UNIVERSITAT POLITÈCNICA DE CATALUNYA
DEPARTAMENT DE LLENGUATGES I SISTEMES INFORMÀTICS
MÀSTER EN COMPUTACIÓ

# TESI DE MÀSTER

# MODELS AND ESTIMATORS FOR MARKERLESS HUMAN MOTION TRACKING

ESTUDIANT: MARCEL ALCOVERRO VIDAL
DIRECTORS: MONTSE PARDÀS FELIU I JOSEP RAMON CASAS PLA
PONENT: MARTA FAIRÉN GONZÁLEZ

DATA: 25 DE JUNY DEL 2009

# Contents

# Chapter 1

# Introduction

The capture and analysis of the motion of a human body is applied in a variety of fields such as bio-mechanical analysis, ergonomics or character animation in films or video games. Currently there are commercial motion capture products that use optical, magnetic or mechanical devices, which are costly and require users to carry markers in the whole body. Motion capture systems without markers are an attractive and non-invasive alternative, which allow an analysis not restricted to the information of the movement of the markers and avoid any inconvenience to users of dressing themselves with special clothes and devices.

In markerless motion capture, the subject should be observed from one or more cameras. The acquired images are processed in order to estimate the position and the pose of the subject at any time. In our case, we develop a system based on an analysis-by-synthesis approach through the use of a 3D human body model. On this basis, the human model is synthesized in an estimated pose, and the similarity between the observed data and the estimated data is measured through a cost function. The estimated poses are improved by an optimization of such function.

In a model based motion tracking system, an important consideration is the high dimensionality of the space of the human poses (i.e, the available combinations of model parameters), together with the fact that the cost functions that we can define have many local minima. These two features condition which strategy we choose to optimize the cost function, and represent the major challenge for any motion tracking system.

## 1.1 Objectives

In this work, we analyze the different components of a model-based motion tracking system. The system consists in: (i) a human body model, (ii) an estimator, and (iii) a likelihood or cost function. We will describe in more detail these components in chapter 2.

A first consideration is the human body representation to be used by the system. The rendering of the model in a specific pose is the way to measure the correspondence between an estimated pose and the captured image data. Thus, the kind of human representation chosen is likely to affect the accuracy of the estimation. We compare the use of two kind of articulated models:

- a model based on rigid cylindrical parts.
- an anthropometric model based on a deformable polygonal mesh.

We describe these models in section 2.1.

The estimator task is to determine the optimal pose at each frame, given the recovered poses at previous frames and the observation at the current frame. The high dimensionality of the pose space and the multimodality of the cost function determine the choice of the estimator. We study two different approaches for the estimation process:

- a single-hypothesis deterministic optimization method, Powell's method.
- a multiple-hypothesis stochastic method, Annealed Particle Filter.

We focus on the precision of the recovered poses, the robustness to track fast movements and the computation cost of both methods, in order to determine the advantages and drawbacks of both strategies. Both estimation methods are explained in section 2.2.

The cost function used to match the representation of the subject to the model is another important aspect. Several image descriptors can be used to build such functions, and we review some of them in section 1.2. Our choice for the cost function is based on 2D silhouettes, which we implement in an efficient manner using graphics hardware. The cost function is evaluated multiple times each frame in order to determine the optimal pose, and an efficient implementation is crucial for the overall performance of the system. We explain the cost function implementation in section 2.3.

The purpose of the study is to combine the distinct models and estimation techniques to draw conclusions about its influence on the results. We use a simple but efficient cost function, that can be equally used with both human models, the cylindrical-based and the deformable mesh.

## 1.2 Related Work

Markerless vision-based human motion analysis has received much interest over the last two decades, and it continues to be an active research domain. In this section, we review the recent literature with a focus on the approaches more related to our work.

Several surveys have been written within the domain of human motion analysis [MG01], [Gav99], and they present different taxonomies. We will follow the taxonomy used by Poppe [Pop07], where the methods are initially subdivided between model-based and model-free approaches, focusing the review in model-based techniques, as they are more related to our work.

In model-free approaches, an explicit human body model is not available. In this cases, the pose estimation is performed either using a database of examples, or by machine learning techniques using training data.

Model-based approaches employ an *a priori* human body, and the pose estimation process consists in a *modelling* phase and an *estimation* phase.

The modelling phase consists in the construction of the likelihood function using image descriptors, the human body model and a matching function. The estimation phase consists in finding the most likely pose using the likelihood function. We will describe the recent work in model-based markerless human motion analysis considering how it performs the modelling and the estimation phases.

### 1.2.1 Modelling Phase

The parameters involved in the construction of the likelihood function are, first, those related to the body model used: the body configuration parameters, body shape and appearance parameters. The likelihood is also related to the kind of observations that we have, the camera viewpoint and the image descriptors that we use. Model-based approaches use a body model

which includes the kinematic structure and the body dimensions. Also, these methods use a function that describes how the human body appears in the image domain, given the model's parameters.

### Human Body Models

The human body models describe the kinematic properties and the shape and appearance.

Most of the models describe the kinematics of the skeleton as a tree, consisting of segments that are linked by joints. Every joint contains a number of degrees of freedom (DOF) depending on the rotations involved with the joint. The number of DOF of the skeleton varies between the methods from 25 DOF until more than 50 when considering hand and feet movements. The range of the rotation angles usually is restricted by kinematic constraints, in order to limit the pose space to human feasible poses. Also are useful the constraints to avoid interpenetration of the body parts, but in this case we also need information of the shape of the body.

The human shape is modeled in different ways. On one side, we have models based on separate rigid shapes, as cylinders [DBR00], ellipsoids [MTHC03], and more generally as a combination of tapered super-quadrics [GD96]. These volumetric shapes depend on only a few parameters.

On the other side, surface-based models employ a single surface for the entire body. These models consist usually on a mesh of polygons than can be deformed by changes on the underlying skeleton. Several studies use a mesh obtained from a 3D scan (Cyberware technology [Cyb]) of the specific individual to track, as in [BC08], [CMC+06]. Vlasic et al. [VBMP08] use either meshes from a 3D scan, or meshes obtained from a 3D multiview stereo reconstruction algorithm. Anguelov et al. [ASK+05] propose the SCAPE method to model deformable surfaces, which is based on models of pose and body shape variation that are learned from a database of 3D scans. The SCAPE model is used for human motion analysis in [BSB+07]. Bandouch et al. [BEB08] use the RAMSIS model [Sol] which is widely used in the automotive industry. Its design has been guided by ergonomic considerations and it is precise in the inner joint locations. Plänkers and Fua [PF01] use a more complex model based on three layers: skeleton; ellipsoidal balls to simulate muscles and fat tissue; and a polygonal surface representation for the skin.

The parameters of the shape model (lengths, widths, etc.) may be assumed fixed. For example, in case of using scanned models there is no need of adjustment. However, when using a generic model, the parameters should be adjusted to the specific individual in order to avoid innaccurate pose estimations. Carranza et al. [CTMS03] propose an initialization phase to adjust these parameters, where the subject has to adopt an specified pose. Plänkers and Fua [PF01] perform the estimation of the parameters during motion over a sequence to determine more precisely the position of the articulations inside the skin surface.

The deformation process of the skin mesh according to the pose parameters is performed using techniques such as *Skeletal Subspace Deformation* (SSD) also called *linear blend skinning* [JRMG07]. SSD determines the new position of a vertex by linearly combining the results of the vertex transformed rigidly with each bone. A scalar weight is given to each bone, and the weighted sum gives the final vertex position. SSD has a number of shortcomings, such as the *"elbow collapsing"* effect. The technique is improved by using data-interpolation techniques using a number of example meshes, as proposed in [LCF00] and [SRIC01], which allow correcting the error in the vertex positions introduced by the SSD. Some models, as for example the SCAPE model [ASK+05], obtain the example meshes from a database of 3D scans of the same individual adopting distinct poses. Then the deformation model is based on a linear blend skinning corrected according to some parameters learned from the database. Vlasic et al. [VBMP08] improve the SSD results using an iterative framework that deform the shape to

approximate better to the silhouette contours of the real image data.

#### Image Descriptors and Matching Functions

We are interested in methods that recover the kinematic configuration of a person, using image data captured from several views. The appearance of people in images varies due to distinct clothing or lighting. To generalize over these varying conditions, motion tracking systems use image descriptors.

Several approaches use silhouettes of the active people in the scene as image descriptors to define the matching functions [DBR00], [BC08], [CTMS03]. Silhouettes are insensitive to variations in the surface such as color, and encode great information to recover 3D poses. The matching functions account for the difference between the projection of the model into a certain view and the silhouette in this view. However, sometimes it is not possible to recover certain DOF, when using only silhouettes, due to the lack of depth information.

Edges are image descriptors also used to construct the matching functions [DBR00]. Edges appear in the image when there is sharp changes in intensity. Within a silhouette usually provide a good outline of visible arms and legs, and are mostly invariant to color and lighting. However, when the subject is wearing baggy clothes edges may lose usefulness. Matching functions take into account the normalized distance between model's synthesized edges and the closest edge found in the image.

Edges and silhouettes lack depth information which makes hard to detect self-occlusions. A 3D reconstruction can be created from the silhouettes of each view, and then use this reconstruction to build a matching function. The visual hull is the volume obtained from the intersection of the silhouette cones [Lau94]. Several methods use matching functions that account for the intersection between the model and the visual hull of the individual [VBMP08], [CMC+06], [MTHC03]. Instead of using the visual hull, Plänkers and Fua [PF01] use depth maps, computed by stereometry using two or more cameras. The matching function in this case is based on closest point distance between the model and the points extracted from the depth map.

Motion measurements such as optical flow are also used as image descriptors. The optical flow information consists in a set of correspondences between pixels of consecutive frames. The pixel displacement in the image is used by Bregler et al. [BMP04]. Ballan and Cortelazzo [BC08] compare the pixel correspondences in consecutive frames with the projection of the vertices of a deformable model in the same consecutive frames.

Color and texture is also used to describe the appearance of individual body parts. Skin color can be a good cue for finding head and hands and is used as feature by López and Casas [LC09].

Image descriptors use to be combined to build more robust likelihood functions. Silhouettes are combined with edges [DBR00], with optical flow [BC08], with depth maps [PF01] or color [LC09]. The combination has to be taken with care as the amount of information varies between descriptors, so a proper weighting for each descriptor should be determined.

### 1.2.2 Estimation Phase

The estimation process is concerned with finding the set of pose parameters that maximize the likelihood function. Often, instead of a likelihood function, we define a cost function or error function, and we search for minima instead of maxima. In the following we will consider that we search for minima.

Cost functions use to have many local minima. Also, the dimensionality of the search space is high. Some methods rely on a estimation based on a single hypothesis, focusing on the efficiency

of a local search. On the other side, we have approaches maintaining multiple hypothesis in order to reduce the probability of getting stuck at a local minimum.

### Single Hypothesis Tracking

Assuming that the time between subsequent frames is small, the distance between body configurations is likely to be small as well. This assumption provides us with an initial pose estimate, using the estimated pose of the previous frame. Single-hypothesis based tracking performs a local search around the initial pose estimate.

Several authors use local-optimization methods to perform the tracking. A common approach is to define an objective function in a least-squares framework [BC08], and then minimize the function using a gradient descent approach, using for example the Levenberg-Marquardt method [PFTV92]. Carranza et al. [CTMS03] use Powell's method and a simple downhill method [PFTV92] that does not need derivatives of the objective function. Delamarre and Faugeras [DF01] perform the search in the image domain, using forces between extracted silhouettes and the projected model to refine the pose estimation.

### Multiple Hypothesis Tracking

Single hypothesis tracking may introduce an accumulation of errors. If a wrong pose is obtained, due to ambiguities such as self-occlusions, maintaining a single hypothesis may propagate the error, and the recovery becomes difficult. To solve this problem, several approaches maintain multiple hypothesis.

Sampling-based approaches, such as particle filtering [IB98], [AMG$^+$02], are able to track non-linear motion, as in the human motion case. In a particle filtering scheme, each particle or hypothesis has an associated weight, that is updated according to the cost function. The particles are propagated in time according to certain dynamics and including a noise component. In the case of human motion, the high dimensionality requires the use of many particles to sample with sufficient density the pose space.

A solution to deal with the high dimensionality is to spread the particles efficiently where a local minimum is more likely. For example, Deutscher et al. [DBR00] use simulated annealing to focus the particles to the global maxima of the posterior. We use these ideas in our system and we will describe them in section 2.2.2.

Another solution to the problem of the dimensionality is to partition the space into a number of lower-dimensional spaces [MI00], [CFCP08], considering the underlying hierarchical structure of the kinematic tree.

# Chapter 2

# Human Motion Tracking: System Components

We address the human motion tracking problem, with an analysis of the effect on the robustness and efficiency of the pose estimation when using distinct sets of system components. We consider a model-based approach, as previous work reported promising results.

In a model-based markerless motion tracking system we can distinguish three main system components: *the human body model*, *the estimation method* and *the likelihood function*. Previous work, as we reviewed in section 1.2, reports an extensive combination of different forms of these three system components. For example, complex human models are combined with deterministic estimation methods in many papers, while other approaches use rather simple models with sophisticated stochastic search schemes. It is hard to draw objective conclusions about the contribution of a certain system component into the overall results, as the reported work showed the results of the system as a whole. We focus the work on an analysis of the performance, robustness and efficiency, of the system when using distinct human models or distinct estimation methods.

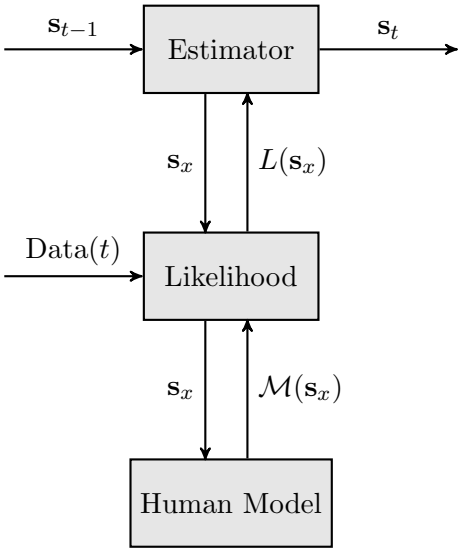



Figure 2.1: System diagram.

The schema of the process performed for each frame is shown in figure 2.1. Generally, the estimator use the pose estimated at the previous frame $\mathbf{s}_{t-1}$ (also with other learned information) to determine an estimated pose for the current frame $\mathbf{s}_t$. To perform this estimation, it evaluates the likelihood function at several poses $\mathbf{s}_x$. The evaluation of the likelihood $L(\mathbf{s}_x)$ depends on the images at instant $t$, Data($t$), and on the 3D representation of the human body model given the pose $\mathcal{M}(\mathbf{s}_x)$.

More precisely, we analyse two kinds of human body model, a cylindrical and a deformable model. Also, we use two kinds of estimators, a deterministic method for optimization (Powell's method) and an stochastic method (Annealed Particle Filter). We implement a likelihood function based on the foreground silhouettes, that is efficient and suitable to use with both body models. We describe in more detail these system components in the following section. Later, in section 3.1, we show how we combine all the components to build the human motion capture system.

## 2.1 Human Body Model

In a model-based motion tracking system we use a human body model to represent our prior knowledge about the human motion process. This model, for example, allows to limit the ranges of possible movements, and it is a basis for the construction of a proper likelihood function.

The human body model describes the kinematic properties of the body by means of an articulated skeleton. The shape and appearance is modelled by a flesh or skin model. For the skin model we use two different representations: a representation based on cylindrical shapes; and a deformable polygonal mesh of anthropometric shape. These two representations provide distinct levels of refinement of the model, and allow to increase the accuracy of the fitting between the model and the subject tracked.

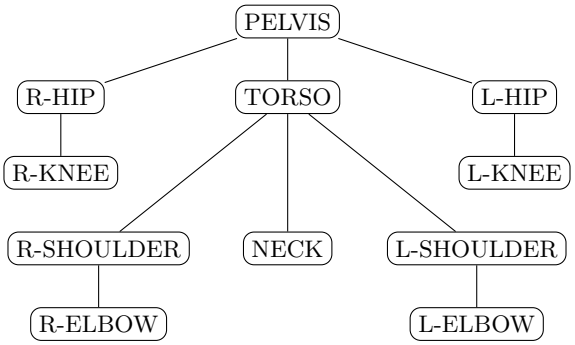
### 2.1.1 Articulated Skeleton Model

We use an articulated skeleton composed by bones and joints that defines the kinematics of the skin model. The skeleton is organized in a hierarchical manner. We choose a configuration that allows a full body tracking, not including hand and foot movements, similar to the models presented in [DBR00] or [CTMS03]. Each joint defines a local coordinate system (CS), generally with the origin in one end point of the bone linked to the joint, and the axis relative to the direction of the bone. Each joint also defines the transformation, i.e translation and rotation, between a parent and a child CS, following the joints hierarchy shown in Figure 2.2.a.

A joint can have up to 3 degrees of freedom (DOF), corresponding to the rotations relative to the axes $x$,$y$ and $z$ of its own CS. Some joints have less DOF, as for example the elbow that has only 1 DOF. The root of the hierarchy is situated at the pelvis, which defines the CS used to translate and rotate the whole model, and results in 6 DOF. Figure 2.2.b shows a table with the DOF associated to each joint.

Consider a joint $j$, we denote by $\mathbf{G}_j$ the matrix that transforms from the $j$'s parent CS to $j$'s CS, when the model is in the rest pose.

Consider the pose is given as an state vector of $D$ angle variables $\mathbf{s} = \{\theta_1, \theta_2, \ldots, \theta_D\}$, each of the angles defining the rotations to apply to each joint. Given a certain pose $\mathbf{s}_x$, to determine the positions of the joints and bones we start from the root of the hierarchy and follow recursively to the children. For each joint $j$ we apply the transformation $\mathbf{G}_j$, and then the rotations from $\mathbf{s}$ that correspond to the joint $j$.




(a) Joints Hierarchy

| Joint | DOF |
|---|---|
| Pelvis (CS of the body) | 6 |
| Neck | 3 |
| Right Shoulder | 3 |
| Left Shoulder | 3 |
| Right Elbow | 1 |
| Left Elbow | 1 |
| Right Hip | 2 |
| Left Hip | 2 |
| Right Knee | 2 |
| Left Knee | 2 |
| Torso | 3 |

(b) Joints DOF

Figure 2.2: Description of the joints hierarchy of the model (a) and the DOF of each joint (b).

### 2.1.2 Cylindrical Skin Model

We approximate the shape of the human body by using an arrangement of cylindrical objects and boxes attached to the limbs and torso. This kind of model is computationally simple and has been used successfully in previous work [DBR00]. The size of the cylinders and boxes is adjusted to the subject tracked during an initialization phase as we describe in section 3.1.1. We place these objects according to the skeleton model bones and joints such that they can be translated and rotated following the bones movements. We use the OpenGL library to model and render tessellated polygonal approximations of the cylinders. Figure 2.3 shows the cylindrical model.

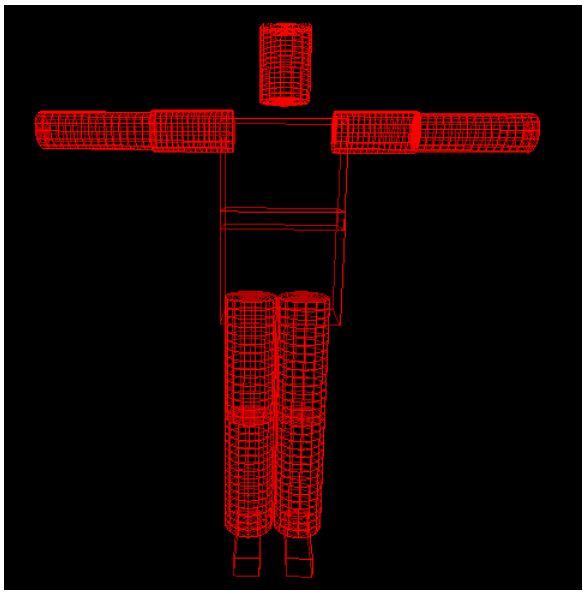


Figure 2.3: Cylindrical model

### 2.1.3 Deformable Skin Model

We use an anthropometric model based on a polygonal mesh deformable according to an underlying skeleton. We adapted the model and the deformation technique of the MakeHuman[1] platform [Mak], [BRM08], in order to adjust them to our needs.

The model is deformed to fit certain height, width or limbs proportions, or more generally the physical characteristics of a person. Similarly, an algorithm modifies the polygonal skin of the human model to obtain the shape corresponding to a certain pose.

The deformation technique is based on the principle of shape interpolation, also called shape blending or target morphing. We distinguish two kinds of operations involved in the deformation: *translational morphing* and *rotational morphing*.

**Translational Morphing** The translational morphing is a linear interpolation on the positions of the vertices of the mesh. We denote $\mathbf{x}_0$ the position of a vertex in the reference model. Let be $\mathbf{x}_m$ the position of the same vertex in a target model. The translational morphing determines the new position of the vertex as

$$\mathbf{x} = \mathbf{x}_0 + \alpha(\mathbf{x}_m - \mathbf{x}_0) \tag{2.1}$$

where $\alpha$ is the morphing coefficient.

**Rotational Morphing** We extend the above approach to rotations. We scale the angle according to the morphing coefficient and generate the rotation matrix that we apply to the vertex position. We obtain $\theta = \alpha\theta_m$, where $\theta_m$ is the angle corresponding to the modeled target, and $\alpha$ is the morphing coefficient. We build a rotation matrix $R_\theta$ according to the rotation axis that drives the deformation. Consider $\mathbf{x}_0$, the position of a vertex in the reference model. Consider $\mathbf{x}_c$ the center of rotation. The new vertex position is then defined as

$$x = R_\theta(\mathbf{x}_0 - \mathbf{x}_c) + \mathbf{x}_c \tag{2.2}$$

#### Multi-Target Morphing

The deformations of the mesh allow to adjust the physical properties of the model to a specific individual. Such kind of deformations use the translational morphing operation. Several instances of the model sculpted by artists serve as targets to drive the morphing. The combination of multiple targets, with different assigned weights for each of them, will determine the actual deformation.

Each target model focus on a certain physical feature, such as tallness, arms length, torso width, etc. The positions of the vertices of each target model respect to the position in the reference model determines translation vectors associated with each vertex. We call *morph vectors* such translation vectors, associated to a certain target and vertex. We denote them as $\mathbf{v}_{ij}$ for the translation vector of the $i$ vertex corresponding to the $j$ morphing target. We modify a vertex position according to a number $N$ of morphing targets as

$$\mathbf{x}_i = \mathbf{x}_i + \sum_{j=1}^{N} \alpha_j \mathbf{v}_{ij} \tag{2.3}$$

where $\alpha_j$ is the morphing weight assigned to each target.

[1] MakeHuman is an open source software for the modelling of 3-dimensional humanoid characters.

Figure 2.4: Deformable model

### Pose Function

The mesh deformation also focus on setting the model into a certain pose. The rotation angles of the joints of an underlying skeleton will drive the deformation. Applying only a rotation of the vertices according to its associated joint will produce undesired results. The obtained mesh shrinks near the rotating joints, producing what is known as the *collapsing elbow* effect [LCF00]. Figure 2.5.a shows the kind of deformation obtained by only rotating the vertices. Instead of performing only a rotation, we obtain the position of the vertices using a translational and a rotational morphing to overcome the shrinking problem. The final position of the limb is modeled by artists, which include muscle bulging or other deformations related to secondary muscles or adjustments. The limb rotation is then achieved by combining a translation and a rotation of the vertices positions. According to a morphing coefficient $\alpha$ we build a rotation matrix $R_\theta$. The modeled target provides a morph vector $\mathbf{v}$. The position of the vertices of the limb is defined as

$$\mathbf{x} = R_\theta(\mathbf{x}_T - \mathbf{x}_c) + \mathbf{x}_c \tag{2.4}$$

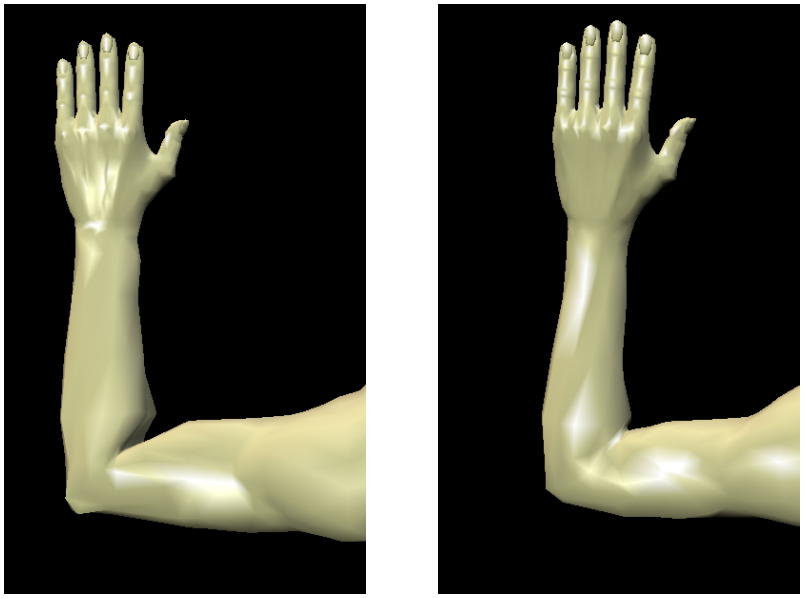
where $\mathbf{x}_c$ is the center of rotation associated to the limb, and $\mathbf{x}_T$ is the translation defined as

$$\mathbf{x}_T = \mathbf{x}_0 + \alpha\mathbf{v}. \tag{2.5}$$

Figure 2.5.b shows the deformation obtained.

### Form Factor

In the creation process of the target models, the modelers work on the reference model. Thus, the transformations are referred to such model. When the model has been subject of a significant transformation, applying the morphing translation using the same precomputed morph vectors would introduce an unexpected behavior. To solve this problem we adjust the morphing according to the dimensions of its area of influence.



(a) (b)

Figure 2.5: Pose deformation. (a) Deformation obtained applying only a rotation, "*collapsing elbow*" effect. (b) Deformation obtained applying rotation and translation, using a modelled example as morphing target.

The vertices that become modified by a particular morphing target can be closed in a bounding box. This bounding box determines a limit height, width and depth of the area of influence of the morphing target. We compare the dimensions of the bounding box in the transformation phase with the dimensions of such box in the original mesh. The relation of both sizes is the *form factor*, which is used as a scale of the morphing coefficient. Thus, we can adapt the transformations to apply them to an already deformed model.

## 2.2 Estimation

In the estimation process we are concerned with finding the pose parameters that minimize the error between observations and the body model projection. Such error function or likelihood function has many local minima, and also is a high dimensional function. These two facts are the main difficulties involved in the estimation process.

We describe, in the following sections, two different approaches. The first one, Powell's method, is a deterministic local search method suitable for multidimensional spaces. The second method, particle filtering, is an statistical technique to maintain multiple hypothesis during the tracking process, and perform a rather global search of the optimal pose.

### 2.2.1 Multidimensional Optimization

Given a single function $f$ that depends on multiple variables, we are interested in finding the values of those variables where $f$ takes a minimum value. Finding a global minimum can be a very difficult task and usually we only assure the achievement of a local minimum. Derivatives of the function use to be of great help in solving optimization problems, but in our case we do not

have an analytical description of the functions we are going to minimize, so we cannot determine the derivatives. We use the *direction-set method* or Powell's method which is a quadratically convergent method that does not need derivatives of the function.

To minimize a function $f$ of $N$ variables we can start at point $P$ in the N-dimensional space, choose a direction vector $\mathbf{v}$, and minimize $f$ along the direction $\mathbf{v}$ using a one-dimensional method. Then, we will choose another direction and minimize along this new direction, and then iteratively continue in the same manner following certain criteria. Thus, our key ingredients are the one-dimensional minimization method and the way we choose the appropriate directions for each iteration.

**One-Dimensional Optimization**

The minimization of functions in one dimension starts by determining an interval of abscissas within we can assure that there is a minimum. In this case, we say that the minimum has been *bracketed*. A minimum is known to be bracketed only where there are three points, $a < b < c$, such that $f(b)$ is less than both $f(a)$ and $f(c)$. In this case we know that the function has a minimum in the interval $(a, c)$.

Once the minimum has been bracketed $(a, b, c)$, the common process would be to choose a new point $x$, either between $a$ and $b$ or between $b$ and $c$. Then we evaluate $f(x)$. If $f(b) < f(x)$ we bracket the minimum as $(a, b, x)$. On the other side, if $f(b) > f(x)$ we bracket as $(b, x, c)$. The bracketing process will continue until the distance between the two outer points is tolerably small.

A main question is the way we choose the point $x$. One option is to choose $x$ such that it minimizes the worst case possibility. In this case we use the so-called *golden ratio* ($\Phi \approx 0.61803$) as shown in [PFTV92]. This means that an optimal bracketing interval $(a, b, c)$ has its middle point $b$ a fractional distance 0.38197 from one end (say, $a$), and 0.61803 from the other (say, $b$).

The golden ratio method handles the worst possible cases, but if the function is sufficiently smooth there are better techniques to choose the middle point and achieve the minimum faster. We use the Brent's method, that assumes that the function may be more or less parabolic near the minimum. It fits a parabola using three points and obtains the abscissa of the minimum of the parabola. With this approach, with a single step we can get very close to the minimum.

As the parabolic approximation may not be always a good solution, Brent's method switches to the golden ratio search approach, whenever the parabolic fitting is not providing good result. To do so, we keep track of six function points:

- $a$ and $b$ bracket the interval.
- $x$ is the point with the very least function value found so far.
- $w$ is the point with the second least function value.
- $v$ is the point with the previous value of $w$.
- $u$ is the point at which the function was evaluated more recently.

At first, the parabolic interpolation is attempted using the points $x$, $v$ and $w$. The step is accepted according to two criterion:

(i) the parabolic step falls within $(a, b)$.

(ii) the movement from $x$ (best current value) is less than the movement of the step before last. This insures convergence.

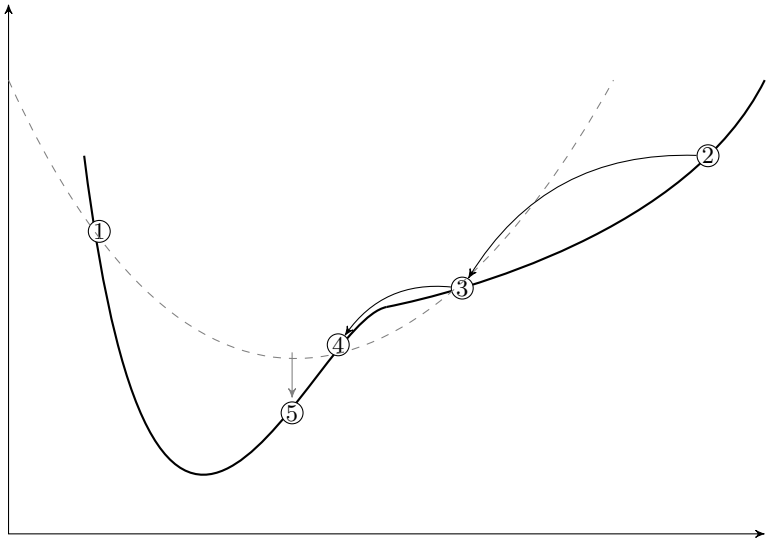



Figure 2.6: One dimensional minimization example. We start bracketing the minimum with points 1, 2 and 3. First, the golden ratio method determines the point 4, giving a new bracketing $(1, 4, 3)$. Then, with parabolic fitting with the points $(1, 4, 3)$, we obtain the point 5, which defines a new bracketing with points $(1, 5, 4)$.

If the parabolic step is not acceptable, we use the golden ratio to choose the next bracketing. We also use a *tolerance* distance such that the function is never evaluated for points that are close enough from an already evaluated point. Also, this tolerance is used to test whether the minimization is done or not. Figure 2.6 shows a graphical example of a minimization, combining golden ratio and parabolic fitting.

**Direction Set Methods for Multidimensional Optimization**

As we already explained above, the general schema for the minimization of multidimensional functions is to perform successive line minimizations. Thus, a simple method would be to take the unit vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N$ and perform one-dimensional minimizations along these directions, until the function stops decreasing. The problem for such approach arises when the function's second derivatives are much larger is some directions than the others. Then we will cycle through successive line minimizations, and the approximation to the minimum will be very slow. For example, consider a 2-dimensional function that defines a long narrow valley at some angle of the coordinate basis. We will perform successively line minimizations along the two coordinate axis directions, while not getting much closer to the minimum. Figure 2.7.a shows this example. The 2-dimensional function is represented by several contour lines, which define a narrow valley at certain angle of the coordinate basis. In this case, if we had chosen the direction of the valley it would have taken us much faster to the minimum.

The *direction set* methods focus on updating the directions while the minimization proceeds, in order to get better directions than the coordinate basis $\mathbf{e}_i$.

The direction set method proposed by Powell is based on the concept of the *conjugate directions*. We explain this concept in the following of this section.

Consider a point $\mathbf{P}$ as the origin of a coordinate system with coordinates $\mathbf{x}$. Consider the

Taylor approximation of the function $f$,

$$f(x) \approx c - b \cdot \mathbf{x} + \frac{1}{2}\mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x} \tag{2.6}$$

where

$$c \equiv f(\mathbf{P}) \quad b \equiv -\nabla f|_{\mathbf{P}} \quad [\mathbf{A}]_{ij} \equiv \frac{\delta^2 f}{\delta x_i \delta x_j}|_{\mathbf{P}}$$

The matrix $\mathbf{A}$ is also called the *Hessian matrix*.

If we minimize the function $f$ along some direction $\mathbf{u}$, at the minimum the gradient of the function must be perpendicular to $\mathbf{u}$. The gradient of $f$ is

$$\nabla f = \mathbf{A} \cdot \mathbf{x} - b \tag{2.7}$$

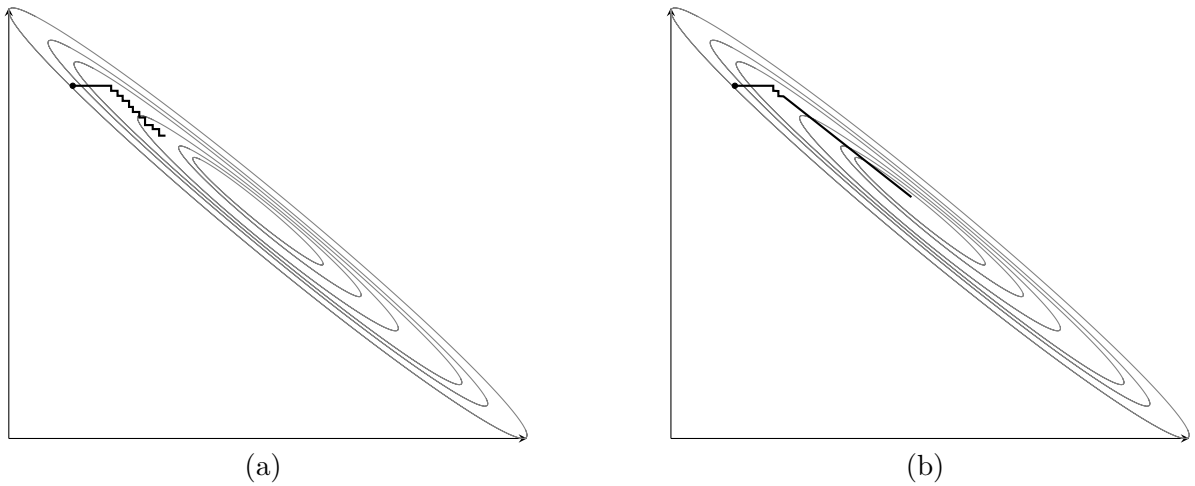
As we move along some direction the gradient changes as

$$\delta(\nabla f) = \mathbf{A} \cdot \delta\mathbf{x} \tag{2.8}$$

Suppose we have moved along the direction $\mathbf{u}$, and now we want to move along a direction $\mathbf{v}$ such that the motion does not break down the minimization we have already done. The condition we have to fulfill is that the change in the gradient must be perpendicular to $\mathbf{u}$. The change in the gradient would be $\delta(\nabla f) = \mathbf{A} \cdot \mathbf{v}$. Thus the condition is

$$0 = \mathbf{u} \cdot \delta(\nabla f) = \mathbf{u} \cdot \mathbf{A} \cdot \mathbf{v} \tag{2.9}$$

If this condition is fulfilled, the two vectors $\mathbf{u}$, $\mathbf{v}$ are said to be *conjugate*. If we do successive line minimizations along a conjugate set of directions we do not need to redo any of the directions. Thus, if we obtain a set of $N$ linearly independent conjugate directions, one pass of $N$ line minimizations will end exactly at the minimum of a quadratic function. For functions that are not exactly quadratic repeated cycles will converge quadratically to the minimum.



(a) (b)

Figure 2.7: (a) Naive multidimensional minimization: we apply succesive line minimizations along the coordinate basis. (b) Powell's method example: we obtain an average direction, and minimize along this direction.

Powell proposed a method to generate a conjugate set of directions after performing $N(N-1)$ line minimizations, but the problem is that it tends to produce sets of directions linearly dependent. We use the slightly modified Powell's method presented in [PFTV92], which finds good directions although they are not necessarily conjugate directions. Thus, it does not have the quadratic convergence property, which is not bad in our case, as we cannot insure that we deal with functions with a form close to quadratic.

The basic idea of the method is as follows:

We initialize the set of directions $\mathbf{u}_i$ to the basis vectors

$$\mathbf{u}_i = \mathbf{e}_i \quad 1, \ldots, N$$

We repeat the following procedure until the function stops decreasing:

- Save the starting position as $\mathbf{P}_0$.
- For $i = 1, \ldots, N$, minimize $f$ starting at $\mathbf{P}_{i-1}$ along the direction $\mathbf{u}_i$. Call the point obtained $\mathbf{P}_i$. Keep track of the direction which makes the *largest decrease* of $f$. We denote by $\mathbf{u}_j$ such direction.
- Obtain $\mathbf{u}_{avg} = \mathbf{P}_N - \mathbf{P}_0$, the average direction.
- Minimize $f$ starting at $\mathbf{P}_N$ along the direction $\mathbf{u}_{avg}$
- Replace the direction of largest decrease by the average direction.
  $\mathbf{u}_j \leftarrow \mathbf{u}_N$ and $\mathbf{u}_N \leftarrow \mathbf{u}_{avg}$.

The idea of replacing the direction of largest decrease is that it is likely to be the major component of the new average direction we are adding, and in this way we avoid building up a linear dependence. The final algorithm also adds an exception, which allows to avoid adding a new direction if it will not give better results. This exception is based on the function value at an extrapolated point using the new average direction. Figure 2.7.b shows an example of the Powell's method applied on the same 2-dimensional function described above.

The linear minimization method used in the algorithm is the Brent's method we described in section 2.2.1. Also note that each linear minimization step needs and initial bracketing of the minimum. This bracketing is done using a point and the direction given by Powell's method. The bracketing method used is the one proposed in (referencia), which is based on a parabolic extrapolation.

### 2.2.2 Particle Filtering

We denote by $\mathbf{x}$ the state vector that defines the model pose and position. Consider the evolution of the state sequence $\mathcal{X}_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ over time, with $\mathbf{x}_t$ the state on the time step $t$. We denote by $\mathcal{Z}_t = \{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$ the set of all available measurements up to time-step $t$. From a Bayesian perspective, the tracking problem is to recursively calculate some degree of belief in the state $\mathbf{x}_t$ at time $t$, given the data $\mathcal{Z}_t$ up to time $t$. Thus, we require the pdf $p(\mathbf{x}_t|\mathcal{Z}_t)$.

We assume that the model dynamics form a temporal Markov chain so that

$$p(\mathbf{x}_t|\mathcal{X}_{t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2.10}$$

and the measurements $\mathbf{z}_t$ are assumed to be independent, both mutually and with respect to the dynamical process. Then, following the Bayes rule, the propagation of the state density over time is

$$p(\mathbf{x}_t|\mathcal{Z}_t) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathcal{Z}_{t-1})}{p(\mathbf{z}_t|\mathcal{Z}_{t-1})} \tag{2.11}$$

where

$$p(\mathbf{x}_t|\mathcal{Z}_{t-1}) = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathcal{Z}_{t-1}) \tag{2.12}$$

as shown in [IB98]. This recursive propagation of the posterior density cannot be determined analytically, except for a restrictive set of cases. For example, the *Kalman filter* is an optimal solution for the recursive Bayesian estimation, in case that the posterior density at every time step is Gaussian, and the dynamics of the model are linear. The observation density $p(\mathbf{z}_t|\mathbf{x}_t)$ in multicamera environments with uncontrolled background is multimodal, due to the chance of detecting erroneous image features. Thus, as the observation density is non-Gaussian, the evolving state density $p(\mathbf{x}_t|\mathcal{Z}_t)$ is also generally non-Gaussian. The Kalman filter has been shown to fail is this kind of tracking environments. The *Particle Filtering* is a more general approach to approximate the Bayesian estimate for arbitrary densities.

In the particle filtering scheme, the posterior density $p(\mathbf{x}_t|\mathcal{Z}_t)$ is represented by a finite set of *normalized weighted particles*, or samples,

$$\{(\mathbf{s}_t^{(0)}, \pi_t^{(0)}) \cdots (\mathbf{s}_t^{(N)}, \pi_t^{(N)})\}.$$

Then, we can estimate the state $\hat{\mathbf{x}}_t$ calculating the sample mean of the posterior density as

$$\hat{\mathbf{x}}_t = \mathcal{E}[\mathbf{x}_t] = \sum_{n=1}^{N} \pi_t^{(n)} \mathbf{s}_t^{(n)}. \tag{2.13}$$

The tracking process of the particle filter is based on the following operations, as summarized in (referencia APF):

- *Resampling*: a weighted particle set is transformed into a set of evenly weighted particles distributed with a concentration dependent on probability density.
- *Propagation*: stochastic movement and dispersion of the particle set in accordance with a motion model to represent the growth of uncertainty during movement of the tracked body.
- *Measurement*: the likelihood function is evaluated at each particle site, producing a new weight for each particle proportional to how well it fits the data. The weighted particle set produced represents the posterior probability density.

Constructing a valid observation model or likelihood $p(\mathbf{z}_t|\mathbf{x}_t)$ as a normalized probability density is difficult and costly to evaluate. Instead of that model, we use an intuitive weighting function $w(\mathbf{z}_t, \mathbf{x}_t)$ that approximates the probabilistic likelihood $p(\mathbf{z}_t|\mathbf{x}_t)$ but requires much less computation cost to evaluate.

The ability of the particle filter to propagate multiple hypotheses provides a robust tracking, as less likely model configurations will not be thrown away immediately and have a chance to prove themselves later on. However, several authors (referencies) have shown problems with the particle filter tracking in high dimensional configuration spaces such as the human motion capture domain. The main reason is that the particle set does not populate adequately the space to represent an arbitrary density when using a reasonable number of particles. In fact, the number of particles required is extremely large and it becomes infeasible.

We use a particle filtering based search, the *annealed particle filter* (APF) proposed by Deutscher et al. (referencia), that overcomes the difficulties of the high dimensional spaces.

**Annealed Particle Filtering**

The annealed particle filter (APF) approach consists on applying the ideas of the simulated annealing [DBR00] to the particle filtering scheme. The weighting functions used for motion capture tend to have several narrow local maxima. The APF perform the evaluations of the particles using several smoothed versions of the weighting functions. In this way the search is not misdirected by local maxima.

Consider the original weighting function $w(\mathbf{z}_t, \mathbf{x}_t)$. A series of weighting functions is set by

$$w_m(\mathbf{z}_t, \mathbf{x}_t) = w(\mathbf{z}_t, \mathbf{x}_t)^{\beta_m}, \tag{2.14}$$

for $\beta_0 > \beta_1 > \cdots > \beta_M$, with $\beta_0 \leq 1$. A large $\beta_m$ produces a peaked weighting function, while low values produce rather broad smooth functions.

At each time step $t$, using the image observations $z_t$, we perform $M$ particle filtering steps. Thus, for each *annealing layer* $m$, starting with $m = M$ and decreasing $m$, we do the following steps:

1. The layer $m$ is initialized by a set of unweighted particles. Each of the particles is assigned a weight proportional to the evaluation of the weighting function $w_m$.

2. We resample the particles according to its weights. The particles are then propagated using a multivariate Gaussian random variable with mean 0 and variance $\mathbf{P}_m$.

3. The set of particles is used to initialize the layer $m - 1$.

From the particle set obtained after the layer 0 we estimate the model configuration at time $t$ using the sample mean $\mathcal{E}[\mathbf{x}_t]$ described in equation (ref). This particle set is propagated to generate the set for the next time step $t + 1$.

The parameter $\beta_m$ that determines the smoothing rate applied to the weighting function, must decrease over the successive layers. We set this parameter as a geometric progression with a ratio $\alpha$, called the annealing rate, such that

$$\beta_m = (\alpha)^m. \tag{2.15}$$

A typical value might be $\alpha = 0.5$.

The covariance used to generate the first annealing layer $\mathbf{P}_0$ has its diagonal elements equal to half the maximum expected movement of the corresponding model parameter over one time step. The covariance should decrease as the resolution of the particle set increases. We set

$$\mathbf{P}_m = (\alpha_P)^{(M-m+1)} \times \mathbf{P}_0 \tag{2.16}$$

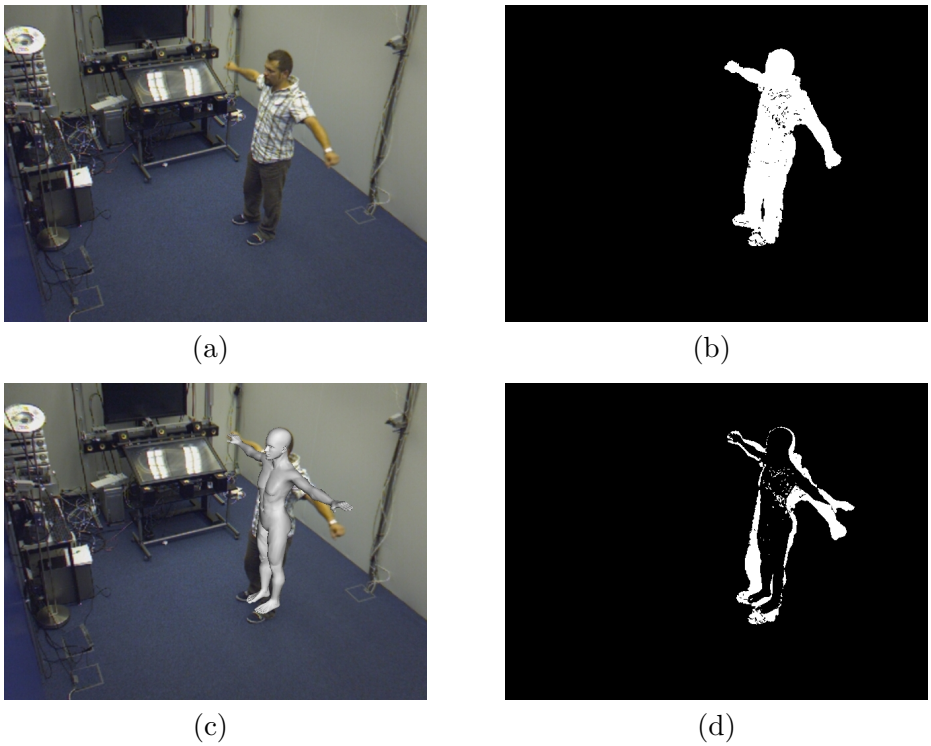
where $\alpha_P$ is the covariance rate, usually set to $\alpha_P = 0.5$.

## 2.3 Likelihood Function

The cost function measures how close is the body model synthesized in a specific pose with respect to the input images. The number of evaluations of the cost function is very high, both in the Powell's method and in the particle filtering scheme. Thus, an efficient cost function is decisive for the system performance. The function we implement performs a pixel-wise XOR

between the input image silhouettes and the rendered model silhouettes. This function can be implemented in graphics hardware as shown in [CTMS03] and [TCMS03].

First, the silhouettes of the input images are obtained by a background substraction technique. We use the *Running Gaussian Average* technique [Pic04], that models the background colors of the scene with a Gaussian distribution per pixel learned during a training period. Then, at each frame a pixel is classified as foreground if the distance between its color intensity and the Gaussian mean is above a threshold. When the pixels are classified as background, the mean and variance of the Gaussians are updated. We remove shadows and highlights with the method proposed by Xu et al. [XLP05]. We consider the chromaciticy and brightness distortion to detect shadowed regions and highlights, and then they are removed by a morphological filter. Figure 2.8.b shows the extracted silhouette from the original image in Figure 2.8.a.



(a) (b)

(c) (d)

Figure 2.8: (a) Image captured by one of the cameras. (b) Silhouette extracted from the image in (a). (c) The human model superposed on the image. (d) The result of applying the XOR between the silhouette and the model's projection.

The cost function operates on the set of multiple cameras. These cameras are calibrated to recover the parameters we need to render the model on each of the views. In the appendix A, we describe the camera model we use, and the relation between the parameters obtained from the calibration process and the OpenGL parameters.

For each of the cameras we perform an XOR between the foreground silhouette from this camera and the model rendered on that view. The sum of all non-zero pixels resulting from this operation are the cost associated for this camera. We use the OpenGL stencil buffer and its associated operations to perform the XOR, so we can exploit the parallel processing of the GPU. Using an 8-bit stencil buffer we can perform the XOR operation for up to 8 cameras, obtaining the result with a single read of the stencil buffer. In this way, we can reduce the

amount of memory transferred from the GPU to the CPU. The algorithm implemented is as follows:

- First, the silhouette images are loaded in the GPU texture units, such that alpha values are 0 for background pixels and different of 0 for foreground pixels. We clear the stencil buffer.
- We perform the following operations for each camera $i$:
  1. We set a mask on the stencil buffer, in order to enable writing only in the bit plane $i$.
  2. We enable the stencil test. We set the stencil test such that it is always passed, and the stencil operation is set to replace the buffer values by 1 on the bit plane $i$.
  3. We set the camera projection matrix with the parameters of camera $i$. We render the model. At this moment, the stencil buffer pixels corresponding to the silhouette of the model have value 1.
  4. We enable the alpha test. We set the stencil test such that it is passed when the buffer bit value equals 1. We set the stencil operation such that it replaces the stencil bit value by 0 when it is 1.
  5. We set an orthographic projection matrix and the viewport considering the image size. We bind the texture associated to camera $i$ and we render a quad. Note that only silhouette pixels pass the alpha test. The stencil buffer values of these pixels are replaced by 0 only in case that the old value is 1, thus performing the XOR operation.
- Once performed the operation for all cameras we transfer the 8-bit stencil buffer from GPU to CPU. We account for the bits that equal 1 in the CPU.

Figure 2.8.d shows the result of the XOR operation for one of the cameras, when rendering the model in the position and pose shown in figure 2.8.c.

# Chapter 3

# Implementation and Results

## 3.1 System Implementation

In this section we describe the motion tracking system implemented. The process is divided in two main phases:

- *Initialization Phase*: this phase proceeds using a single frame, where the actor adopts an specific pose. An optimization procedure obtains the pose and shape parameters that best adjust the deformable model to the input images.
- *Tracking Phase*: the tracking phase is executed each frame during all the sequence. We estimate the pose of the actor at the current frame, given the pose estimated at the previous frame. Note that the estimator may maintain information related to the whole sequence until the current frame.

The following sections describe in more details these two phases, and how we combine the system components explained in sections 2.1, 2.2 and 2.3 to obtain the final algorithms.

### 3.1.1 Initialization Phase

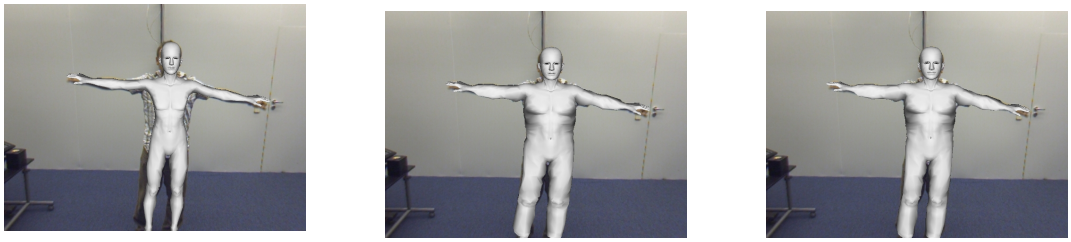
The goal of the initialization phase is to obtain a human body model as much adjusted as possible to the specific individual we are going to track. We use the cost function based on silhouettes described in section 2.3, and the projection of the deformable skin model explained in section 2.1.3. We build a function of the pose and shape parameters that is minimized using the Powell's method (section 2.2.1).

In the initialization phase, the pose is known approximately as the subject adopts an specified position. However, we must adjust the pose parameters of the model skeleton in order to determine correctly the parameters related to the shape. The parameters we use to adjust the shape correspond to the multi-target feature of the deformable skin model 2.1.3. We call these parameters the *morphing parameters*. Thus, the pose parameters together with the morphing parameters form a high dimensional space which is based on variables of different nature. An optimization over the whole dimensionality leads to bad results due to the multimodality of the function. To overcome this problem we perform an heuristic subdivision of the space, and then we perform local optimizations in these subspaces iteratively.

We build several functions:

- *global translation function*: it depends only on the translation $x, y, z$ of the whole model.
- *global rotation function*: it depends only on the rotations of the whole model along the axis $x, y, z$.

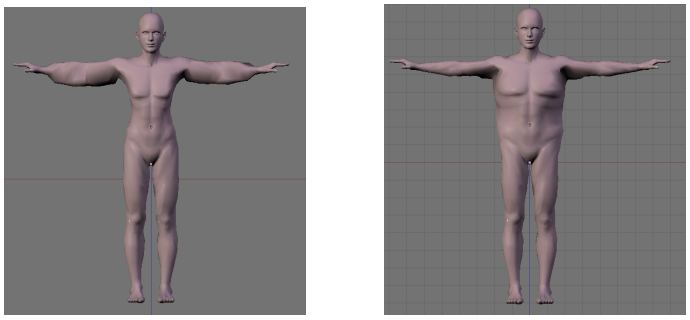
- *pose functions*: we build several functions according to a hierarchical decomposition of the pose space. We have a pose function depending on the torso rotational parameters. Then, a different function associated to each of the limbs.
- *morphing function*: it depends on parameters related to limbs and torso deformations, affecting the limbs and torso length and width, and general anthropometric features.



1 iteration (only pose) 2 iterations 3 iterations

Figure 3.1: Results of the initialization phase. At the first iteration we optimize only the pose. The other two iterations optimize pose and shape morphing parameters.

We obtain the optimal parameters using Powell's method applied sequentially over the functions. We start with the translation function, then the global rotation, the torso pose, arms, legs and morphing function. This optimization sequence can be applied several times. Note that each of the subspace optimizations will converge due to the tolerance values of the Powell's optimization method (section 2.2.1). We noticed that more than 2 optimization iterations do not improve the results significantly. Figure 3.1 show the results of the initialization phase.



(a) (b)

Figure 3.2: Morphing targets modelled to simulate parts of a dressed body. Simulate a deformation due to loose-fitting clothes, either on arms (a) or torso (b).

### Modelling of Morphing Targets

In order to optimize the parameters related to the limbs and torso deformations, we need several instances of the model, that should be sculpted to represent certain anthropometric features. These distinct instances of the model, called morphing targets, will define the translational

vectors that guide the displacement of the vertices of the skin mesh. The different morphing targets should cover a wide range of human shapes, such that when applying the multi-target morphing operation, explained in section 2.1.3, we can fit the model to any specific individual.

We use several morphing targets obtained from the MakeHuman platform [Mak], that allow the adjustment of limbs length, and overall proportions. However, the generic human model and the targets modelled by the makehuman community focus on the modelling of a nude body. For example, many of the targets model the muscles sizes. In our case, the actors wear clothes, so we need deformations of the reference mesh to adjust the shape to a dressed individual.

We model with Blender[1] several targets that simulate parts of a dressed body. We start from the reference model, and we deform the mesh to simulate an individual wearing loose-fitting clothes. When the modelling has finished, the difference between the vertex positions of the obtained model and the vertex positions of the reference model determine translational vectors for each vertex. We store these vectors to use them as morphing targets during the optimization process. Figure 3.2 show two of these targets, modelled to simulate the effect of loose-fitting clothes on the arms and torso appearance.

1 iteration 2 iterations 4 iterations

Figure 3.3: Results of the estimated pose obtained by a sequence of Powell's optimizations, performing 1, 2 or 4 iterations. Images show the deformable model and the skeleton superposed over a single frame, while the actor was moving the arms down.

### 3.1.2 Tracking Phase

The tracking phase estimates the pose vector of the individual frame by frame. In this case, we studied several distinct approaches in order to determine advantages and drawbacks of each of the methods. Following the notation of the diagram in figure 2.1, we implemented different versions of the system, either using the Powell's method or the Annealed Particle Filter as Estimators, and then combined with the Cylindrical or the Deformable model as Human Body Models. In all cases we use the cost function described in section 2.3.

#### Tracking with Powell's Method

During the tracking phase, at each frame the Powell's optimization process starts with the pose vector estimated at the previous frame. As we already explained in the initialization phase description (section 3.1.1), an optimization over the whole pose space using Powell's

[1]Blender is an open source 3D content creation suite. http://www.blender.org

method does not provide good results. During this tracking phase also apply a hierarchical decomposition of the pose space. So, we perform sequentially the Powell's optimization of the global translation, rotation and pose functions described in section 3.1.1.

The full sequence of optimizations is repeated to improve the results, as generally a single iteration is not accurate. Note that we adopt the estimated pose of the previous frame as starting point. Thus, when optimizing within a certain subspace, the rest of the pose vector remain fixed with the previous pose. Consider for example the optimization of the global translation of the body. In this case, we maintain fixed the rest of the pose vector, with the pose estimated at the previous frame. So we estimate the global translation with the current image data, where the pose has changed, with the body rendered with the previous pose. This leads to inaccurate results as shown in Figure 3.3. In this case, the actor is moving the arms down, so the estimation of the global translation ends with a lower value of the $z$ coordinate. The estimation improves when performing 2 or 4 iterations. We used the deformable model for these examples.

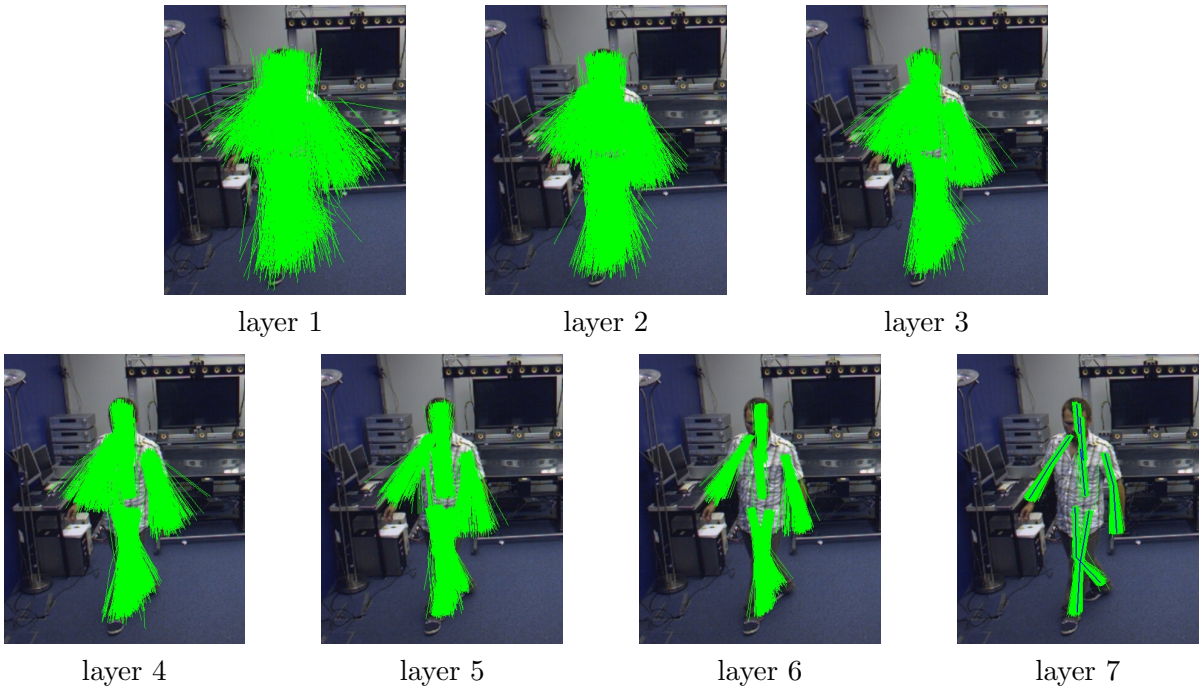



Figure 3.4: Evolution of the particle set during consecutive annealing layers. The last layer shows the estimated pose in blue.

### Tracking with Annealed Particle Filtering

The tracking based on the annealed particle filter maintains multiple sample poses, corresponding to the particles, during the whole sequence. The particles are assigned a weight according to its evaluation with the cost function. The whole set of particles is evaluated, resampled and propagated at each frame, along several annealing layers. Then the particle set resulting after the last layer is used as initial set for the next frame.

The variance of the random Gaussian variable used during the propagation step conditions the exploration of the pose space. For the initial layer, the variances are set to cover the

maximum displacement of the skeleton bones between consecutive frames. Then the variances are reduced along the consecutive layers following the expression in equation 2.16.

Figure 3.4 shows the evolution of the particle set during consecutive annealing layers. The particles are shown as skeleton samples in green color. For the last layer, the estimated pose using the sample mean $\mathcal{E}[\mathbf{x}_t]$ (equation 2.13) is shown in blue. We used the cylindrical model for these experiments.

### Cylindrical and Deformable Model Comparison

The error between the projection of the model and the silhouettes depends on two facts: (i) the difference between the pose synthesized and the actor's pose ; (ii) the shape of the model with respect to the shape of the actor. The human model representation chosen has an influence on this last fact, but it is not obvious to split the contribution on the error that is due to the model shape and the part that is due to the estimated pose. For example, we can have a situation where a given pose minimizes the error using the cylindrical model, but when using the deformable model, this given pose is not a minimum of the cost function. Then we can compare either the error when keeping the pose fixed for both models, or the error with the pose that results in a minimum for each of the models. In both cases the comparison is not fair.

We compare the behavior of each of the models by analyzing the results of the estimation. We use the Powell's method as estimator. As we will show below in next section, this method is more precise in the final estimation than the annealed particle filter, when working with slow movements relative to the frame rate. Thus, the use of the Powell estimator provides more reliable results to analyze the models influence on the final estimation.

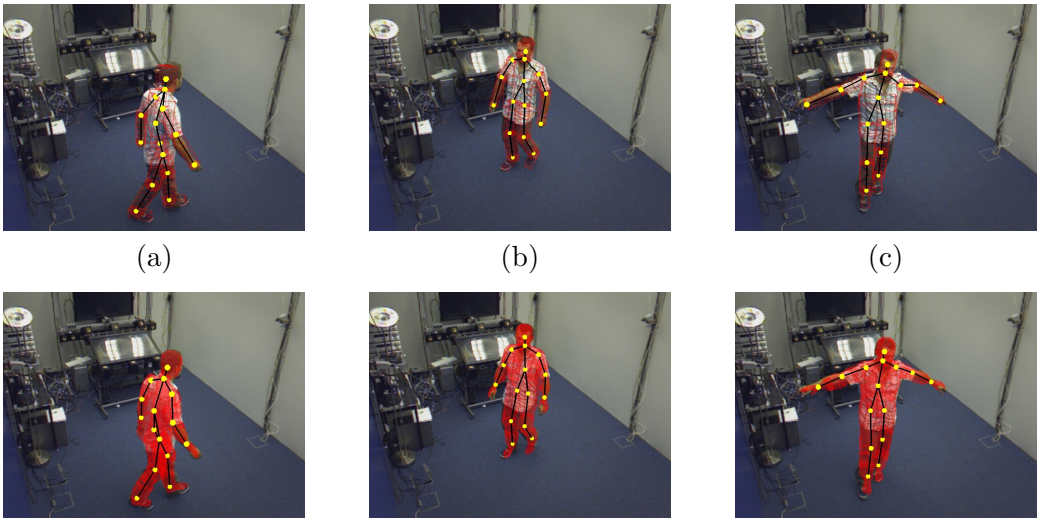



Figure 3.5: Tracking during a walking sequence using Powell's optimization. Images a, b and c show results using the cylindrical model. Images d, f, g show results using the deformable model.

Figure 3.5 shows the pose estimation for a walking sequence when using the cylindrical model or the deformable model. Note that the initialization phase is the same for both approaches. In this initialization phase the optimization process explained in section 3.1.1 obtains the shape

parameters of the deformable model, and this parameters are used to adjust the sizes of the cylinders and boxes of the cylindrical model. Limbs are approximately cylindrical, thus the shape of the limbs is very similar for both models. For this reason, the adjustment to the subject with respect to the limbs does not appear clearly different between both models. The torso shape differ more clearly between both representations. The boxes used for the torso and pelvis do not adjust well to the subject when adopting poses where there is a torso rotation. The deformable model fits better with the subject when the torso and pelvis rotate, and this rotations affect on the positions and rotations of the dependent joints in the kinematic structure. As we can see in the images of the figure 3.5, the shoulder joints position estimated is more precise when using the deformable model. This is mainly due to a better estimation of the torso position and rotation. Due to the hierarchical body structure, this difference on the shoulder positions causes a difference on the arms rotation angles, because both models adjust well the arms position with respect to its shoulder position.

The tracking system with the cylindrical model has lower computational cost than when using the deformable model. Using the Powell's method, the processing time for a single frame is $\backsim$ 2.5 seconds with the cylindrical model, and $\backsim$ 40 seconds with the deformable model.

We performed some tests combining both models. In this case, the idea is to benefit from the lower computational cost of the cylindrical model in a first estimation. Then, in a second step, we refine the estimation using the deformable model. The results obtained did not have enough precision. The reason is the lack of precision of the cylindrical model, as we described above, which is mainly focused on the upper nodes of the kinematic tree (pelvis and torso). When the joint angles are mapped to the deformable model, the pose obtained is not accurate enough, and the estimation with Powell's method do not refine the pose. Even though, this can be further studied, with other estimation techniques.

#### Powell's method and Annealed Particle Filter Comparison

We distinguish two main features related to the estimator method performance: the precision of the estimated pose and the ability to track fast movements relative to the frame rate. To analyze the performance of the Powell's method and the annealed particle filtering we adopt for both approaches the cylindrical model representation, mainly because the synthesis of the model has a lower computational cost.

Consider a sequence where the actor is moving relatively slow with respect to the frame rate, for example, a walking sequence at 25 frames/second. In this case, the Powell's method uses as starting point a pose very similar to the current frame pose, and the minimization process is able to fit the model to the image data with precision. In general, as the pose is similar from frame to frame, if the local minima achieved at the previous frame was good, at the current frame, the minimization process is able to remain also in a good local minima. In figure 3.5, we already showed an example of tracking at 25 frames/second with Powell's method.

On the other hand, the annealed particle filter estimates the pose with the weighted mean of the particle set. The precision of the estimation is very related to the discriminative ability of the cost function. The weighted mean will be precise if the particles representing an incorrect pose have a sufficiently low weight such that they do not disturb the particles close to the correct estimation. The cost function based on silhouettes that we use is not much discriminative. The reason is that silhouettes tend to be ambiguous from certain views, such that we cannot distinguish between a set of poses for a given silhouette. For example, when the actor adopts a pose with the arms in front of the body and the view is also frontal, using the silhouette we cannot distinguish the pose of the arms. The same apply in case of self-occlusions. In these situations, wrong poses may have high weights for certain views, and this will affect to the

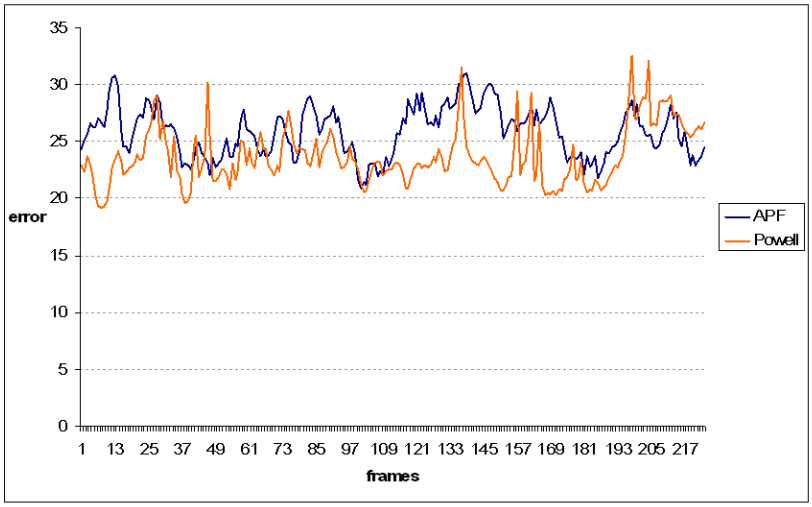



Figure 3.6: Reprojection error during a walking sequence at 25 frames/second.

weighted mean estimation, leading to a lack of precision.

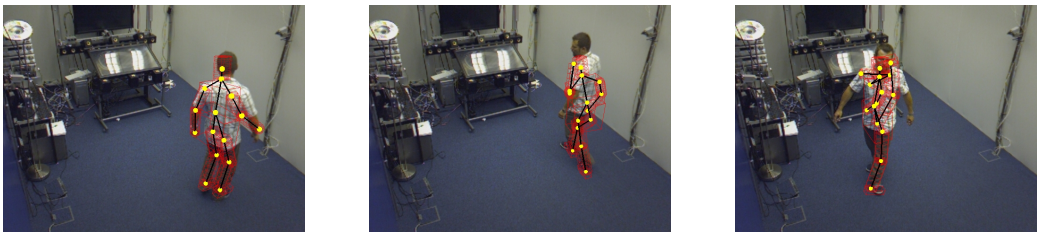
We do not have ground truth of our test sequences, thus we cannot measure the precision of the estimation of the joints positions or angles. We use, as indicator of the precision of the estimation, the error of reprojection of the model in the estimated pose with respect to the detected foreground silhouettes. We define this error for a camera $i$ as

$$\mathbf{error}_i = \frac{\text{XOR}(\text{model}, \text{foreground silhouette})}{\sharp \text{ of foreground pixels}} \tag{3.1}$$

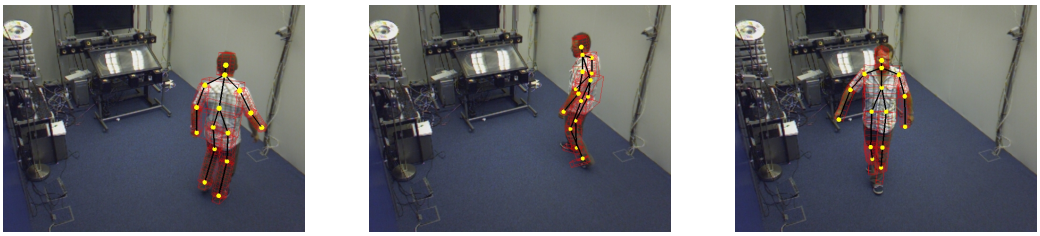
Thus, we compute this error with the XOR operation described in section 2.3 for each of the cameras, and then, we obtain the mean over all cameras. Figure 3.6 shows the evolution of this error during the walking sequence at 25 frames/second. We show the error when using the annealed particle filter and when using Powell's method, in both cases with the cylindrical model. In general, the error is lower for Powell's method, except at certain frames where the annealed particle filter tracks better. This corresponds with the observations obtained when printing the estimated joints over the image data, where we can determine also better precision of Powell's method.

When the movement relative to the image capturing frame rate is fast, it is more difficult to track the motion. In this kind of situation, the tracking system using Powell's method is not able to obtain correct poses. The poses between consecutive frames differ substantially, so the minimization process gets stuck in local minima as the sequence progress. At the end, this becomes an unrecoverable error. Figure 3.7 (a, b and c) show three frames of the walking sequence at a frame rate of 8 frames/second, with the results of the estimation using Powell's method. The system is not able to follow the movement, ending in a total failure.

The annealed particle filter performs an exploration of the pose space when it spreads the particle set. The range of the exploration is determined by the covariances of the Gaussian noise in the propagation step (section 2.2.2). In the case of fast movements we need a wider range of exploration, so higher covariances, to follow the movements from frame to frame. Also a higher range of exploration requires a greater number of particles, such that we have enough sampling density to compute proper measurements. The particle filter maintains a population



(a) (b) (c)



(d) (f) (g)

Figure 3.7: Tracking during a walking sequence at 8 frames/second. Images a, b and c show results using Powell's method. Images d, f, g show results using annealed particle filtering.

of particles along the sequence, so that wrong hypothesis are not directly ignored, and have a chance in successive frames. This fact provides robustness to failures, so the system can recover the track after committing estimation errors. In figure 3.7 (d, e and f) we show the results of the annealed particle filter estimation in the same sequence at 8 frames/second that Powell's method fails. A drawback of the particle filter estimation is that the poses shake from frame to frame during the sequence. This fact is a conjunction of the random nature of the sampling and that the particle weights are not enough discriminative. For example, consider a limb that remains static during several frames, the position of the limb is estimated close but slightly different at each frame, and the overall result is that the limb shakes during the sequence. This is not the case for the Powell's method because if the limb remains static the estimation will end at the same local minima.

The cost of the annealed particle filter estimation with 400 particles and 7 annealing layers is $\backsim$ 10 seconds for a single frame. Using the Powell's method, the cost is $\backsim$ 2.5 seconds. In these cases, we used the cylindrical model.

## 3.2 Conclusions

In this work, we analyzed the different components of a model-based motion tracking system, the model, the estimator and the likelihood function. We compared different instances of these components in order to infer its influence in the system results.

On one hand, we compared the cylindrical and the deformable human body models. The cylindrical body model has a low computational cost and the tracking of the positions of the limbs joints is precise. The model is not so precise in estimating the torso, head and shoulder positions, and this has an influence on the estimation of several angles. This is important for applications that need precision, such as ergonomics or bio-mechanical studies, so the cylindrical model may not be appropriate for such applications. Even though, the precision on the

estimation of the limbs positions might be enough for human-computer interaction applications.

The deformable model is more precise in the overall estimation but has a higher rendering cost. The movements captured have a higher realism, which is a valuable feature, for example, for realistic animation. Also, the model obtained is close to a 3D reconstruction of the actor, which makes it suitable for free viewpoint video applications. In free viewpoint video applications the goal is to obtain virtual views of the subject while he is in action. Usually for these applications, geometric and photometric information is used to build a 3D reconstruction of the subject, and then the reconstructed shape is rendered on the virtual view. In this cases, the shape obtained is restricted to the visual hull [Lau94], or the photo hull [KS00], which might not be accurate enough if the number of cameras available is not high. The deformable model can be an alternative, if we provide also a texturing technique that combines the images captured by the cameras.

On the other hand, we analyzed the estimation with Powell's method and the estimation with the annealed particle filter. We adopted a hierarchical approach for the deterministic optimization with Powell's method. The estimation is precise with relatively slow movements, but have the risk of getting stuck in a wrong local minima, which may lead to an unrecoverable error. The annealed particle filter is able to track faster movements, but with a higher computational cost. It is also more robust to errors, as it is able to recover the track after committing estimation errors. The annealed particle filter may be used in cluttered environments to track fast movements, while Powell's method is more precise and suitable to track the motion in controlled environments.

The cost function based on silhouettes is fast, and provides good information in an 8 cameras environment. Even though, it has ambiguities that are more significant as the number of cameras decreases. Besides, these ambiguities affect on the precision of the final estimation. Other image descriptors using color, motion or 3D information can increase the robustness and precision of the system.

The initialization phase is able to adjust the model to the general proportions of the specific individual. Even though, more precision may be required for certain applications, so the morphing to adjust the model to loose-fitting clothes should be further improved.

The system using the deformable model can be extended to track hand gestures, but this implies a dramatic increase of the dimensionality of the pose space, so it needs a further study of the estimation technique.

# Appendix A

# Camera Model

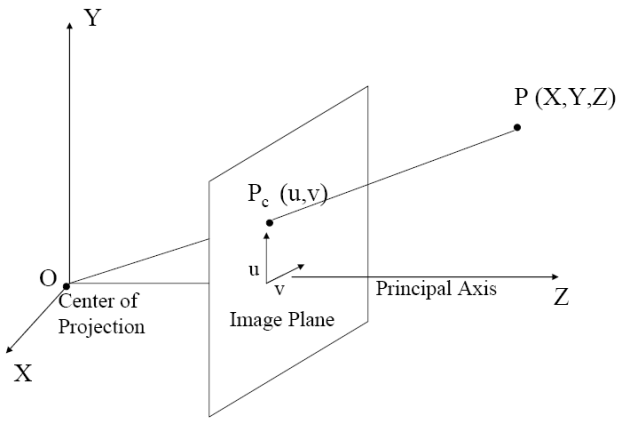



Figure A.1: Pinhole camera model.

In this section we describe the camera model used. Then, we discuss the relation of the camera parameters obtained after the calibration process, and the parameters we use to render the models using the OpenGL pipeline.

We model the cameras as pinhole cameras, where central projection of points in space into a plane (image plane) is considered. The camera calibration process recovers camera parameters from 2D images. We use an algorithm implemented in Matlab by Jean-Yves Bouguet [Bou04] to calibrate the cameras. These camera parameters determine the process of image formation and they describe the internal camera characteristics (*intrinsic parameters*) and the 3D position and orientation of the camera frame (*extrinsic parameters*).

Figure A.1 shows a camera with the center of projection $O$ and the principal axis parallel to $Z$ axis. Image plane is at focus and hence focal length $f$ away from $O$. A 3D point $\mathbf{P} = (X, Y, Z)$ is imaged on the camera's image plane at coordinate $\mathbf{P}_c(u, v)$. We will first find the calibration matrix which maps 3D $\mathbf{P}$ to 2D $\mathbf{P}_c$. We can find $\mathbf{P}_c$ using Thales theorem as

$$\frac{f}{Z} = \frac{u}{X} = \frac{v}{Y}$$

which give us

$$u = \frac{fX}{Z}$$

$$v = \frac{fY}{Z}$$

Using homogeneous coordinates for $\mathbf{P}_c$, we can write this as

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{A.1}$$

Note that this indeed generates the point $\mathbf{P}_c = (u, v, w) = (\frac{fX}{Z}, \frac{fY}{Z}, 1)$.

The origin of the 2D image coordinate system does not coincide with where the $Z$ axis intersects the image plane. So, we need to translate $\mathbf{P}_c$ to the desired origin. Let this translation be defined by $(t_u, t_v)$. Hence, now $(u, v)$ is

$$u = \frac{fX}{Z} + t_u$$
$$v = \frac{fY}{Z} + t_v$$

This can be expressed in a similar form as equation A.1 as

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & t_u & 0 \\ 0 & f & t_v & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{A.2}$$

Now, in equation A.2, $\mathbf{P}_c$ is expressed in distance units (i.e, cm). We need to express it in pixels, so we will need to know the resolution of the camera in pixels/cm. For a general case, we assume rectangle pixels with resolution $m_u$ and $m_v$ pixels/cm in $u$ and $v$ direction respectively. Therefore, to measure $\mathbf{P}_c$ in pixels, its $u$ and $v$ coordinates should be multiplied by $m_u$ and $m_v$ respectively. Thus

$$u = m_u \frac{fX}{Z} + m_u t_u$$
$$v = m_v \frac{fY}{Z} + m_v t_v$$

This can be expressed in matrix form as

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} m_u f & 0 & m_u t_u & 0 \\ 0 & m_v f & m_v t_v & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{A.3}$$

and we can write concisely the *intrinsic parameter matrix*, $\mathbf{K}$, as

$$\mathbf{K} = \begin{pmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.4}$$

which depends on the intrinsic parameters of the camera like the focal length or the principal axis.

Usually the camera does not have its center of projection at $(0, 0, 0)$ and is oriented in an arbitrary fashion. In a multicamera environment, we set a common *world coordinate system* for all the cameras. Then, we need a rotation and translation to transform the world coordinate system to the camera coordinate system, such that it coincide with the configuration in figure

A.1. Let the camera translation be given by $(T_x, T_y, T_z)$. The rotation applied to coincide the principal axis with the $Z$ axis is given by a $3 \times 3$ matrix $\mathbf{R}$. The transformation can be expressed by a matrix in homogeneous coordinates

$$\mathbf{E} = \begin{pmatrix} & & & T_x \\ & \mathbf{R} & & T_y \\ & & & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.5}$$

called the *extrinsic parameter* matrix.

Then $\mathbf{P}_c$, the projection of $\mathbf{P}$ is given by

$$\mathbf{P}_c = \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{E}\mathbf{P} \tag{A.6}$$

The camera calibration process provides the matrices $\mathbf{K}$ and $\mathbf{E}$. We are interested in using the recovered camera parameters from the real images to render the virtual models, such that we can compare the model projection with the image data. We use the OpenGL specification to render the models in order to exploit the graphics card capabilities. The OpenGL image formation pipeline is slightly different than the image formation pipeline that we described above, which is the one commonly used in computer vision. Thus, we need the correspondence between the parameters used in both pipelines. In this way, we will set for the virtual camera the projection transformation of the real camera.

We consider first the transformation from the world coordinate system to the camera coordinate system. From the calibration process, we obtain the matrix $\mathbf{E}$, that contains the information of the translation and rotation needed. In the OpenGL pipeline, the Modelview matrix ($\mathbf{M}_{mv}$) defines this transformation. The OpenGL specification defines the $Z$ axis in the opposite direction of the one used in the calibration process, shown in figure A.1. Thus, we set the Modelview matrix with the parameters of the extrinsics matrix $\mathbf{E}$, considering this distinct configuration. The matrix is

$$\mathbf{M}_{mv} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & T_x \\ -r_{21} & -r_{22} & -r_{23} & -T_y \\ -r_{31} & -r_{32} & -r_{33} & -T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.7}$$

where $r_{ij}$ and $(T_x, T_y, T_z)$ are the components of the rotation matrix $\mathbf{R}$ and the translation vector of the extrinsic matrix, equation A.5.

The equivalent of the intrinsics matrix $\mathbf{K}$ in the OpenGL pipeline is the Projection matrix ($\mathbf{M}_p$). In this case, the Projection matrix includes more information to define the far and near clipping planes. This planes limit the viewing frustum in the $Z$ axis, and they are not considered in a real camera. Another difference is the image coordinate system origin. The origin considered in the calibration process is at the top left corner of the image, while in the OpenGL pipeline is the bottom left corner. Taking this into account, the Projection matrix is set as

$$\mathbf{M}_p = \begin{pmatrix} \frac{\alpha_x}{w/2} & 0 & 1 - \frac{u_0}{w/2} & 0 \\ 0 & \frac{\alpha_y}{h/2} & \frac{v_0}{h/2} - 1 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2 \cdot f \cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{A.8}$$

where $f$ and $n$ are the far and near clipping planes, $w$ and $h$ are the image width and height.

We do not consider the distortion in the rendering pipeline. We use the distortion parameters obtained from the calibration algorithm to undistort the captured images in a preprocessing step.

# Bibliography


[AMG+02] MS Arulampalam, S. Maskell, N. Gordon, T. Clapp, D. Sci, T. Organ, and SA Adelaide. A tutorial on particle filters for online nonlinear/non-GaussianBayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[ASK+05] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: shape completion and animation of people. *Proceedings of ACM SIGGRAPH 2005*, 24(3):408–416, 2005.

[BC08] Luca Ballan and Guido Maria Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3DPVT*, Atlanta, GA, USA, June 2008.

[BEB08] J. Bandouch, F. Engstler, and M. Beetz. Accurate Human Motion Capture Using an Ergonomics-Based Anthropometric Human Model. In *Articulated Motion and Deformable Objects: 5th International Conference, Amdo 2008, Port D'andratx, Mallorca, Spain, July 9-11, 2008, Proceedings*, page 248. Springer, 2008.

[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.

[Bou04] J.Y. Bouguet. Camera calibration toolbox for matlab. 2004.

[BRM08] M. Bastioni, S. Re, and S. Misra. Ideas and methods for modeling 3D human figures: the principal algorithms used by MakeHuman and their implementation in a new approach to parametric modeling. In *Proceedings of the 1st Bangalore annual Compute conference*. ACM New York, NY, USA, 2008.

[BSB+07] AO Balan, L. Sigal, MJ Black, JE Davis, and HW Haussecker. Detailed Human Shape and Pose from Images. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pages 1–8, 2007.

[CFCP08] Cristian Canton-Ferrer, Josep R. Casas, and Montse Pardàs. Exploiting structural hierarchy in articulated objects towards robust motion capture. In Francisco J. Perales López and Robert B. Fisher, editors, *AMDO*, volume 5098 of *Lecture Notes in Computer Science*, pages 82–91. Springer, 2008.

[CMC+06] S. Corazza, L. Mündermann, A.M. Chaudhari, T. Demattio, C. Cobelli, and T.P. 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.

[CTMS03] J. Carranza, C. Theobalt, M.A. Magnor, and H.P. Seidel. Free-viewpoint video of human actors. *ACM Transactions on Graphics (TOG)*, 22(3):569–577, 2003.

[Cyb] Cyberware®. www.cyberware.com.

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

[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.

[Gav99] D.M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[GD96] DM Gavrila and LS Davis. 3-D model-based tracking of humans in action: a multi-view approach. In *1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96*, pages 73–80, 1996.

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

[JRMG07] D. Jacka, A. Reid, B. Merry, and J. Gain. A comparison of linear skinning techniques for character animation. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 177–186. ACM New York, NY, USA, 2007.

[KS00] K.N. Kutulakos and S.M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[Lau94] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Trans. PAMI*, 16(2):150–162, 1994.

[LC09] Adolfo López and Josep R. Casas. Feature-based annealing particle filter for robust body pose estimation. In Alpesh Ranchordas and Helder Araújo, editors, *VISSAPP (2)*, pages 438–443. INSTICC Press, 2009.

[LCF00] JP Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 2000.

[Mak] MakeHuman. www.makehuman.org.

[MG01] T.B. Moeslund and E. Granum. A Survey of Computer Vision-Based Human Motion Capture. *Computer Vision and Image Understanding*, 81:231–268, 2001.

[MI00] J. MacCormick and M. Isard. Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 3–19. Springer-Verlag London, UK, 2000.

[MTHC03] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–223, 2003.

[PF01] R. Plankers and P. Fua. Tracking and modeling people in video sequences. *Computer Vision and Image Understanding*, 81(3):285–302, 2001.

[PFTV92] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical recipes in C: the art of scientific computing*. 1992.

[Pic04] M. Piccardi. Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 4, 2004.

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

[Sol] RAMSIS Community. Human Solutions. www.ramsis.de.

[SRIC01] P.P.J. Sloan, C.F. Rose III, and M.F. Cohen. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143. ACM New York, NY, USA, 2001.

[TCMS03] Christian Theobalt, Joel Carranza, Marcus A. Magnor, and Hans-Peter Seidel. A parallel framework for silhouette-based human motion capture. In Thomas Ertl, editor, *VMV*, pages 207–214. Aka GmbH, 2003.

[VBMP08] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3):1–9, 2008.

[XLP05] L.Q. Xu, JL Landabaso, and M. Pardas. Shadow removal with blob-based morphological reconstruction for error correction. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05)*, volume 2, 2005.