}_i + \Pi(\theta_0\hat{\boldsymbol{\xi}}_0\mathbf{v}_i) \times \mathbf{n}_i + \sum_{j=1}^{n_{k_i}} \Pi(\theta_{o_{k_i}(j)}\hat{\boldsymbol{\xi}}_{o_{k_i}(j)}\mathbf{v}_i) \times \mathbf{n}_i) = w_i\mathbf{m}_i. \tag{A.6}$$

Relax the summation, such that the summation is over all joints, instead on only the joints influencing the motion of $\mathbf{v}_i$. If the transformation of vertex $\mathbf{v}_i$ is not influenced by joint j and is not on the kinematic chain up to $\mathbf{v}_i$, then we can set $\hat{\boldsymbol{\xi}}_j = 0$. Also, move the weight w_i inwards and move the fixed term $w_i\Pi(\mathbf{I}V_i) \times \mathbf{n}_i$ to the right side of the equation.

$$w_i\Pi(\theta_0\hat{\boldsymbol{\xi}}_0\mathbf{v}_i) \times \mathbf{n}_i + w_i\sum_{j=1}^{k}(\Pi(\theta_j\hat{\boldsymbol{\xi}}_j\mathbf{v}_i) \times \mathbf{n}_i) = w_i\mathbf{m}_i - w_i\Pi(\mathbf{I}\mathbf{v}_i) \times \mathbf{n}_i. \tag{A.7}$$

Since θ_j is a scalar, we can write $(\Pi(\theta_j\hat{\boldsymbol{\xi}}_j\mathbf{v}_i) \times \mathbf{n}_i)$ as $\theta_j(\Pi(\hat{\boldsymbol{\xi}}_j\mathbf{v}_i) \times \mathbf{n}_i)$ and $\mathbf{I}\mathbf{v}_i = \mathbf{v}_i$, giving us:

$$w_i\Pi(\theta_0\hat{\boldsymbol{\xi}}_0\mathbf{v}_i) \times \mathbf{n}_i + w_i \begin{bmatrix} \Pi(\hat{\boldsymbol{\xi}}_1\mathbf{v}_i) \times \mathbf{n}_i) \\ \Pi(\hat{\boldsymbol{\xi}}_2\mathbf{v}_i) \times \mathbf{n}_i) \\ \vdots \\ \Pi(\hat{\boldsymbol{\xi}}_j\mathbf{v}_i) \times \mathbf{n}_i) \end{bmatrix}^T \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_k \end{bmatrix} = w_i\mathbf{m}_i - w_i\Pi(\mathbf{v}_i) \times \mathbf{n}_i. \tag{A.8}$$

Before we rewrite the entire equation to the proper form, we rewrite $\Pi(\hat{\boldsymbol{\xi}}_i\mathbf{v}_i)$

$$\begin{aligned} \Pi(\hat{\boldsymbol{\xi}}_i\mathbf{v}_i) &= \Pi \begin{bmatrix} 0 & -\omega_z & \omega_y & v_x \\ \omega_z & 0 & -\omega_x & v_y \\ -\omega_y & \omega_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\ &= \Pi \begin{bmatrix} -\omega_z y + \omega_y z + v_x \\ \omega_z x - \omega_x z + v_y \\ -\omega_y x + \omega_x y + v_z \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -\omega_z y + \omega_y z + v_x \\ \omega_z x - \omega_x z + v_y \\ -\omega_y x + \omega_x y + v_z \end{bmatrix}. \end{aligned} \tag{A.9}$$

In the first term of Equation (A.8), the entire twist$(\theta_0, \hat{\boldsymbol{\xi}}_0)$ is unknown. Therefore we will find the value for the entire unknown twist. The found twist will be an unnormalized twist $\hat{\boldsymbol{\xi}}_0$ from which the scalar θ can be extracted. First, lets rewrite the first term

$$
\begin{aligned}
& w_i \Pi(\hat{\boldsymbol{\xi}}_0 \mathbf{v}_i) \times \mathbf{n}_i \qquad (A.10)\\
& = w_i \begin{bmatrix} -\omega_z y + \omega_y z + v_x \\ \omega_z x - \omega_x z + v_y \\ -\omega_y x + \omega_x y + v_z \end{bmatrix} \times \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \\
& = w_i \begin{bmatrix} (\omega_z x - \omega_x z + v_y) n_z - (-\omega_y x + \omega_x y + v_z) n_y \\ (-\omega_y x + \omega_x y + v_z) n_x - (-\omega_z y + \omega_y z + v_x) n_z \\ (-\omega_z y + \omega_y z + v_x) n_y - (\omega_z x - \omega_x z + v_y) n_x \end{bmatrix} \\
& = w_i \begin{bmatrix} \omega_z x n_z - \omega_x z n_z + v_y n_z + \omega_y x n_y - \omega_x y n_y - v_z n_y \\ -\omega_y x n_x + \omega_x y n_x + v_z n_x + \omega_z y n_z - \omega_y z n_z - v_x n_z \\ -\omega_z y n_y + \omega_y z n_y + v_x n_y - \omega_z x n_x + \omega_x z n_x - v_y n_x \end{bmatrix} \\
& = w_i \begin{bmatrix} 0 & n_z & -n_y & -z n_z - y n_y & x n_y & x n_z \\ -n_z & 0 & n_x & y n_x & -x n_x - z n_z & y n_z \\ n_y & -n_x & 0 & z n_x & z n_y & -y n_y - x n_x \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \\
& = w_i \mathbf{D}_0 \mathbf{x}_0.
\end{aligned}
$$

In the second term of Equation (A.8), the entire term $\Pi(\hat{\boldsymbol{\xi}}_j \mathbf{v}_i) \times \mathbf{n}_i$ is known. So we can write the second term of the same form as the first term

$$
w_i \begin{bmatrix} \Pi(\hat{\boldsymbol{\xi}}_1 \mathbf{v}_i) \times \mathbf{n}_i) \\ \Pi(\hat{\boldsymbol{\xi}}_2 \mathbf{v}_i) \times \mathbf{n}_i) \\ \vdots \\ \Pi(\hat{\boldsymbol{\xi}}_j \mathbf{v}_i) \times \mathbf{n}_i) \end{bmatrix}^T \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_k \end{bmatrix} = w_i \mathbf{D}_j \mathbf{x}_j. \qquad (A.11)
$$

Now we can rewrite Equation (A.8), using Equation (A.10) and Equation (A.11) as

$$
w_i \begin{bmatrix} \mathbf{D}_0 & \mathbf{D}_j \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_j \end{bmatrix} = w_i (\mathbf{m}_i - \Pi(\mathbf{V}_i) \times \mathbf{n}_i), \qquad (A.12)
$$

This is one of the equations that is going to be solved by the weighted least squares problem, where w_i is the weight on the diagonal of $\mathbf{W}$, $[\mathbf{D}\ \mathbf{D}_j]$ contains three rows of $\mathbf{A}$, $\mathbf{x}_0$ and $\mathbf{x}_j$ are all unknowns in $\mathbf{x}$ and the right term of the equation are three rows of the observation vector $\mathbf{b}$. Converting all correspondences into this form and putting them into the matrices, will give the system of linear equations that has to be solved.

Bibliography

[1] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel, "Motion capture using joint skeleton tracking and surface estimation," *Conference on Computer Vision and Pattern Recognition*, 2009.

[2] Y. Liu, C. Stoll, J. Gall, H.-P. Seidel, and C. Theobalt, "Markerless motion capture of interacting characters using multi-view image segmentation," *Conference on Computer Vision and Pattern Recognition*, 2011.

[3] R. Poppe, "Vision-based human motion analysis: An overview," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, 2007.

[4] C. Bregler, J. Malik, and K. Pullen, "Twist based acquisition and tracking of animal and human kinematics," *International Journal of Computer Vision*, vol. 56, no. 3, 2004.

[5] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated mesh animation from multi-view silhouettes," *ACM Transactions on Graphics*, vol. 27, no. 3, 2008.

[6] S. Corazza, L. Mundermann, A. M. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi *Annals of Biomedical Engineering*, vol. 34, no. 6, pp. 1019–1029, 2006.

[7] B. Daubney and X. Xie, "Tracking 3d human pose with large root node uncertainty," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, Conference on Computer Vision and Pattern Recognition, 2011.

[8] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Transactions on Graphics*, vol. 27, no. 3, 2008.

[9] J. Gall, B. Rosenhahn, and H.-P. Seidel, "An introduction to interacting simulated annealing," *Human Motion - Understanding, Modeling, Capture and Animation, Klette R., Metaxas D., and Rosenhahn B. (Eds.), Computational Imaging and Vision*, vol. 36, 2008.

[10] C. Cagniart, E. Boyer, and S. Ilic, "Free-form mesh tracking: A patch-based approach," *Conference on Computer Vision and Pattern Recognition*, pp. 1339–1346, 2010.

[11] N. Hasler, C. Stoll, B. Rosenhahn, T. Thorählen, and H.-P. Seidel, "Technical section: Estimating body shape of dressed humans," *Computer and Graphics*, vol. 33, no. 3, 2009.

[12] A. O. Bălan and M. J. Black, "The naked truth: Estimating body shape under clothing," in *Proceedings of the 10th European Conference on Computer Vision: Part II*, European Conference on Computer Vision, 2008.

[13] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," *ACM Transactions on Graphics*, vol. 24, no. 3, 2005.

[14] K.-M. G. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette across time part i," *International Journal of Computer Vision*, vol. 62, no. 3, 2005.

[15] K.-M. G. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette across time part ii," *International Journal of Computer Vision*, vol. 63, no. 3, 2005.

[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[17] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[18] M. Mason, *Mechanics of Robotic Manipulation*. Cambridge University Press, 2001.

[19] L. Kavan, S. Collins, J. Zára, and C. O. Sullivan, "Geometric skinning with approximate dual quaternion blending," *ACM Transactions on Graphics*, 2008.

[20] I. Baran and J. Popovic, "Automatic rigging and animation of 3d characters," *ACM SIGGRAPH*, vol. 26, no. 72, 2007.

[21] M. Slater, A. Steed, and Y. Chrysanthou, *Computer Graphics and Virtual Environments*. Addison Wesley, 2001.

[22] D. C. Lay, *Linear Algebra and its Applications*. Addison Wesley, 2003.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, 2004.

[24] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, 2002.

[25] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, 1993.

[26] R. Douc, "Comparison of resampling schemes for particle filtering," *In 4th International Symposium on Image and Signal Processing and Analysis*, 2005.

[27] J. Gall, B. Rosenhahn, T. Brox, and H.-P. Seidel, "Optimization and filtering for human motion capture," *International Journal of Computer Vision*, vol. 87, no. 1, 2010.

[28] X. Pennec and N. Ayache, "Uniform distribution, distance and expectation problems for geometric features processing," *Journal of Mathematical Imaging and Vision*, vol. 9, 1998.

[29] G. Pons-Moll, A. Baak, T. Helten, M. Müller, H.-P. Seidel, and B. Rosenhahn, "Multisensor-fusion for 3d full-body human motion capture," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 0, 2010.

[30] A. Baak, T. Helten, M. Müller, G. Pons-Moll, B. Rosenhahn, and H.-P. Seidel, "Analyzing and evaluating markerless motion tracking using inertial sensors," in *European Conference on Computer Vision (ECCV Workshops)*, vol. 0, 2010.

[31] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commununications of the ACM*, vol. 27, no. 3, 1984.

[32] T. Larsson and T. Akenine-Möller, "A dynamic bounding volume hierarchy for generalized collision detection," *Computer and Graphics*, vol. 30, no. 3, 2006.

[33] J. Kuffner, K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue, "Self-collision detection and prevention for humanoid robots," in *in Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[34] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2007.

[35] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 22, 1955.

[36] G. W. Taylor, L. Sigal, D. J. Fleet, and G. E. Hinton, "Dynamical binary latent variable models for 3d human pose tracking," *Conference on Computer Vision and Pattern Recognition 2010*, 2010.

[37] S. M. LaValle, *Planning Algorithms*. The MIT Press, 2006.