University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Exchange Student at the Department of Electrical and Computer Engineering Home Institution: University of Applied Sciences Rapperswil HSR (Switzerland)	Autumn semester, 2011 Open access
Writer: Patrick Fleischmann	…… (Writer's signature)
Faculty supervisor: Assoc. Prof. Dr. Ing. Ivar Austvoll External supervisor(s):	
Title of thesis: **Particle Swarm Optimization with Soft Search Space Partitioning for Video-Based Markerless 3D Human Pose Tracking**	
Credits (ECTS): 27	
Keywords: **Articulated Human Motion Capture, Kinematic Tree, 3D Model, Multiple view, Analysis-by-synthesis, Pose Estimation, Particle Swarm Optimization**	Pages: 83 Stavanger, 06.01.2012

This document was typeset with LaTeX.

Abstract

Video-Based Markerless Motion Capture

The task for this master's thesis was to develop, implement, and evaluate an algorithm for markerless human pose tracking. The algorithm should be based on the freely available HumanEva framework to enable a quantitative comparison to the state of the art. This framework uses an articulated body model to estimate the pose in an analysis by synthesis approach. Some of the hard problems of markerless motion capture come from self-occlusions and 3D-2D mapping ambiguities. These problems are alleviated by using multiple cameras. But the hardest problem remains: The high number of parameters that define the pose of the body model.

Soft Partitioning Particle Swarm Optimization

This thesis proposes a new algorithm called soft partitioning particle swarm optimization (SPPSO) which formulates pose tracking as an optimization of the 31 parameters that define the pose of the body model. The optimization objective is a fitness function which represents the match between the body model and the video frames. To tackle the dimensionality problem, SPPSO divides the optimization into two stages that exploit the hierarchical structure of the model. The first stage only optimizes the six most important parameters that define the global orientation and position of the model. In contrast to hard hierarchical partitioning schemes, soft partitioning refines the estimation of these parameters in the second optimization stage. In addition to presenting SPPSO, the thesis also provides a literature review of the current research in the field with an emphasis on approaches that use particle swarm optimization.

Better Tracking at Low Frame Rates

The performance of SPPSO was evaluated in various tracking experiments on the Lee walk sequence, a standard dataset from the HumanEva framework which contains multi-view video and ground truth motion capture data. The most important result of these experiments is that SPPSO performs better than the annealed particle filter, a common benchmark algorithm, at a frame rate of 20fps, and equally well at 60fps. The better performance at the lower frame rate is attributed to the explicit exploitation of the hierarchical model structure. The experiments also showed that SPPSO performs better than a single-stage global optimization and better than a variant with hard partitioning. An important conclusions from the literature review is that any future developments should combine a stochastic global optimization, such as SPPSO, with a local refinement stage to make tracking more accurate. The global optimization stage allows such an algorithm to recover from tracking failure and the local refinement efficiently improves the tracking accuracy.

Acknowledgements

First, I want to thank my supervisor Ivar Austvoll for the warm welcome to Norway and the support during this project. Thanks also to Bogdan Kwolek for the good discussions during his research stay at UiS.

Furthermore, I want to thank all the people who made my exchange semester possible, especially my advisor at HSR, Guido Schuster, and of course my parents.

Det var veldig hyggelig i Norge og jeg håper at jeg vil komme tilbake annen gang.

Accompanying DVD

The accompanying DVD contains the following folders:

`doc`	This thesis as a pdf file and LaTeX source code.
`literature`	The cited literature as pdf files.
`matlab`	The Matlab scripts and data sets used to perform the experiments, including the tracking results shown in this thesis. See appendix A for instructions on how to run the scripts.
`visualStudio`	The Visual Studio project for compiling a required MEX-file.

The file `3D Model Human Pose Tracking.mm` is an overview over the cited literature in the form of a mindmap. It can be opened by the program FreeMind.
download: `http://freemind.sourceforge.net/wiki/index.php/Download`

Contents

1 Acronyms

APF Annealed Particle Filter [DR05]

CSS Covariance Scaled Sampling [ST01]

DoF Degrees of Freedom

GA Genetic Algorithm [SP94]

GLAPSO Global Local Annealed Particle Swarm Optimization [KKW11b]

GLPSO Global Local Particle Swarm Optimization [KKW11a]

ISA Interacting Simulated Annealing [GPS+07]

KLT Kanade Lucas Tomasi [LK81, TK91]

NSF Niching Swarm Filtering [ZS11]

PF Particle Filter [AMGC02]

PSO Particle Swarm Optimization [KE95]

ROI Region of Interest

SIS Sequential Importance Sampling [AMGC02]

SPPSO Soft Partitioning Particle Swarm Optimization

2 Task Description

2.1 Task

The task for this master's thesis was to develop, implement, and evaluate an algorithm for markerless human motion capture based on multiple cameras. The original idea for the algorithm was to segment the observed images into individual body parts and use dense optical flow to propagate the segmented images. The segmentation would facilitate hierarchical optimization for the pose estimation step and thus make the whole tracking process much faster. However, using optical flow for tracking always introduces drift and early experiments clearly showed this. Consequently, using optical flow was abandoned and the further work was focused on the soft partitioning aspect of SPPSO. Moreover, an extensive study of the related work in the area of pose tracking with articulated models and also tracking with optical flow was carried out to provide a profound base for the presented and future work.

2.2 Implementation

The proposed algorithm should be implemented in Matlab, using the HumanEva framework [BSB05, SBB10]. This enabled a quantitative comparison to a state of the art tracking algorithm.

2.3 Supervisor

The supervisor for this thesis was Ivar Austvoll (Assoc. Prof. at the University of Stavanger UiS, Norway). Many ideas for the proposed algorithm were developed in collaboration with Bogdan Kwolek (Assoc. Prof. at the Rzeszów University of Technology, Poland) during his research stay at UiS in autumn 2011.

3 Introduction

Outline

This thesis proposes an algorithm for markerless video-based human motion capture called *Soft Partitioning Particle Swarm Optimization* (SPPSO). This chapter first gives a short general overview of the motion capture process, its taxonomy, and applications. Furthermore, it lists the main challenges of motion capture and outlines SPPSO. The remainder of this thesis is structured as follows: Chapter 4 **Related Work** establishes the context by reviewing relevant contributions to markerless motion capture, chapter 5 **Soft Partitioning Particle Swarm Optimization** describes SPPSO in detail, and chapter 6 **Experiments** reports the experimental results. Finally, chapter 7 **Conclusion** summarizes the gained insights and gives some suggestions for future work. Appendix A explains implementation details.

3.1 Video-Based Human Motion Capture

Surveys

The aim of video-based human motion capture is to estimate the pose and position of a human subject at consecutive time instants. The following surveys give broad overview of the subject: [MG01, MHK06, Pop07, BSF10]. The taxonomy of Moeslund and Granum [MG01] is used throughout this thesis. Sigal and Black recently compiled a short overview over the current state of the art in pose tracking [SB10].

Markerless motion capture

Commercially available motion capture systems are most often marker-based. These markers are attached to the subject and tracked using multiple video cameras. This method works well and has been used by the film industry for years. But in many cases it is inconvenient or even impossible to attach markers to the subject, e.g. in surveillance. *Markerless* motion capture is therefore an active research topic. From now on the term tracking always means markerless tracking.

Process overview

Figure 3.1 shows a flow diagram of the whole human body motion analysis process according to Moeslund and Granum. This thesis only treats the *tracking* and *pose estimation* step. The *initialisation* is done by using ground truth data and the *recognition* step is not considered.

Taxonomy

SPPSO belongs to the category of *multiple view 3D pose estimation* algorithms. It estimates the body location (*tracking*) and the limb configuration (*pose estimation*) simultaneously within the same framework. Therefore, the term *pose tracking* is used to refer to the whole process throughout this thesis. The term *subject* is henceforth used to denote a human tracking subject, multi-person tracking is not discussed.

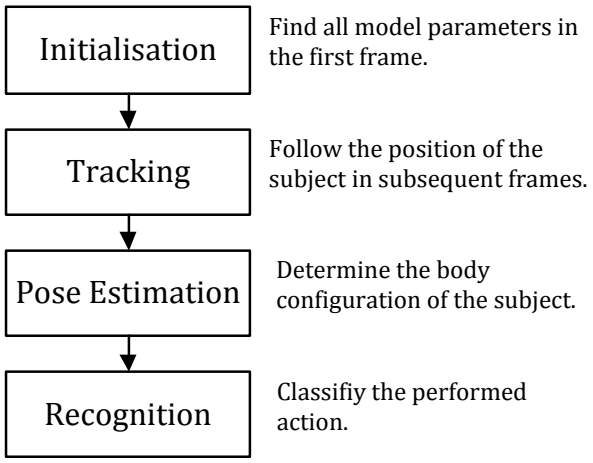


Figure 3.1 The process of human body motion analysis [MG01].

3.2 Applications of Human Motion Capture

There are many applications of markerless motion capture, ranging from computer games to medical gait analysis. Moeslund et al. group the applications into three sectors: *surveillance applications*, *control applications*, and *analysis applications* [MHK06]. Brubaker et al. emphasize the use of motion capture in perceptive environments and man-machine interfaces [BSF10]. The ubiquitous presence of computer vision hardware in the modern world, e.g. in smart phones or surveillance cameras, offers many opportunities for video based human motion capture.

3.3 Challenges in Human Motion Capture

Markerless Human motion capture from video is a hard problem due to various reasons. Four major difficulties are listed below:

Ambiguities arise from the mapping of 3D poses to 2D images. Two mirrored poses are for example indiscernible in a 2D silhouette image.

Self occlusions are also caused by the 3D to 2D mapping. They prevent the observation of body parts.

Clothing and Appearance variability are a major challenge in real-world applications. This problem especially applies to the initialisation step.

High dimensionality of the parameter space is a major problem in all approaches that use articulated body models. The required number of parameters for a full body model is often over 30, even for coarse models.

Multiple cameras

Motion capture from monocular video is a largely unsolved problem [BSF10]. Therefore, most algorithms use multi-view videos to alleviate the ambiguity- and self occlusion problems. SPPSO always uses four cameras.

3.4 Outline of SPPSO

Requirements

Moeslund and Granum compiled a list of common assumptions made by motion capture systems [MG01]. The proposed algorithm relies on the following of these assumptions:

Movement Assumptions

- The subject remains inside the workspace
- No camera motion
- Only one person in the workspace at the time
- No external occlusion (only self-occlusion)

Appearance Assumptions

- Constant lighting
- Static background
- Known camera parameters
- Known start pose

Background subtraction

The static cameras, background, and lighting enable a simple background subtraction. SPPSO uses precomputed background-subtracted images that are included in the used dataset.

Camera parameters

The camera parameters are also included in the dataset. They are required for the projection of the body model into the four camera views. The term *view* is henceforth used to denote the image produced by one of the four used cameras.

Articulated model

SPPSO uses a freely available articulated body model, introduced by Balan et al. [BSB05]. It is based on a kinematic tree with 31 parameters to approximate the skeleton. The shape is approximated by 10 truncated cones.

Optimization formulation

Pose tracking is hereafter formulated as an optimization problem. The objective is to maximize a *fitness* that indicates how well certain image features of the model and the observation match. Namely the background subtracted image, the *silhouette*, and an image gradient based *edge map*.

Search space partitioning

SPPSO uses a *soft search space partitioning* with two stages. In the first stage, only the position and orientation (6 parameters) are optimized. The second stage is a global optimization of all parameters but the position parameters are constrained to a narrower range. Both stages are optimized using Particle Swarm Optimization (PSO) This scheme achieves a good tracking accuracy with few fitness evaluations and avoids error accumulation.

3.4.1 Summary of Contributions

The major contributions of this thesis are:

- A literature survey of articulated pose tracking algorithms, focusing on algorithms that use PSO.
- The SPPSO algorithm and study of soft search space partitioning for optimization of hierarchical models with PSO.
- An experimental examination of different parameter configurations for PSO in pose tracking.

4 Related Work

This chapter is a literature survey in the field of pose tracking. It first gives a general overview of the two fundamentally different approaches to pose tracking: *model free* methods (section 4.1) and *direct model use* methods (section 4.2). After that the commonly used objective functions, i.e. fitness/likelihood functions, are discussed in section 4.3. Section 4.4 then categorizes pose tracking algorithms according to the used optimization method. Section 4.5 discusses the HumanEva framework, which consists of a tracking algorithm and various video sequences with synchronised motion capture data. The freely available Matlab implementation of the HumanEva tracking algorithm is used as the base for SPPSO and the experiments in chapter 6 were performed using the Lee sequence from this framework. Finally, section 4.6 shows how optical flow is used in various pose tracking algorithms.

4.1 Model Free Pose Estimation

Two main categories

Model free pose estimation algorithms take an observed image and find the best matching pose in a database of examples. Model free means that they do not use an explicit geometric model of the body. They can be categorized into two categories [MHK06]: *Probabilistic assemblies of parts* use a classifier to locate possible positions of individual body parts in the observed image and then find the most probable body configuration [FH05, FMJZ08, ARS09, BKSS10]. *Example-based methods* directly use a classifier to match an observed image to a database with possible poses [AT06, GEJ+08, GPZ+11, GLS11].

Limited to examples

All the model free approaches share one inherent limitation: they only recognize the poses or motions in their example database. These databases are commonly produced by rendering motion capture data using a 3D model. *Direct model use* algorithms do not exhibit this limitation in general.

4.2 Direct Model Use Methods

4.2.1 Kinematic Tree

In contrast to the model free methods (discriminative), *direct model use* (generative) algorithms incorporate a 3D model in an analysis-by-synthesis fashion. This model approximates the shape, appearance, and kinematic structure of a human body [MHK06]. The kinematic structure is usually modelled by a kinematic tree with the joint angles as the variable parameters during tracking. The 3D position

and orientation of the root node of the tree is parametrised with six additional parameters. Table 4.1 lists the number of parameters for various sources. A kinematic tree for a full body model requires around 30 parameters, this high number of degrees of freedom (DoF) makes pose estimation and tracking a very hard problem. See Table 4.2 for a list of the used acronyms.

Table 4.1 Number of parameters in the human model in various references.

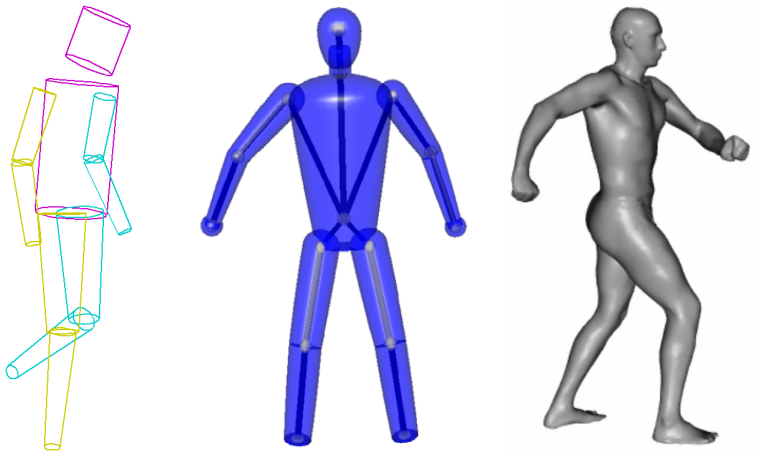
Reference	Algorithm	DoF
Deutscher et al. [DR05]	APF	29
Balan et al. [BSB05]	APF	31
Bandouch et al. [BEB08]	PS, APF	41
John et al. [JTI10]	HPSO	31
Sigal et al. [SBB10]	APF	34
Zhang et al. [ZHW+10]	APSOPF	31
Krzeszowski et al.[KKW11a]	GLPSO	26

Table 4.2 Acronyms of various particle based algorithms and the first reference that applies the algorithm to full body pose tracking.

Acronym	Algorithm	Reference
APF	Annealed Particle Filter	[DBR00]
PS	Partitioned Sampling	[BEB08]
HPSO	Hierarchical Particle Swarm Optimization	[JTI10]
APSOPF	Annealed PSO based Particle Filter	[ZHW+10]
GLPSO	Global Local Particle Swarm Optimization	[KKW11a]

4.2.2 Shape Model

The shape model represents the outer geometric shape of the human body. Its grade of detail may vary from the coarse model with 15 cylinders used in this thesis [BSB05] to the very detailed model used by Kehl et al. [KBVG05] (Figure 4.1). The shape model is often initialised manually for every new subject and usually not adapted during tracking. Balan et al. reported successful automatic recovery of a full human body model only from multi-view image data [BSB+07]. They use the SCAPE model [ASK+05], which is a detailed but low-dimensional parametric model of the human shape. Gall et al. propose a two-stage skeleton-tracking and surface estimation approach where the estimated skeleton is used to initialise the surface estimation stage [GSDA+09]. However, the estimated skeleton does not *restrict* the surface estimation stage, which allows the algorithm to accurately model wide clothing. The main drawback of highly detailed shape models is their high computational cost.



(a) (b) (c)

Figure 4.1 3D shape-models of the human body with different levels of detail. (a) Model with 15 truncated cones used in this thesis, based on the model of Balan et al. [BSB05]. (b) Model based on superellipsoids used by Kehl et al. [KG06]. (c) SCAPE model [ASK+05], image taken from a video from `http://ai.stanford.edu/~drago/Projects/scape/scape.html`.

4.2.3 Appearance Models

Mapping

An appearance model defines a mapping from the body model and the observation to a common representation. For example, the observed silhouette is computed by performing foreground-background segmentation on the observed image and the model silhouette (projected silhouette) is obtained by projecting the 3D body model into the image plane. The two silhouettes can then be compared to determine the fitness of the model.

Surface texturing

Surface texturing models the colour (or grey-value) and texture of individual limbs or areas. Because the appearance can change rapidly due to lighting changes or shadows, the model is usually made adaptive. A simple way to do this is to use the colour of pixels that lie inside the projection of the model as a template [WN97, SBF00, MH03]. In other words, the 3D model at time t is textured with the pixels that lie inside the projection of the model at time $t-1$. This approach relies heavily on an exact pose estimation at time $t-1$ and is therefore prone to error accumulation. Kehl et al. implement an adaptive colour model with variable learning rate [KBVG05]. Gall et al. circumvent the error accumulation problem by using a static texturing [GRS08]. As mentioned above, this approach becomes problematic if the appearance of the subject changes, e.g. due to lighting variability.

Edges

Edges are an important cue in appearance modelling because they can be extracted reliably and are invariant to illumination. Often, a distance map of the observed edges is computed. This map can then be used to determine how well the edges produced by the model fit the observation [DR05].

Silhouette

The *silhouette* is a binary image where all pixels belonging to the foreground (i.e. the tracking subject) are 1, and pixels in the background are 0. The foreground-background segmentation is commonly performed by classifying each pixel with a statistical model of the background and the foreground. The background is most often modelled by a mixture of Gaussians (MoG) [MHK06], whereas the foreground may be modelled by a uniform distribution [BSB05]. This can be seen as an inverse appearance model because the appearance of the background is modelled instead of the foreground. Background subtraction works well in controlled indoor scenarios, but is more difficult in outdoor scenarios where the background may vary over time. The main drawback of this kind of background subtraction is the requirement for stationary cameras.

Visual hull

Silhouettes from multiple views can be used to construct the visual hull, an approximation of the 3D shape of the subject. The visual hull can then be used to match a body model directly in 3D [KG06, CMC+06, MCA07]. The main drawback of this approach is the computational cost of computing a visual hull, a result of the large number of required voxels. Moreover, it requires a relatively large number of cameras to compute an accurate visual hull (8 cameras in [CMC+06, MCA07], 4-11 in [KG06]).

4.2.4 The Pose Tracking Process

Pose tracking is the process of sequentially estimating the pose in a sequence of images. These image sequences are typically produced by video cameras at a frame rate of 10 to 60 frames per second (fps). In the first frame, the pose must be *initialised*. This includes locating the subject in the image and estimating the pose. Initialisation can be a very difficult task when there is only little prior information. Pose tracking, on the other hand, is simpler because the pose estimation from the previous frame can be used as a starting point.

Action specific motion model

When the type of motion (e.g. walking) is known, a strong (action specific) motion model can be used to predict possible poses in the next frame [SBF00]. Successful algorithms for monocular tracking all rely on a strong motion model because it can alleviate the occlusion- and ambiguity-problem [MHK06, ARS10, Fle11].

General motion model

When the type of motion is unknown, a weak motion model must be used. The most simple weak model is zero motion with additional Gaussian noise [BSB05, SBB10]. This works well at high frame rates, but it inevitably breaks down at low frame rates because of the high-dimensional search space of possible poses. Finding better general motion models is an active research topic [LH05, LM07, Fle11].

4.2.5 Bayesian Problem Formulation

Posterior estimation

The objective of all the *direct model use* algorithms is to fit the model as closely as possible to the observations. There are two common formulations of this objective. The first is the Bayesian tracking formulation. Here, the goal is to estimate the

posterior probability distribution $p(x_t|y_{1:t})$, where x_t is the current state of the model (i.e. the true body pose) and $y_{1:t}$ are all the observations up to time t (i.e. the current and past images).

Probabilistic model

With the two assumptions that the underlining process is a first-order Markov process where the current state only depends on the previous state

$$p(x_t|x_{1:t-1}) = p(x_t|x_{t-1}) \tag{4.1}$$

and that the current observation only depends on the current state

$$p(y_t|x_{1:t}, y_{1:t-1}) = p(y_t|x_t) \tag{4.2}$$

the posterior can be formulated recursively as follows:

$$p(x_t|y_{1:t}) \propto p(y_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}. \tag{4.3}$$

This process model is known as a Hidden Markov Model, see Figure 4.2 for a graphical representation.

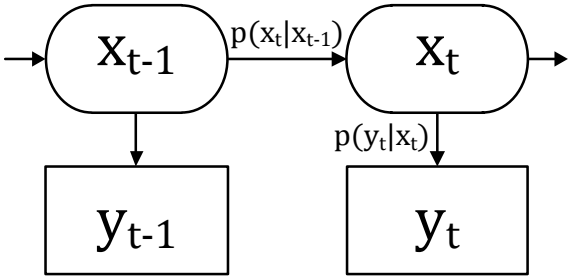


Figure 4.2 Bayesian network of the hidden Markov model (HMM) underlying the Bayesian tracking formulation.

Predict and update

The tracking process consists of two steps: In the *predict* step, the previous estimate $p(x_{t-1}|y_{1:t-1})$ is transformed using the motion model (motion prior) $p(x_t|x_{t-1})$. In the *update* step, this prediction is weighted by the likelihood of the current observation $p(y_t|x_t)$. The likelihood indicates how well a pose x fits the observed image y [AMGC02].

4.2.6 Optimization Formulation

Sequential optimization

The *true* prior and observation distributions are unknown in pose tracking. However, a fitness function based on image features can be constructed easily [GPS+07]. It is therefore convenient to formulate the pose tracking problem as an optimization with two steps. The *predict* step uses a motion model to predict the new pose at time t based on the previous estimations: $\overline{x}_t = f_{\text{motion}}(\widehat{x}_{1:t-1})$. This prediction is then used as the initial value for the second step, the actual optimization. Here, the optimizer searches for an $\widehat{x}_t$ that maximizes the fitness $f(\widehat{x}_t, y_t)$. The fitness indicates how

well a candidate pose $\widehat{x_t}$ fits the observation y_t, it corresponds to the likelihood in the Bayesian formulation. This two-step optimization process is repeated for every new frame.

Simplicity

The main advantage of the optimization formulation is its simplicity. Where the Bayesian formulation requires the estimation of a probability distribution in a high-dimensional state space, the optimization formulation only searches the state space for a pose that maximizes the fitness. No attempt is made to describe the probability distribution.

Ambiguities

In this simplicity lies also the major drawback of the optimization formulation. It is not able to represent pose ambiguities. The Bayesian formulation is in principle able to represent multimodal posterior distributions where the pose estimation is ambiguous. In other words: It can propagate multiple hypotheses. This would make the tracker more robust. In practice however, the complete representation of the posterior of a high-dimensional articulated 3D model becomes infeasible within the commonly used particle filtering framework due to the exponential growth of the required number of particles [DR05].

4.3 Fitness/Likelihood Functions

Objective function

Regardless of whether the problem is directly formulated as an optimization or as Bayesian inference, all the pose tracking algorithms that are discussed in the following section 4.4 try to optimize some sort of objective function (in SPPSO called fitness function). There are two fundamentally different classes of optimization objectives in use: edge/silhouette matching and correspondences.

4.3.1 Edge/Silhouette Matching

Silhouette matching

The objective in silhouette matching approaches is to maximize the overlap between the observed and projected silhouette. A simple way of doing this is to sample the observed silhouette image at discrete points, located on the projected silhouette. Then the number of points that are inside the observed silhouette can be counted [DBR00]. However, one of the most important properties of a silhouette fitness is that it not only rewards the proportion of the projected silhouette that lies inside the observed (as just described), but also the proportion of the observed silhouette that lies inside the projected [ST02, SBB10]. Such a *bidirectional* or *symmetric* silhouette fitness is used for SPPSO. The most simple bidirectional silhouette fitness is obtained by simply XOR-ing the two silhouettes and inverting the remaining area [BEB08].

Edge matching

The edges of the model and the observation are also often matched. They especially help to align body parts that do not contribute to the contour of the observed silhouette. An important step for edge matching is to smooth the observed edges, for example by a Gaussian filter [SBB10] or by computing a distance map [KKW11b]. SPPSO uses the same edge fitness as Sigal et al. [SBB10]. A different way to match

edges was introduced by Sidenbladh and Black. They use directional filters aligned to the edges of the model limbs to determine the fitness [SB03]. Balan et al. showed that tracking with an articulated model fails if only an edge fitness is used [BSB05]. This can be explained by the fact that an edge fitness has generally more local maxima than a silhouette fitness (See experiment 6.10). But another result of their experiments is that a combination of edge and silhouette fitness makes tracking more accurate.

No gradient

The common drawback of silhouette and edge matching is that the gradient of the fitness function can not be derived in a closed form. This is no problem for gradient free optimization methods like PSO or APF (See section 4.4). But gradient based optimization using these fitness functions must smooth the fitness to make sure the gradient can be estimated robustly [ST02].

4.3.2 Correspondences

Closed form gradient

The most important advantage of a fitness function based on *correspondences* is that the gradient can be derived in a closed form as a function of the model parameters [BC08]. This facilitates employing gradient based optimization methods which are generally much faster than gradient free methods [BKMM+04].

Concept

Correspondences are essentially displacement vectors that indicate where the model should be moved to match the observation. In 2D, the starting point of a correspondence lies on the projection of the model, and the end point lies on the corresponding point on the observation. For example, given a set of starting points on the contour of the model projection, correspondences could be obtained by searching for the closest points on the contour of the observation [DF01, BC08, GRS08]. Another way of obtaining correspondences is using optical flow [GRS08].

Finding good correspondences

Once the correspondences are found, the optimization objective is simply to minimize the length of all the correspondences. The main problem is to find good correspondences, especially with fast motion. The closest points on the contour for example do not coincide with the *true* corresponding points when the model is far off the observation. Therefore, correspondence based methods are inherently local optimization methods. However, they yield impressive results for full body tracking [KBVG05, BC08, GRS08].

4.4 Pose Tracking Algorithms

Outline

This section reviews different pose tracking algorithms using articulated body models. The algorithms are categorized by their optimization method as follows: First, subsection 4.4.1 discusses gradient based algorithms. Then, a few applications of genetic algorithms are shown in 4.4.2. After that, subsection 4.4.3 reviews one of the most common algorithms for pose tracking: particle filtering (and why it is regarded

as an optimization method here). Finally, the related work on tracking with particle swarm optimization is discussed in subsection 4.4.5.

4.4.1 Gradient Based Optimization

Speed

Gradient based optimization is attractive for pose tracking because it is much faster than the gradient free methods discussed below. But gradient based methods can only perform local optimization. Thus, the used fitness function should be as smooth as possible and not have spurious maxima. However, such a fitness function is hard to find in practice and therefore a robust optimization method must be used, e.g. a trust region method [ST01] or stochastic meta-descent (SMD) [BKMM+04].

Fitness functions

Gradient based methods are very attractive for correspondence based fitness functions because the gradient can be derived in closed form [BKMM+04, BC08]. Sminchisescu and Telea even developed an elaborate silhouette fitness that is smooth enough to enable gradient based optimization [ST01].

Local refinement

Another approach is to use a gradient based method for the local refinement stage in a population based search method. Zhang and Seah use the Levenberg-Marquardt method [Mar63] to refine the estimate obtained by a PSO based optimization. Sminchisescu and Triggs employ a trust region method to find multiple modes of the fitness function [ST01]. Here, the local optimization is initialised with multiple starting points obtained by a stochastic sampling method. The strategy of using a population based global search method as the first stage, followed by a local refinement stage, seems to be the most powerful approach to pose tracking [GRBS10].

4.4.2 Genetic Algorithms

Terminology

Genetic algorithms (GA) are a group of stochastic global optimization methods based on the principle of evolution in nature. A GA works with a population of individuals (PSO terminology: swarm of particles), which have a candidate solution encoded in their *genes*, i.e. the parameter set. In every iteration, the fittest individuals of the swarm are selected for mutation and crossover to produce the next population.

Few applications

There are very few examples of genetic algorithms applied to pose tracking or estimation [OK94, ZL08]. But the crossover operator, introduced by genetic algorithms, has been shown to improve pose tracking with the annealed particle filter [DR05].

Building-block hypothesis

The crossover operator builds on the hypothesis that the parameter set can be split into building blocks, and that a combination of good building blocks from different individuals produces a good parameter set. This is called the building-block hypothesis [SP94]. In pose tracking, the parameter set can, for example, be split into: body position, arm configuration, and leg configuration. Combining a good arm configuration and a good leg configuration will produce a very fit individual.

Therefore, the building-block hypothesis holds for pose tracking with kinematic tree models.

4.4.3 Particle Filtering

Bayesian framework

Variants of the basic particle filtering (PF) algorithm (also known as sequential importance sampling (SIS) [AMGC02] or condensation [IB98]), have often been applied to the pose tracking problem. The particle filter builds on the Bayesian problem formulation, i.e. estimating the posterior $p(x_t|y_{1:t})$ of the pose parameters, given the priors and observations. The main idea is to approximate the posterior as a discrete set of weighted particles. The approximation is then refined iteratively by evaluating the likelihood of all the particles and resampling concentrated near more likely poses.

Partitioned sampling

The straightforward application of particle filtering to pose tracking with high-dimensional (> 10 DoF) articulated models suffers from the problem that the number of required particles to approximate the posterior grows exponentially [DR05], known as the *curse of dimensionality*. A way to solve this, is partitioned sampling [MI00]. Partitioned sampling is the statistical equivalent of hierarchical search. It is based on the assumption that the likelihood (fitness) on higher hierarchy levels can be evaluated independently from lower hierarchy levels. A similar approach is proposed by Bandouch et al. [BEB08]. The problem is that this assumption does not hold under realistic conditions. For example: With imperfect human models and noisy observations it is impossible to localize the torso exactly, without knowing the true configuration of the arms. In any hierarchical method, errors on higher hierarchical levels propagate to the final pose estimate. This error is then propagated to the next frame and and so on. This finally results in catastrophic failure [DDR01]. The hierarchical approach may produce good results with a perfect model [BEB08], or in single pose estimation, where error accumulation is less of a problem.

Covariance scaled sampling

Sminchisescu and Triggs [ST01] proposed Covariance Scaled Sampling (CSS) to overcome the curse of dimensionality in stochastic sampling. Their approach is based on a sophisticated set of priors and a local approximation of the *cost surface*, i.e. the likelihood function. With this approximation, the search space can be tightly focused on more promising areas. This algorithm is able to perform full body tracking with 30 DoF using only monocular video.

4.4.4 Annealed Particle Filtering and Interacting Simulated Annealing

Optimization method

Particle filtering aims to approximate the posterior distribution, which is possibly multimodal. However, interacting simulated annealing (ISA), which is based on particle filtering, only aims to find the global maximum of the posterior. It is therefore an optimization method and not a particle filter in the strict sense. The basic annealed particle filter (APF) is the predecessor of ISA and included in ISA as a special case.

Basic APF

The basic APF was introduced by Deutscher et al. in 2000 [DBR00]. In 2005, an improved version was reported which includes soft partitioning and the crossover operator [DR05]. This improved APF is the core of the baseline algorithm of the HumanEva framework described in section 4.5 and is used as a benchmark for SPPSO in section 6.5.

Annealing

APF modifies the standard particle filter algorithm in three important ways. First, it introduces an annealing scheme inspired by simulated annealing [KGV83]. At the first annealing layer, the particles are sampled from a broad distribution, which is gradually narrowed down on subsequent layers. This ensures that the search space is explored thoroughly on the first annealing layer and the algorithm does not get stuck on local maxima of the likelihood function. On later layers the particles are more concentrated around interesting regions and the estimate is refined.

Soft partitioning

Second, the distribution is adapted separately for every parameter, based on how well the parameter has been localized yet. Parameters on high hierarchy levels (for example the global body position) can be localized faster than parameters lower in the hierarchy because they have more influence on the likelihood function. The separate adaptation of the sampling distribution is a *soft partitioning* of the search space. More important parameters are estimated first, and less important parameters after, without completely relying on the initial estimation of the important parameters. This is probably the most important contribution of the annealed particle filter.

Crossover

The third modification to the particle filter is the introduction of a crossover parameter. Deutscher et al. clearly show the benefit of a crossover operator when applied to articulated models with sections that have some degree of independence and can therefore be optimized in parallel (for examples the arms and legs). However, the important point is that the annealed particle filter is successful because it does not *completely* trust this assumption.

Interacting simulated annealing

The annealed particle filter is one of the most successful pose tracking algorithms and is often used as a benchmark [BEB08, SBB10, KKW11a]. Gall et al. conducted a through mathematical study of the algorithm and showed that APF converges to the global maximum of the likelihood function under some assumptions [GPS+07]. In the same paper, they introduced a new algorithm called interacting simulated annealing (ISA), which adds a stochastic selection operator to the basic APF (APF is the special case of ISA where 100% of the particles are selected). In a recent comparison of different optimization approaches for pose tracking, ISA was found to outperform APF [SGS+09]. This comparison used 200 particles and 15 iterations for both algorithms, a detailed body model, and 5 cameras with a frame rate of 50fps. The performance of ISA can be improved further when it is combined with a local optimization step [GRBS10]. In this case, the local refinement is performed by an iterative closest point algorithm (ICP) [Zha94].

4.4.5 Particle Swarm Optimization

Dimensionality problem

Particle Swarm Optimization (PSO) is a stochastic optimization method that is well suited for parameter-optimization problems like pose estimation (See section 5.3.1 for a detailed description). It was first applied to markerless pose estimation by Ivekovic and Trucco in 2006 [IT06]. PSO is based on a swarm of particles, each particle represents a candidate pose and moves through the space of possible poses. Unfortunately, the standard PSO also suffers from the *curse of dimensionality*, which has led to many variants specifically adapted for pose tracking.

Hierarchical PSO

Ivekovic and Trucco first used PSO only for static pose estimation of the upper body [IT06]. In two recent papers, Ivekovic et al. describe a hierarchical approach using PSO for full body pose tracking [JTI10, IJT10]. They use the articulated human model of the HumanEva framework and divide the 31-dimensional parameter space into 12 (!) hierarchical subspaces to overcome the problem of high dimensionality. This approach is flawed because the optimization can not escape from local maxima found in preceding hierarchical levels, the final solution tends to drift away from the true pose, especially at low frame rates. This drifting behaviour can be seen in the error graphs in the above-mentioned sources and it could be replicated in experiments (See section 6.6).

Global local PSO

Krzeszowski et al. propose a global local PSO (GLPSO) [KKW11a] where the PSO is divided into two stages. The first stage is a global optimization of the pose. The second stage is a local refinement of the limb configuration. This is done for the legs and arms separately. They also use skin colour detection for body segmentation. The GLPSO performs better than the standard PSO. In another paper, Kowlek et al. [KKW11b] combine the global-local approach with a modified PSO named global local annealed PSO (GLAPSO). The most notable property of this variant of PSO is the quantization of the fitness function. Instead of one global best candidate *gbest* the algorithm maintains a pool of candidate *gbest* candidates, which improves the algorithm's ability to explore the search space. This modification improves the tracking performance and allows the use of fewer fitness evaluations.

Hierarchical fitting

Robertson and Trucco use a different hierarchical approach which they call *hierarchical fitting* [RT06]. They use this approach to fit a model of the upper body to 3D data from stereo cameras. The optimization proceeds as follows: The position of the root node is optimized first (3 DoF), while keeping the other parameters fixed. In the second stage, the position and orientation of the root node is optimized (6 DoF). In the third stage, the position and orientation of the root node plus the clavicle joints is optimized (10 DoF). The subsequent stages always optimize a superset of the parameters of the preceding stages. This approach is very similar to SPPSO and exploits the hierarchical structure of the body model while avoiding the error accumulation problem of other hierarchical approaches.

Global refinement

Another way to overcome the error accumulation problem is suggested in a paper of Ivekovic and Trucco [IT06]. They divide the fitting of an articulated model of the upper body with 20 DoF into seven hierarchical steps. After the hierarchical optimization, a global refinement step is used to correct the errors accumulated

during the hierarchical steps. In the global refinement, the inertia parameter of the PSO is set to a low value to restrict the search space. This makes sense because the optimization result from the hierarchical steps should be near the global optimum.

Soft partitioning

Hierarchical optimization, as well as global local PSO, divide the optimization into multiple stages, in which a subset of the parameters is optimized while the rest of the parameters is fixed. This is a *hard partitioning* of the search space. The term *soft partitioning* was introduced by Deutscher et al. to describe the way the annealed particle filter automatically adjusts the sampling variance of individual parameters [DR05]. In contrast to hard partitioning, soft partitioning means that some parameters are allowed more variance than others, but no parameters are completely fixed. The annealed particle filter adjusts the variance fully automatic, it uses no prior information about the hierarchical structure of the body model and is therefore a very general approach. SPPSO on the other hand, explicitly exploits the hierarchical structure. See Figure 5.6 for an illustration of the soft partitioning principle in SPPSO.

PSO PF hybrid

Zhang et al. proposed a hybrid of APF and PSO to introduce *swarm intelligence* to the annealed particle filter [ZHW+10]. This means, that the particles can exchange information about the current global best candidate solution *gbest*. This approach yields better results than both, the standard PSO and APF algorithms. In a more recent paper, Zhang et al. [ZS11] introduce yet another hybrid algorithm which they call Niching Swarm Filtering (NSF). Furthermore, they use the Levenberg-Marquardt (LM) algorithm [Mar63], a gradient based algorithm for local optimization, to refine the result from NSF.

PSO GA hybrids

Deutscher et al. clearly showed that the crossover operator can improve human pose tracking by exploiting the independence of different body parts. It seems therefore promising to introduce the crossover operator from genetic algorithms into the PSO framework. Such PSO-GA hybrids [LRK01, Jua04, PTA07] have, to the authors knowledge, not yet been applied to human pose tracking. Another promising direction is the introduction of cooperating sub-swarms [VdBE04]. PSO with sub-swarms has been introduced to battle the *curse of dimensionality* and seems suitable for the high-dimensional pose tracking problem.

Number of particles

PSO is generally insensitive to variations of swarm size. Bratton and Kennedy reported that swarms of 20 to 100 particles produced comparable results on a set of standard benchmark sequences [BK07]. But this result was obtained with optimizations over 300'000 fitness evaluations, much more than the commonly used 1000 evaluations per frame in pose tracking [BSB05, SBB10]. Table 4.3 lists the number of particles and iterations used by various full body pose tracking algorithms. All the listed algorithms use either particle filtering or PSO to fit a full body model to observations from multiple cameras. See Table 4.2 for a list of the used acronyms.

Number of evaluations

The number of fitness or likelihood evaluations per frame is the number of particles times the number of iterations. For multi-stage (e.g. hierarchical) optimizations, the number of evaluations must be summed over all stages (some algorithms use

Table 4.3 Number of particles and iterations of markerless full body pose tracking algorithms. For multi-stage (e.g. hierarchical) optimizations with different swarm sizes, the largest swarm size on a single stage is given. For the APF based methods, the number of iterations is the number of resampling layers.

Reference	Algorithm	Particles	Iterations	Stages
Deutscher et al. [DR05]	APF	400	10	1
Balan et al. [BSB05]	APF	200	5	1
Bandouch et al. [BEB08]	PS, APF	1000	10	5
John et al. [JTI10]	HPSO	10	60	12
Sigal et al. [SBB10]	APF	200	5	1
Zhang et al. [ZHW+10]	APSOPF	50	20	1
Krzeszowski et al.[KKW11a]	GLPSO	200	20	3

Table 4.4 Number of evaluations per frame and per second of markerless full body pose tracking algorithms.

Reference	Algorithm	fps	Eval/frame	Eval/s
Deutscher et al. [DR05]	APf	?	4000	?
Balan et al. [BSB05]	APF	60	1000	60'000
Bandouch et al. [BEB08]	PS, APF	25	ca. 20'000	ca. 500'000
John et al. [JTI10]	HPSO	60	7200	432'000
Sigal et al. [SBB10]	APF	60	1000	60'000
Zhang et al. [ZHW+10]	APSOPF	60	1000	60'000
Krzeszowski et al. [KKW11a]	GLPSO	24	6000	144'000

a different number of evaluations at different stages). Table 4.4 lists the number of evaluations per frame and per second. Note that the algorithms with a hard partitioning [BEB08, JTI10, KKW11a] require the most evaluations per frame and also per second. This is probably because these algorithms require a very high accuracy at every optimization hierarchical stage to minimize error propagation.

4.5 The HumanEva Framework

Quantitative comparison

Balan et al. initiated the effort for a quantitative comparison of markerless articulated human pose tracking by releasing the Matlab implementation of their algorithm in 2005, along with the used motion capture dataset. In their paper [BSB05] they established a standard error measure for pose tracking, based on the 3D error of 15 marker joints.

State of the art

In 2010, Sigal et al. published an improved algorithm and a new dataset with a larger variety of motions. In the same year, Sigal and Black compiled a representative list of papers to give an overview over the current state of the art of markerless motion capture [SB10]. This section gives an overview over the HumanEva framework because SPPSO uses its body model and its standard error measure.

Standard error measure

Given the true pose x and an estimated pose $\widehat{x}$, the 3D error D is computed as

$$D(x, \widehat{x}) = \frac{1}{15} \sum_{i=1}^{15} ||m_i(x) - m_i(\widehat{x})|| \tag{4.4}$$

where $m_i(x)$ returns the 3D location of marker i in mm, given the pose x. The ground truth model parameters were acquired using a marker-based, commercial motion capture system. Figure 4.3 Depicts the locations of the 15 marker joints. All pose errors presented in this thesis are computed using this formula. For a sequence of T frames, the *mean* and *max* errors are defined as follows:

$$D_{mean} = \frac{1}{T} \sum_{i=1}^{T} D(x_t, \widehat{x}_t) \tag{4.5}$$

$$D_{max} = \max_t D(x_t, \widehat{x}_t), \quad t \in [1, 2, .., T] \tag{4.6}$$

Cylinder model

In their first paper [BSB05], Balan et al. used the standard error measure only to evaluate their own tracking algorithm, which is based on the annealed particle filter. As the body model, they used an articulated model with 13 joints, 31 parameters, and 10 truncated cones (See Figure 4.3). SPPSO, presented in chapter 5, uses this body model with some minor modifications. The Matlab code for the model and the algorithm can be downloaded from `http://www.cs.brown.edu/~alb/download.htm`.

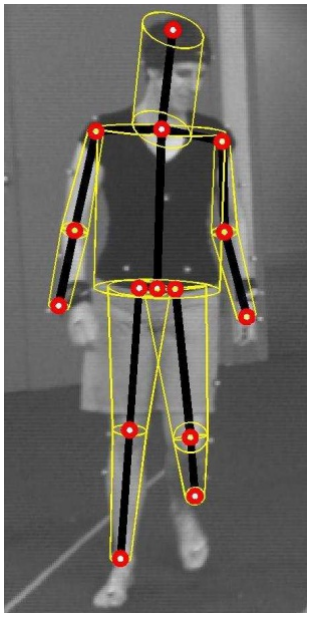


Figure 4.3 The 15 marker joints for the standard error measure [BSB05]. The ground truth markers (red), kinematic tree (black), and cylinder model (yellow) are superimposed on a frame of the Lee walk sequence.

4.5.1 HumanEva Datasets

Lee walk

The first HumanEva dataset is called Lee walk [BSB05], it consists of video and motion capture data of one human subject walking in a circle. The subject is filmed by four greyscale cameras with a resolution of 644x484 at 60fps (in total 532 images per camera). The recorded motion capture data was used to compute ground truth parameters for the body model. Figure 4.4 shows frame 190 of the Lee walk from all four views. The dataset can be downloaded from: http://www.cs.brown.edu/~alb/download.htm.

(a) view 1 (b) view 2

(c) view 3 (d) view 4

Figure 4.4 Frame 190 of the Lee walk sequence (total 532 frames), seen from all four views. The image resolution is 644x484 pixels and the frame rate is 60fps.

HumanEva-I

The HumanEva-I dataset was released in 2006 and contains synchronised video and motion capture data of 4 subjects, performing a standard set of six actions including walking and jogging [SB06]. It contains about 50'000 frames of video in colour and greyscale. It is mainly intended as a training dataset.

HumanEva-II

The HumanEva-II dataset was released in 2010 [SBB10]. It contains colour videos and motion capture data of 2 subjects which perform a different set of actions. This dataset is intended as a validation set. The HumanEva-I and II datasets can be downloaded from http://vision.cs.brown.edu/humaneva/.

Used datasets

Table 4.5 shows the used evaluation datasets in several references. The tracking results in section 6 were all obtained on the Lee walk dataset.

Table 4.5 Evaluation datasets used in various references.

Reference	Algorithm	Dataset
Deutscher et al. [DR05]	APF	own, natural, various movements
Balan et al. [BSB05]	APF	Lee walk
Bandouch et al. [BEB08]	PS, APF	own, synthetic , walking in circle
John et al. [JTI10]	HPSO	Lee walk
Sigal et al. [SBB10]	APF	HumanEva-II
Zhang et al. [ZHW+10]	APSOPF	Lee walk
Krzeszowski et al.[KKW11a]	GLPSO	own, natural, walking straight

4.5.2 Baseline Algorithm

Body model

The most recent paper by Sigal et al. [SBB10] describes the HumanEva-II dataset, contains a short survey of pose tracking algorithms and presents the *baseline algorithm*, which is intended as a benchmark for future developments. The body model for the baseline algorithm comprises 34 parameters and 15 cylinders. It is more detailed than the first body model [BSB05] that only comprised 31 parameters and 10 cylinders. SPPSO uses the older model with 10 cylinders because it is simpler and therefore requires less computational power.

Optimization

The baseline algorithm is a further development of the original algorithm of Balan et al. [BSB05]. It mostly differs in the more detailed body model and in the new bidirectional likelihood function. The core of the algorithm, the optimization, is still the annealed particle filter.

4.6 Using Optical Flow for Tracking

Definition

Optical flow (OF) is the apparent motion between two images. It can be either described as a dense flow field, where every pixel has an associated displacement vector, or it can be a sparse set of displacement vectors. Such sparse flow fields are computed by tracking feature points, for example SIFT [Low04] or KLT features [LK81, TK91]. Examples for dense OF estimation algorithms are: Lucas-Kanade [LK81], Horn-Schunck [HS81], and Brox [BBPW04]. Note that OF only represents the *apparent motion*, it may not always describe the true motion because of problems such as the aperture problem, variable lighting, and occlusions.

Motion segmentation

One way OF is used in tracking applications is for motion segmentation. This means that the dense OF field is segmented into clusters of similar motion vectors. For one thing these clusters indicate the position of moving objects in the scene and

for another thing they are also used for velocity estimation [WA96, HN99, TMS01, HTWM04].

Error accumulation in tracking

Because OF contains the motion information, it is an obvious feature candidate for tracking and it is often used to track rigid objects and even articulated models. However, every tracking algorithm that *only* relies on OF suffers from the same problem: error accumulation. A tracker based on OF uses the last known position of the tracking subject and propagates its position with the OF between the last and the current frame. This process is repeated for every new frame and therefore integrates the small errors in the OF which are unavoidable.

Rigid objects

Articulated body models are made up by rigid objects, connected by joints. It is therefore interesting to look at rigid object tracking with OF. The following survey gives an overview over the subject [LF05]. It concludes that OF-based tracking has the following problems:

- As explained above, the main problem is drift. It must be corrected by some drift correction mechanism.
- Furthermore, the OF estimation is problematic under variable lighting conditions.
- And lastly, OF estimation may fail with fast movements or low frame rates.

4.6.1 Articulated Body Models

Without drift correction

Some early approaches of articulated body model tracking with OF simply use OF to propagate the model to the next frame and do not employ a drift correction. For example Pentland and Horowitz use a model with *soft* joints, modelled as springs [PH91]. Yamamoto et al. [YSK+98] use a model with real joints. The two approaches are interesting sources, even without drift correction, because they explain how the model parameters can be inferred from OF.

Model initialisation

Meyer et al. use motion segmentation to initialise an articulated model for gait analysis. Clusters of similar flow vectors are used to determine the contours of the head, torso, and legs, the arms are not modelled. These contours are then used to track the body in subsequent frames. The resulting body model is only very coarse and the approach only works when the subject walks in a defined direction across the camera view.

Motion model

Daubney et al. use sparse OF obtained with KLT features [DGC09]. They employ a trained motion model for individual limbs and the full body to classify the observed motion field. This approach shows a good performance and was evaluated on the HumanEva dataset. However, it is restricted to the trained walking model.

Correspondences

Another approach is defining an objective function based on *correspondences* [BC08]. These correspondences are essentially displacement vectors in 2D where the starting point of each vector is allocated to a point on the body model at time t. The objective of the optimization is then to minimize the length of the displacement vectors

by adapting the body model parameters. The major advantage of this objective function is that its gradient can be derived in closed form and the optimization can be efficiently performed with gradient descent. Ballan and Cortelazzo [BC08] use sparse OF from KLT features to obtain a part of their correspondence set. The other part is obtained by finding closest points on the contours of the projected and observed silhouettes. These contour correspondences are updated during the optimization and help to correct the drift of the OF correspondences. This algorithm could track complex motion very accurately with video from four cameras running at 21pfs and a body model with 46 DoF and deformable skin.

Drift correction

Gall et al. also use correspondences for rigid object and articulated model tracking [GRS08]. They use correspondences obtained from sparse OF for their prediction step in the pose estimation. Afterwards, the estimate is refined with new correspondences obtained in two different ways. For one thing, they compute optical flow from the projection of a static textured 3D model to the observed image. For another thing they use closest points on the projected and the observed contour. By using twists [BMP04] for the model parametrisation, they are able to estimate the parameters by solving a linear system.

5 Soft Partitioning Particle Swarm Optimization

Summary

This chapter describes Soft Partitioning Particle Swarm Optimization (SPPSO) in detail. SPPSO is a *direct model use* algorithm for markerless, full body pose tracking in multi-view video. It works in an analysis-by-synthesis fashion with an articulated body model. The skeleton of this model consists of a kinematic tree with 13 joints and 31 parameters which determine the global body position and the relative joint angles. Attached to the kinematic tree are 10 truncated cones (referred to as cylinders), to model the torso, head, and limbs. The cylinder model is used to project the silhouettes and the edges to the four camera views, which can then be compared with the silhouette and the edges that were extracted from the four videos using image processing methods. A two-stage PSO is used to maximize the similarity (fitness) of a set of candidate poses to the observations.

Outline

This chapter is structured as follows: Section 5.1 describes the body model and its parametrisation, section 5.2 explains the silhouette and edge fitness functions, and section 5.3 explains the optimization step of SPPSO, after a short review of the basics of PSO in section 5.3.1.

5.1 Body Model

Source

The body model is a modification of the model used by Balan et al. [BSB05]. This model is implemented completely in Matlab and can be downloaded from `http://www.cs.brown.edu/~alb/download.htm`. There exists a newer and more accurate human model in the HumanEva framework [SBB10], but the older model was chosen because of its much simpler Matlab source code, which makes modifications easier and rendering faster. Furthermore, the older model has only 31 instead of 34 parameters. The main deficiency of the older model is that it does not model the hands and feet.

5.1.1 Kinematic Tree

Parametrisation

The kinematic tree, a coarse approximation of the human skeleton, is the base of the body model (See Figure 5.1(a)). The root node of the kinematic tree is at the centre of the pelvis. Its position and orientation are the first six variable parameters and determine the global position and orientation of the model. The rest of the total 31 variable parameters are relative joint angles. Table 5.1 lists all variable

parameters in the parameter vector of the kinematic tree. The lengths of the limbs are kept constant throughout tracking. They were determined by using motion capture data.

Implementation

All computations in 3D space (*world* coordinates) are performed in 4D homogeneous coordinates. This allows the implementation of all necessary operations for rendering, such as translation, rotation, and scaling, as matrix multiplications. The joint angle parameters are Euler angles that can be easily transformed to rotation matrices. Their ordering is chosen so that the last parameter of joints with three DoF (hips, shoulders, and neck) is a rotation around the axis of the outer skeleton segment (transformed z axis). For example: the third parameter of the left shoulder joint, x_{19}, defines the rotation around the axis of the left upper arm. After computing the transformation matrices for all joints, the position and orientation of individual limbs in the kinematic tree is computed by chaining these matrices according to the tree structure.

5.1.2 Cylinder Model

Truncated cones

The cylinder model is a modified version of the original 10-cylinder model used by Balan et al. [BSB05]. It defines the outer shape of the body with a set of truncated cones (henceforth called cylinders), which are fixed to the kinematic tree. The cylinders for the head and torso have elliptical cross-sections and the cylinders representing the limbs have circular cross sections. Figure 5.1(b) shows the projection of the cylinder model into view 1. The dimensions of the cylinders were determined by Balan et al. using motion capture data and are kept constant during tracking.

Cylinder allocation

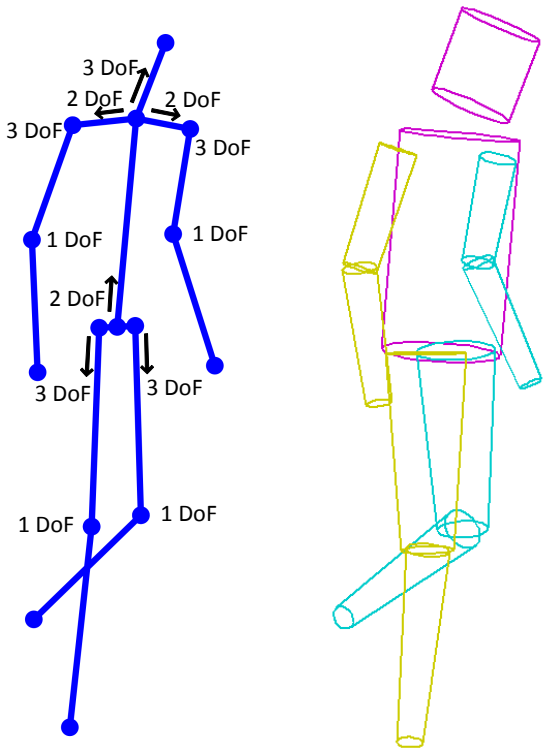
All cylinders are fixed to a certain joint of the kinematic tree. Table 5.2 lists the allocation of all cylinders. Note that the torso cylinder is fixed to the pelvis joint, i.e. the global orientation of the kinematic tree. This means that the spine joint influences the position of the head- and arm-cylinders but not the torso cylinder.

Modifications

Initial experiments showed that some modifications of the original cylinder model improve the tracking results on the Lee walk sequence. The original model was modified as follows:

- The lower limbs were lengthened by a factor of 1.2 to cover the wrists and heels.
- The head was shortened by a factor of 0.6 and offset from the torso by half its length. This is a more accurate model of the head.
- The torso was lengthened by a factor of 1.05 to align its upper edge to the observed edges in the shoulder area.
- The upper legs were thickened by a factor of 1.2 to better model the loose shorts of the subject in the Lee walk.

Figure 5.2 shows a comparison of the original and the modified cylinder model.



(a) (b)

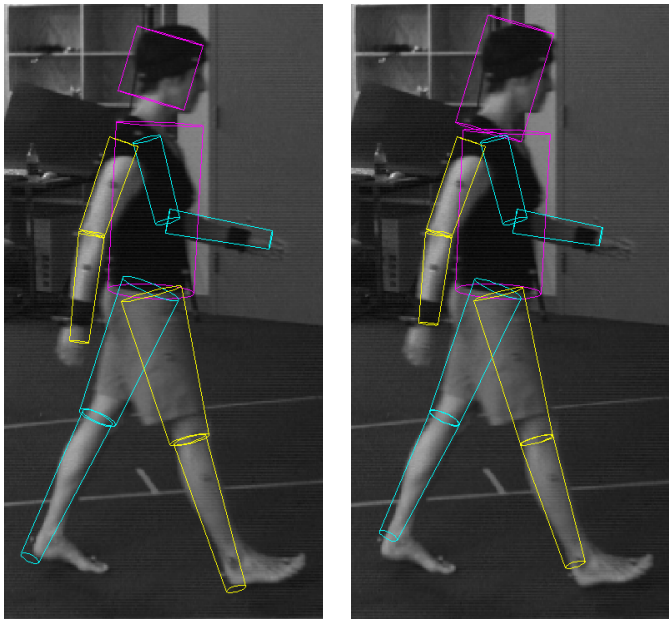
Figure 5.1 (a) The kinematic tree of the body model with the respective number of DoF for all joints. (b) Cylinder model projected into view 1. The right limbs are always shown in yellow, the left limbs in cyan.

Table 5.1 Parametrisation of the kinematic tree (only the 31 variable parameters). Angle parameters are in radians.

Parameter	**Joint**
$x_1 - x_3$	Pelvis orientation (Global orientation)
$x_4 - x_6$	Pelvis position (Global position, in mm)
$x_7 - x_9$	Left hip
x_{10}	Left knee
$x_{11} - x_{13}$	Right hip
x_{14}	Right knee
$x_{15} - x_{16}$	Left clavicle
$x_{17} - x_{19}$	Left shoulder
x_{20}	Left elbow
$x_{21} - x_{22}$	Right clavicle
$x_{23} - x_{25}$	Right shoulder
x_{26}	Right elbow
$x_{27} - x_{28}$	Spine orientation
$x_{29} - x_{31}$	Head Orientation

Table 5.2 Allocation of the cylinders to the joins of the kinematic tree.

Nr.	Cylinder	Joint
1	Torso	Pelvis (Global orientation and position)
2	Left thigh	Left hip
3	Left calf	Left knee
4	Right thigh	Right hip
5	Right calf	Right knee
6	Left upper arm	Left shoulder
7	Left lower arm	Left elbow
8	Right upper arm	Right shoulder
9	Right lower arm	Right elbow
10	Head	Neck



(a) (b)

Figure 5.2 (a) The modified cylinder model used in this thesis. (b) The original cylinder model [BSB05]. Both models projected into view 1.

5.2 Fitness Function

Observations

The above described body model is used to compute the fitness of candidate poses, defined by the parameters $x_1 - x_{31}$. The fitness indicates how well a candidate pose matches the observations. Since the fitness is only computed for individual time instants, the observations are the images from all four views at the time instant.

Composition

The fitness $f = f_s + f_e$ is the sum of two terms: the *silhouette fitness* f_s and the *edge fitness* f_e. Both terms are normalized to lie in the range between 0 and 1, where 0 means no match and 1 means complete match. Both partial fitness terms are defined on a single observed image. The total fitness is computed by first averaging

the partial fitness values of all four views and then summing the two averaged partial fitness values.

5.2.1 Edge Fitness

Robust feature

Edges are an image feature that can be computed robustly under lighting and background changes, and human subjects usually produce strong edges along the outline of the body and individual limbs. Edges are therefore a valuable feature for motion tracking [DR05]. The edge fitness f_e is computed similar to the *edge likelihood* used by Sigal et al. [SBB10].

Edge map

The edges in an observed image are detected by convolving the image with a Sobel operator and thresholding. The binary edge image is first masked with the dilated silhouette to remove spurious edges in the background, then it is blurred with a Gaussian kernel and rescaled to the range [0,1] to produce the *edge map*. The Gaussian blurring is used to approximate a distance map at a low computational cost. Balan et al. compared different edge detection methods and distance maps for motion tracking. They concluded that a more sophisticated edge map does not improve the tracking when combined with a silhouette fitness [BSB05].

Fitness computation

To compute the edge fitness f_e for a candidate pose, the edge map is sampled at discrete points along the visible edges of the candidate pose (as introduced by Deutscher et al. [DBR00]). Let $p(i)$ denote the value of the edge map at pixel i and S the set of sampling points along the candidate pose's edges. The edge fitness is then

$$f_e = \frac{1}{|S|} \sum_{k \in S} p(k). \tag{5.1}$$

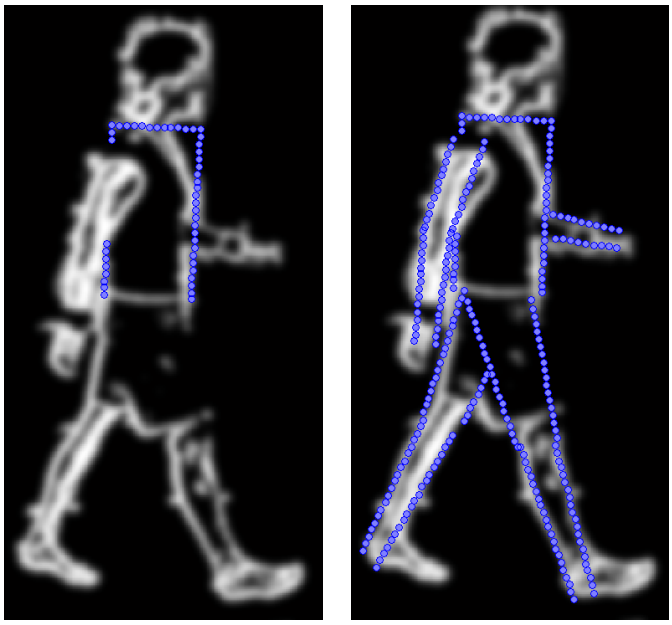
f_e is approximately 1 if all sampling points lie on the detected edges and 0 if all points are far away from edges.

Discrete points

The discrete sampling points on the candidate pose are computed using the framework of Balan et al. [BSB05]. The edge fitness can be computed for all, or for individual cylinders. At the first stage of SPPSO, only the torso cylinder is considered for the edge fitness.

Special cylinders

Normally, only the edges parallel to the cylinder axes of the model are considered. The torso is an exception from this rule. Its upper edge is also sampled because it provides a valuable hint for the z-location of the model (See section 6.10). The head cylinder is never used for the edge fitness because its shape is only a very crude approximation of the head's shape. Figure 5.3 depicts the sampling points for the edge fitness overlaid on the edge map.



(a) (b)

Figure 5.3 Sampling points for the edge fitness function, overlaid on the edge map. (a) Only the torso cylinder at the first stage of SPPSO. (b) All cylinders except the head at the second stage of SPPSO.

5.2.2 Silhouette Fitness

Overlap

The silhouette fitness measures the overlap of the observed silhouette and the projected silhouette. The *observed silhouette* is a binary image, obtained by foreground-background segmentation of the observed image. And the *projected silhouette* is obtained by projecting the cylinders of the candidate pose into the respective view.

Bidirectionality

A good silhouette fitness must be bidirectional (i.e. symmetric) [ST02, SBB10]. This means that it must measure how much of the projected silhouette falls into the observed, as well as how much of the observed silhouette falls into the projected. This is necessary to prevent unreasonably high fitness values for poses that have overlapping limbs (See [SBB10] for a detailed explanation.

Computation

The used silhouette fitness is based on the bidirectional silhouette log-likelihood used by Sigal et al. [SBB10]. It is computed as follows: Let $M_o(i)$ represent the observed silhouette image at pixel i and $M_p(i)$ the projected silhouette. Furthermore, let **R**ed, **B**lue, and **Y**ellow, be the following sums over all pixels in the binary silhouette images:

$$R = \sum_i M_o(i)(1 - M_p(i)) \tag{5.2}$$

$$B = \sum_i M_p(i)(1 - M_o(i)) \tag{5.3}$$

$$Y = \sum_i M_o(i)M_p(i)) \tag{5.4}$$

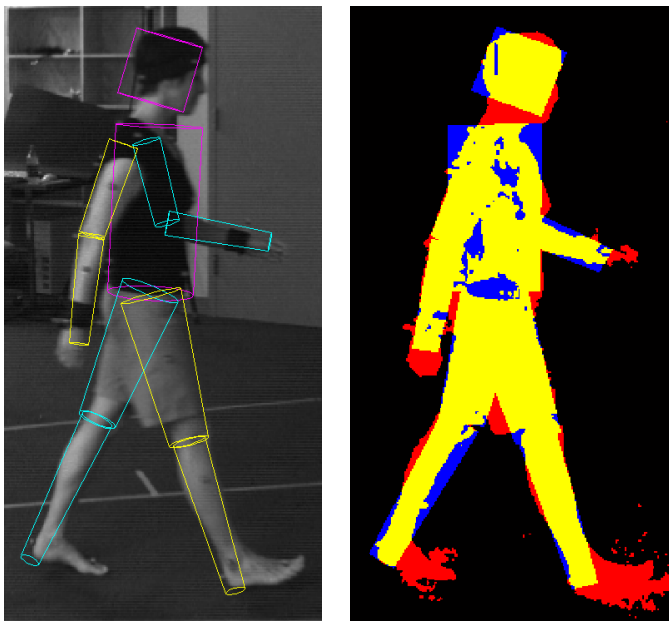
In words: R is the area that lies in the observed, but not in the projected silhouette, B the area that lies in the projected, but not in the observed silhouette, and Y the overlap area of both silhouettes (See Figure 5.4 for an illustration of this segmentation.). The silhouette fitness is then computed as

$$f_s = (1-a)\frac{Y}{B+Y} + a\frac{Y}{R+Y}, \quad a = \frac{1}{2}. \tag{5.5}$$

Hence, f_s is 1 when the two silhouettes are identical, and 0 when there is no overlap. The formula would allow an asymmetric weighting of the R and B by changing a, but the above definition is used throughout this thesis. f_s is equivalent to the inverse of the log-likelihood of Sigal et al.: $f_s = 1 + \log p^d(\mathbf{y}_t|\mathbf{x}_t)$. Figure 5.5 shows an overview of the whole fitness computation with edge- and silhouette fitness.

Full body evaluation

Note that a bidirectional silhouette fitness only makes sense when evaluated over the full body. For example: when only a single arm cylinder would be projected for the evaluation, it would normally yield a higher fitness value when it is projected onto the torso. This is because the model and the observed silhouette are not perfect. Consequently, the arm cylinder always protrudes from the observed arm silhouette at some places, yielding a slightly higher fitness than when the arm is projected onto a bigger body part in the observation.



(a) (b)

Figure 5.4 Silhouette fitness f_s. (a) Projected cylinders of the body model. (b) Image segmentation for the silhouette fitness. Red: in observed silhouette but not in projected, blue: in projected but not in observed, yellow: overlap of both silhouettes.

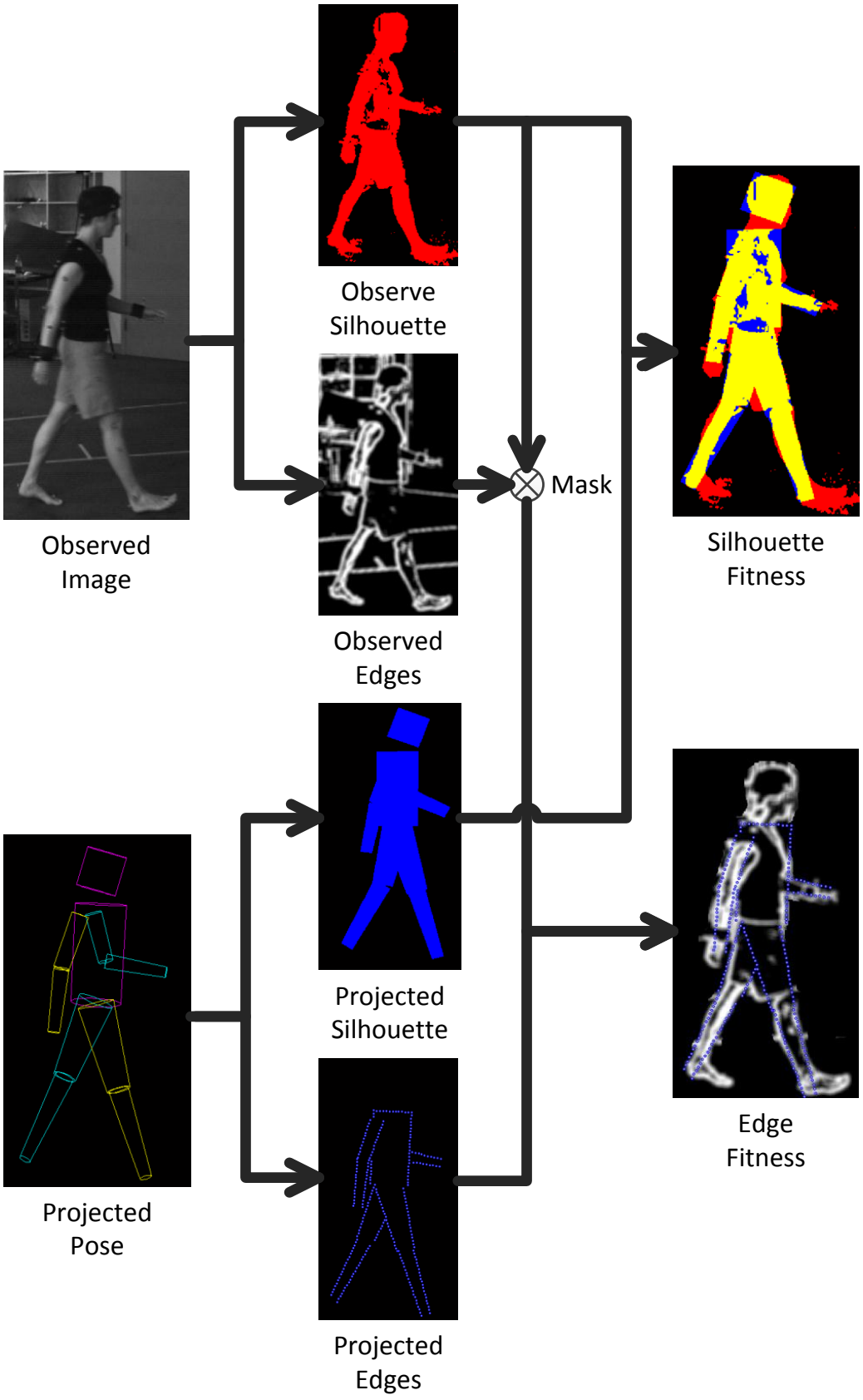


Figure 5.5 Overview over the computation of the silhouette and edge fitness.

5.3 Optimization

Outline

The basics of Particle Swarm Optimization (PSO) are summarized in the following subsection 5.3.1 because it constitutes the core of Soft Partitioning PSO (SPPSO). The subsequent subsections give the details of the full SPPSO algorithm.

5.3.1 Particle Swarm Optimization

Algorithm

PSO is a stochastic method for global optimization (also called a metaheuristic), inspired by the flocking behaviour of birds and other animals. It has been used in a great variety of applications because of its simplicity and ability to optimize nonlinear and multidimensional problems [ES01]. The PSO Algorithm works with a set of particles, called swarm, which move through the search space. Each particle consists of three D-dimensional, real-valued vectors: a position x_i, the best position it has found so far p_i, and its velocity v_i (Note that the commonly used term *velocity* for v_i is misleading, it is a translation vector). Additionally, the algorithm keeps track of the global best particle position p_g. To start the optimization, the positions and velocities are randomly initialised. The positions are often distributed uniformly over the search space $\mathbb{S}^D \subset \mathbb{R}^D$. After initialisation, the particles move through the search space by updating their position in every dimension d according to the following equations:

$$v_{id} = wv_{id} + c_1\epsilon_1(p_{id} - x_{id}) + c_2\epsilon_2(p_{gd} - x_{id}), \tag{5.6}$$

$$x_{id} = x_{id} + v_{id}. \tag{5.7}$$

That means that they move according to their own previous velocity (term one in (5.6)), but are also draw towards their own best and the global best position (term two and three). The parameters w, c_1, and c_2 control the algorithm's two contradictory tendencies of exploring the search space and converging to a local optimum. ϵ_1 and ϵ_2 are independent and uniformly distributed random variables in the interval $[0, 1]$, they are uniquely generated for every iteration. Furthermore, it is common practice to limit the components of the particle velocity to a constant vector V_{max}. Algorithm 1 summarizes the basic PSO algorithm.

Algorithm 1 The PSO update process [BK07].

```
for each iteration do
  for each particle i in the swarm do
    update position x_i using (5.6) and (5.7)
    calculate particle fitness f(x_i)
    update p_i and p_g
  end for
end for
```

History

Particle Swarm Optimization was introduced by Kennedy and Eberhart in 1995 [KE95]. Soon after that, Shi and Eberhart added the inertia weight parameter w

to the original algorithm [SE98]. This PSO with inertia weight is the most widely known version of the algorithm. It uses the update equations (5.6) and (5.7). In 2002, Clerc and Kennedy [CK02] published their findings on the convergence behaviour of PSO and suggested to modify the velocity update equation as follows:

$$v_{id} = \chi(v_{id} + c_1\epsilon_1(p_{id} - x_{id}) + c_2\epsilon_2(p_{gd} - x_{id})) \tag{5.8}$$

where the constriction factor χ is calculated as:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \quad \varphi = c_1 + c_2. \tag{5.9}$$

The constricted PSO will consistently converge to a local optimum if $\varphi > 4$, with faster convergence for larger φ. Common parameter values are: $c_1 = c_2 = 2.05$, resulting in $\chi = 0.72984$. Note that the constricted PSO update rule is algebraically identical to the PSO with inertia weight. The constriction factor just ensures convergence.

Communication topology

The original algorithm uses a global communication topology called *gbest*, meaning that every particle in the swarm "knows" about the global best particle p_g and is drawn towards it. But shortly after their original paper [KE95], Kennedy and Eberhart proposed the *lbest* topology [EK95]. In this algorithm variant, a particle only knows about the best particles within a certain neighbourhood. A simple example of such a neighbourhood is a ring topology where every particle has only two neighbours.

Standard PSO

There exist many variations of the PSO algorithm, and the update rule with inertia weight (5.6) is still widely used as a starting point for new developments. In 2007, Bratton and Kennedy [BK07] suggested a more modern standard definition of the algorithm with the following properties:

- a ring communication topology (*lbest*),
- the constricted velocity update rule as shown in equation (5.8),
- 50 particles,
- boundary conditions wherein a particle is not evaluated when it exits the search space.

Convergence speed

The *lbest* variant of PSO is more immune to local optima than the *gbest*, but this comes at the price of slower convergence [EK95]. Bratton and Kennedy also compared the performance of these two variants, and proposed the *lbest* variant for the standard PSO because it achieves better fitness values for standard benchmark functions [BK07]. However, the fitness values using *lbest* are often worse than with *gbest* for tens of thousands of iterations. Because fitness evaluations are computationally very expensive in human pose tracking, the *lbest* topology is a bad choice. Most pose tracking algorithms use between 1000 and 10'000 fitness evaluations and a *gbest* topology [IT06, ZHW+10, KKW11b].

5.3.2 Optimization Formulation

Sequential optimization

Pose tracking with SPPSO is done by maximizing the fitness function for every new frame. The estimated pose from the previous frame is used to initialise the optimization. Hence, the tracking process is a series of static optimizations. Furthermore, these optimizations are divided into two stages and both stages use a constricted PSO.

Candidate poses

Each particle in the PSO constitutes a candidate pose. Its position vector consists of the variable parameters of the body model (i.e. the position and angles of the kinematic tree). The initial particle positions x_i^t are sampled from a multivariate normal distribution centred around the estimated pose from last frame $\hat{x}^{t-1}$.

$$x_i^t \leftarrow \mathcal{N}(\hat{x}^{t-1}, \Sigma), \quad \Sigma = \begin{pmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_{31}^2 \end{pmatrix}, \quad \sigma = \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_{31} \end{pmatrix} \tag{5.10}$$

Σ is the same diagonal covariance matrix as used for the first annealing layer in the annealed particle filter by Balan et al. [BSB05]. The standard deviations σ_d in Σ are equal to the maximum absolute inter-frame differences of the body angles in a training set of motion capture data at 60fps. For example: σ_4 (x-translation) is 13.7mm an σ_{10} (left knee angle) is 0.093 rad. The distribution $\mathcal{N}(\hat{x}^{t-1}, \Sigma)$ can be interpreted as a prior probability for the parameters at time t. It is therefore reasonable to sample the initial particle set from this distribution.

Motion model

The used training set focuses primarily on walking motions. Therefore, this covariance matrix can be regarded as a weak model for walking motions. This bias towards walking motions could be removed by using a training set with more diverse motions. But this would enlarge the search space and therefore make the tracking of a walking subject more difficult. For experiments at slower frame rates than 60fps, σ is always upscaled accordingly. That is, at a frame rate of 20fps σ is multiplied by three.

V_{max}

The particle velocity is limited to two times the standard deviation in every dimension. This is because it was found that the PSO produces many unreasonable poses when the velocity is not limited, especially during the first few iterations. The initial particle velocities are sampled from a uniform distribution over the range $+/-\sigma$.

Constraints

The optimization is subject to two constraints:

- The angles must remain inside anatomical joint limits.
- The limbs may not inter-penetrate.

These constraints are equal to the *hard priors* of Balan et al. [BSB05]. They were found to improve the tracking performance significantly by Balan et al. because they reduce the search space. The constraints are enforced by resampling the particle velocity until either the constraints are met or the maximum number of 10 attempts is exceeded.

Single stage optimization

Algorithm 2 shows the PSO that is used at the two stages of SPPSO. The coefficients $c_1(k)$ and $c_2(k)$ are increased linearly from 2.05 to 2.15 during the optimization to gradually increase the algorithm's tendency to converge. Consequently, $\chi(k)$ is adapted according to equation 5.9 for every iteration. This can be seen as an annealing scheme which was introduced to enforce swarm convergence even with a limited number of iterations N.

Algorithm 2 Constricted PSO with enforced constraints for one stage of SPPSO.

sample particle positions $x_i \leftarrow \mathcal{N}(\hat{x}^{t-1}, \Sigma)$
sample particle velocities $v_i \leftarrow \mathcal{U}(-\sigma, \sigma)$
calculate particle fitness: $f(x_i) = f_s(x_i) + f_e(x_i)$
update particle best p_i and global best p_g
for each iteration $k = 2$ **to** N **do**
  for each particle i in the swarm **do**
    repeat
      for each dimension d **do**
        $v_{id} = \chi(k)(v_{id} + c_1(k)\epsilon_1(p_{id} - x_{id}) + c_2(k)\epsilon_2(p_{gd} - x_{id}))$
      end for
      limit abs(v_i) to 2σ
      $x_i = x_i + v_i$
    until x_i meets constraints
    calculate particle fitness: $f(x_i) = f_s(x_i) + f_e(x_i)$
    update particle best p_i and global best p_g
  end for
end for

5.3.3 Soft Partitioning Stages

The optimization of the pose is divided into two hierarchical stages. Both stages are complete optimizations with the above described PSO and the estimated pose from the first stage is used as the initialisation for the second stage.

Error accumulation

Pose estimation which is divided into hierarchical stages with hard partitions suffers from error accumulation. This happens because the fitness function for one stage cannot be evaluated completely independently from subsequent stages. The fitness function based on silhouettes and edges cannot be evaluated separately for individual body parts because there is no segmentation of these parts. Edges produced by the left lower arm, for example, cannot be discerned from edges produced by the torso. Therefore, the torso cylinder cannot be localized unambiguously without localizing the lower arm cylinder.

Soft partitioning

To avoid error accumulation, SPPSO uses a soft partitioning scheme. Figure 5.6 illustrates the principle of soft partitioning compared to hard (hierarchical) partitioning and global optimization. As in hierarchical schemes, the search space is partitioned according to the model hierarchy. The most important parameters are optimized first, while the less important are kept constant. The crucial difference to hard partitioning is that the previously optimized parameters are allowed some variation in the following stage. Soft partitioning reduces the search space not as

much as hard partitioning but the search space is much smaller than in a global optimization. This allows a much more efficient optimization.

Partitions

SPPSO has two hierarchical stages: In the first stage, only the six first parameters $x_1 - x_6$ (global orientation and position) are optimized. The second stage is a global optimization over all parameters. But the standard deviations for $x_4 - x_6$ (global position) are reduced to one tenth. Experiments showed that the tracking performance is not significantly increased when the optimization is further divided into three stages. However, the soft partitioning scheme performs much better than global optimization or hard partitioning.

Soft partitions with PSO

Standard PSO is a global optimization method. This means that the particles can generally explore the whole search space, given enough iterations. To divide the search space into soft partitions, the movement of the particles most be constrained in the required dimensions. In SPPSO the partitioning is done by downscaling the standard deviation vector σ for these dimensions. This influences the PSO in three ways:

- The initial sampling distribution is narrower in the downscaled dimensions.
- The initial particle speed is smaller in the downscaled dimensions.
- By limiting the particle velocity to 2σ and through the limited number of iterations, the part of the search space that a particle can explore during the optimization is narrower in the downscaled dimensions.

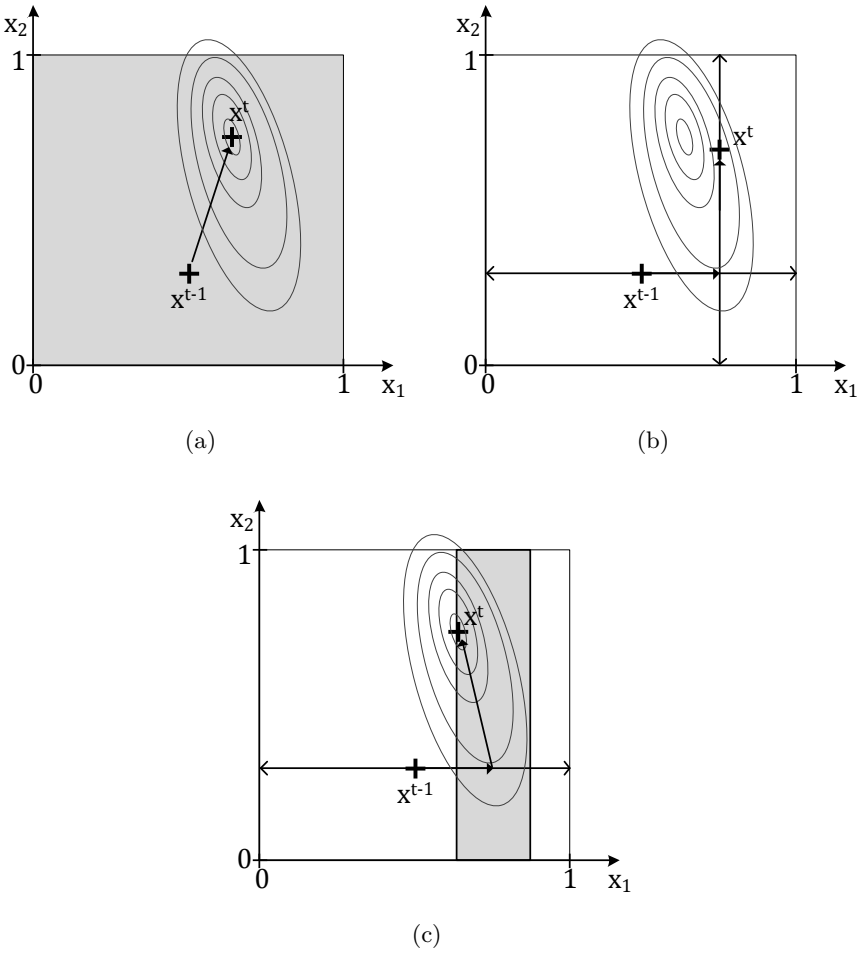


Figure 5.6 Illustration of different partitioning schemes by the example of a optimization with two parameters. x^{t-1} denotes the initial and x^t the new estimate. (a) Global optimization. Here, the optimizer searches the whole search space (grey) at once. (b) Hierarchical optimization. At the first stage, x_1 is optimized while x_2 is kept constant. At the second stage, x_1 is kept constant while x_2 is optimized. Consequently, the optimizer cannot correct the suboptimal estimate of x_1 from the first stage. (c) Soft partitioning. The first stage is identical to the hierarchical scheme, but x_1 is allowed some variation at the second stage. Therefore, the optimizer finds a better estimate.

6 Experiments

This chapter presents the experimental evaluation of SPPSO. First, SPPSO is compared to the standard benchmark algorithm, the annealed particle filter (APF). SPPSO outperforms the APF at a frame rate of 20fps and performs equally well at 60fps. After this comparison, the maximum obtainable tracking accuracy with the used body model is established. Furthermore, the effect of various algorithm parameters of SPPSO, such as number of particles, iterations, and stages is examined. The impact of different fitness functions is also shown in comparative experiments. Moreover, the convergence behaviour of the PSO inside SPPSO is shown. And finally, the required computation time for different parts of the SPPSO algorithm is analysed.

Summary

6.1 Experimental Setup

Base configuration

The base configuration of SPPSO with two stages, shown in the following Table 6.1, was used for all the experiments unless otherwise specified.

Table 6.1 Base configuration for SPPSO.

Stage	Particles	Iterations	Edge fitness	Silhouette fitness
1	10	20	only torso	full body
2	20	40	full body	full body

1000 evaluations

The base configuration requires 1000 fitness evaluations per frame (particles · iterations). Keeping the number of evaluations fixed allows a fair comparison to other algorithms because fitness evaluations (including rendering) dominate the total processing time (See 6.11). 1000 evaluations per frame is the standard number of evaluations in the HumanEva framework [BSB05, SBB10].

Dataset

The experiments were performed on the Lee walk sequence of the HumanEva framework (See section 4.5). Figure 6.1 illustrates the action of the subject during the sequence that is 8.8s long. The subject walks counter-clockwise in a circle, interrupted by a short period of standstill from frame 330 to 430. The maximum frame rate of the Lee walk is 60fps, but it is downsampled to 20fps for many experiments.

Error graphs

The tracking results are presented as error graphs over the first 450 frames of the

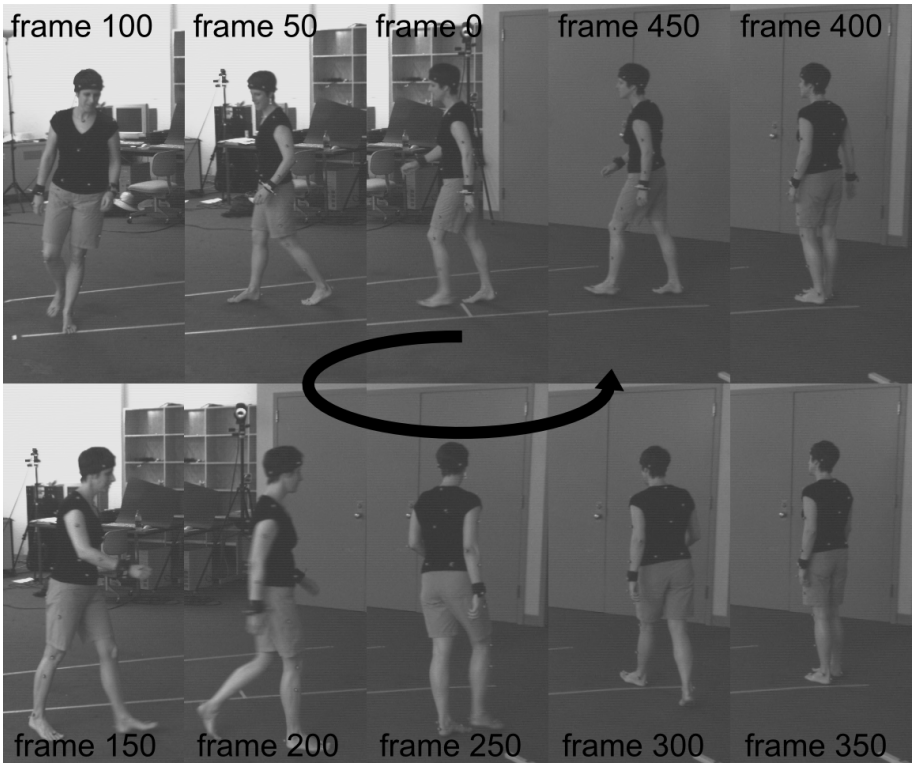


Figure 6.1 10 cropped frames of the lee walk sequence from view 1.

Lee walk. This is a sufficiently long sequence to show that a tracking algorithm does not exhibit drift. The error values are computed using the standard error measure (4.4) and denoted by D. For experiments at 60fps, the errors are computed for every frame. For experiments that estimate the pose at 20fps, the errors are only computed for the frames that were used for pose estimation, i.e. every third frame. Full error graphs are more informative than only average errors over the whole sequence because they show how the error behaves during different actions (e.g. walking or standstill). The error of SPPSO at frame 0 is always zero because the algorithm is initialised with the ground truth pose.

Multiple runs

Because SPPSO is a stochastic algorithm, the tracking results vary for every run (One run is one tracking of the sequence). Several runs were therefore carried out for most experiments. Most often the mean and the maximum error over the performed runs are shown. This allows conclusions about the accuracy and robustness of the tested algorithms.

Head Rotation

The parameter x_{31}, which determines the rotation of the head cylinder around its own axis, was kept constant throughout all experiments. This is because the head cylinder is only a very crude approximation of the head's shape and the head rotation can therefore not be estimated accurately with this model. Furthermore the head rotation has no influence on the error measure because the marker is located on the axis of the head cylinder (see Figure 4.3).

6.2 General Results

Accuracy

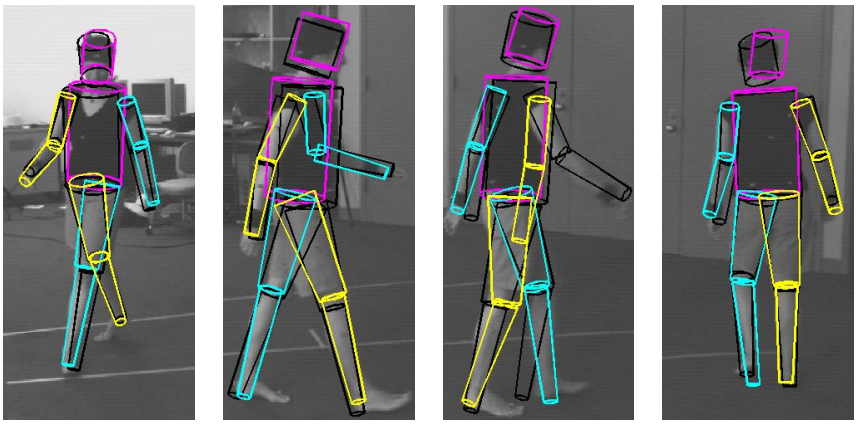
Figures 6.2 to 6.4 give an overview of the tracking results obtained with SPPSO. Figure 6.2 depicts results with the base configuration at 20fps. The tracking is clearly not perfect, but it can generally follow the body configuration. However, the right arm is completely lost in frame 216 and then reacquired in frame 279. This is a typical tracking error at the low frame rate of 20fps. In Figure 6.3, still with the base configuration but at 60fps, the arm is not lost in the depicted frames. The tracking is also generally more accurate (for example the head in frame 279). Finally, with 4000 evaluations per frame at 60fps, the tracking is very accurate in Figure 6.4. Generally, the tracking gets more accurate with higher frame rates and more evaluations per frame, i.e. with a higher evaluation rate.

Lost limbs

Lost limbs, like in frame 216 of Figure 6.2, appear as outliers in the error graphs (for example in Figure 6.5(b)). Very high error values (> 80mm) are normally the result of confused legs. Section 6.7 shows the errors of individual marker joints at different frame rates.

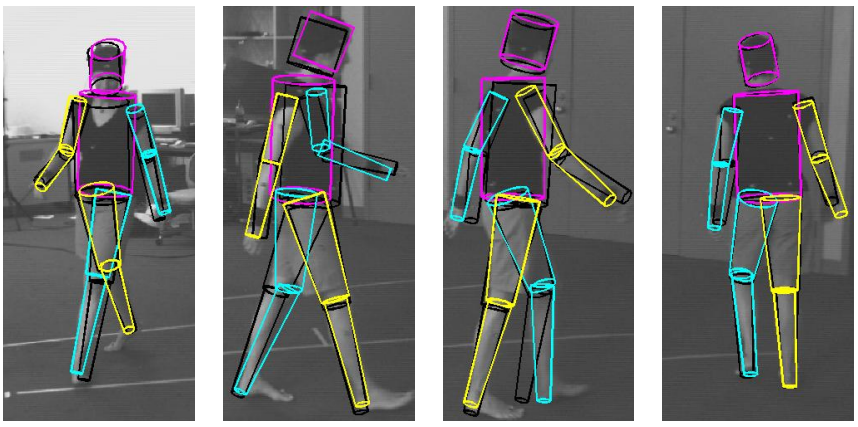
Ground truth

The ground truth pose, depicted in black in Figures 6.2 to 6.4, is not perfect. The left lower leg in frame 216 is for example far off from the real position. This is one of the reasons why the minimally obtainable error of SPPSO is not closer to zero. See section 6.4 for experiments to determine the minimally obtainable error.



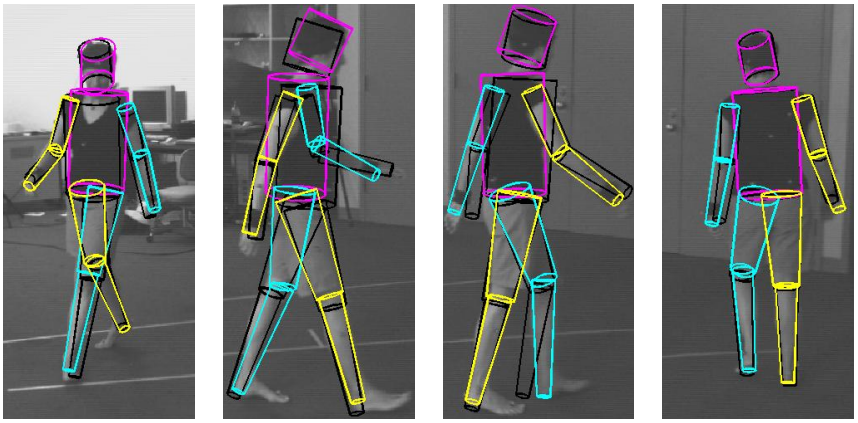
(a) f 81, D=34mm (b) f 186, D=42mm (c) f 216, D=76mm (d) f 279, D=53mm

Figure 6.2 SPPSO tracking results at 1000 evaluations per frame and 20fps. Ground truth cylinders are shown in black, estimated cylinders are coloured to distinguish left and right limbs. Results are shown at frames 81, 186, 216, and 279. D denotes the tracking error at the depicted frame.



(a) f 81, D=36mm (b) f 186, D=41mm (c) f 216, D=43mm (d) f 279, D=29mm

Figure 6.3 SPPSO tracking results at 1000 evaluations per frame and 60fps.

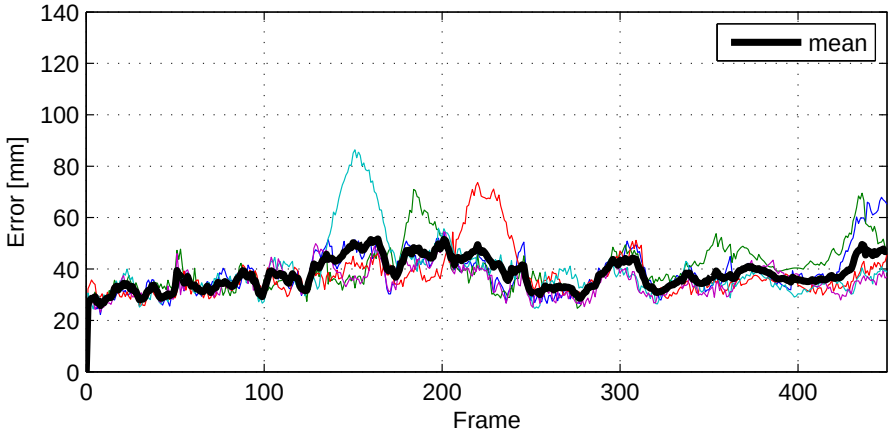


(a) f 81, D=35mm (b) f 186, D=57mm (c) f 216, D=37mm (d) f 279, D=29mm

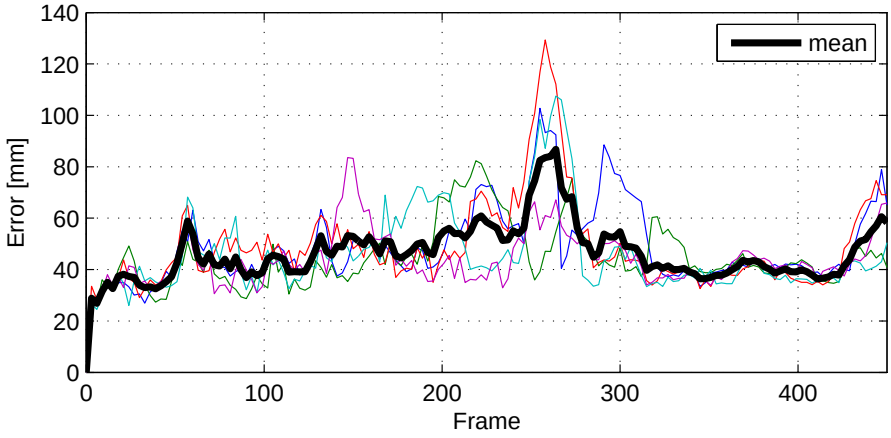
Figure 6.4 SPPSO tracking results at 4000 evaluations per frame and 60fps.

6.3 Base Configuration

The error graphs produced by SPPSO with base configuration at 60fps and 20fps are shown in Figure 6.5. The mean error is a little smaller at 60fps and the tracking is generally more stable. The outliers in the graph at 60fps come from a temporarily lost arm. At 20fps, the maximum error is much higher because SPPSO often looses track of the legs around frame 250 and also looses arms frequently. Figure 6.6 illustrates how SPPSO looses multiple limbs but reacquires them after some frames at 20fps. The ability to recover from tracking failures is an important feature for pose tracking algorithms. As expected, the tracking is always very good during the standstill period (frame 330 to 430).

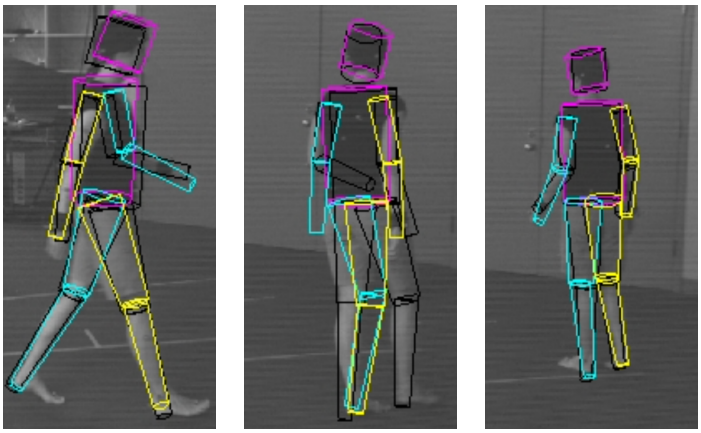


(a) 60fps, D: mean = 38.0 mm, max = 86.4 mm



(b) 20fps, D: mean = 46.3 mm, max = 129.4 mm

Figure 6.5 3D tracking error of SPPSO with base configuration (1000 evaluations per frame) for the Lee walk sequence. The graphs show five individual runs and the mean error.



(a) f 192, D=48.1mm (b) f 249, D=90.7mm (c) f 318, D=34.0mm

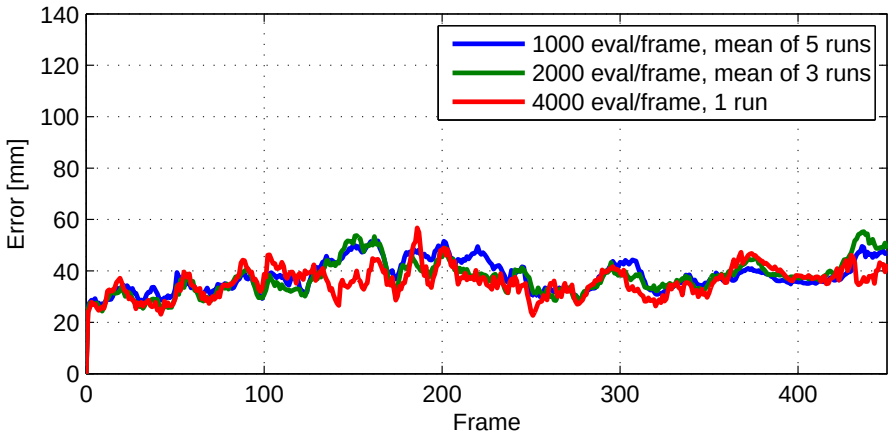
Figure 6.6 SPPSO tracking results with the base configuration at 20fps. The tracker temporarily looses the legs and one arm but can recover in later frames.

6.4 Minimal Error at 60fps

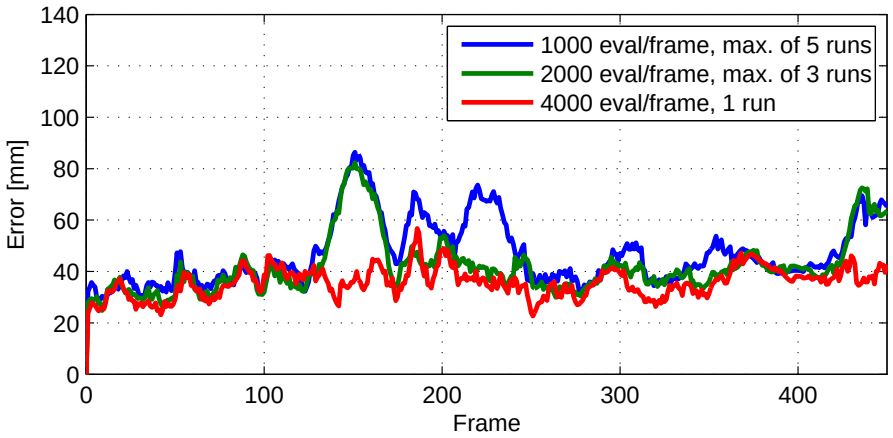
The minimal obtainable tracking error of SPPSO is mainly limited by two things. First, the ground truth poses are not perfectly accurate. And second, the body model is very coarse. To establish the minimal error with the used body model, SPPSO was run with high numbers of evaluations at the maximum frame rate of 60fps. Figure 6.7 shows the results for 1000, 2000, and 4000 evaluations per frame. There are still some limited tracking errors at 1000 and 2000 eval/frame visible in Figure 6.7(b), but at 4000 eval/frame, the tracking has reached the minimal error. The mean tracking error, depicted in Figure 6.7(a), is almost the same for all three configurations, it is not significantly decreased by more evaluations. The minimal mean error is reached at about 35mm (See Table 6.2).

Table 6.2 Accuracy of SPPSO at 60fps with different evaluation rates. The table shows mean and maximum 3D error on the first 450 frames of the Lee walk sequence.

eval/frame	**1000**	**2000**	**4000**
runs	**5**	**3**	**1**
mean error [mm]	38.0	37.3	35.7
max error [mm]	86.4	82.3	56.7



(a)



(b)

Figure 6.7 Mean and maximum 3D tracking error of SPPSO at 60fps and different evaluation rates for the Lee walk sequence.

6.5 Comparison of SPPSO to APF

Standard benchmark

SPPSO was compared to APF in this experiment, which is the benchmark algorithm of the HumanEva framework [BSB05, SBB10]. See subsection 4.4.4 for a discussion of APF. The comparison shows that SPPSO performs better than APF at 20fps and equally at 60fps.

Same body model

Both algorithms used the same body model and the same fitness function. In the particle filter terminology, the fitness is called negative log likelihood. It was simply computed as

$$-\log p(y|x) = f_e + f_s. \tag{6.1}$$

Results

The experiment was performed with the standard number of 1000 evaluations per frame for both algorithms. Figure 6.8 shows the resulting mean error graph at 60fps. The two algorithms perform about equally well at this frame rate. For the

experiment at 20fps, the sampling covariance Σ of the APF was upscaled exactly as for SPPSO. Figure 6.9 shows that SPPSO performs significantly better at this frame rate. The better performance probably comes from the direct exploitation of the hierarchical model in SPPSO. APF on the other hand, relies on an automatic soft partitioning.

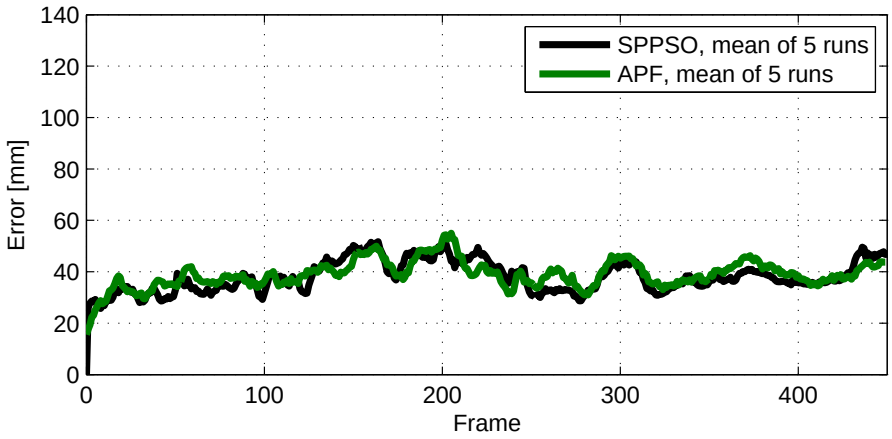
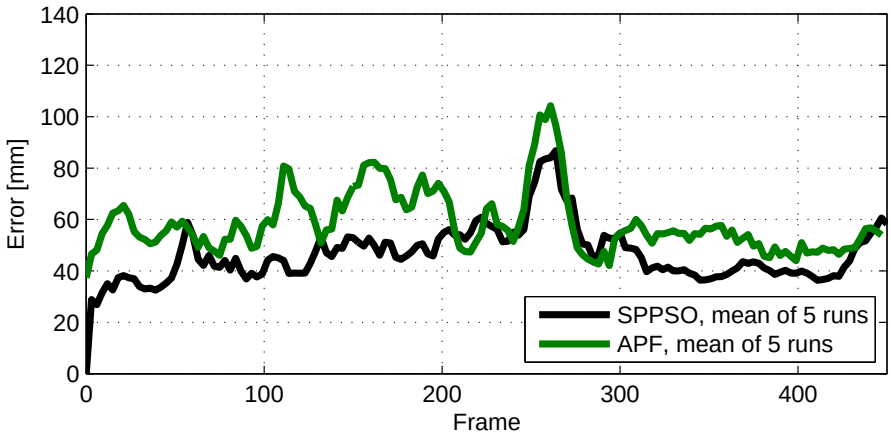
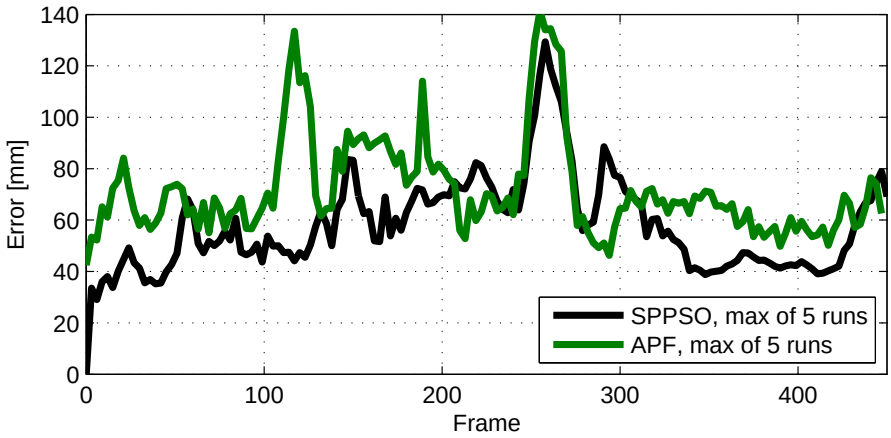


Figure 6.8 Comparison of the mean 3D tracking error of APF and SPPSO at 1000 evaluations per frame and 60fps for the Lee walk sequence.



(a)



(b)

Figure 6.9 Comparison of the mean and maximum tracking error of SPPSO and APF at 1000 eval/frame and 20fps for the Lee walk sequence.

6.6 Partitioning Schemes

The following experiments have been carried out to demonstrate the superiority of soft partitioning for articulated pose tracking compared to global optimization and hard partitioning.

Hard partitioning

Figure 6.10 shows the performance of SPPSO compared to a hard partitioning algorithm. The two algorithms are completely identical, except that the first six parameters (global orientation and position) are kept constant in the second stage for the hard partitioning algorithm. As a result, the tracking performance gets much worse. The torso is often badly located and even the legs get confused. While the leg confusion could be prevented by more accurate anatomical constraints, the bad localization of the torso clearly comes from the hard partitioning.

Global optimization

Figure 6.11 shows the comparison of SPPSO to a global optimization with 25 par-

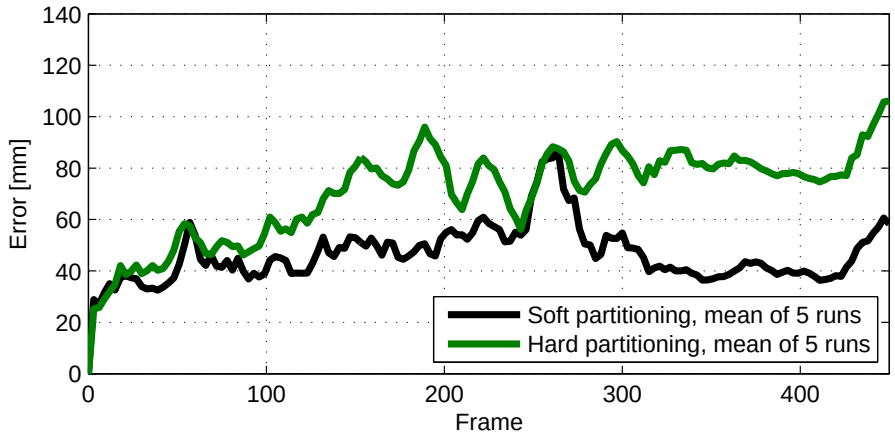


Figure 6.10 SPPSO compared to hard partitioning with two stages at 20fps.

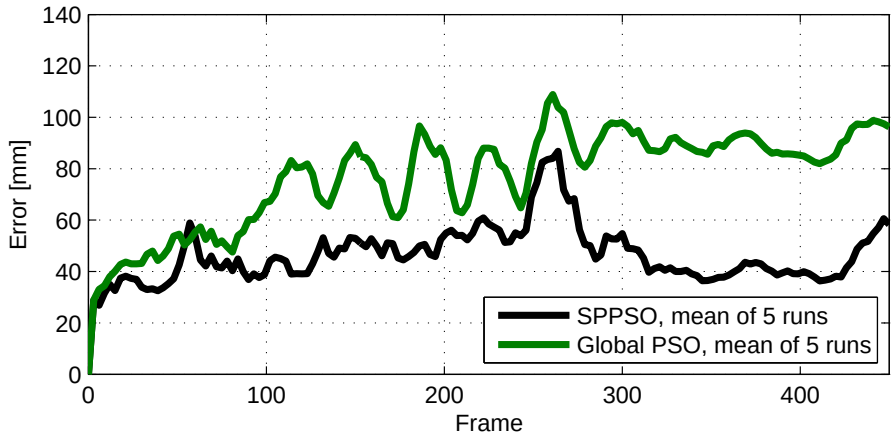


Figure 6.11 SPPSO compared to global optimization at 20fps.

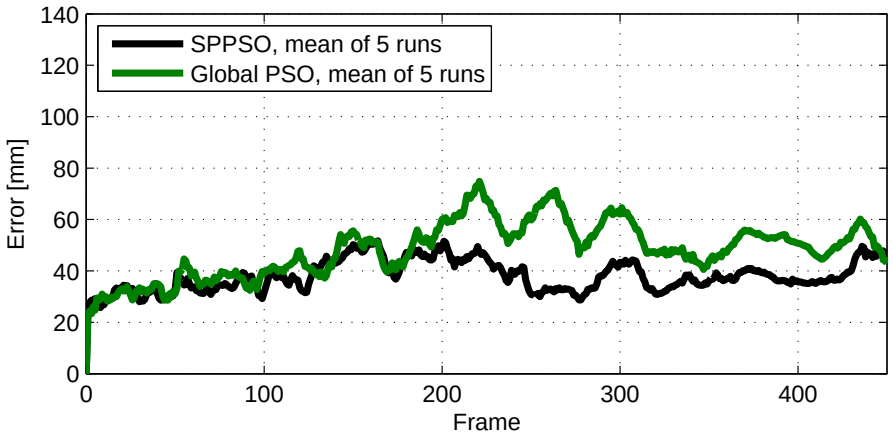
ticles and 40 iterations at 20fps. Again, SPPSO performs much better at 20fps. However, the difference is less pronounced at 60fps (See figure 6.12. This is probably because the first stage of SPPSO is especially useful for handling the larger displacements of the subject between frames at lower frame rates.

Three partitions

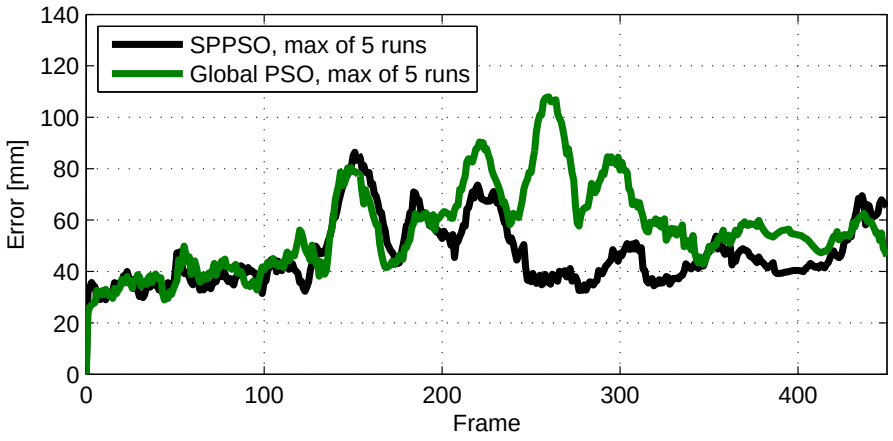
The concept of SPPSO can be extended to more than two partitions. Figure 6.13 shows the results of an experiment with three partitions compared to the SPPSO base configuration. Table 6.3 shows the used number of particles and fitness functions a the individual stages for the three-partition SPPSO. The optimization sequence goes as follows:

1. Optimization of the six parameters for global orientation and position.
2. Optimization of all parameters except the ones that only affect the lower limbs $(x_9, x_{10}, x_{13}, x_{14}, x_{19}, x_{20}, x_{25}, x_{26})$.
3. Global optimization.

The soft partitioning is done by scaling σ for the tree stages as follows (the scaling is always relative to the standard σ):



(a)



(b)

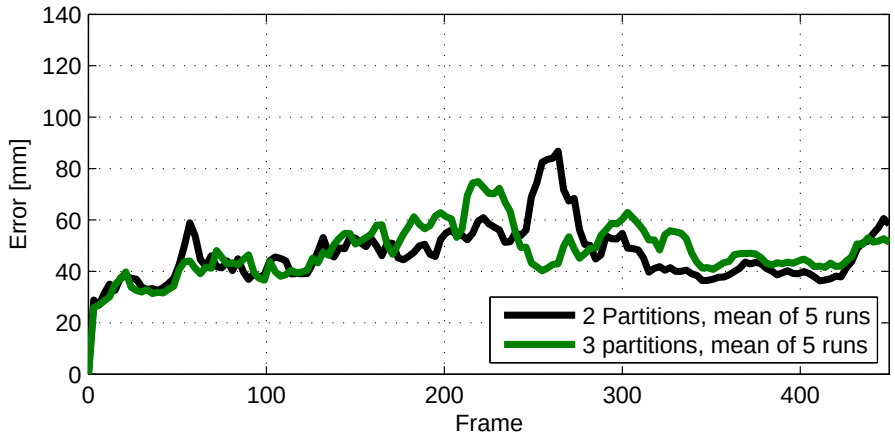
Figure 6.12 SPPSO compared to global optimization at 60fps.

1. Standard σ
2. downscale $\sigma_4 - \sigma_6$ by ten (global position).
3. downscale $\sigma_1 - \sigma_6$ by ten (global orientation and position), downscale the parameters optimized in the previous stage by two, upscale σ for the remaining parameters by $\sqrt{2}$.

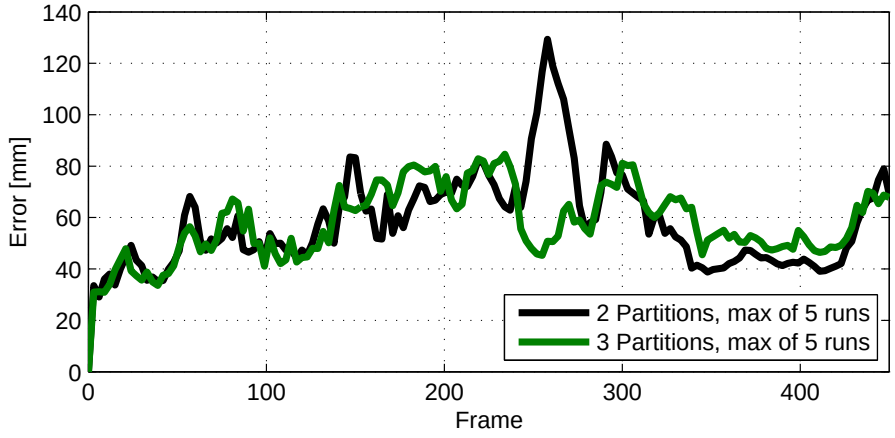
These scaling factors were determined empirically and could certainly be optimized. As can be seen in Figure 6.13, the three-partition SPPSO has about the same accuracy over the whole sequence as the base configuration with the same number of fitness evaluations. The robustness is even better (lower maximum error), but the accuracy during the standstill period is a bit worse.

Table 6.3 SPPSO with 3 partitions

Stage	Particles	Iterations	Edge fitness	Silhouette fitness
1	10	20	only torso	full body
2	10	40	torso + upper limbs	full body
3	10	40	full body	full body



(a)



(b)

Figure 6.13 SPPSO with two partitions (base configuration) compared to SPPSO with three partitions at 20fps. Both configurations require 1000 evaluations per frame.

12 partitions

Figure 6.14 shows a comparison of the SPPSO base configuration to an algorithm with 12 hard partitions as used by John et al. [JTI10] at 60fps. Each of the 12 partitions uses 10 particles and 60 iterations, resulting in 7200 fitness evaluations per frame. Even with such a high number of evaluations, the hard partitioning performs worse than SPPSO with only 1000 evaluations. The pose estimate tends to drift away from the true pose with hard partitioning and the effect is even more pronounced at lower frame rates. This can also be seen in the graphs published by John et al. [JTI10]. A hard partitioning may work with a more detailed body model and more precise silhouettes [BEB08] but with a simple model and imperfect silhouettes the soft partitioning clearly performs better.

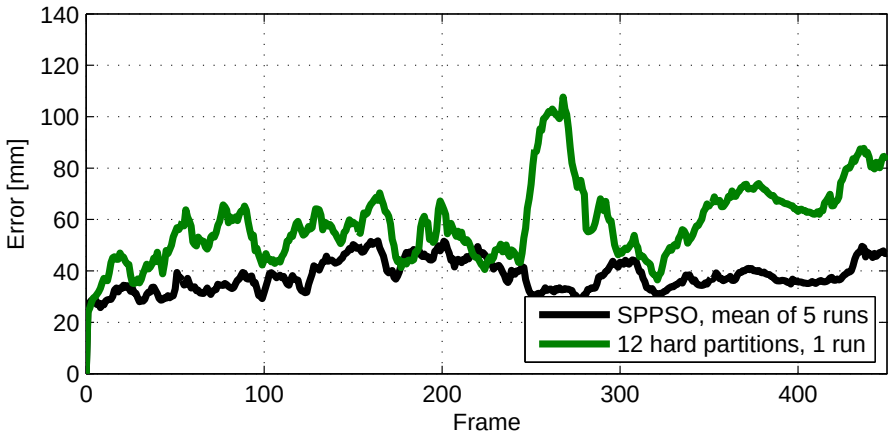
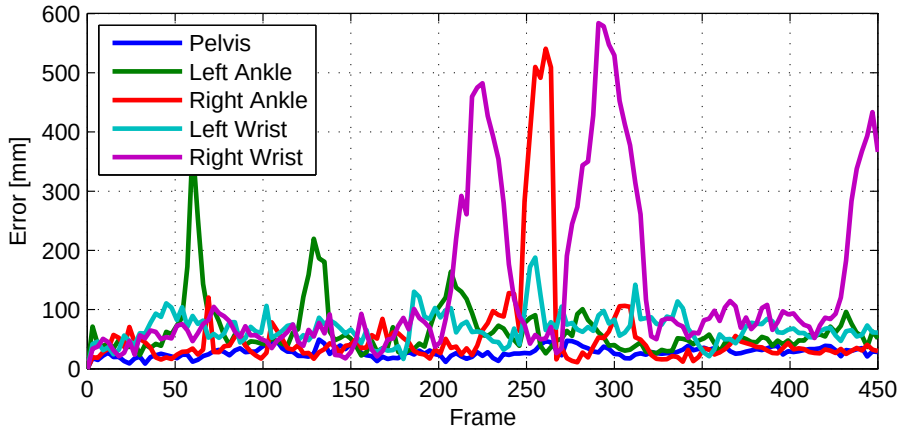


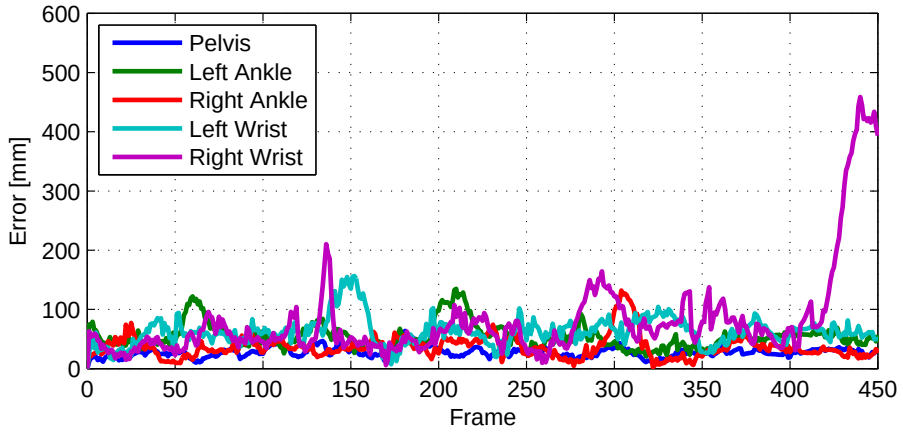
Figure 6.14 SPPSO base configuration compared to 12 hard partitions at 60fps, the 12 partitions are the same as used by John et al. [JTI10] and require 7200 fitness evaluations.

6.7 Individual Marker Errors

This experiment shows how some of the 15 individual markers contribute to the mean error at different frame rates. As can be seen in Figure 6.15, some lower limbs are repeatedly lost and reacquired at 20fps whereas the pelvis is always tracked accurately. At 60fps no limbs are completely lost during tracking, except for the right lower arm from frame 420 on. The problem here seems to be that the rotation parameter of the right arm x_{25} can not be estimated correctly during the standstill period because the arm is stretched out. When the subject starts to move again, the arm is bended but the tracker can not follow the bending correctly because the rotation parameter is too far away from the true value. This is a general problem because the rotational parameters are not observable while the limbs are stretched out. A very accurate body model and accurate silhouettes would be required to overcome this problem.



(a) 20fps



(b) 60fps

Figure 6.15 Individual marker errors during a single run of SPPSO at different frame rates. At 20fps some lower limbs are repeatedly lost and reacquired.

6.8 Number of Particles vs. Iterations

PSO is generally robust against changing the swarm size [BK07]. To show this, SPPSO was run with different swarm sizes while keeping the total number of evaluations at 1000. The first stage was always run with 10 particles and 20 iterations while the second stage was run with 2 - 400 particles. As expected, the algorithm performs well in the medium range of 10 - 40 particles (See Figure 6.16). With very low or high swarm sizes, PSO looses its swarm behaviour and can not track the subject accurately. Table 6.4 shows how many runs were performed for the different settings to produce figure 6.16.

Table 6.4 SPPSO with different numbers of particles for the second stage, number of performed runs.

Particles	Iterations	runs
2	400	1
10	80	3
20	40	5
40	20	3
400	2	1

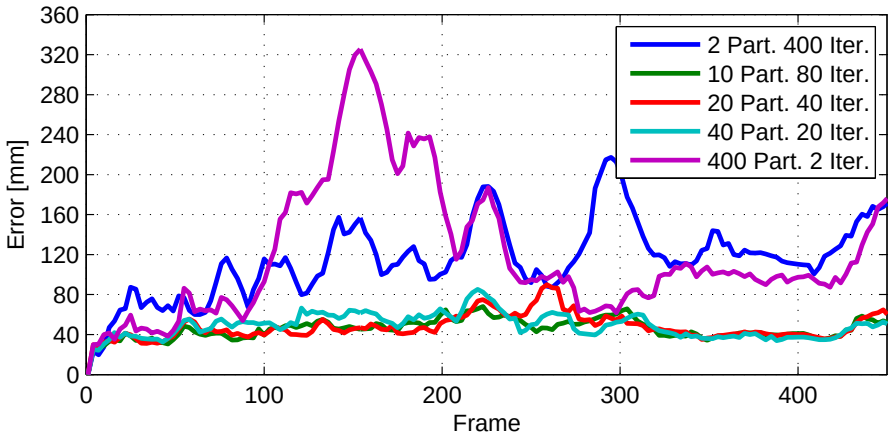
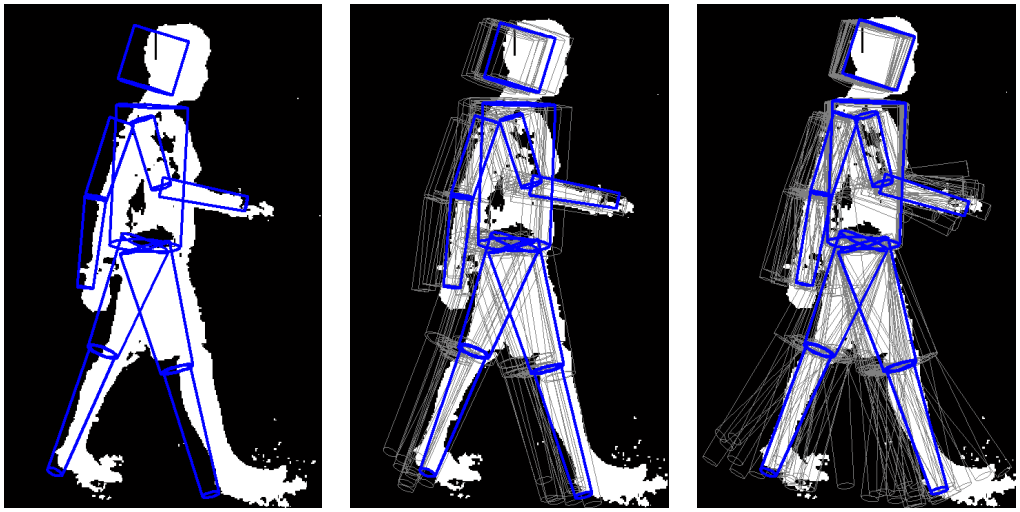


Figure 6.16 Mean error of SPPSO with different swarm sizes for the second stage (The total number of evaluations per frame is always 1000). The algorithm is robust against changing the swarm size. Table 6.4 shows how many runs were performed for the different settings.

6.9 Swarm Convergence

SPPSO Stages

Figure 6.17 illustrates visually how the SPPSO stages work. The first stage optimizes the global orientation and location parameters while the joint angles are kept constant. The second stage mainly optimizes the joint angles.



(a) (b) (c)

Figure 6.17 Illustration of the two SPPSO stages. The estimated poses are depicted in blue and the initial particle distribution in grey. (a) previous pose estimate, (b) after stage 1, (c) final pose estimate after stage 2.

Parameter convergence

Figure 6.18 shows how the variances of the individual parameters develop throughout the two optimization stages of SPPSO. Iterations 0-19 are in stage one and the further iterations in stage two. The graph depicts the standard deviations of the parameters over the swarm, normalized to one in the first frame they are optimized. The factors c_1 and c_2 of the PSO are increased linearly from 2.05 to 2.15 during the optimization in both stages to ensure the convergence that can be seen in Figure 6.18.

Individual parameters

The standard deviation of the position parameters $x_4 - x_6$ is reduced to about one tenth by the PSO during the first stage. In the second stage these parameters are resampled from a distribution downscaled by one tenth to enforce this convergence. The global orientation parameters $x_1 - x_3$ are resampled from the same distribution for both stages because initial experiment showed that they did not converge to a good estimate in the first stage. The joint parameters $x_7 - x_{31}$ (relative angles) are kept constant during the first stage.

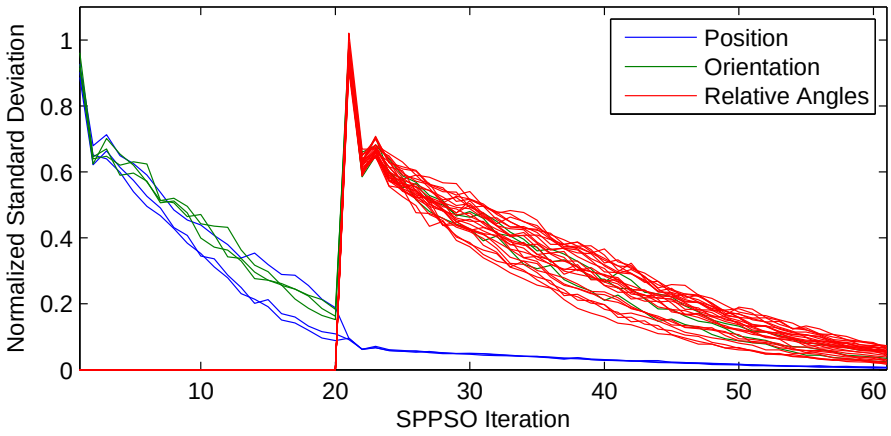


Figure 6.18 Normalized standard deviation of individual parameters averaged over 50 SPPSO optimizations. The standard deviation is estimated over all particles at every SPPSO iteration. The 50 SPPSO optimizations are successive pose estimations on the Lee walk sequence at 20fps.

6.10 Different Fitness Functions

One parameter varied

To show the different properties of the silhouette and edge fitness, the following experiment was conducted: Starting from the ground truth pose, a single parameter was varied with all other parameters kept constant and the silhouette and edge fitness (f_s and f_e) was evaluated at every parameter value. The offset range of the parameter equals the standard deviation of the sampling distribution at 20fps. The total fitness was computed as $f = f_s + f_e$.

Fitness properties

The three fitness graphs are depicted in Figure 6.19, each graph was normalized separately to allow an easier comparison. The silhouette fitness f_s is much smoother than the edge fitness f_e which has significant local maxima in the depicted range. On the other hand, the edge fitness allows a more accurate localization of the global maximum. Note that the maxima of f_s and f_e coincide only roughly. This can be explained by the coarse body model and the noisy observations.

Only silhouette

Figure 6.21 shows a comparison of SPPSO with base configuration to SPPSO where only the silhouette fitness is used. The tracking becomes more robust robust over all, as can be seen by the lower maximum error. On the other hand, the accuracy is worse during the standstill period where the subject is standing with the arms hanging down. This can be explained by the ability of the edge fitness to locate the torso and the arms better when the arms are close to the torso or in front of it.

No upper edge

Figure 6.22 shows that the tracking gets much worse when the upper edge of the torso is omitted in the edge fitness. When the limbs are roughly aligned with the torso, the edge fitness without the upper edge becomes invariant to changes of the vertical position. This is because the model limbs do not cover the full length of

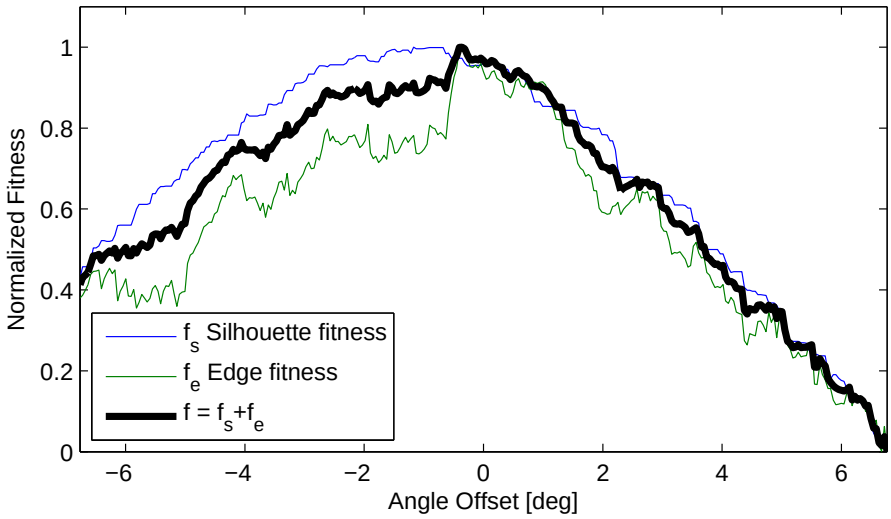
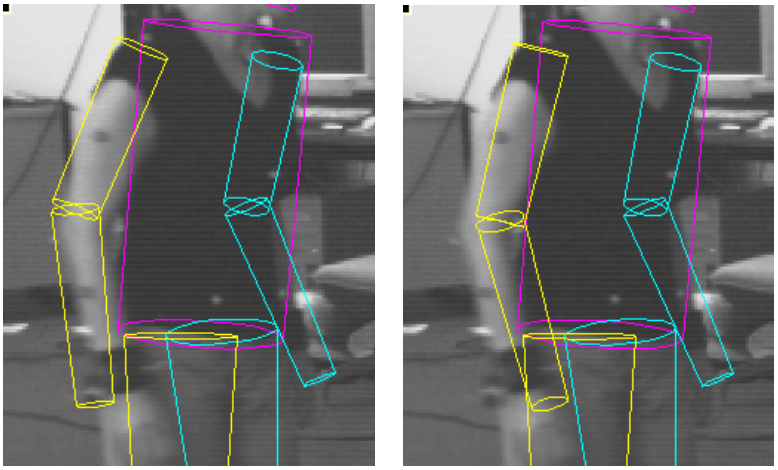


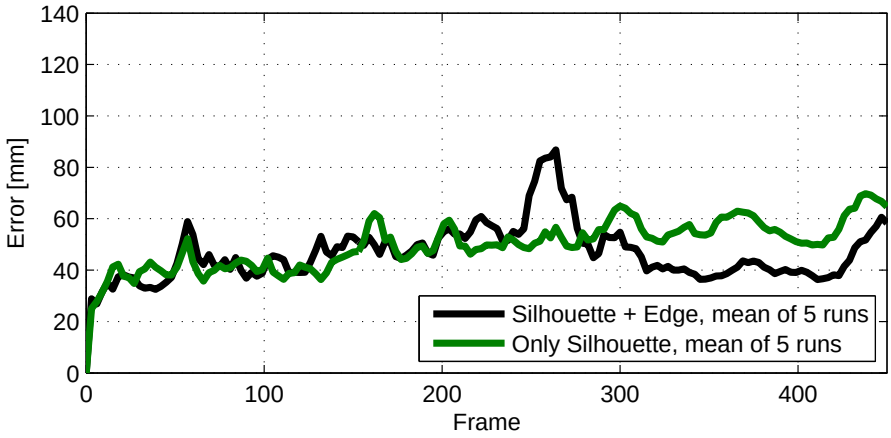
Figure 6.19 Normalized Fitness functions evaluated at different values of the parameter x_{24}. All other parameters are kept constant. The varied parameter controls the forward-backward angle of the right shoulder joint. Figure 6.20 depicts the body model at the two extreme positions projected into view 1. The maximum offset of the parameter equals the standard deviation of the sampling distribution at 20fps.



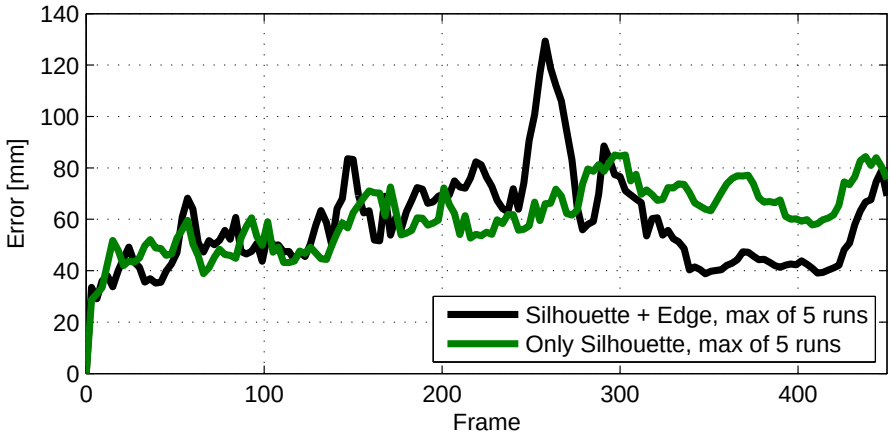
(a) (b)

Figure 6.20 Body model with parameter x_{24} varied. The parameter controls one angle of the right shoulder joint. (a) Offset -6.8°, (b) offset +6.8°.

the subject's limbs and can therefore *slide* up and down without affecting the edge fitness. This experiment shows that even small changes in the fitness functions can have a big effect on the tracking accuracy with the used coarse body model.



(a)



(b)

Figure 6.21 Comparison of SPPSO at 20fps with base configuration to SPPSO where only the silhouette fitness is used. The tracking becomes more robust (lower maximum error) but the accuracy is worse during the standstill period.

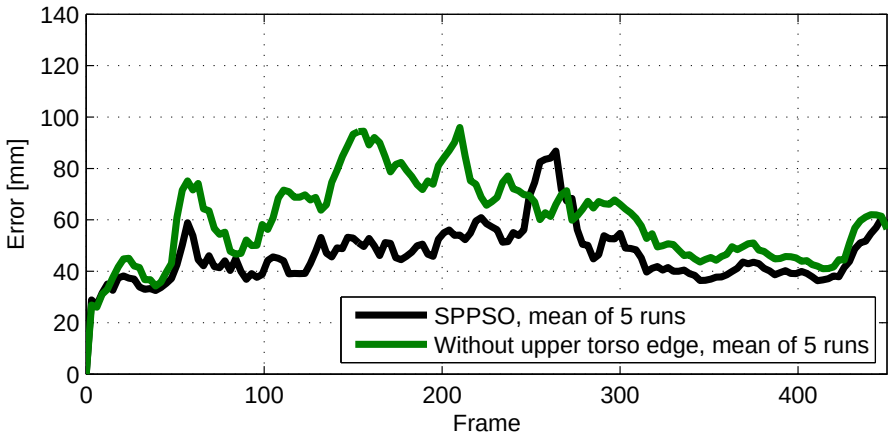


Figure 6.22 SPPSO with and without using the upper edge of the torso at 20fps.

6.11 Computation Time

Major problem

The task of this project was not to develop a *real-time* implementation for SPPSO, and the algorithm is almost entirely implemented in Matlab. However, computation time is a major problem for pose tracking algorithms. It usually takes seconds to minutes to estimate the pose in one frame for Matlab implementations [BSB05, SBB10, JTI10]. This means that tracking an entire sequence may take hours. It is therefore very time consuming to evaluate different algorithm configurations.

Optimizations

To enable a faster evaluation of different algorithm configurations, the Matlab implementation was optimized in the following ways: (See Appendix A for more details.)

- The polygon filling function of Matlab, used to render the cylinders, was replaced by a MEX-file.
- ROI processing was added to the silhouette fitness function to avoid summing over the whole images.
- A cache was added to the cylinder rendering function to avoid re-rendering cylinders that have not moved.

Code parts

With these optimizations, SPPSO spends about half of the total computation time rendering the cylinders, i.e. projecting the 3D model to the camera views. Fitness computations account for 39% of the time. And finally, the kinematic tree requires 11% of the time for computing the 3D locations of the cylinders from the angle parameters. Table 6.5 lists the percentage of time spent in the code parts that dominate the time consumption.

Table 6.5 Time consumption of individual parts of the Matlab implementation of SPPSO. Results from a run with 1000 evaluations per frame.

Code part	**Time/frame [s]**	**%**
Rendering	10.0	48
Edge fitness	4.6	22
Silhouette fitness	3.5	17
Kinematic tree	2.4	11
Rest	0.5	2
Total	**21**	**100**

7 Conclusion

7.1 SPPSO

Soft partitioning

One of the main insights gained during this project is that the refinement stages in a hierarchical pose tracking algorithm should be global in some way to avoid error accumulation. The soft partitioning scheme of SPPSO is able to correct small errors from the first hierarchical stage in the second stage and thus does not suffer from error accumulation. A hard hierarchical partitioning only works with a very precise model and noise free observations [BEB08].

Comparison to APF

SPPSO performs better than APF at a frame rate of 20fps with the same number of fitness evaluations. At 60fps, the performance is equal. The better performance at slow frame rates probably comes from the explicit exploitation of the hierarchical model structure in SPPSO. APF relies on an *automatic* soft partitioning where the algorithm must detect the more important parameters by itself, whereas SPPSO has predefined partitions based on the model structure.

7.2 PSO for Tracking Articulated Body Models

Young method

PSO is a relatively new optimization method for pose tracking (The first source known to the author is [IT06]). And there exist only few, more or less successful, attempts to video-based *full body* tracking [JTI10, ZHW+10, KKW11a]. It seems that the methods of John et al. and Krzeszowski et al., which use a hard partitioning, require more fitness evaluations to minimize the problem of error accumulation.

Soft partitioning

All of the discussed methods use different hierarchical approaches to battle the curse of dimensionality and none of them seems to be clearly superior. With the soft partitioning scheme, SPPSO proposes yet another approach and it has been shown to perform well. A similar approach has been shown to work by Robertson and Trucco [RT06], but they used only an upper body model and 3D data.

7.3 Future Work

PSO algorithm

Future work in PSO based pose tracking may explore the use of newer variants of the PSO algorithm such as the sub-swarms method [VdBE04]. Another promising area is the introduction of operators from genetic algorithms such as the crossover

operator, which has been shown to improve tracking for the annealed particle filter [DR05].

Detailed model The simple ten-cylinder model of Balan et al. [BSB05] was used for SPPSO because it has relatively few parameters and is freely available. However, it does not model the human body sufficiently accurate. This could be shown by experiment 6.10 where the upper edge of the torso was not used for the fitness function. This minor change of the fitness function resulted in a significantly worse tracking performance. A more detailed model, which models the torso and head more accurately and also includes the hands and feet, should be less sensitive to changes of the fitness function. But a more elaborate model requires a faster rendering method.

Fast rendering SPPSO is based on the HumanEva framework an therefore almost completely implemented in Matlab, including the rendering of the 3D model. Consequently, the algorithm needs 20 seconds to process one frame and is therefore far from being real-time, which would be necessary for many applications. But more importantly, the long processing time means that testing new algorithm settings is a very lengthy task. It is not sufficient to run the tracking for only a few frames because drift problems may only show after many frames. Moreover, several runs are required to really evaluate a new setting because SPPSO is a stochastic algorithm. Consequently, the most processor intensive tasks should be outsourced for future developments to speed up testing cycles. The most processor-intensive tasks in SPPSO are model rendering and fitness evaluation. They could be performed very fast by graphics processing hardware.

New benchmark APF is outdated as a benchmark algorithm for pose tracking. Interacting simulated annealing (ISA) has a better tracking performance and should therefore be the benchmark for new developments. Moreover, it has been shown, that a two stage pose estimation with a global optimization stage and a local refinement stage achieves a significantly better accuracy than only a global optimization [GRBS10].

Local refinement The important local refinement stage could for example employ a gradient based method such as stochastic meta descent (SMD) [BKMM+04] or an iterative closest point algorithm (ICP) [Zha94].

Global optimization The global optimization step in a two-stage pose estimator is necessary to enable recovering from wrong estimates. This can not be achieved by approaches that rely on basic correspondences such as closest points on silhouette contours or optical flow. An interesting option is to incorporate a body part detector in the first pose estimation stage [BKSS10] because part detector based approaches are inherently global optimizations. They could also be used for initializing a model based algorithm [SB10].

7.4 Optical Flow for Tracking Articulated Body Models

Original algorithm idea The original idea for the algorithm was to segment the observed images based on

the estimated pose at time $t-1$ an then propagate the segmentation using dense optical flow between the frames $t-1$ and t. This segmented image would then be used in a PSO based pose estimation at time t. It turned out that this approach introduces new problems such as error accumulation, which is a problem for all OF based approaches. Furthermore, there are more efficient ways of exploiting the information in OF.

Correspondences

When OF is used for tracking, the best way of using the information in OF seems to be the concept of correspondences (See section 4.3). The most important advantage of correspondences is that the model parameters can be estimated much more efficiently than with conventional fitness functions such as silhouette based ones. However, care must be taken to find valid correspondences, i.e. reliable OF, and a correspondence-based approach must include a drift correction mechanism [GRS08].

8 Bibliography

[AMGC02] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002. 7, 17, 21

[ARS09] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1014–1021. IEEE, 2009. 13

[ARS10] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:623–630, 2010. 16

[ASK+05] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. *ACM Trans. Graph.*, 24:408–416, 2005. 14, 15, 76

[AT06] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *IEEE transactions on pattern analysis and machine intelligence*, 28:44–58, 2006. 13

[BBPW04] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004*, volume 3024, pages 25–36. Springer, 2004. 28

[BC08] L. Ballan and G.M. Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *Proceedings of the Fourth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2008. 19, 20, 29, 30

[BEB08] J. Bandouch, F. Engstler, and M. Beetz. Evaluation of hierarchical sampling strategies in 3d human pose estimation. In *Proceedings of the 19th British Machine Vision Conference (BMVC)*, 2008. 14, 18, 21, 22, 25, 28, 56, 65

[BK07] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 120–127. IEEE, 2007. 24, 39, 40, 58, 80

[BKMM+04] M. Bray, E. Koller-Meier, P. Müller, L. Van Gool, and N.N. Schraudolph. 3d hand tracking by rapid stochastic gradient descent using a skinning model. In *In 1st European Conference on Visual Media Production (CVMP*, pages 59–68, 2004. 19, 20, 66

[BKSS10] M. Bergtholdt, J. Kappes, S. Schmidt, and C. Schnörr. A study of parts-based object class detection using complete graphs. *International journal of computer vision*, 87(1):93–117, 2010. 13, 66

[BMP04] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *International Journal of Computer Vision*, 56(3):179–194, 2004. 30

[BSB05] A.O. Balan, L. Sigal, and M.J. Black. A quantitative evaluation of video-based 3d person tracking. In *Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 349–356. Citeseer, 2005. 8, 11, 14, 15, 16, 19, 24, 25, 26, 27, 28, 31, 32, 34, 35, 41, 45, 51, 64, 66, 76, 81

[BSB+07] A.O. Balan, L. Sigal, M.J. Black, J.E. Davis, and H.W. Haussecker. Detailed human shape and pose from images. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 14

[BSF10] M.A. Brubaker, L. Sigal, and D.J. Fleet. Video-based people tracking. In *Handbook of Ambient Intelligence and Smart Environments*, pages 57–87. Springer, 2010. 9, 10

[CK02] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, 2002. 40

[CMC+06] S. Corazza, L. Mündermann, AM Chaudhari, T. Demattio, C. Cobelli, and TP Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering*, 34(6):1019–1029, 2006. 16

[DBR00] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *cvpr*, page 2126. Published by the IEEE Computer Society, 2000. 14, 18, 22, 35

[DDR01] J. Deutscher, A. Davison, and I. Reid. Automatic partitioning of high dimensional search spaces associated with articulated body motion capture. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–669. IEEE, 2001. 21

[DF01] Q. Delamarre and O. Faugeras. 3d articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, 81(3):328–357, 2001. 19

[DGC09] B. Daubney, D. Gibson, and N. Campbell. Monocular 3d human pose estimation using sparse motion features. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1050–1057. IEEE, 2009. 29

[DR05] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205, 2005. 7, 14, 15, 18, 20, 21, 22, 24, 25, 28, 35, 66

[EK95] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995. 40

[ES01] R.C. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation*, volume 1, pages 81–86. Piscataway, NJ, USA: IEEE, 2001. 39

[FH05] P.F. Felzenszwalb and D.P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005. 13

[Fle11] David J. Fleet. Motion models for people tracking. In Thomas B. Moeslund, Adrian Hilton, Volker Krüger, and Leonid Sigal, editors, *Visual Analysis of Humans*, pages 171–198. Springer London, 2011. 16

[FMJZ08] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 13

[GEJ+08] S. Gammeter, A. Ess, T. Jäggli, K. Schindler, B. Leibe, and LJV Gool. Articulated multi-body tracking under egomotion. In *European Conference on Computer Vision*, volume 66, pages 657–662, 2008. 13

[GLS11] T. Greif, R. Lienhart, and D. Sengupta. Monocular 3d human pose estimation by classification. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011. 13

[GPS+07] J. Gall, J. Potthoff, C. Schnörr, B. Rosenhahn, and H.P. Seidel. Interacting and annealing particle filters: Mathematics and a recipe for applications. *Journal of Mathematical Imaging and Vision*, 28(1):1–18, 2007. 7, 17, 22

[GPZ+11] M. Germann, T. Popa, R. Ziegler, R. Keiser, and M. Gross. Space-time body pose estimation in uncontrolled environments. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*, pages 244–251. IEEE, 2011. 13

[GRBS10] J. Gall, B. Rosenhahn, T. Brox, and H.P. Seidel. Optimization and filtering for human motion capture. *International journal of computer vision*, 87(1):75–92, 2010. 20, 22, 66

[GRS08] J. Gall, B. Rosenhahn, and H.P. Seidel. Drift-free tracking of rigid and articulated objects. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 15, 19, 30, 67

[GSDA+09] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1746–1753. IEEE, 2009. 14

[HN99] M. Haag and H.H. Nagel. Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences. *International Journal of Computer Vision*, 35(3):295–319, 1999. 29

[HS81] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 28

[HTWM04] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, 2004. 29

[IB98] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998. 21

[IJT10] S. Ivekovic, V. John, and E. Trucco. Markerless multi-view articulated pose estimation using adaptive hierarchical particle swarm optimisation. In *Applications of Evolutionary Computation*, pages 241–250. Springer, 2010. 23

[IT06] S. Ivekovic and E. Trucco. Human body pose estimation with pso. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1256–1263. IEEE, 2006. 23, 40, 65

[JTI10] V. John, E. Trucco, and S. Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image and Vision Computing*, 28(11):1530–1547, 2010. 14, 23, 25, 28, 56, 57, 64, 65, 77

[Jua04] C.F. Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(2):997–1006, 2004. 24

[KBVG05] R. Kehl, M. Bray, and L. Van Gool. Full body tracking from multiple views using stochastic sampling. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 129–136. IEEE, 2005. 14, 15, 19

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995. 7, 39, 40

[KG06] R. Kehl and L.V. Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 104(2-3):190–209, 2006. 15, 16, 76

[KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671, 1983. 22

[KKW11a] T. Krzeszowski, B. Kwolek, and K. Wojciechowski. Model-based 3d human motion capture using global-local particle swarm optimizations. In *Computer Recognition Systems 4*, pages 297–306. Springer, 2011. 7, 14, 22, 23, 25, 28, 65

[KKW11b] B. Kwolek, T. Krzeszowski, and K. Wojciechowski. Swarm intelligence based searching schemes for articulated 3d body motion tracking. In *Advances Concepts for Intelligent Vision Systems*, pages 115–126. Springer, 2011. 7, 18, 23, 40

[LF05] V. Lepetit and P. Fua. *Monocular model-based 3D tracking of rigid objects*. Now Publishers Inc, 2005. 29

[LH05] X. Lan and D.P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 470–477. IEEE, 2005. 16

[LK81] BD Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Image Understanding Workshop*, pages 121–130. Carnegie-Mellon University, 1981. 7, 28

[LM07] N.D. Lawrence and A.J. Moore. Hierarchical gaussian process latent variable models. In *Proceedings of the 24th international conference on Machine learning*, pages 481–488. ACM, 2007. 16

[Low04] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 28

[LRK01] M. Lovbjerg, T.K. Rasmussen, and T. Krink. Hybrid particle swarm optimiser with breeding and subpopulations. In *Proceedings of the third Genetic and Evolutionary computation conference*, volume 1, pages 469–476. Citeseer, 2001. 24

[Mar63] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. 20, 24

[MCA07] L. Mundermann, S. Corazza, and T.P. Andriacchi. Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007. 16

[MG01] T.B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001. 9, 10, 11, 76

[MH03] J. Mitchelson and A. Hilton. Simultaneous pose estimation of multiple people using multiple-view cues with hierarchical sampling. Technical report, Centre for Vision, Speech, and Signal Processing, University of Surrey, Guildford, UK, 2003. 15

[MHK06] T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2-3):90–126, 2006. 9, 10, 13, 16

[MI00] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Computer Vision - ECCV 2000*, pages 3–19. Springer, 2000. 21

[OK94] J. Ohya and F. Kishino. Human posture estimation from multiple images using genetic algorithm. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 750–753. IEEE, 1994. 20

[PH91] A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(7):730–742, 1991. 29

[Pop07] R. Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1-2):4–18, 2007. 9

[PTA07] M. Pant, R. Thangaraj, and A. Abraham. A new pso algorithm with crossover operator for global optimization problems. In *Innovations in Hybrid Intelligent Systems*, pages 215–222. Springer, 2007. 24

[RT06] C. Robertson and E. Trucco. Human body posture via hierarchical evolutionary optimization. *BMVC06*, 3:999, 2006. 23, 65

[SB03] H. Sidenbladh and M.J. Black. Learning the statistics of people in images and video. *International Journal of Computer Vision*, 54(1):183–209, 2003. 19

[SB06] L. Sigal and M.J. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Technical Report CS-06-08, Brown Univertsity, 2006. 27

[SB10] L. Sigal and M.J. Black. Guest editorial: state of the art in image-and video-based human pose and motion estimation. *International Journal of Computer Vision*, 87(1):1–3, 2010. 9, 25, 66

[SBB10] L. Sigal, A.O. Balan, and M.J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1):4–27, 2010. 8, 14, 16, 18, 22, 24, 25, 27, 28, 31, 35, 36, 45, 51, 64

[SBF00] H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Computer Vision - ECCV 2000*, pages 702–718. Springer, 2000. 15, 16

[SE98] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73. IEEE, 1998. 40

[SGS+09] M. Shaheen, J. Gall, R. Strzodka, L. Van Gool, and H.P. Seidel. A comparison of 3d model-based tracking approaches for human motion capture in uncontrolled environments. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–8. IEEE, 2009. 22

[SP94] M. Srinivas and L.M. Patnaik. Genetic algorithms: a survey. *Computer*, 27(6):17–26, 1994. 7, 20

[ST01] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3d body tracking. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–447. IEEE, 2001. 7, 20, 21

[ST02] C. Sminchisescu and A. Telea. Human pose estimation from silhouettes. a consistent approach using distance level sets. In *10th International Conference on Computer Graphics,Visualization and Computer Vision (WSCG '02)*, volume 10 of *1-2*, 2002. 18, 19, 36

[TK91] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, School of Computer Science, Carnegie Mellon University, 1991. 7, 28

[TMS01] H. Tsutsui, J. Miura, and Y. Shirai. Optical flow-based person tracking by multiple cameras. In *Multisensor Fusion and Integration for Intelligent Systems, 2001. MFI 2001. International Conference on*, pages 91–96. IEEE, 2001. 29

[VdBE04] F. Van den Bergh and A.P. Engelbrecht. A cooperative approach to particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, 8(3):225–239, 2004. 24, 65

[WA96] Y. Weiss and E.H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 321–326. IEEE, 1996. 29

[WN97] S. Wachter and H.H. Nagel. Tracking of persons in monocular image sequences. In *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, pages 2–9. IEEE, 1997. 15

[YSK+98] M. Yamamoto, A. Sato, S. Kawada, T. Kondo, and Y. Osaki. Incremental tracking of human actions from multiple views. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 2–7. IEEE, 1998. 29

[Zha94] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13:119–152, 1994. 22, 66

[ZHW+10] X. Zhang, W. Hu, X. Wang, Y. Kong, N. Xie, H. Wang, H. Ling, and S. Maybank. A swarm intelligence based searching strategy for articulated 3d human body tracking. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 45–50. IEEE, 2010. 14, 24, 25, 28, 40, 65

[ZL08] X. Zhao and Y. Liu. Generative tracking of 3d human motion by hierarchical annealed genetic algorithm. *Pattern Recognition*, 41(8):2470–2483, 2008. 20

[ZS11] Z. Zhang and H.S. Seah. Real-time tracking of unconstrained full-body motion using niching swarm filtering combined with local optimization. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 23–28. IEEE, 2011. 7, 24

9 List of Figures

10 List of Tables

11 List of Algorithms

A Matlab Implementation

Source

The SPPSO implementation is based on the Matlab implementation of the annealed particle filter by Balan et al. [BSB05]. The code and the Lee walk dataset can be downloaded from `http://www.cs.brown.edu/~alb/download.htm`. Both are also contained in the zip archive `Lee_Tracking_original.zip` in the directory `matlab` on the accompanying DVD. The used Matlab version is R2011a.

Setup

To run SPPSO, first copy the whole folder `matlab` from the DVD to a location on your hard drive. The Lee walk dataset has already been unpacked and is in the folder `matlab\WebData`, it has to remain in this exact location. The m-files for SPPSO are in the folder `matlab\SPPSO`. To start tracking, simply run `my_TrackPSO_tb.m`. The algorithm parameters can all be adjusted in this file. When the variable `DEBUG` is set to 0, only console output will be produced to show the progress of tracking. When it is set to 3, there will be multiple figures that illustrate the fitness evaluations and the tracking results of the SPPSO stages.

Tracking results

The tracking results will be saved in a .mat file in the folder `matlab\SPPSO \outTrackPSO`. A subfolder will be created with the name specified by the variable `experimentName` in `my_TrackPSO_tb.m`. A video and images of the tracking result will also automatically be saved in this subfolder. The error graphs shown in this thesis were produced by the m-file `my_ErrorGraphDoc.m`. The tracking results of the shown experiments can be found in the folder `matlab\SPPSO \outTrackPSO`.

OpenCV

The original implementation of the model rendering used the Matlab function `poly2mask()` to fill the polygons that are produced when the cylinders are projected into 2D. To speed up the rendering, the MEX-file `overlayCylinders.mexw32` was compiled, which uses the function `cvFillConvexPoly()` from OpenCV. This is the only part of the implementation that not only uses Matlab. In case the MEX-file has to be recompiled, the visual studio project is in the folder `visualStudio` on the DVD. For instructions on how to compile MEX-files that call OpenCV, refer to the file `visualStudio\OpenCV_And_MEX_Files_quick_guide.pdf`. OpenCV can be downloaded from `http://opencv.willowgarage.com/wiki/`.

A.1 Program Flow and Variables

Data structures

This section gives some hints for an easier understanding the Matlab code. Most of the important variables are the same as in the original implementation.

A.1.1 Variables in `my_Track_PSO.m`

The structure `PRM` contains the global algorithm configuration, such as: number of frames to process, subsampling factor, and location of the dataset. The number of particles and iterations for each stage are defined in the cell array `PRM.OPTIMIZING_SEQ`.

The structure `MODEL_PRM` contains the lengths and diameters of the cylinders in the body model as well as the ground truth parameters in `MODEL_PRM.angles`.

The 31 variable model parameters are kept in a row vector, the indexing can be seen in Table 5.1. The estimated parameters from `my_performPSOAdvanced()` are saved in the matrix `gbestAngles`.

A.1.2 Variables in `my_performPSOAdvanced.m`

The four binary image silhouettes are in the cell array `bgs`, the edge maps in `edgeMaps`, and the original frames in `img`.

The cell array `cyls`, produced by `my_ComputeCylArray()`, contains a struct for every cylinder of the body model. This struct contains the dimensions of the cylinder and the transformation matrix that defines its location in 3D. This matrix is 4x4 because all 3D computations are done in homogeneous coordinates.

The cell array `cylpts`, produced by `my_computeCylptsCell(cyls)`, contains the four corner points of the projections of the cylinders for all four camera views.

A.1.3 Program Flow

The main loop of the program is in `my_Track_PSO.m`. After initialising the model and tracking parameters, the function `my_performPSOAdvanced.m` is called once for every frame to perform the two step optimization. in `my_performPSOAdvanced.m` the subfunction `partialPSO()` is called for the individual optimization stages. This is the subfunction that contains the actual PSO implementation. The subfunction `computeFitness()` is called from `partialPSO()` to compute the fitness for all particles of the swarm in one call.

B Declaration Of Authorship

I, Patrick Fleischmann, declare that this thesis and the work presented in this thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signature and Date: