CHALMERS

Human Interaction Solutions for Intuitive Motion Generation of a Virtual Manikin

Master's thesis in Applied Mechanics

LUCA CALTAGIRONE

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2014
Master's thesis 2014:60

MASTER'S THESIS IN APPLIED MECHANICS

Human Interaction Solutions for Intuitive Motion Generation of a Virtual Manikin

LUCA CALTAGIRONE

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2014

Human Interaction Solutions for Intuitive Motion
Generation of a Virtual Manikin
LUCA CALTAGIRONE

Master's thesis 2014:60
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Intelligently Moving Manikins (IMMA). CAD models courtesy of Volvo Cars.

Chalmers Reproservice
Göteborg, Sweden 2014

Human Interaction Solutions for Intuitive Motion
Generation of a Virtual Manikin
Master's thesis in Applied Mechanics
LUCA CALTAGIRONE
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

The aim of this work was to develop a motion capture algorithm for the Kinect sensor, which can provide robust and real time tracking, even in those situations where the Kinect algorithm performs poorly. The proposed method belongs to the family of model based algorithms which work by establishing correspondences between an acquired point cloud and a custom-built body model. Specifically, an extension to articulated bodies of the point cloud registration algorithm known as iterative closest point (ICP) was used in combination with the Gauss-Newton minimization algorithm.

The virtual manikin IMMA (Intelligently Moving Manikins) allows for the conduction of ergonomic studies in a simulated assembly line. To perform motion verification of the solutions found by the simulation software, a motion capture system could be integrated within its platform. Furthermore, this could facilitate the analysis of some complex assembly operations which may be troublesome to solve with the current version of IMMA. Most of the available commercial devices are expensive, difficult to operate, and require specialized equipment and software. However, in recent years, Microsoft has introduced to the market an RGB-D camera, the Kinect 1.0, which provides 3D information without the need for triangulation, and furthermore integrates a sophisticated motion tracking system based on machine learning algorithms. Unfortunately, this system performs rather poorly in the settings commonly found in assembly lines where self-occlusions and occlusions are commonplace.

By measuring the normalized residual error per point (NREP), one can determine how well the articulated iterative closest point (AICP) system matches the body model to the point cloud acquired from the sensor. The obtained results show that the AICP system is more robust than the Kinect algorithm by producing an NREP approximately seven times smaller on average relative to a set of 30 unconstrained motion sequences involving occlusions and self-occlusions. Furthermore the developed system allows the tracking of a wider range of motions in an extended range. This makes it a potentially better solution for performing motion capture in assembly lines.

Keywords: Computer vision, RGB-D sensor, Gauss-Newton algorithm, articulated ICP, nearest neighbor search

Acknowledgements

I would like to thank my supervisors Niclas Delfs, Peter Mårdberg, and Fredrik Ekstedt for the stimulating discussions and very useful advice. I am grateful to FCC for allowing me to work on this project and providing me with the necessary equipment and space. I would also like to thank my examiner Mattias Wahde for his feedback and guidance. Last but not least, my special thanks goes to my beloved wife for her invaluable help, support, and patience.

Contents

1 Introduction

1.1 Background

One of the programs developed at the Fraunhofer-Chalmers Research Centre for Industrial Mathematics (FCC), in collaboration with its industrial partners and academia, is the virtual manikin IMMA (Intelligently Moving Manikins) (see Figure 1.1), which allows for the conduction of ergonomic studies in a simulated assembly line. This is accomplished by optimizing the comfort function during the execution of a simulated assembly operation. To verify the solutions found by the simulation software, a motion capture system could be used to track the movements of a real worker performing the same operation. On the market there are many available systems, but most of them require expensive equipment, long installation time, and specialized software. Another option is to utilize the relatively cheap sensor produced by Microsoft, the Kinect 1.0, which incorporates a color camera and a depth sensor, thus providing 3D information without the calibration and triangulation phases which are required by optical systems. The Kinect is also equipped with a motion capture system; unfortunately, however, it is mainly designed for playing video games and controlling external devices remotely, and therefore it is not so sensitive to finer movements and it performs rather poorly when motions involve self-occlusions or occlusions.

1.2 Purpose

The goal of this thesis is to design and develop a motion capture system that can provide robust and real time tracking, even in those situations where the Kinect algorithm performs poorly, while at the same time maintaining its simplicity of use. The main focus is on motions involving self-occlusions, given that they commonly occur while performing manual work, and occlusions caused by objects or people, another common situation in assembly lines. Furthermore, the system should be easily extendable to perform motion capture with multiple Kinects.

1.3 Outline

In Chapter 2 a brief introduction to the field of motion capture is given, presenting the various technologies available, their advantages, and their limitations. Afterward, the features of the Kinect sensor and its proprietary skeleton tracking algorithm are described. Finally, in the last section, an overview of different methods used in computer vision for motion capture is presented. Chapter 3 covers the theoretical background of this work. In the first section, a description of how to mathematically model a skeleton and its corresponding body surface is provided, followed by an introduction of the point cloud registration method called ICP in both its standard version and in its generalization to articulated bodies (AICP). Chapter 4 presents a detailed description of all the phases of the AICP motion capture system: data acquisition and sub-sampling, subject segmentation, body model generation, and point cloud - body model matching. In Chapter 5 the results are shown and the AICP and Kinect capture system performances are compared. In Chapter 6 a final discussion, the conclusions, and future work are presented.

1.4 Scope and limitations

The implementation of a machine learning system to perform motion capture would have required a large amount of time and resources, particularly the phase of collecting a representative training set; therefore the scope of this work was limited to a more classical algorithm that works by model fitting through dense correspondences. Given that the system was intended to work on consumer hardware and in real time, the motion capture was restricted to only one subject in order to reduce the computing resources needed. Another important requirement was simplicity of use, therefore the option of wearing markers or special suits to help the tracking process was not considered.

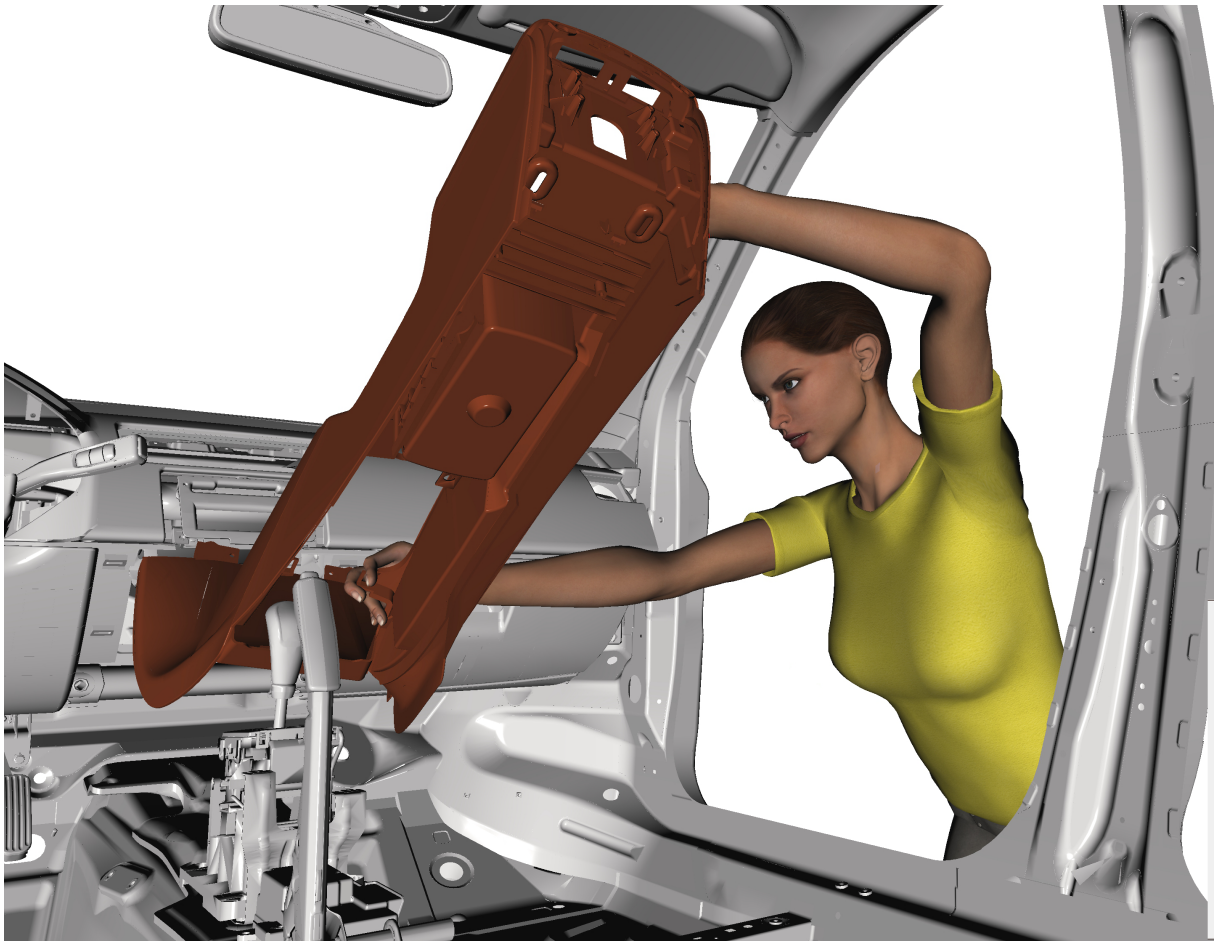

Figure 1.1: *Assembly operation performed by a virtual manikin within IMMA software. CAD models courtesy of Volvo Cars.*

2 Motion capture

Motion capture, as the name suggests, is the process of recording patterns of movement. Its applications include computer animation in the movie and video-game industries [1, 2], human-machine interaction [3, 4], surveillance [5], and entertainment [6]. In the following, a brief overview of the most common systems used to perform human motion capture is given.

Optical systems

Optical systems make use of multiple color cameras to obtain 2D images, which are then combined, by means of triangulation, to recover 3D information. The number of cameras used depends on the specific application but it can range from a minimum of two, up to 40 or more, with a refresh rate that can reach 1000 frames per second [7]. Generally the subject wears markers over his body, in specific locations, to facilitate the identification of landmarks. There exist two kinds of markers: passive ones, which are simply reflective objects, and active ones, which are light-emitting diodes (LED). By setting a brightness threshold for the cameras, it is then possible to exclude all other objects which do not emit or reflect as much light. The passive markers do not require using cables as the the active ones do, however the active markers allow for larger capture volumes. These systems necessitate long set-up times and can only be used indoors.

Inertial systems

Inertial systems make use of miniature inertial sensors to determine a subject's movements. An example of this system is the one developed by Xsens [8] that consists of 17 sensors, each of which comprises 3D gyroscopes, accelerometers, and magnetometers. After a set-up and calibration phase during which body dimensions and body-to-sensor alignment is performed, the system determines motion information by combining data fusion algorithms and bio-mechanical models. The basic working principle behind this system is to determine the angular velocities of the various body segments, which are then integrated to determine the rotation angles and consequently the segments' orientations. These systems only provide relative motion information, so external devices are needed to obtain absolute positioning. As opposed to optical systems, they are not affected by lighting conditions or occlusions, and are much more portable.

Mechanical systems

In this case, the relative position of a subject is directly measured by wearing an exoskeleton [9]. Each joint is equipped with a potentiometer whose resistance changes as a function of the rotation angle; by measuring the voltage at its terminals, it is then possible to determine the joint angle. Just like inertial systems, mechanical systems are not affected by occlusions and offer unlimited capture volume.

2.1 Kinect sensor

All the above systems make use of special hardware and software, and are generally expensive and complex to set up and operate. However, in recent years, with the introduction to the market of cheap RGB-D (color and depth) sensors, a lot of interest has been directed at developing marker-less motion capture systems. The Kinect sensor was first released by Microsoft in 2010 to allow the control of the Xbox video-game console by using gestures and spoken commands. It is equipped with a color camera, a depth sensor, and a microphone array. The depth sensor consists of an infrared laser projector combined with an infrared camera; depth data is obtained by analyzing how a known pattern of infrared light, emitted from the projector, is deformed by the environment in the sensor's field of view; a technique known as structured light[10]. Its depth and color resolutions are respectively 320x240 and 1024x620 pixels with a refresh rate of 30 frames per second. One of the main features of the Kinect sensor is its skeleton tracking system which can capture the motion of up to two people simultaneously.

The algorithm that performs the skeleton tracking has been developed by Shotton *et al.* [11]. It consists of a deep randomized decision forest classifier[1] trained with hundreds of thousands of images covering a wide spectrum of body types, positions, and clothing. The classifier associates each pixel with a body part and this information is then pooled to generate a proposal for the positions of 3D skeletal joints. The performance of this system is rather impressive; it runs at 30 frames per second on consumer hardware with a good level of accuracy, it does not need an initialization phase, and it recovers quickly from tracking failures. However, this applies only in specific settings: the user should be facing directly toward the sensor, or with limited deviation; no objects should be in the field of view between the user and the sensor; and no objects should be handled such as holding tools. Another strict requirement is that at least the full upper body should be visible. If one or more of the these requirements is not met, the motion tracking can become very poor or fail altogether.

2.2 Motion capture algorithms

Motion capture and analysis is a very active research field and many approaches have been developed to perform this task [13]. The algorithm implemented by Shotton *et al.* [11] belongs to the family of learning based methods. These algorithms directly map the data acquired from a sensor (i.e., color camera, RGB-D sensor, etc.) to the joint space by means of a classifier. As previously mentioned, they can perform rather well but they also have some important limitations: the breadth of the training set restricts the range of motions that can be accurately tracked, adding new motions is time consuming because it involves re-training the classifier and acquiring tracking information for the training set by other means, and the motion tracking cannot incorporate temporal and kinematic coherence.

To avoid these limitations, model based methods could be used instead [14, 15, 16, 17]. In this case, a model of the subject being tracked is created during an initialization phase and then matched to the data to determine the subject's position. These algorithms can, in principle, track all possible motions, they usually incorporate constraints such as bone lengths and joint angles limits in the body model, and they can make use of temporal information to facilitate tracking during brief occlusions or failures of the matching algorithm. On the downside, the need to generate a body model makes these systems less user-friendly and slower to initiate. Additionally, the dense correspondences fitting needed for pose estimation is computationally demanding, which limits their use in real time applications, and recovery from tracking failures is generally more difficult.

A combination of learning and model based algorithms is, of course, possible; for example, in [18], the authors developed an adaptive body model that enforces kinematic constraints and eliminates odd poses to complement the OpenNI [19] motion tracking algorithm. Another approach is to detect key-points such as anatomical landmarks on the image (or depth map) and then use inverse kinematics to determine the joint orientation. Zhu and Fujimura [20] made use of a deformable template to detect the head, neck, and trunck of a subject; Schwarz *et al.* [21] built a graph-based representation of the depth data in order to measure geodesic distances, allowing them to localize body landmarks. Other interesting approaches have been developed but a complete review is beyond the scope of this work; for more details we refer the interested reader to [13].

[1] The basic component of a random forest classifier is a decision tree. A decision tree is a tree-shaped structure with nodes, branches, and leaves. Each node represents a feature and each branch leaving that node corresponds to one of its possible values. The classification process consists of visiting all nodes until a leaf, with its associated class label, is reached. A random forest is an ensemble of decision trees, each of which is trained using only a randomly chosen subset of features. For a detailed description see [12].

3 Theory

The approach adopted in this work to perform motion capture belongs to the family of model based algorithms. As briefly mentioned in the previous chapter, these algorithms work by establishing and then minimizing correspondences between a point cloud and a body model. The first part of this chapter illustrates in detail the components of the implemented body model; in the second part, a description of the algorithm used to align the body model with a point cloud is presented.

3.1 Body model

A body model is an abstract and simplified representation of a human body. Its basic components are a skeleton and a body surface.

3.1.1 Skeleton model

Just like a real skeleton, a model skeleton is composed of bones (links) and joints connecting them and allowing motion. The considered model consist of 16 links and 16 joints. Furthermore, a set of 11 body landmarks placed in correspondence with anatomical areas of interest is defined. The links are segments of fixed length, with no mass or volume. Joints are modeled as mathematical points with only rotational degrees of freedom. Each joint j_i is associated to a cartesian coordinate system $\mathfrak{R}_i$, x_i-y_i-z_i, that, by convention, has the z_i-axis oriented in the same direction as the link extending from it (see Figure 3.1). The state of joint j_i is then fully defined by the Tait–Bryan angles $(\alpha_i, \beta_i, \gamma_i)$, where $i = 0, \dots, 15$, indicating the amount of rotation about the x_i, y_i, and z_i-axes, respectively. In this work, rotations are applied sequentially to the x_i, y_i, and z_i-axes, in that order (x_i-y_i'-z_i''). These basic rotations can be expressed in matrix form as:

$$\mathcal{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{3.1}$$

$$\mathcal{R}_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{3.2}$$

$$\mathcal{R}_z(\gamma) = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

As Figure 3.2 illustrates, the skeleton can be organized in a tree structure which clarifies the connections among its elements. Multiple joints can have the same parent, therefore the operator $\Lambda(\cdot)$ [16] is introduced, which takes as input the index of a joint and returns the index of its parent. Furthermore, the operator $\Xi(\cdot)$ is defined; this takes as input the index of a body landmark and returns the index of its parent joint. Rigid body transformations can be utilized to perform transformations between coordinate systems. Given that reflections change the right-hand axis convention to a left hand, only transformations belonging to the Special Euclidean group $SE(3)$ are considered. Each element in $SE(3)$ is defined by a pair $(\mathcal{R}, \mathbf{t})$, where $\mathcal{R}$ is a rotation followed by a translation $\mathbf{t}$, so that $x \mapsto \mathcal{R}x + \mathbf{t}$. Two transformations can be combined as follow:

$$(\mathcal{R}_3, \mathbf{t}_3) = (\mathcal{R}_2, \mathbf{t}_2) \bullet (\mathcal{R}_1, \mathbf{t}_1) = (\mathcal{R}_2\mathcal{R}_1, \mathcal{R}_2\mathbf{t}_1 + \mathbf{t}_2) \tag{3.4}$$

In robotics, it is common to utilize homogeneous coordinates in order to unify the description of rotations and translations [22]. This requires augmenting the space from $\mathbb{R}^3$ to $\mathbb{R}^4$ and appending a 1 at the end of all vectors. The rotation $\mathcal{R}$ and the translation $\mathbf{t}$ can then be combined in a single matrix as follow:

$$(\mathcal{R}, \mathbf{t}) \mapsto \begin{pmatrix} \mathcal{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.5}$$

Given that $\mathcal{R}$ is an orthogonal matrix (i.e., $\mathcal{R}^{-1} = \mathcal{R}^T$), the inverse transformation of a rigid transformation in homogeneous coordinates can be easily determined:

$$\begin{pmatrix} \mathcal{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathcal{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathcal{R}\mathcal{A} & \mathcal{R}\mathbf{b} + \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbb{1} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \iff \mathcal{A} = \mathcal{R}^T, \mathbf{b} = -\mathcal{R}^T\mathbf{t} \tag{3.6}$$

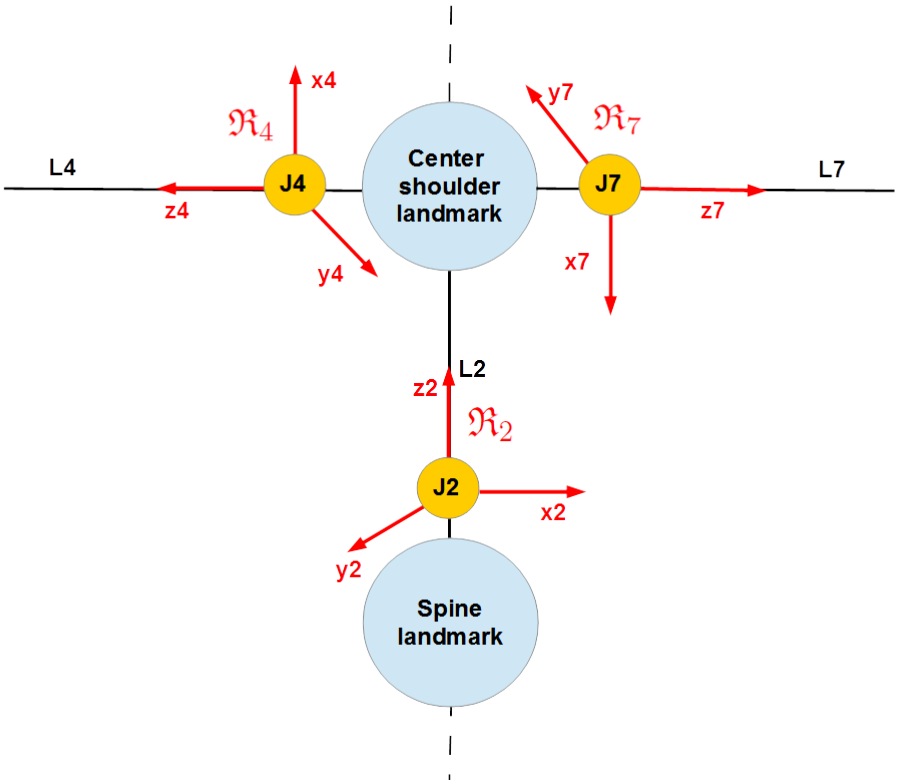

Figure 3.1: *Local coordinate systems. Each joint is associated to a local coordinate system with the z-axis oriented in the same direction of the link extending from it.*

Furthermore, it is straightforward to verify that relation 3.4 is still valid; in fact by multiplying two rigid transformations it follows that:

$$\begin{pmatrix} \mathcal{R}_2 & \mathbf{t}_2 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathcal{R}_1 & \mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathcal{R}_2\mathcal{R}_1 & \mathcal{R}_2\mathbf{t}_1 + \mathbf{t}_2 \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.7}$$

According to mapping 3.5, the basic rotation matrices in homogeneous coordinates can be redefined as follow:

$$\mathcal{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

$$\mathcal{R}_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.9}$$

$$\mathcal{R}_z(\gamma) = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

Furthermore, it is possible to define a general translation matrix $\mathcal{D}$ as:

$$\mathcal{D}(x, y, z) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

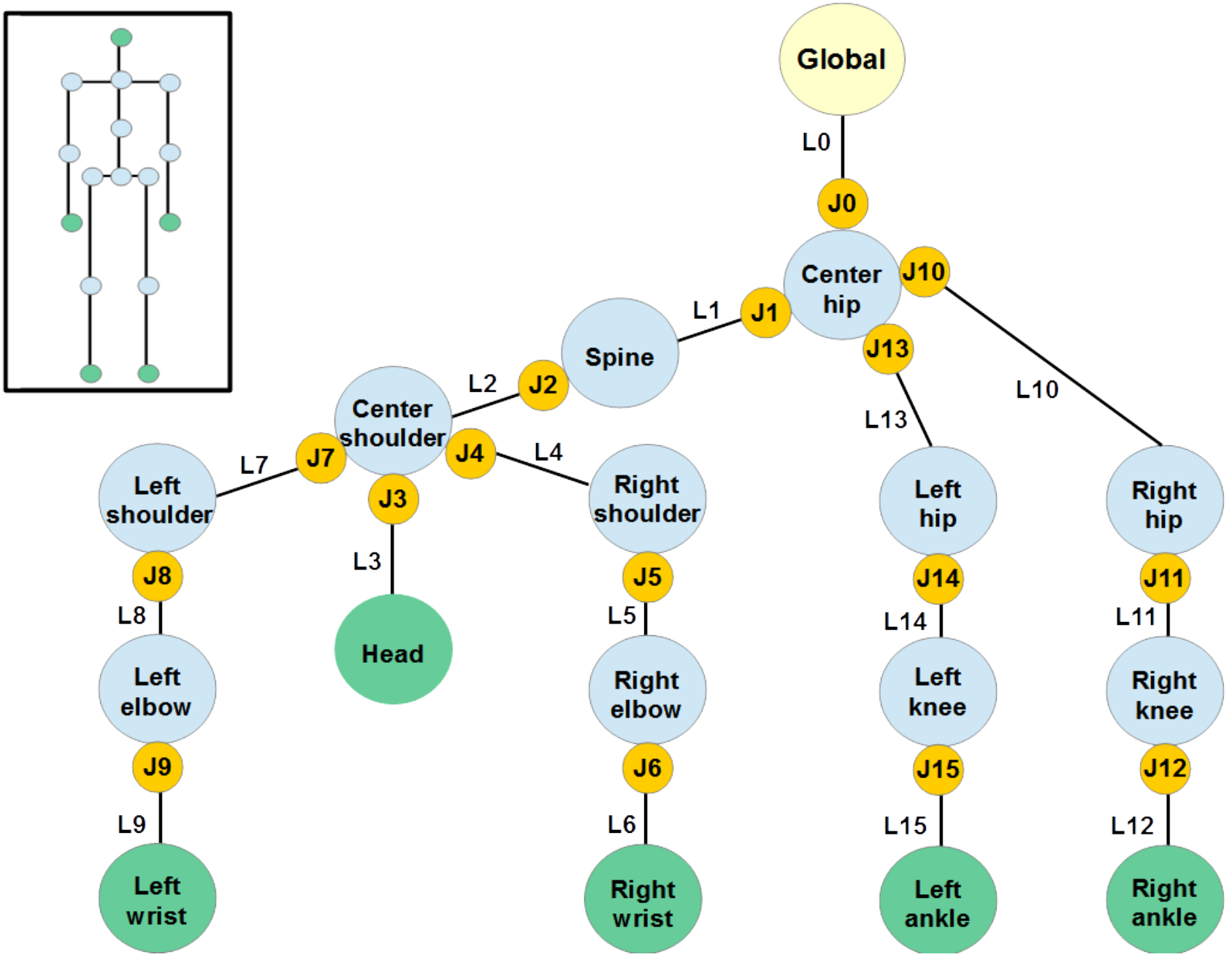

Figure 3.2: *Tree structure of the skeleton model. The global node represents the world coordinate system, while the remaining nodes represent the body landmarks tracked during motions. Links and joints are depicted as solid lines and smaller circles, respectively. The top left figure shows the skeleton model in its default state, that is, when all joint angles are zero.*

A homogeneous rigid transformation $\mathcal{M}_i$ is then associated to each joint j_i, expressed as a combination of the basic rotations and translation matrices previously defined, establishing the coordinate transformation from $\mathfrak{R}_i$ to $\mathfrak{R}_{\Lambda(i)}$. Moreover, transformation matrices $\mathcal{M}_i^W$ are defined as:

$$\mathcal{M}_i^W = \mathcal{M}_0 \cdot \ldots \mathcal{M}_{\Lambda(i)} \cdot \mathcal{M}_i \tag{3.12}$$

establishing the transformations from $\mathfrak{R}_i$ to the world coordinate system.

Remark

Given that the lengths of the links are set at initialization time and do not change afterwards, the variables determining the skeleton configuration are only the joint angles and link $\mathbf{l}_0$ representing the displacement from the world coordinate system. These can be arranged in a single variable $\mathbf{k} = [\boldsymbol{\Theta}_0, \ldots, \boldsymbol{\Theta}_{15}; \mathbf{l}_0]$, where $\boldsymbol{\Theta}_i = [\alpha_i, \beta_i, \gamma_i]$.

3.1.2 Body surface

The Kinect sensor only provides information about the surface of the body, therefore it is necessary to build a surface model to relate the skeleton to the point cloud. For this purpose, a total of 11 major body parts are identified: upper and lower arm (left and right), upper and lower leg (left and right), head, chest, and abdomen, which need to be generated and connected to specific links of the skeleton.

Link	**Body part name**	**Body part label**
L1	Abdomen	B1
L2	Chest	B2
L3	Head	B3
L5	Right upper arm	B4
L6	Right lower arm	B5
L8	Left upper arm	B6
L9	Left lower arm	B7
L11	Right upper leg	B8
L12	Right lower leg	B9
L14	Left upper leg	B10
L15	Left lower leg	B11

Table 3.1: Correspondences between skeleton links and body parts.

As Table 3.1 illustrates, some of the links do not have any associated body part. For this reason the operator $\Gamma(\cdot)$ is introduced, which takes as input the index of a body part and returns the index of the link it is connected to. Each B_m is a set of 3D points defined in the local frame of reference of the link they are attached to. How to generate them will be described in detail in the next chapter. To obtain the body parts in world coordinates, the transformation matrices $\mathcal{M}_i^W$ can be utilized. Then it follows that:

$$B_m^W = \mathcal{M}_{\Gamma(m)}^W B_m \tag{3.13}$$

The full body surface B^W is then obtained by joining all the subsets B_m^W:

$$B^W = (\mathcal{M}_{\Gamma(1)}^W B_1) \cup (\mathcal{M}_{\Gamma(2)}^W B_2) \cup \ldots \cup (\mathcal{M}_{\Gamma(11)}^W B_{11}) = \bigcup_{i=1}^{11} \mathcal{M}_{\Gamma(m)}^W B_m \tag{3.14}$$

3.2 Iterative closest point

As previously mentioned, model based algorithms perform motion capture by matching a body model of the subject with the data acquired from a sensor. To perform this task, a modified version of a popular point registration algorithm known as ICP has been implemented. In the following sections, an introduction to the ICP algorithm in its standard version will be introduced, followed by its generalization to articulated bodies.

3.2.1 Standard ICP

The ICP is an algorithm employed to minimize the difference between a target point cloud, $T = \{t_1, \ldots, t_n\}$, which is kept fixed, and a source point cloud, $S = \{s_1, \ldots, s_m\}$, which is iteratively transformed through rigid transformations. Let us describe the algorithm's steps:

1. For each point in the source point cloud, find the closest point in the target point cloud. This set of points is referred to as $C = \{c_1, \ldots, c_m\}$.

2. Estimate the rigid transformation $\mathcal{M}(\alpha, \beta, \gamma, x, y, z) = \mathcal{D}(x, y, z)\mathcal{R}_z(\gamma)\mathcal{R}_y(\beta)\mathcal{R}_x(\alpha)$ that minimizes the least-squares cost function E:

$$E(\alpha, \beta, \gamma, x, y, z) = \sum_{i=1}^{m} \|\mathcal{M}s_i - c_i\|^2 \tag{3.15}$$

3. Transform the source points by applying M. That is:

$$S \to \mathcal{M}S \tag{3.16}$$

4. Repeat steps (1,2,3) until the stopping criterion is met (for example, when the residual of the cost function is below fixed threshold)

3.2.2 Articulated ICP

The standard ICP algorithm is applied to point clouds that can be registered through a single rigid transformation (i.e., point clouds representing rigid bodies), in which case convergence to a local minimum and closed-form solution of the minimization problem in (3.15) are guaranteed [23]. The process of human motion capture, on the other hand, requires registering point clouds with an underlying structure of an articulated body, which can undergo multiple and simultaneous rigid transformations. For this reason, a procedure inspired by the standard ICP, which can be extended to the case of articulated bodies [16, 17, 24, 25], has been developed.

Let us consider a body model with its associated body parts $B = \{B_1, B_2, \ldots, B_{11}\}$, world transformations $\mathcal{M}^W = \{\mathcal{M}^W_1, \ldots, \mathcal{M}^W_{15}\}$, and configuration $\mathbf{k}_0$. As before, let us call T the target point cloud. Then the algorithm proceeds in the following steps (see Figures 3.3, 3.4, 3.5, 3.6):

1. Compute body parts in world coordinates, $B^W_m = \mathcal{M}^W_{\Gamma(m)}(\mathbf{k}_0)B_m$, for $m = 1, \ldots, 11$.

2. Partition T by finding closest body parts, $T = \{T_1, \ldots, T_{11}\}$ where $T_m = \{t^m_1, \ldots, t^m_{l_m}\}$.

3. For each T_m compute the set $C_m = \{c^m_1, \ldots, c^m_{l_m}\}$ of its closest neighbors in the body model surface B^W. Then determine the corresponding set in local coordinates, $C^*_m = \{c^{*m}_1, \ldots, c^{*m}_{l_m}\}$.

4. Find the configuration $\mathbf{k}^\dagger$ which minimizes the cost function E:

$$E(\mathbf{k}) = \frac{1}{2}\sum_{m=1}^{11} \left\| \mathcal{M}^W_{\Gamma(m)}(\mathbf{k})C^*_m - T_m \right\|^2 = \frac{1}{2}\sum_{m=1}^{11}\sum_{j=1}^{l_m} \left\| \mathcal{M}^W_{\Gamma(m)}(\mathbf{k})c^{*m}_j - t^m_j \right\|^2 \tag{3.17}$$

5. Update the body model configuration, $\mathbf{k}_0 \to \mathbf{k}^\dagger$.

6. Repeat steps (1-5) until stopping criterion is met.

Remarks

The cost function can be rewritten in vector form by rearranging its terms in a column vector, $\mathbf{r}(\mathbf{k})$, so that:

$$E(\mathbf{k}) = \frac{1}{2}\mathbf{r}(\mathbf{k})^T\mathbf{r}(\mathbf{k}) \tag{3.18}$$

Step 3 of the AICP algorithm requires performing a nearest neighbor search. A simple but rather slow approach to solve this task would be to compute all the mutual distances between target and source point cloud, and then

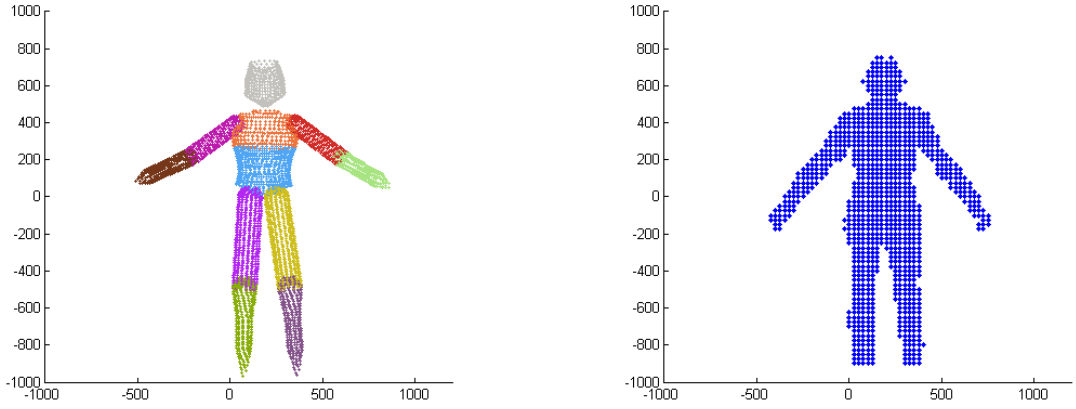

Figure 3.3: *Step 1 of AICP: calculation of the body model in world coordinates. Left panel: body model with color coded body parts, B_i^W. Right panel: target point cloud, T.*

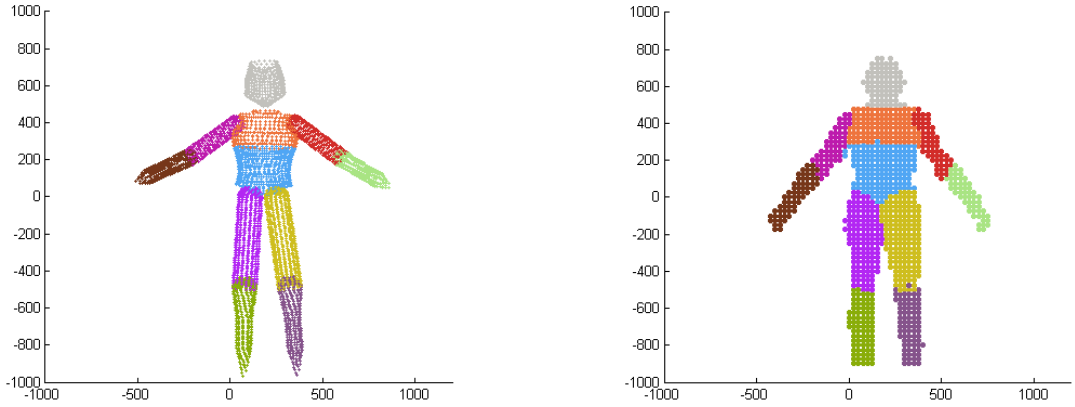

Figure 3.4: *Step 2 of AICP: partitioning of the target point cloud. Left panel: body model with color coded body parts, B_i^W. Right panel: the target point cloud has been partitioned by finding closest body part in the body model.*

to select the pairs of points that have the smallest relative distance. Another approach, which usually allows for a faster search, is to build and query a data structure, such as a k-d tree. For this purpose an external library called ANN [26] was utilized.

The problem in step 4 does not have a closed-form solution as in the case of the standard ICP, so another minimization method should be employed. The chosen algorithm is the well known Gauss–Newton algorithm (see Appendix B), which approximates a given function with its second order Taylor expansion, and iteratively corrects an initial guess $\mathbf{k}_0$ with the following update rule:

$$\mathcal{J}_0^T \mathbf{r}_0 = -\mathcal{J}_0^T \mathcal{J}_0 (\mathbf{k} - \mathbf{k}_0) \tag{3.19}$$

where $\mathbf{r}_0 = \mathbf{r}(\mathbf{k}_0)$ and $\mathcal{J}_0$ is its Jacobian matrix.

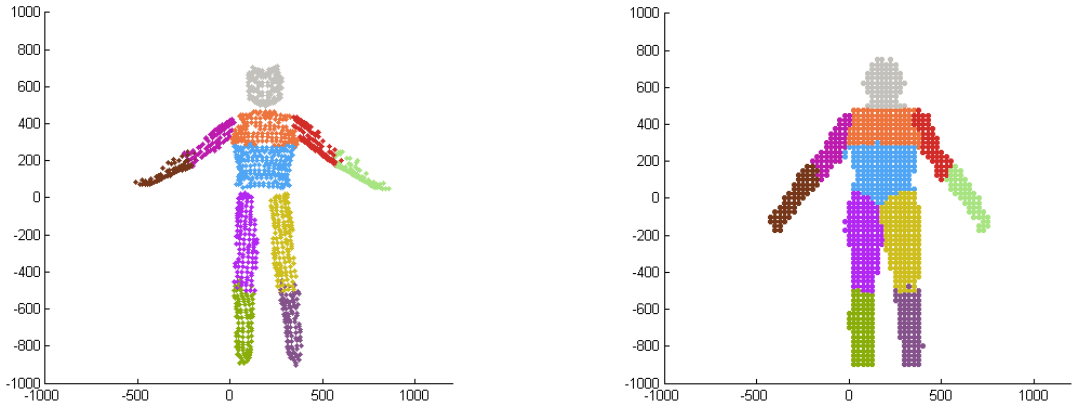

Figure 3.5: *Step 3 of AICP. By performing a nearest neighbor search, correspondences between the body model and the target point cloud are established. The left panel shows the target sets, T_i. The right panel shows the source sets, C_i.*

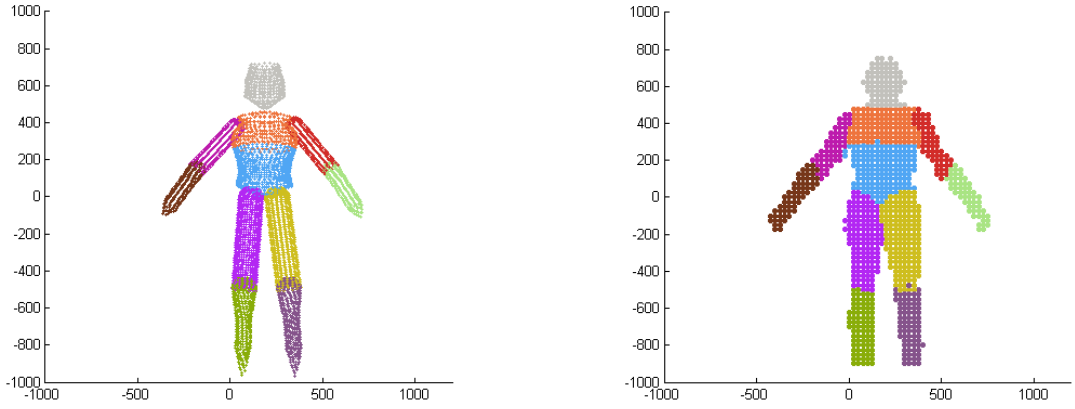

Figure 3.6: *Body model and target point cloud after execution of AICP. The left panel shows the body model in its updated configuration. The right panel shows the target point cloud.*

4 Method

In this chapter, all the stages of the AICP motion capture system will be presented and detailed. As will become clear, the process of gaining high-level information, such as the position of the subject's body landmarks, from a partial 3D surface reconstruction involves the interplay of many components.

4.1 Data acquisition

As mentioned in Chapter 2, the Kinect is equipped with a depth sensor and a color camera. Our system only makes use of the depth information from which a point cloud of the scene is obtained. The depth map resolution is 320x240 pixels; it is important to point out that the depth accuracy decreases with distance from the sensor. The Kinect for Windows has two working modalities: near mode and default. In near mode, the distance range is between 0.4 and 3.0 meters, while in default mode it is between 0.8 and 4.0 meters. This limits the volume of space where a subject's full-body can be tracked to a narrow region. On the software side, two options are available for accessing the Kinect hardware: the opensource APIs OpenNI [19] and the proprietary Microsoft SDK. The chosen approach was to utilize the Microsoft SDK, which offers better support and functionality.

4.2 Sub-sampling

Given that the depth map is a two-dimensional grid, a simple procedure to obtain a uniform sub-sampling of factor k is to store every k^{th} pixel while iterating through its elements (see Figures 4.1 and 4.2).

4.3 Subject segmentation

The point cloud at this stage is a representation of the full scene in the field of view of the Kinect. Therefore, beside the points belonging to the body surface of the subject, there are usually present a large number of points associated with the floor, walls, and other objects, which are unnecessary for the motion capture and need to be removed (see Figure 4.1). The chosen method to perform subject segmentation utilizes the information provided by the Kinect SDK in the depth map, where each pixel is a structure with two fields, one storing the depth in millimeters, and the other the player index. By knowing the player index of the subject, it is then possible to select only his associated pixels. To determine the subject's player index, the positions of all the skeletons found by the Kinect tracking system are checked and the one that is closest to the subject is chosen. Their relative distance, however, should be smaller than a threshold (set to 0.5 meters); this is to avoid the body model being captured by another person in the field of view of the Kinect should the subject leave the scene momentarily.

The above segmentation procedure fails entirely when the subject is farther than four meters from the Kinect, and has some problems during occlusions. Therefore, in those situations, a different segmentation algorithm is used. This method stems from the consideration that two consecutive frames will generally be very similar to each other (assuming that the subject is not moving too quickly) and therefore the previous skeleton position will be rather close to the current subject's point cloud. Under this assumption, the points that are within a certain range from the skeleton can be then selected and associated to the subject, while all other points are discarded. The range should be chosen carefully; if it is allowed to be too large it would cause many erroneous points to be assigned to the subject, and if set too small it would produce an incomplete subject point cloud, possibly leading to tracking failure.

4.4 Initialization body model

As explained in Chapter 2, modelbased systems require the use of a model to perform motion capture. The following sections present an explanation of how the body model is generated.

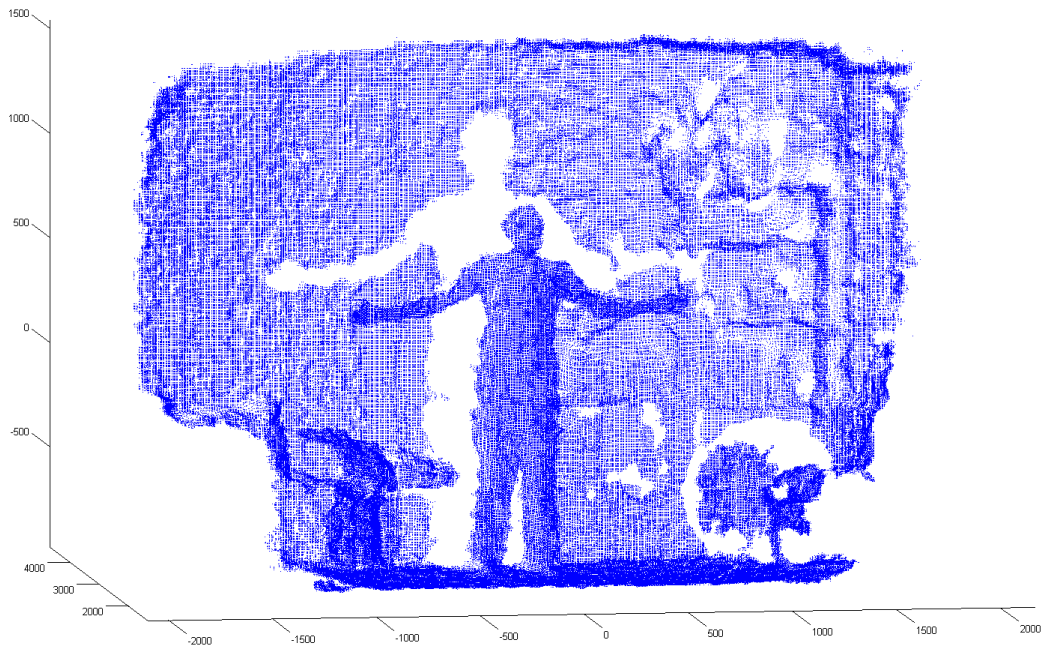

Figure 4.1: *Raw point cloud as obtained by the Kinect sensor consisting of 70.000 points.*

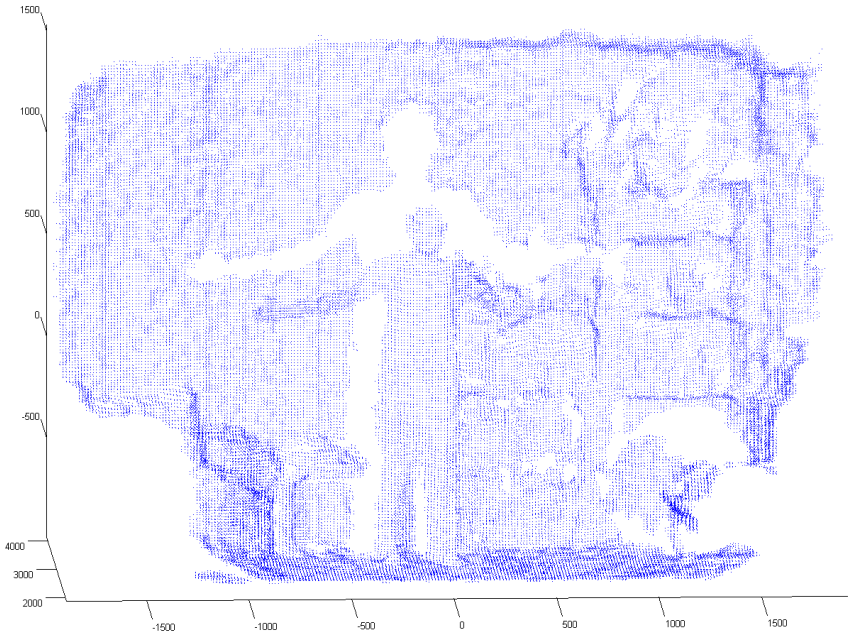

Figure 4.2: *Sub-sampled point cloud by factor $k = 2$. This is accomplished by keeping every 2^{nd} pixel while iterating through the depth-map elements.*

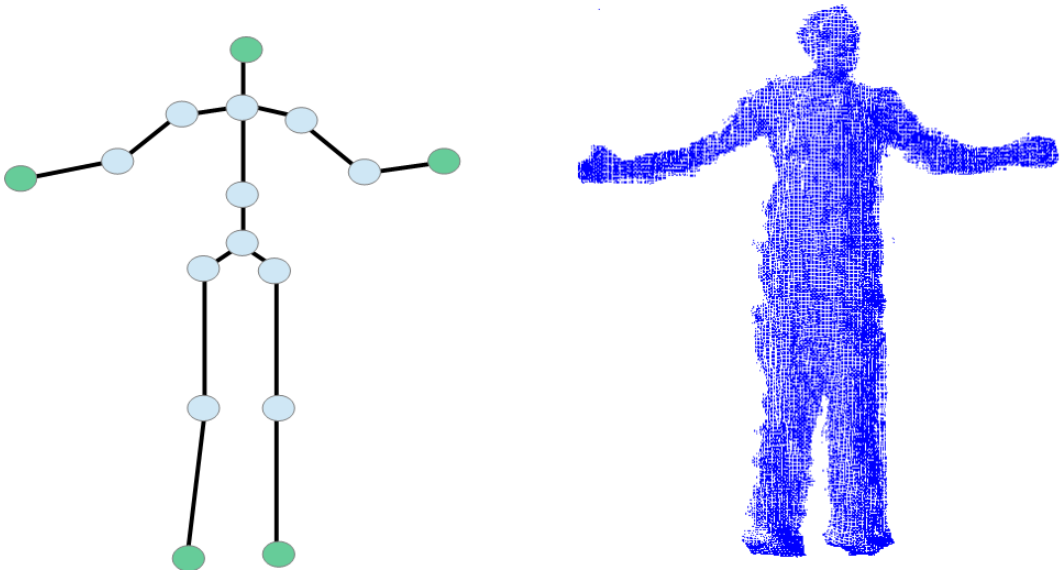

Figure 4.3: *Skeleton model after initialization and segmented point cloud of the subject in the T-pose. During the initialization phase, the link lengths and joint angles are computed by utilizing the body landmark positions returned by the Kinect tracking system.*

4.4.1 Skeleton model

As illustrated in Figure 3.2, the skeleton model has 15 links with lengths that need to be determined according to the subject's actual body size. The chosen method of measurement requires the subject to stand straight in front of the Kinect sensor in the T-pose (see Figure 4.3) for a short time interval (see Appendix A for a discussion of this procedure). Within this time frame, the body landmark positions are acquired from the Kinect system and their relative distances are computed and averaged to obtain the link lengths. The links belonging to the extremeties are also symmetrized. Afterward, the last acquired positions of the body landmarks are utilized to compute the joint angles such that the skeleton model is in the same pose as the Kinect skeleton. This is accomplished by computing the rotations about the x-axis and y-axis such that the positions of the body landmarks, as returned by the Kinect tracking system, have only one non-zero component along the z-axis when expressed in the local coordinate system of their parent joint. Let us call $\mathbf{b}_m = [b_x^m, b_y^m, b_z^m]$, the position of body landmark m, as returned by the Kinect. Then for $m = 2, \ldots, 16$, the coordinates of each $\mathbf{b}_m$ are converted in the local coordinate system of joint j_i, where $i = \Xi(m)$, so that: $\mathbf{b}_m \mapsto \mathbf{b}_m^* = [b_x^{*m}, b_y^{*m}, b_z^{*m}]$. The joint angles are then given by:

$$\alpha_i = -\arctan_2 \frac{b_y^{*m}}{\sqrt{(b_x^{*m})^2 + (b_z^{*m})^2}} \tag{4.1}$$

$$\beta_i = \arctan_2 \frac{b_x^{*m}}{b_z^{*m}} \tag{4.2}$$

The functions $\arctan_2$ takes two arguments in order to determine the quadrant of the computed angles that can be in the range $(-\pi, \pi]$.

4.4.2 Body surface model

A common method utilized to model a body surface involves using simple geometric shapes such as cylinders, spheres, and ellipsoids, which are then combined in a simplified representation of the subject's body. The chosen approach is of this kind, where the basic geometric shape is the elliptical cone frustum, parametrized by five quantities: height h, base radii: r_x^b, r_y^b, and top radii: r_x^t, r_y^t. Each body part (i.e., upper arm, lower arm, etc.) can be decomposed into an arbitrary number of these elements, which are then fitted to the segmented point cloud to approximate the real shape of the subject.

In the utilized model, the extremities (arms and legs) are symmetrical around their links, therefore their basic geometric elements only require three parameters to be fully determined: top and bottom radii, and

height. Moreover, continuity between elements is imposed, that is, two contiguous elliptical cone frustums must have the same radii where they meet. This applies to elements within the same body part, and to elements belonging to contiguous body parts, which further reduces the total numbers of parameters to be determined. Once the number of basic elements is chosen, the link is split accordingly in equally long sub-segments. Then, for each subsegment's end points, a set of nearest neighbors in the point cloud is computed and their distances averaged. These values are then assigned to the radii of the elliptical cone frustums (see Figure 4.4). A similar procedure is used when generating the abdomen, trunk, and head, but given that these body parts are not symmetric, the lateral radii also need to be determined. The trunk and abdomen are chosen to be equally broad and of fixed radius, which is set to be equal to the length of the link connecting the center shoulder to the left or right shoulder. The head width is set to 0.8 the radius just described.

4.5 Tracking algorithm

4.5.1 Matching

The first step of the ICP algorithm establishes correspondences between body model and point cloud. The standard implementation of the ICP algorithm [23, 27] and many of its extensions to articulated bodies [24, 16, 17] match the body model to the destination point cloud. However, in this work, an inverse assignment has been utilized; that is, the target points are matched to their nearest neighbors in the body model.

4.5.2 Minimization cost function and body model update

Once correspondences between body model and target point cloud have been established, the next step is to find the rigid transformations that minimize the cost function (3.17) by using the Gauss-Newton algorithm. To start the minimization process, a guess for the parameters vector is required, and a natural choice is to use the solution found in the previous iteration, $\mathbf{k}_0$. In this implementation, only one step in the descent direction is performed, where the step size is found by using a simple backtracking line search. The body model is then updated to the new configuration $\mathbf{k}^\dagger$ just found. Matching and minimization are then repeated until the cost function's relative error, $\Delta E = \frac{E(\mathbf{k}_0)-E(\mathbf{k}^\dagger)}{E(\mathbf{k}_0)}$, is smaller than a fixed threshold (see Table C.3).

4.6 Tracking failure recovery

The AICP algorithm can sometimes undergo tracking failures which would require a reinitialization of the system. To avoid this, a recovery routine has been implemented which makes use of the tracking information provided by the Kinect. The algorithm outline is the following:

1. At each frame, acquire Kinect skeleton data. The skeleton data contains body landmark positions and body landmark tracking status: tracked, inferred, not tracked.

2. If all body landmark have the status 'tracked', then:

   (a) Compute distances between each body landmark as returned by the Kinect and our system.

   (b) If any of the distances are consistently (at least 30 consecutive frames) larger than a set threshold, then re-position our skeleton according to the Kinect system.

This algorithm works under the reasonable assumption that the Kinect tracking algorithm is reliable when all body parts are clearly visible.

4.7 Occlusions and self-occlusions handling

The AICP can become unstable when one or more body parts are largely occluded. To avoid tracking failures in those situations, a routine has been implemented that disables the joints corresponding to the involved body parts. This is accomplished by checking the cardinalities, l_m, of each target set T_m and by comparing them with their values at initialization time, l_m^0, when all the body parts were clearly visible. Whenever the condition $l_m < l_m^0/K$, where K is a user-defined parameter, is verified, the corresponding joint $\Gamma(m)$ may be disabled,

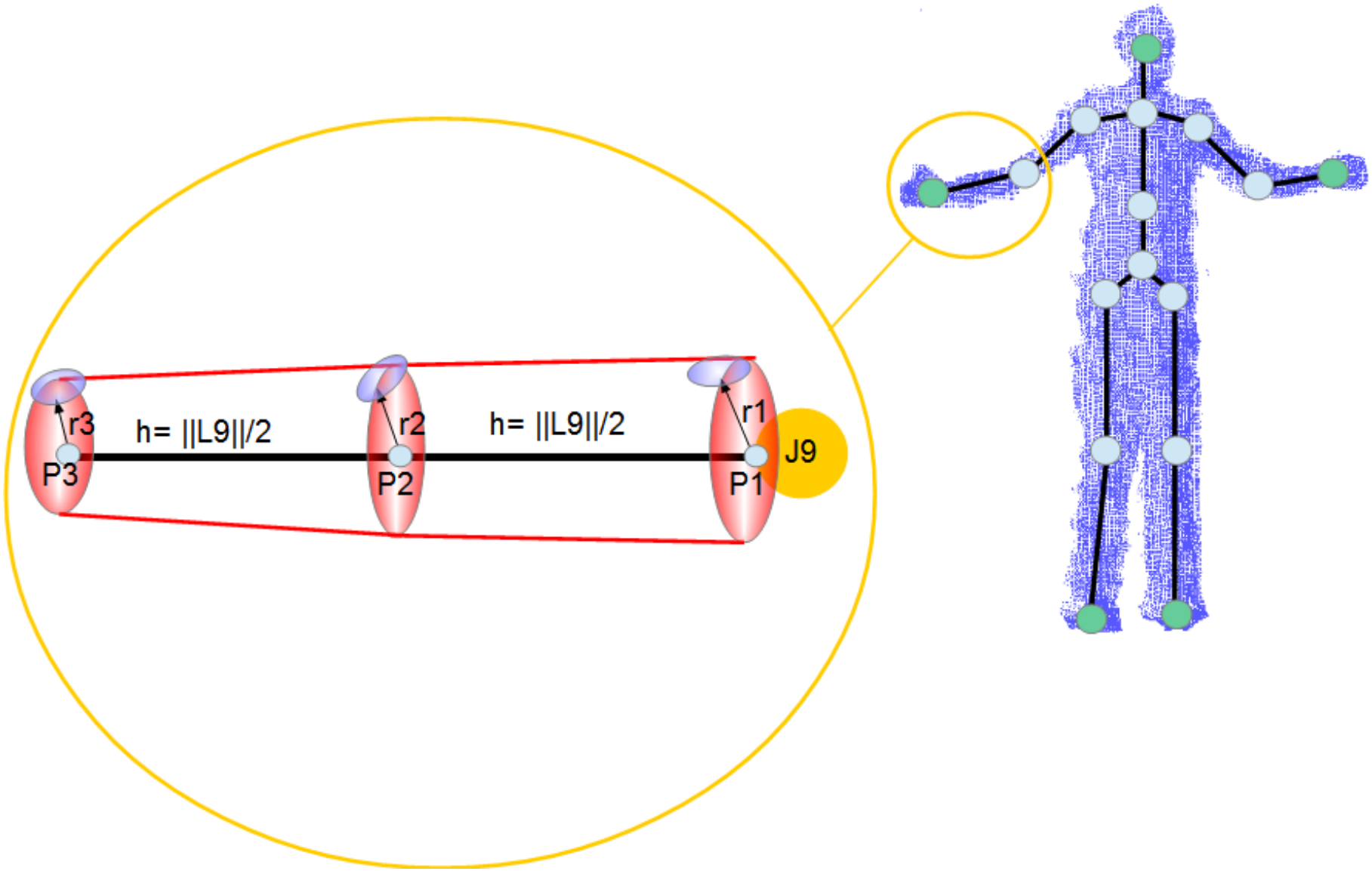

Figure 4.4: *Example showing the process of modeling a body part. In this case, the right lower arm has been modeled by joining two basic elements. Its corresponding link is split in two equally long sub-segments. For each of their end points: P1, P2, and P3, a set of nearest neighbors in the point cloud is computed and their distances averaged. These values are then assigned to the radii of the elliptical cone frustums.*

depending on whether the remaining body parts in its child branch (see Figure 3.2) are also occluded. For example, if B_7 (left lower arm) is occluded, joint j_9 in the left elbow will be disabled. On the other hand, if B_6 (left upper arm) is occluded, j_8 will be disabled only if B_7 is also occluded. This is because the tracking algorithm can determine the correct configuration of a joint even when its linked body part is entirely hidden as long as the visible body parts contain sufficient information.

5 Results

The standard procedure to test a newly developed motion capture algorithm is to compare its predicted movements against a ground truth, usually obtained by using one of the motion capture systems described in Chapter 2. In this way, it is possible to accurately measure the deviations from the correct positions. Unfortunately, in this work, access to such a system was not available, and therefore the obtained results are rather qualitative. In the following, some of the most common situations where the Kinect tracking system encounters problems are presented, showing, side-by-side, how the body landmarks were detected by the AICP and the Kinect systems. Each figure is followed by a plot showing the normalized residual error per point (NREP) obtained while performing the represented pose a few times. This quantity is computed at the end of the optimization process when the body model has been aligned with the point cloud, and it is the residual of the cost function divided by the number of points in the point cloud and normalized by subtracting a base line value. The NREP does not give a direct measure of the accuracy of the motion capture, but provides information about how well the point cloud is covered by the body model. When the motion capture is working poorly, the offset between body model and point cloud will grow larger, and therefore the NREP is expected to increase accordingly.

Figure 5.1 shows the residual errors per point (REP) before normalization in the case of a subject standing still in the T-pose. This pose is optimal in the way that all body parts are clearly visible and well separated from each other, and therefore it is considered as baseline for the measurements to follow. Each of the cases presented begins with the subject holding this pose for several seconds. Both systems present an offset which is expected given that the body model surface only approximates the subject's real shape. The Kinect offset is larger, which is also expected given that its tracking system does not perform body model - point cloud matching. To make the results more readable and easier to compare, the utilized unit of measurement is taken to be the average standard deviation of the REP relative to the baseline phase of various datasets collected in this work, which is referred to as $\Delta = 2.4 \cdot 10^{-2}$ mm.

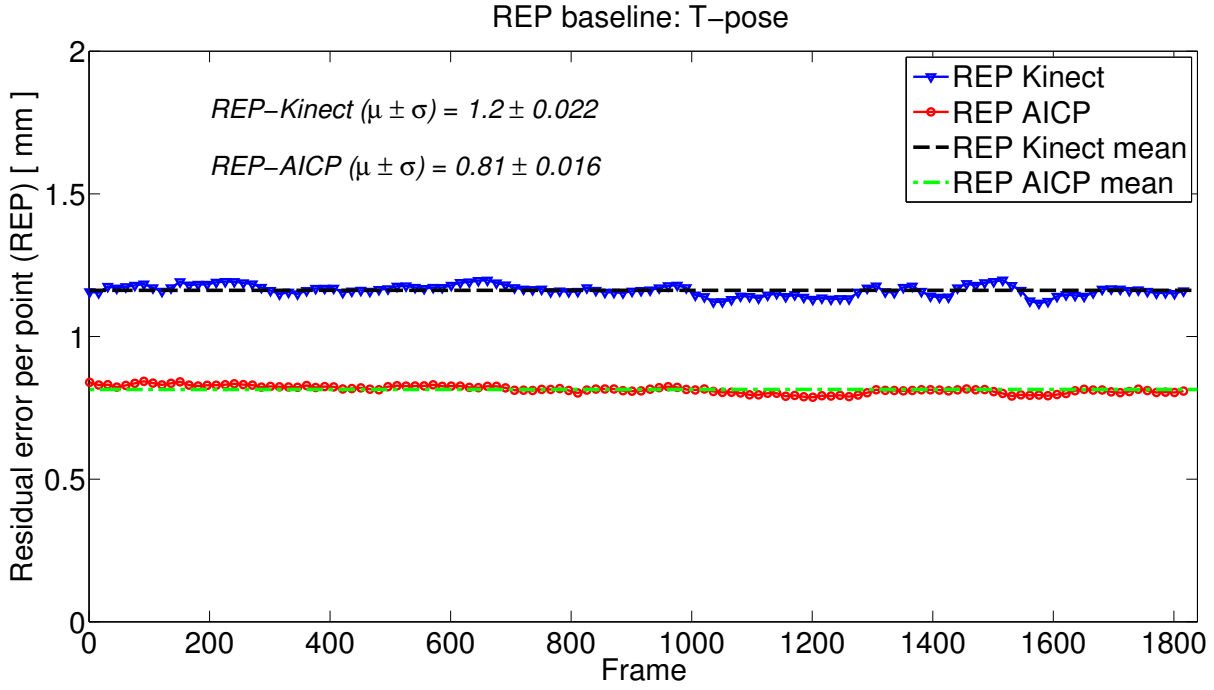

Figure 5.1: *Residual error per point relative to subject in the T-pose.*

5.1 Self-occlusions

Figures 5.2 to 5.8 illustrate how the AICP algorithm and the Kinect tracking system responded to various situations involving self-occlusion.

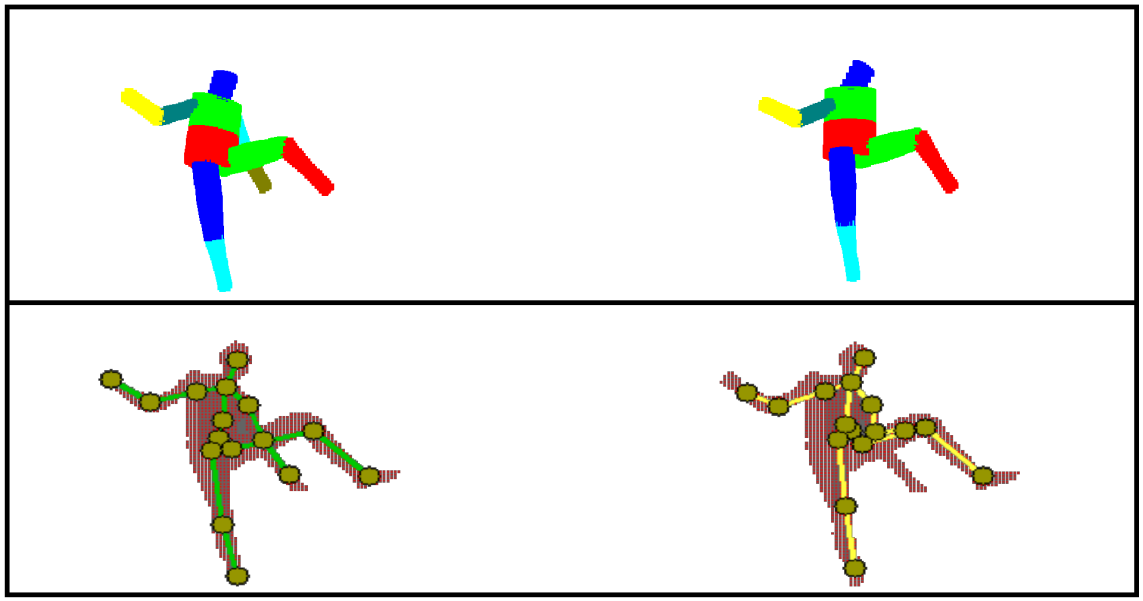

Figure 5.2: *Example of a self-occlusion involving one arm and one leg. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

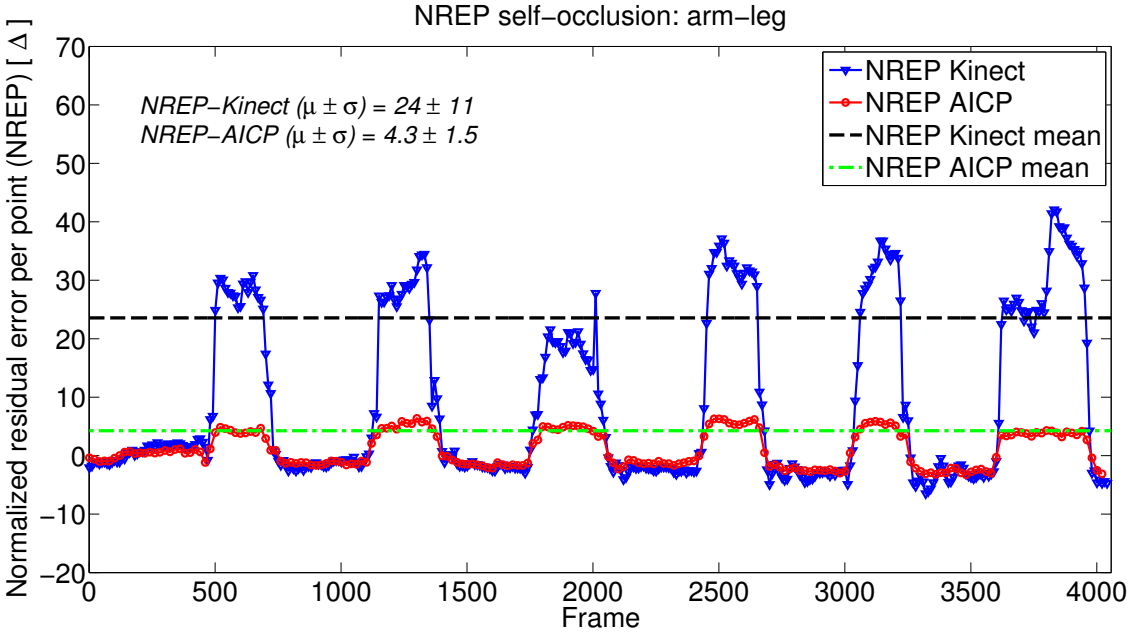

Figure 5.3: *NREP during self-occlusions involving one arm and one leg. Six neutral and motion phases are alternated. The neutral phase corresponds to the subject in the T-pose. The mean values shown are relative to the motion phase.*

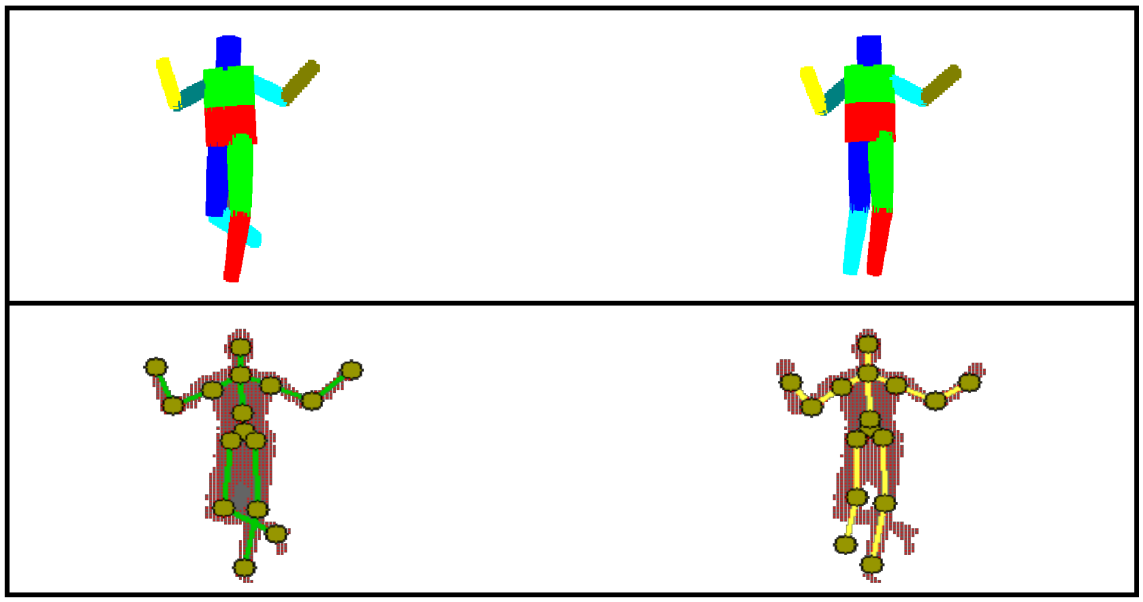

Figure 5.4: *Example of a self-occlusion involving the legs. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

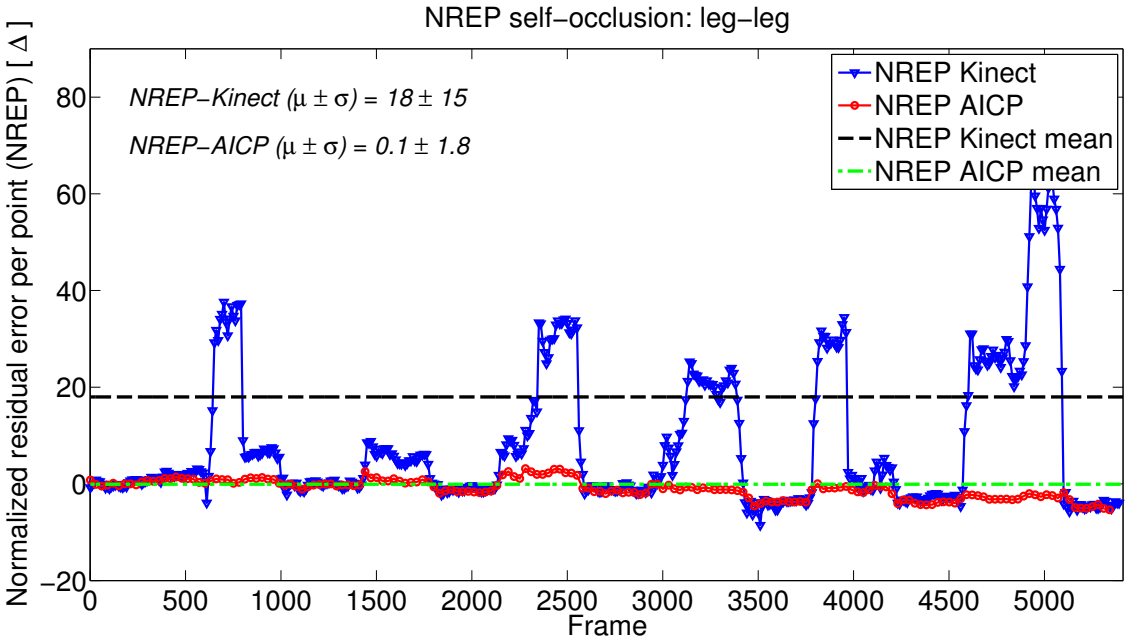

Figure 5.5: *NREP during self-occlusion involving the legs. Six neutral and motion phases are alternated. The neutral phase corresponds to the subject in the T-pose. The mean values shown are relative to the motion phase.*

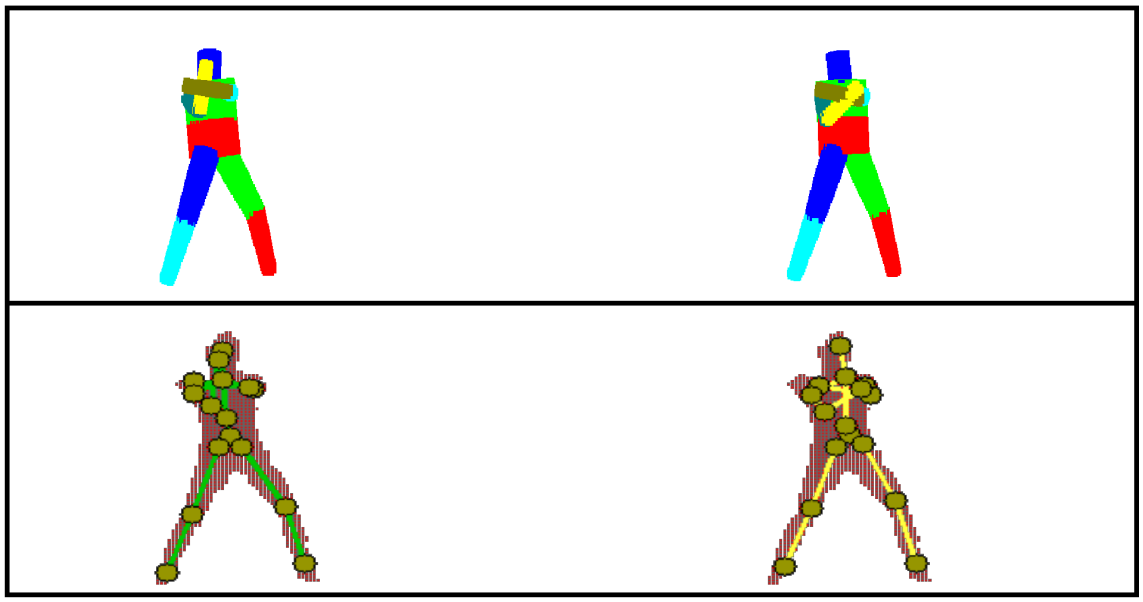

Figure 5.6: *Example of a self-occlusion involving the arms. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

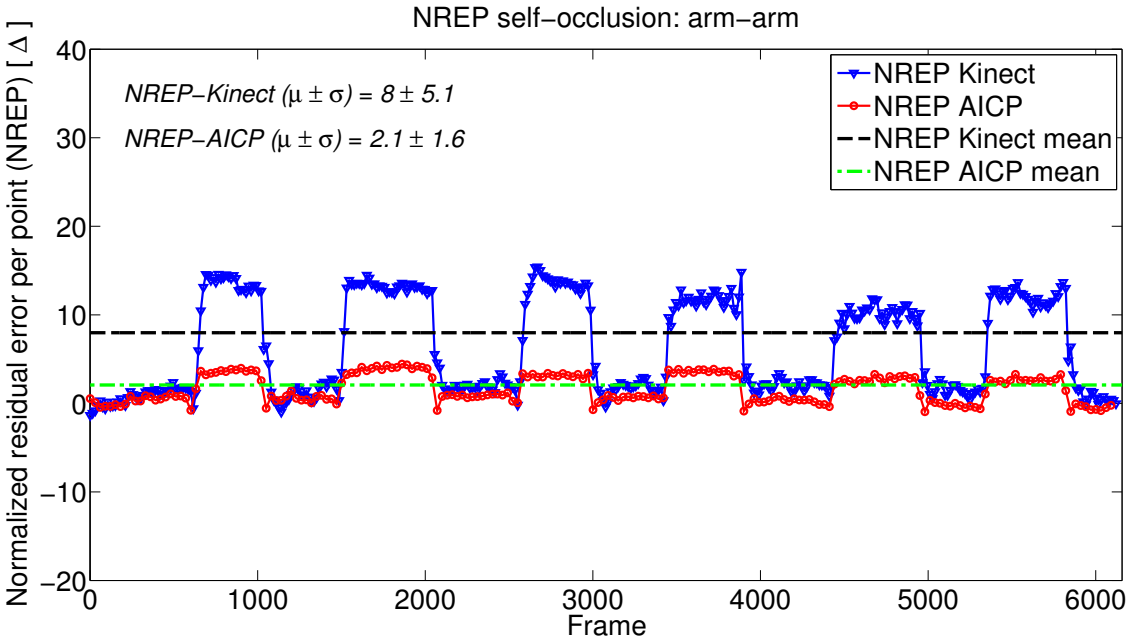

Figure 5.7: *NREP during self-occlusion involving arms. Six neutral and motion phases are alternated. The neutral phase corresponds to the subject in the T-pose. The mean values shown are relative to the motion phase.*

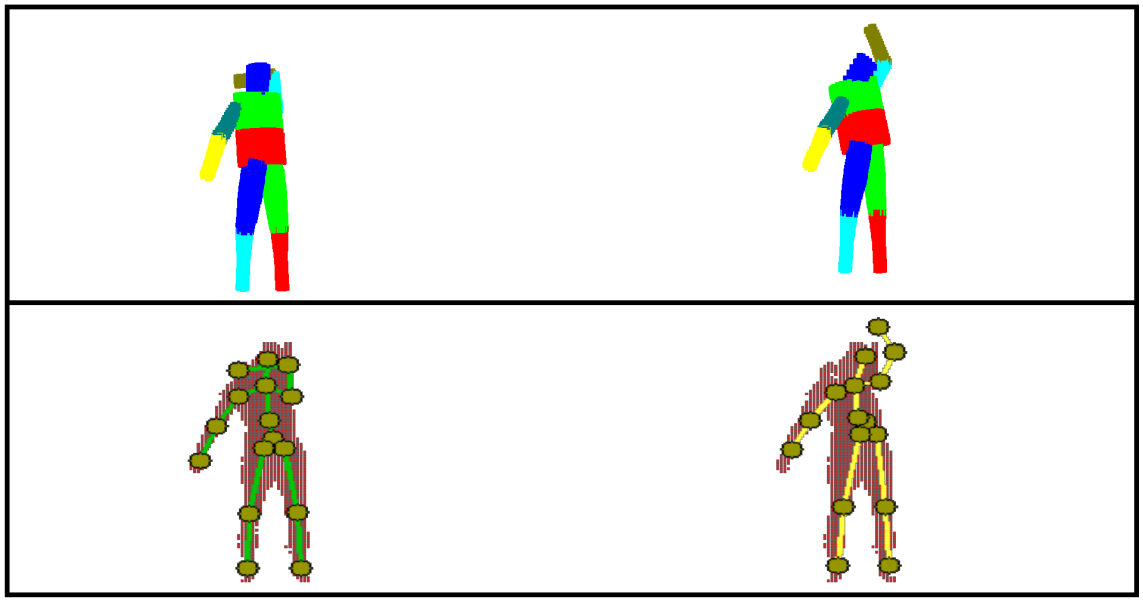

Figure 5.8: *Example of a self-occlusion involving left arm and head. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

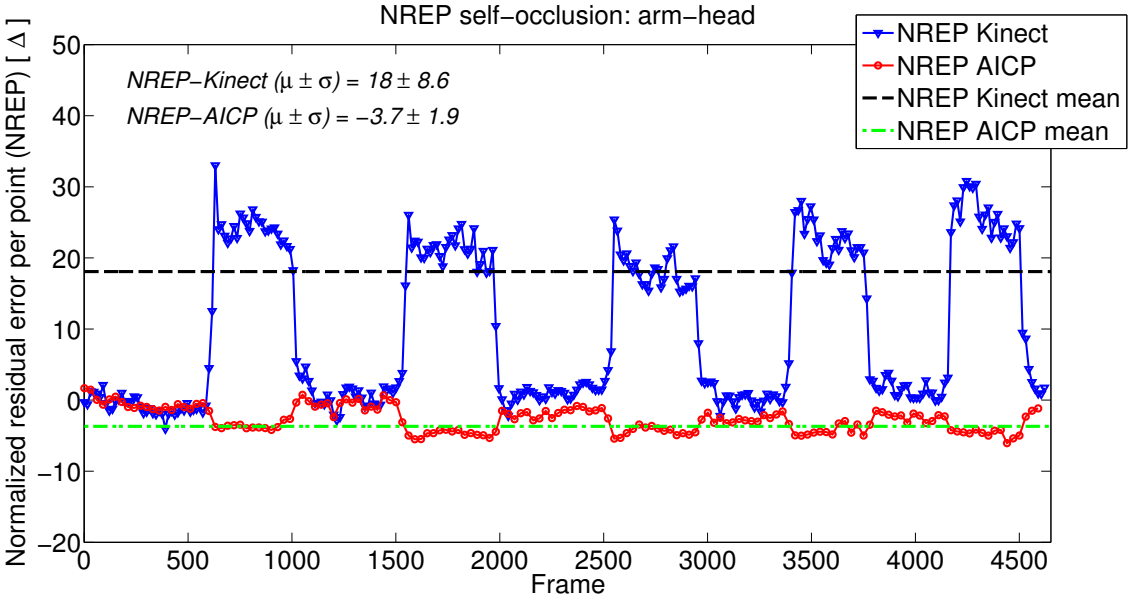

Figure 5.9: *NREP during self-occlusion involving left arm and head. Five neutral and motion phases are alternated. The neutral phase corresponds to the subject in the T-pose. The mean values shown are relative to the motion phase.*

5.2 Less common poses

There are certain poses that the Kinect tracking system fails to identify correctly. This is likely due to the fact that its training set did not contain any similar poses. Figures 5.10 to 5.15 show some examples of this problem.

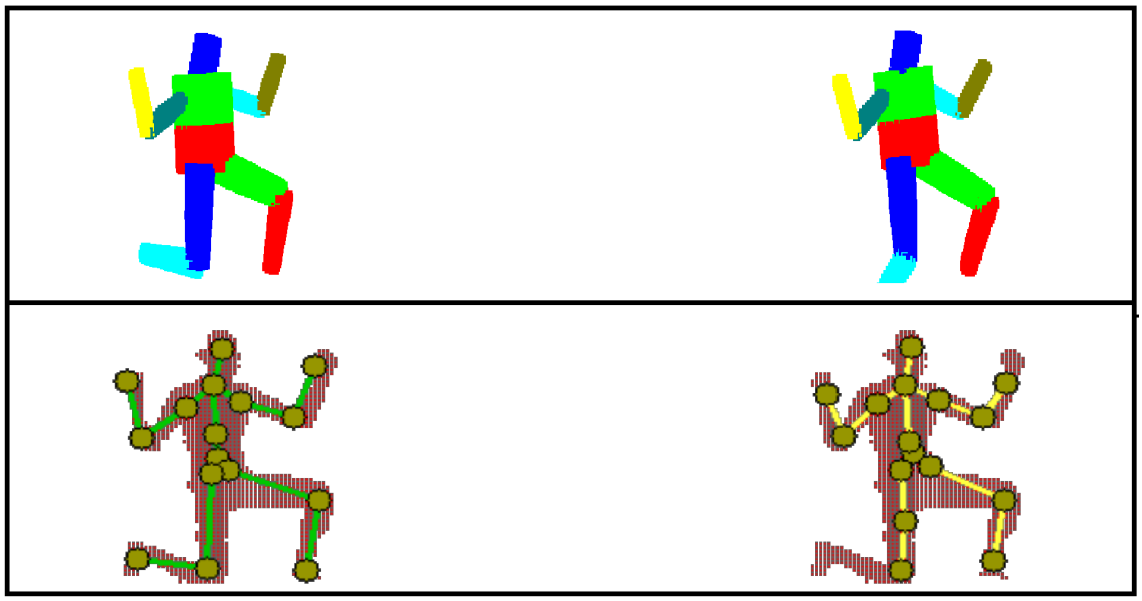

Figure 5.10: *Example of a less common pose. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

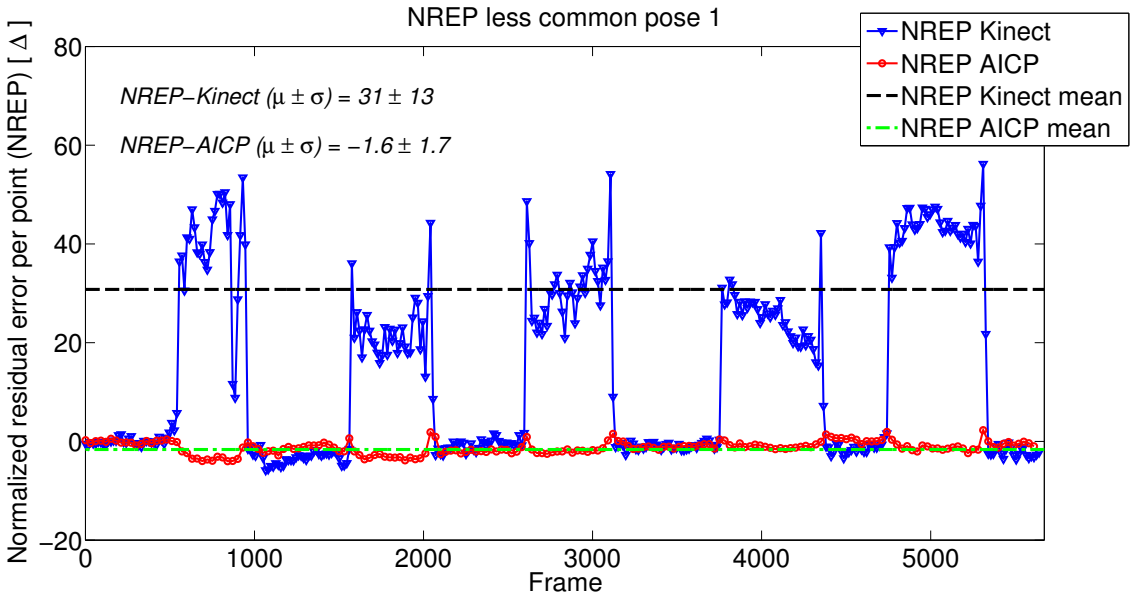

Figure 5.11: *NREP while performing a less common pose. Five neutral and motion phases are alternated. The neutral phase corresponds to the subject in the T-pose. The mean values shown are relative to the motion phase.*

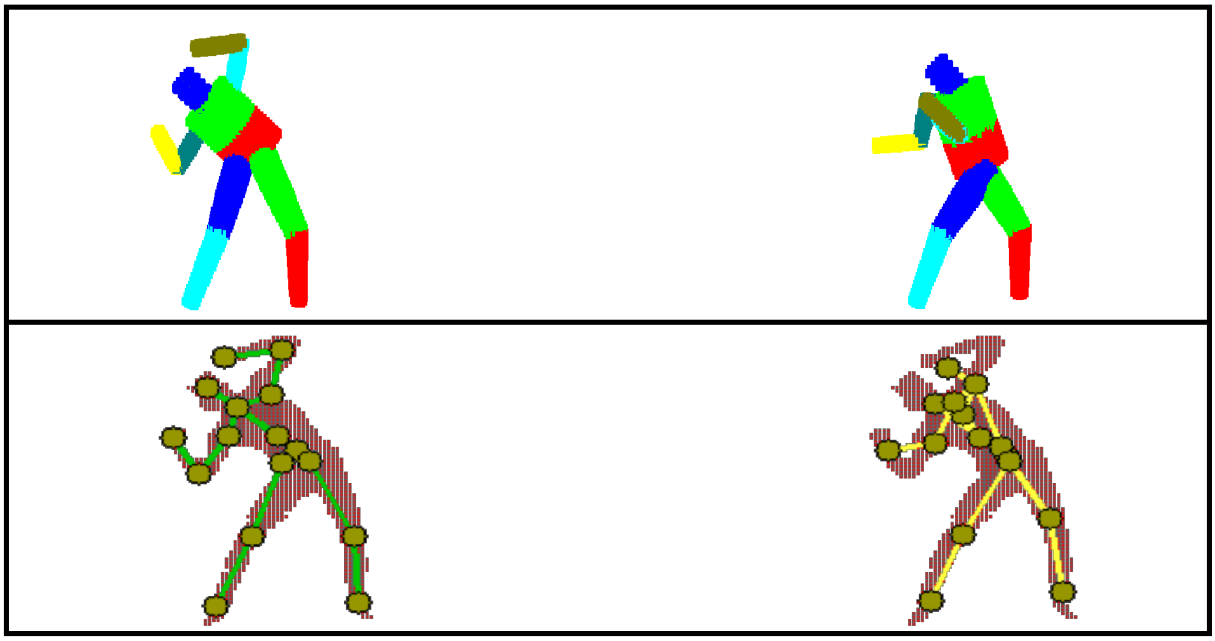

Figure 5.12: *Example of a less common pose. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

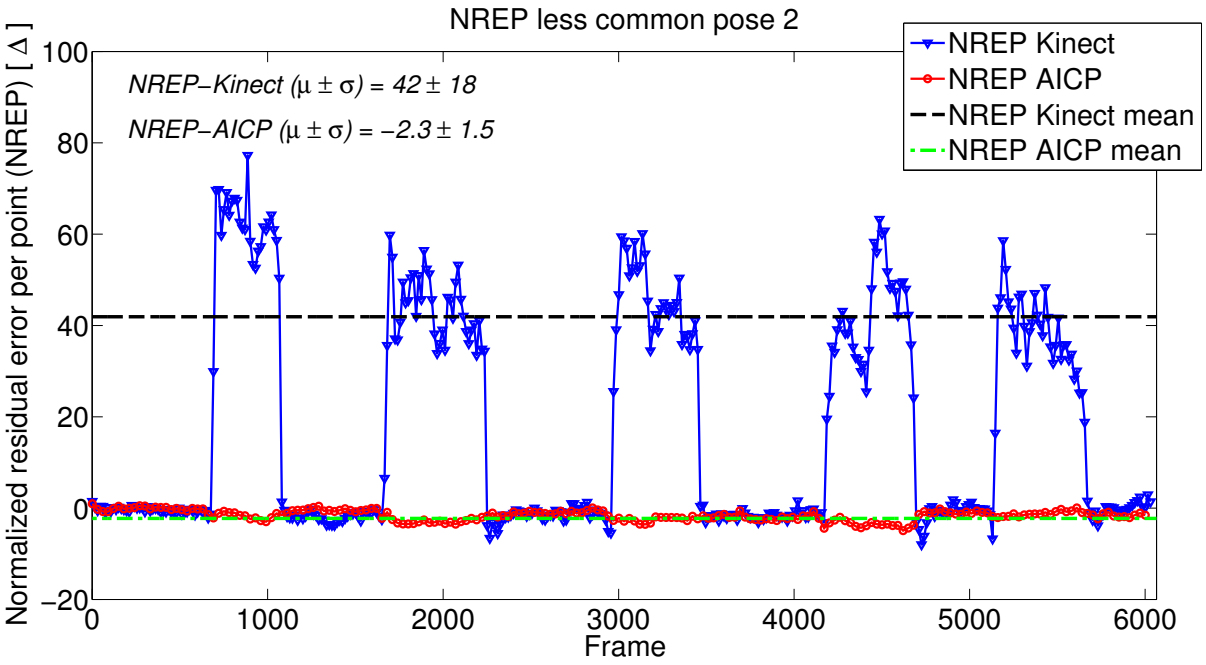

Figure 5.13: *NREP while performing a less common pose. Five neutral and motion phases are alternated. The neutral phase corresponds to the subject in the T-pose. The mean values shown are relative to the motion phase.*

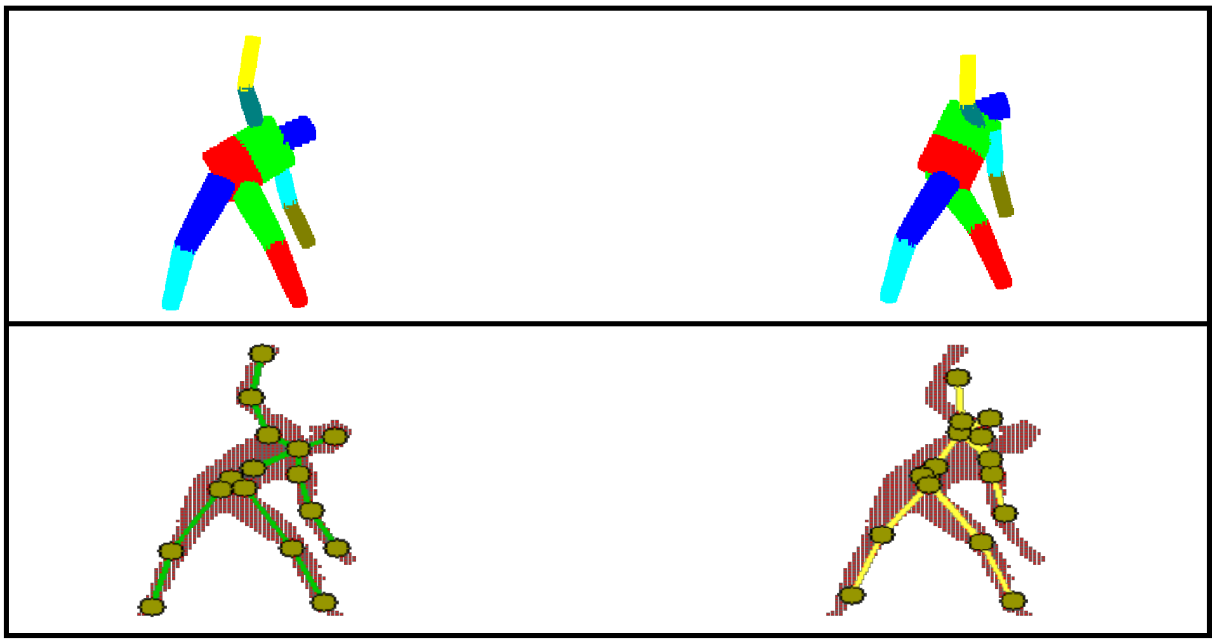

Figure 5.14: *Example of a less common pose. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

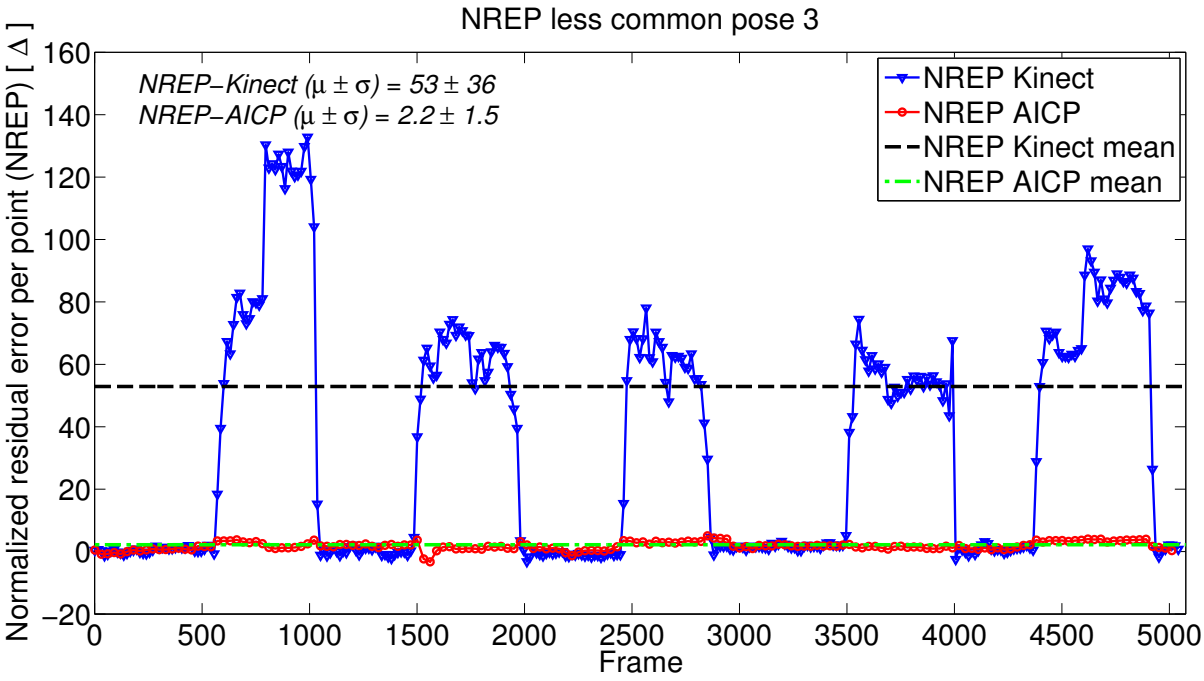

Figure 5.15: *NREP while performing a less common pose. Five neutral and motion phases are alternated. The neutral phase corresponds to the subject in the T-pose. The mean values shown are relative to the motion phase.*

5.3 Free motions

Figure 5.16 presents the NREP relative to a simple free motion sequence: subject facing the Kinect sensor while performing slow movements not involving self-occlusions. Figure 5.17 shows the case of an unconstrained motion sequence involving self-occlusions, quick movements, walking, turning away from the sensor, squatting, bending the back, etc. In Figure 5.18, the average NREP (ANREP) relative to 30 unconstrained motion sequences lasting one minute each is presented (see Table C.1 for the numerical results).

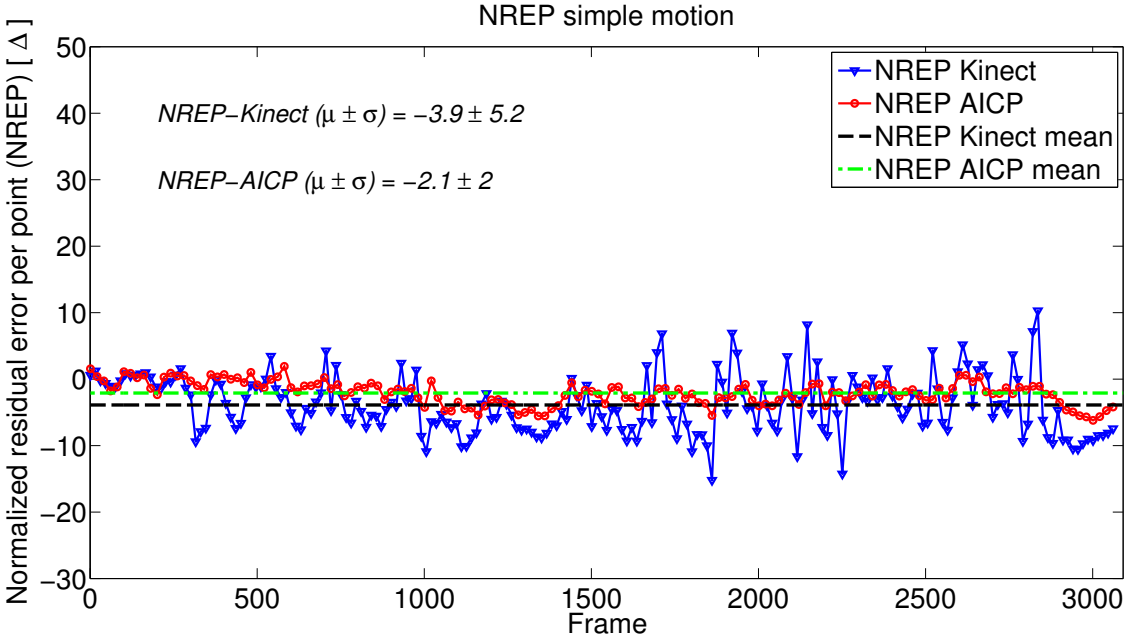

Figure 5.16: *NREP while performing a simple motion: slow movements, no self-occlusions, always facing the sensor.*

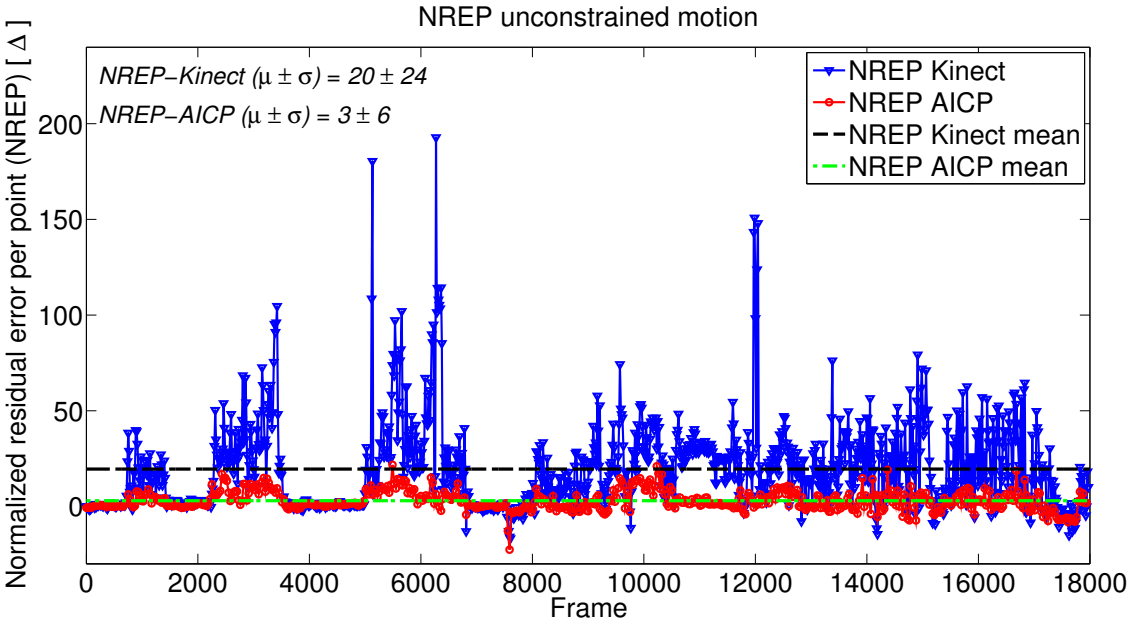

Figure 5.17: *NREP while performing an unconstrained motion: normal or quick movements, self-occlusions, squatting, turning away from the sensor, etc.*

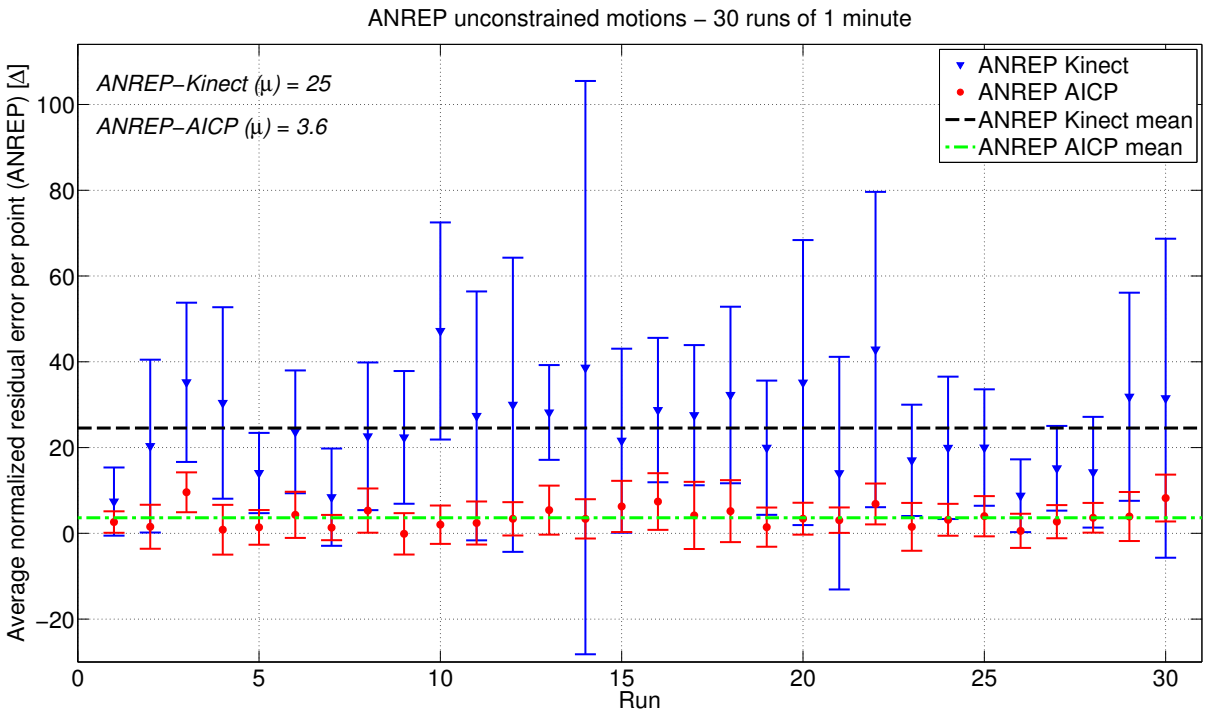

Figure 5.18: *ANREP unconstrained motion 30 runs. Average NREP relative to 30 unconstrained motion sequences lasting one minute each.*

5.4 Occlusions, extended range, and rear facing

Figure 5.19 illustrates how the AICP algorithm and the Kinect tracking system respond when presented with occlusions covering a large part of the body surface. In Figure 5.20, the ability of the AICP system to work in an extended range (distance > 4 m) is demonstrated. Figure 5.21 presents an additional problem associated with the Kinect tracking system which cannot distinguish between front and back of the subject.

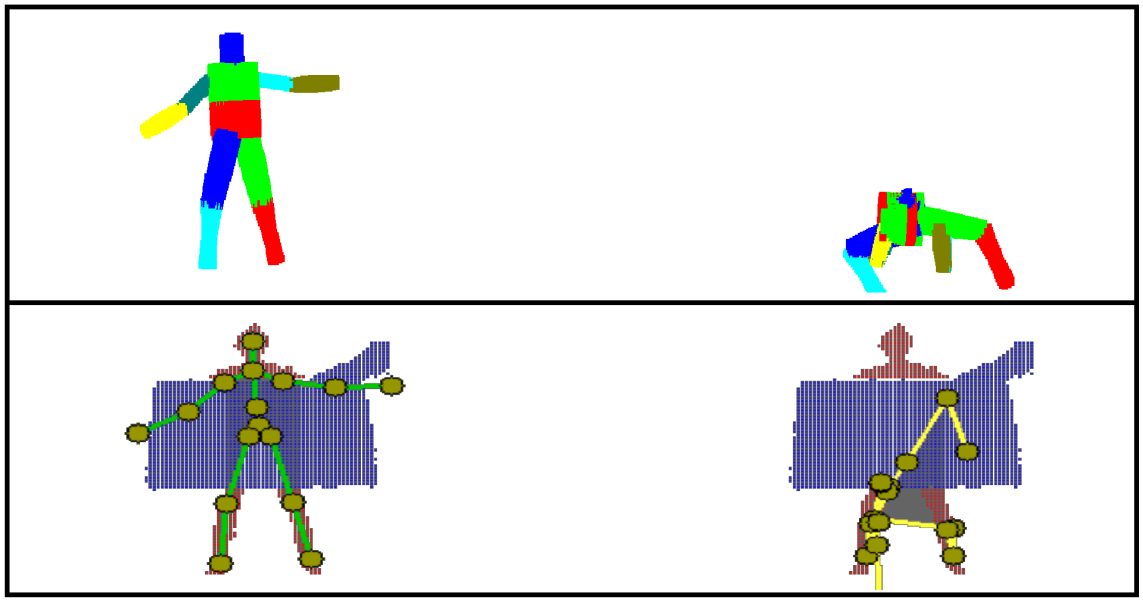

Figure 5.19: *Example of an occlusion caused by a book held in front of the sensor. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

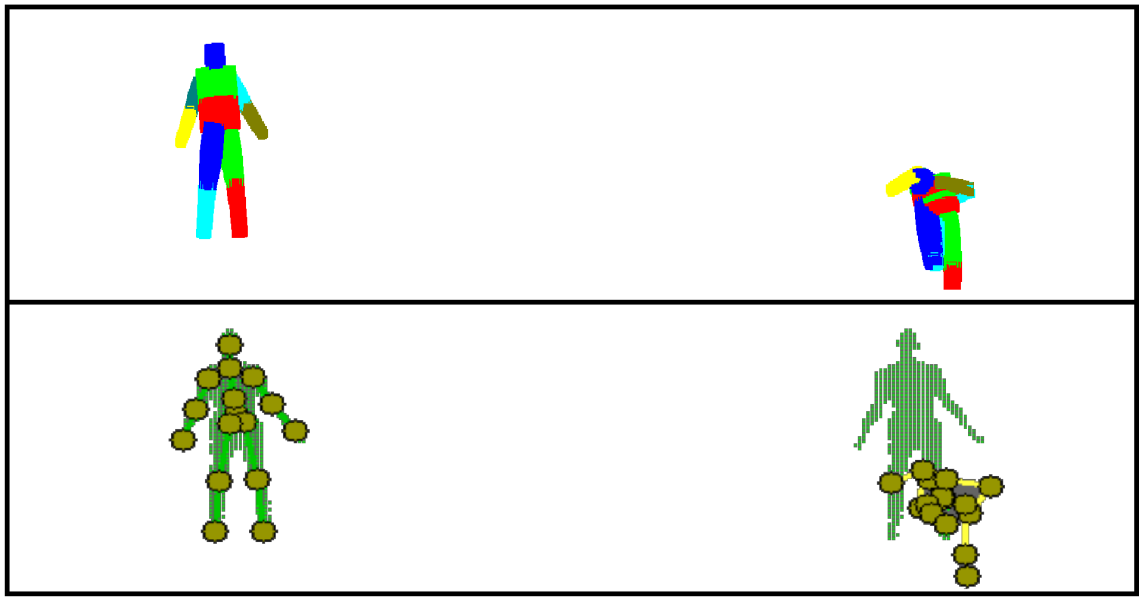

Figure 5.20: *Example of subject in the extended region (distance > 4 meters). Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

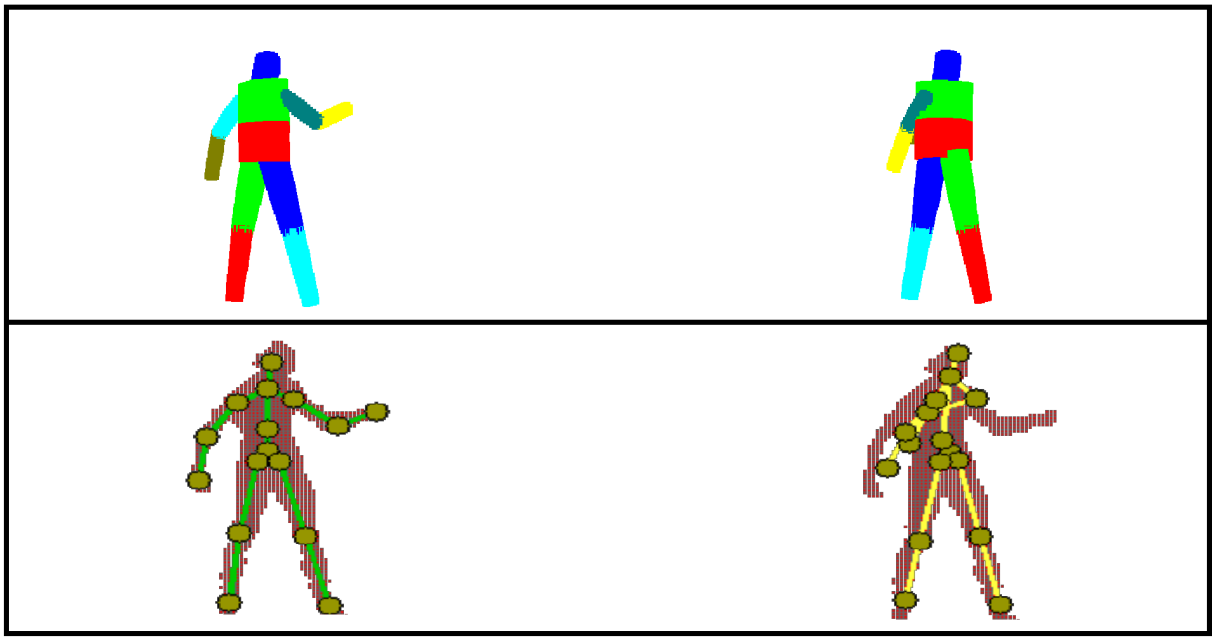

Figure 5.21: *Example of subject turned away from the sensor. The Kinect system cannot differentiate between front and back of the subject. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

5.5 Known problems of the AICP

There are certain situations that can cause tracking failures of the AICP system such as fast motions, closely positioned body parts, and large occlusions. Fast motions and closely positioned body parts are problematic for the same reason; as previously explained, the AICP works by matching the point cloud with the body model by finding the pairs of closest points. When body parts get too close or the subject's motions become too quick, it becomes more likely that wrong correspondences are established. Figures 5.22, 5.23, and 5.24 show two typical situations presenting this problem.

Larger occlusions can cause problems for a different reason; the unavoidable noise and discrepancy between body model and real shape of the subject are averaged out when the subject is clearly visible or undergoing small occlusions. However, during large occlusions only few points are visible and the AICP minimization process can produce erratic behaviors (see Figure 5.25). Figure 5.26 shows an example of this situation.

Figure 5.22: *Example of AICP tracking failure due to fast motion and close body parts. This picture shows the starting situation: the left arm is in contact with the chest. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

Figure 5.23: *Example of AICP tracking failure due to fast motion and close body parts. The subject quickly moved his arm away from the chest and the matching phase associated the points belonging to the arm to the body model abdomen, thus resulting in a wrongly estimated pose. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

Figure 5.24: *Example of AICP tracking failure due to body parts too close to each other. In this case the subject was walking sideways with his arms against the body, when eventually the right arm was absorbed by the trunk. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

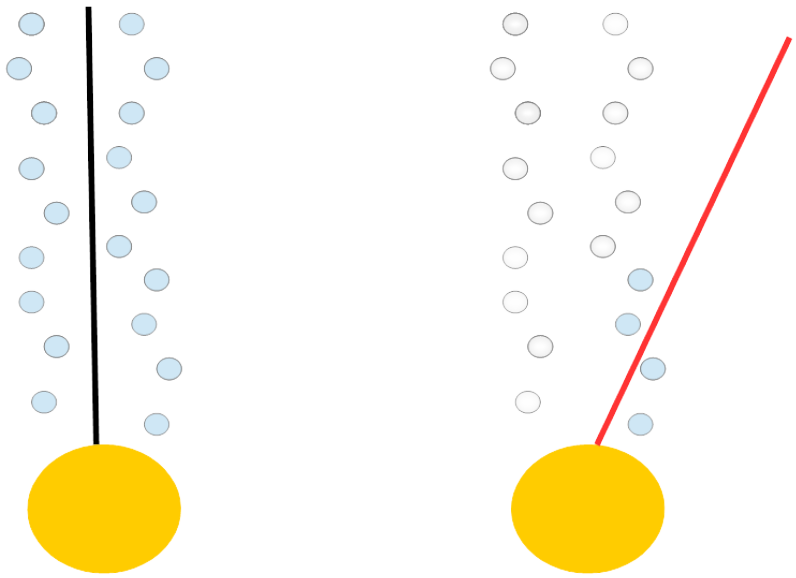

Figure 5.25: *Illustration of instability due to large occlusion. The joint tries to minimize the relative distances between the points and its link. The left panel shows the joint in the optimal configuration which minimizes the link distance from the point cloud. The right panel shows how the joint repositions the link when a large subset of points are hidden (shown in lighter color) by an occlusion.*

Figure 5.26: *Example of ICP tracking failure due to large occlusions. The subject is turned away from the sensor, causing some body parts to be severely occluded. The few points visible produce erratic behavior of the body model. Top panel: the left side shows the body model in the pose determined by the AICP, while the right side shows the body model in the Kinect pose. Bottom panel: AICP skeleton and Kinect skeleton superimposed on the depth map, on the left and right side, respectively.*

6 Discussion

In this work, a motion capture system to be used in combination with the Kinect sensor has been developed and implemented. The task was rather demanding due to the strict requirements of performing real-time tracking while maintaining a good level of accuracy and allowing for motions where occlusions and self-occlusions are frequent. Other systems available on the market, such as the proprietary Microsoft Kinect algorithm, can only meet some of these requirements and only in specific conditions, despite the considerable resources invested in research and development. The developed system works by establishing and minimizing dense correspondences between point clouds acquired from the Kinect and a body model previously generated. It only makes use of low-level information to perform motion capture, with the exceptions of the initialization phase and when recovering from tracking failures, during which it acquires the body landmark positions from the Kinect tracking algorithm.

The core routine of the implemented system is an extension to articulated bodies of the ICP algorithm, which iteratively revises a group of rigid transformations in order to minimize the residual distance between the body model and the point cloud. However, the idea of using the ICP algorithm for motion tracking is not new; in fact, some papers have already been published discussing various implementations of it [24, 16, 25]. But whereas these authors optimize the cost function one joint at the time in a hierarchical fashion, in this work it is minimized over the full configuration space at once. Furthermore, an inverted matching procedure [17] is utilized, that is, correspondences are established from the point cloud to the body model. Intuitively this can be seen as having the point cloud pulling the body model instead of having the body model chasing the point cloud. During occlusions and self-occlusions, matching the body model with the point cloud could result in several erroneous correspondences, because all the points of the body model need to be matched, even though their corresponding body part in the point cloud is partially or entirely hidden. This could be avoided, to a limited extent, by introducing a distance threshold for the matched pairs (i.e., discarding pairs with relative distances larger than a set value), but at the price of possibly losing valuable correspondences in situations involving quick motions. The inverted matching does not incur these problems because in this case, if a body part in the point cloud were occluded, the corresponding part in the body model would simply not be active.

In the previous chapter, the NREP produced by the AICP and the Kinect systems were compared while performing various motion sequences. Beside producing a lower mean NREP, thus providing a better alignment between body model and point cloud, the AICP was also more stable than the Kinect algorithm, as shown by its significantly smaller NREP standard deviation in most of the analyzed situations. The main assumption made when using an ICP algorithm is that the point clouds to be matched are almost aligned, otherwise the algorithm could get stuck in a local optimum. In the context of this work, that means assuming that the movements of the subject are not excessively fast; if this condition is not met, the matching phase could produce several erroneous correspondences and thereby result in tracking failures. This is especially true with fast motions involving body parts that are close to each other (see Figures 5.22, 5.23, and 5.24). A possible solution to these problems could be to utilize markers to help the matching algorithm. For example, by wearing colored wrist bands it could be possible to avoid the situations shown in the previous chapter, where the lower arm section of the point cloud was erroneously matched with the body model trunk.

The processing speed of the system is affected by a few factors, the most obvious being the hardware running the program. The sizes of body model and point cloud play another important role and that is why a sub-sampling routine was incorporated in the pipeline (see Table C.2). It is important to point out that a low frame rate is not only unpleasant to view on the screen but it also makes the system much more likely to fail; given that the previous configuration is the initial guess to calculate the current one, if too many frames are skipped, the body model could result poorly aligned with the current frame, thus causing the AICP to get stuck in a local minimum.

The AICP can become unstable when one or more body parts are largely occluded (see Figures 5.25 and 5.26). In those situations, the noise in the point cloud and the unavoidable discrepancy between the body model and the real shape of the subject, in conjunction with the fact that the occluded parts are represented by very few points, can result in erratic motions of the skeleton. For this reason, a routine to disable the joints associated with severely occluded body parts was implemented. Anyhow, the AICP system could be easily extended to work with multiple Kinects, therefore substantially reducing the occlusions problem.

7 Conclusions and future work

The goal of this work was to develop a motion capture algorithm that could obviate some of the inherent limitations of the Kinect tracking system while providing accurate and real-time tracking. As shown in the previous chapter, the developed system proves to be more robust in dealing with occlusions and self-occlusions by producing an NREP approximately seven times smaller on average relative to a set of 30 unconstrained motions, it can track a wider range of poses, and it can work in an extended range. This makes it a potentially better solution for performing motion capture in an assembly line setting, where self-occlusions commonly occur while performing manual work and occlusions are sometimes unavoidable due to the presence of other workers or robotic tools.

The segmentation of the subject is a crucial phase of the overall tracking algorithm which needs careful attention, especially when using an inverted matching procedure. The current routines could be extended to use more information in the segmentation process, such as color intensity or markers. Another option could be to integrate or substitute them with more advanced machine learning techniques for human detection such as the one available with OpenCV. The main bottleneck of this pipeline is the nearest neighbor search in the AICP algorithm. The current implementation utilizes the ANN library developed by Arya and Mount [26], which is highly optimized but runs on a single core. A substantial improvement in tracking speed could be achieved by developing a GPGPU implementation to parallelize the search. The Jacobian used in the Gauss-Newton algorithm is computed numerically by means of forward finite difference. However, it could be possible to compute it analytically or by automatic differentiation, thus obtaining exact derivatives in shorter time. The Kinect for Windows SDK supports up to four simultaneous Kinects for skeletal tracking. However, the tracking accuracy is still bound by the accuracy of the best positioned Kinect (relative to the subject), which, as has been shown, can be rather poor in the event of occlusions or self-occlusions even when the user is facing the sensor. The AICP system, on the other hand, could combine the point clouds acquired from each Kinect to obtain a more complete view of the subject, and therefore providing more accurate and stable tracking.

List of Figures

List of Tables

References

[1] A. J. D. *et al.* "Markerless Motion Capture of Complex Full-Body Movement for Character Animation". *Proceedings of the Eurographics Workshop on Animation and Simulation*. Springer-Verlag LNC, 2001.

[2] A. S. *et al.* Motion Capture process, techniques and applications. *IJRITCC* **1**.4 (2013). ISSN: 2321 – 8169.

[3] K. Yamane and J. Hodgins. "Simultaneous Tracking and Balancing of Humanoid Robots for Imitating Human Motion Capture Data". *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009.

[4] S. C. *et al.* "Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning". *Proceedings of Australasian Conference on Robotics and Automation: 3-5 Dec 2012, Victoria University of Wellington, New Zealand*. 2012.

[5] S. Oh et al. "A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video". *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 3153–3160. ISBN: 978-1-4577-0394-2. DOI: 10.1109/CVPR.2011.5995586. URL: http://dx.doi.org/10.1109/CVPR.2011.5995586.

[6] J. C. P. C. *et al.* A Virtual Reality Dance Training System Using Motion Capture Technology. *IEEE Transactions on Learning Technologies* **4**.2 (2011), 187–195. ISSN: 1939-1382. DOI: http://doi.ieeecomputersociety.org/10.1109/TLT.2010.27.

[7] *Phase space motion capture*. URL: http://www.phasespace.com/impulse_motion_capture.html.

[8] D. Roetenberg, H. Luinge, and P. Slycke. Xsens MVN: full 6DOF human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep* (2009).

[9] *Exo-skelelton motion capture*. URL: http://www.metamotion.com/gypsy/gypsy-motion-capture-system-mocap.htm.

[10] L. Zhang, B. Curless, and S. M. Seitz. "Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming". *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*. Padova, Italy, June 2002, pp. 24–36.

[11] J. S. *et al.* Real-Time Human Pose Recognition in Parts from Single Depth Images. *IJACT* **4**.11 (2012). DOI: 10.4156/ijact.vol4.issue11.23.

[12] L. Breiman. Random Forests. English. *Machine Learning* **45**.1 (2001), 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: http://dx.doi.org/10.1023/A:1010933404324.

[13] C. S. *et al.* Skeleton Tracking using Kinect Sensor and Displaying in 3D Virtual Scene. *IJACT* **4**.11 (2012). DOI: 10.4156/ijact.vol4.issue11.23.

[14] S. Knoop, S. Vacek, and R. Dillmann. "Sensor fusion for 3D human body tracking with an articulated 3D body model". *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. May 2006, pp. 1686–1691. DOI: 10.1109/ROBOT.2006.1641949.

[15] L. Zhang et al. "Real-time human motion tracking using multiple depth cameras". *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. Oct. 2012, pp. 2389–2395. DOI: 10.1109/IROS.2012.6385968.

[16] S. Pellegrini, K. Schindler, and D. Nardi. "A Generalisation of the ICP Algorithm for Articulated Bodies." *BMVC*. Citeseer. 2008, pp. 1–10.

[17] D. Droeschel and S. Behnke. "3D Body Pose Estimation Using an Adaptive Person Model for Articulated ICP". *Intelligent Robotics and Applications*. Ed. by S. Jeschke, H. Liu, and D. Schilberg. Vol. 7102. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 157–167. ISBN: 978-3-642-25488-8. DOI: 10.1007/978-3-642-25489-5_16. URL: http://dx.doi.org/10.1007/978-3-642-25489-5_16.

[18] L. V. Calderita et al. Model-Based Reinforcement of Kinect Depth Data for Human Motion Capture Applications. *Sensors* **13**.7 (2013), 8835–8855. ISSN: 1424-8220. DOI: 10.3390/s130708835. URL: http://www.mdpi.com/1424-8220/13/7/8835.

[19] *OpenNI API*. 2014. URL: http://structure.io/openni.

[20] Y. Zhu and K. Fujimura. "Bayesian 3D Human Body Pose Tracking from Depth Image Sequences". *Proceedings of the 9th Asian Conference on Computer Vision - Volume Part II*. ACCV'09. Xi'an, China: Springer-Verlag, 2010, pp. 267–278. ISBN: 3-642-12303-1, 978-3-642-12303-0. DOI: 10.1007/978-3-642-12304-7_26. URL: http://dx.doi.org/10.1007/978-3-642-12304-7_26.

[21] L. A. Schwarz et al. Human Skeleton Tracking from Depth Data Using Geodesic Distances and Optical Flow. *Image Vision Comput.* **30**.3 (Mar. 2012), 217–226. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2011.12.001. URL: http://dx.doi.org/10.1016/j.imavis.2011.12.001.

[22] T. B. Jadran Lenarcic and M. M. Stanisic. *Robot Mechanisms*. Springer, 2012. ISBN: 978-9400745216.
[23] P. Besl and N. D. McKay. A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **14**.2 (Feb. 1992), 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791.
[24] D. Moschini and A. Fusiello. "Tracking Human Motion with Multiple Cameras Using an Articulated Model". *Computer Vision/Computer Graphics CollaborationTechniques*. Ed. by A. Gagalowicz and W. Philips. Vol. 5496. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 1–12. ISBN: 978-3-642-01810-7. DOI: 10.1007/978-3-642-01811-4_1. URL: http://dx.doi.org/10.1007/978-3-642-01811-4_1.
[25] S. Corazza et al. Markerless Motion Capture through Visual Hull, Articulated ICP and Subject Specific Model Generation. English. *International Journal of Computer Vision* **87**.1-2 (2010), 156–169. ISSN: 0920-5691. DOI: 10.1007/s11263-009-0284-3. URL: http://dx.doi.org/10.1007/s11263-009-0284-3.
[26] *Approximate Nearest Neighbor Searching*. 2010. URL: http://www.cs.umd.edu/~mount/ANN/.
[27] S. Rusinkiewicz and M. Levoy. "Efficient variants of the ICP algorithm". *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. 2001, pp. 145–152. DOI: 10.1109/IM.2001.924423.

Appendices

A Bone length

The Kinect tracking system does not set any constraints on bone lengths, which can therefore largely fluctuate from frame to frame, depending on the particular motion the subject is performing. This could present a problem, given that the initialization phase of the AICP system relies on the Kinect algorithm to determine the body proportions of the body model. For this reason, a small study was undertaken to understand what poses or motions are best suited for acquiring reliable bone length data. In the following, the results accounting for three different cases are presented: subject standing in the T-pose (see Figure A.1), subject facing the Kinect while performing a slow periodic motion which involved waving his arms and squatting (see Figure A.2), and subject performing unconstrained movements (see Figure A.3). Instead of studying individually each one of the fifteen links, the time evolution of their average (average bone length, ABL) is considered.

The T-pose does not involve motion, therefore its corresponding average standard deviation σ_N, is taken as the noise level of the Kinect system (see Table A.1). As Figures A.1), A.2), and A.3), and Tables A.1, A.2, and A.3 illustrate, the final mean values of the ABL are compatible with each other, but the slow and unconstrained motions running averages take longer to settle about their mean values within the noise level. Therefore, it appears preferable to initialize the skeleton model by standing in the T-pose.

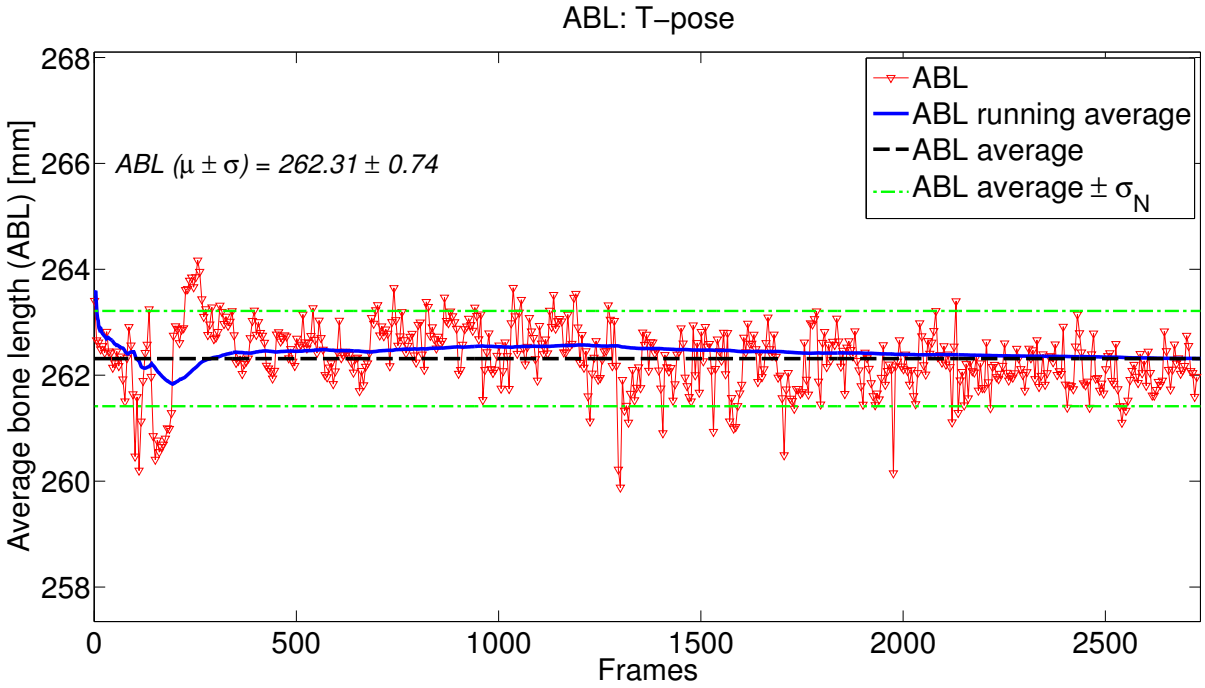

Figure A.1: *Average bone length for subject standing in the T-pose.*

	T-Pose	
Run	**Mean[mm]**	**Std[mm]**
1	263.4	0.8
2	263.8	1.0
3	265.1	0.7
4	264.9	1.2
5	264.2	0.6
Average	264.3	0.9

Table A.1: Table reporting ABL mean and standard deviation relative to T-pose. Values obtained performing the pose for one minute each run.

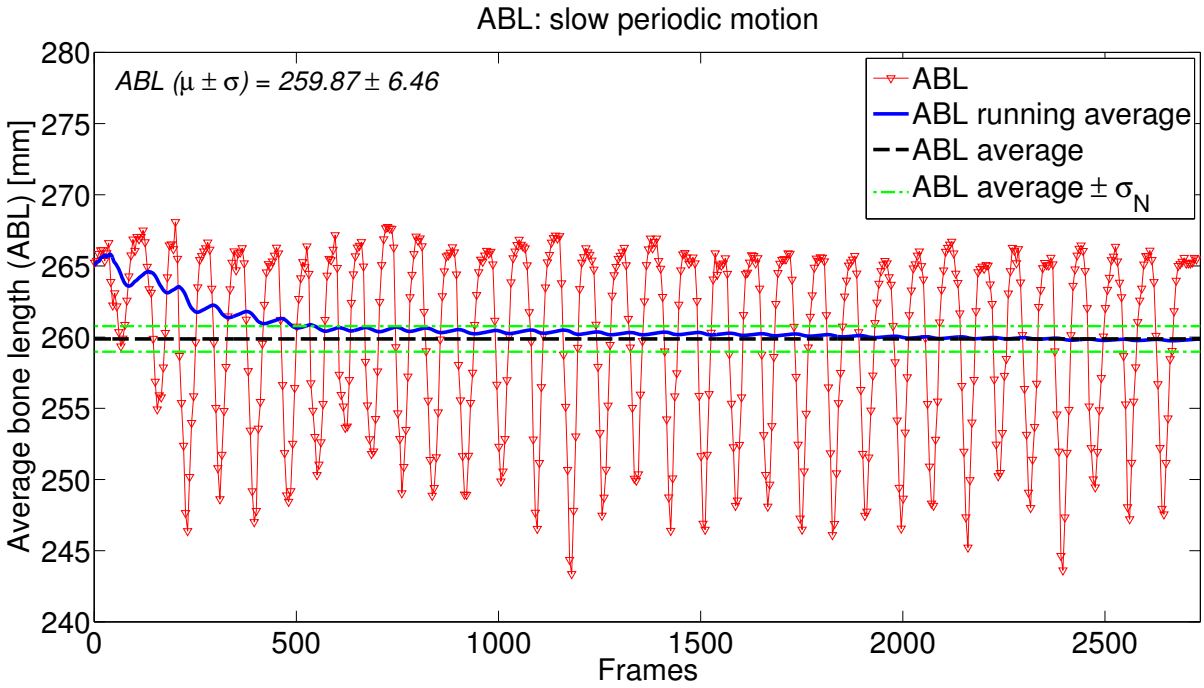

Figure A.2: *Average bone length during slow periodic motion. In this case the subject is performing a periodic and slow motion involving waving his arms and squatting.*

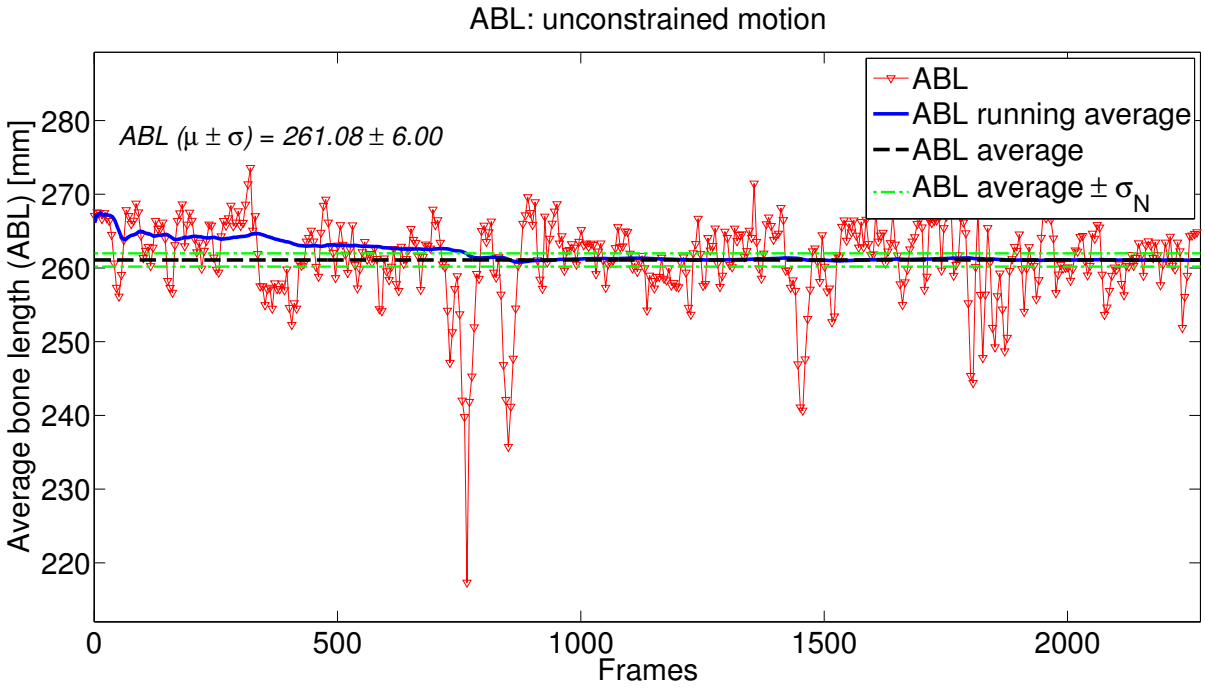

Figure A.3: *Average bone length during unconstrained motion.*

	Slow motion	
Run	**Mean[mm]**	**Std[mm]**
1	261.2	9.3
2	260.5	9.2
3	259.4	10.7
4	262.0	9.0
5	260.3	10.1
Average	260.7	9.6

Table A.2: Table reporting ABL mean and standard deviation relative to slow motion. Values obtained performing the motion for one minute each run.

	Unconstrained motion	
Run	**Mean[mm]**	**Std[mm]**
1	262.5	6.8
2	263.4	4.8
3	264.3	4.4
4	261.5	6.2
5	263.8	5.1
Average	263.1	5.5

Table A.3: Table reporting ABL mean and standard deviation relative to unconstrained motion. Values obtained performing the motion for one minute each run.

B Gauss-Newton algorithm

A common method used to solve non-linear least squares problems is the Gauss-Newton algorithm, which will be presented in the following. Let us consider the observation pairs $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ and a model curve f which is parametrized by $k = [k_1, \ldots, k_m]$. Then the objective function is defined:

$$E(k) = \frac{1}{2}\sum_{i=1}^{N}[y_i - f(x_i, k)]^2 \tag{B.1}$$

By setting $r_i(k) = y_i - f(x_i, k)$, the previous expression can be rewritten as:

$$E(k) = \frac{1}{2}\sum_{i=1}^{N} r_i(k)^2 = \frac{1}{2}r(k)^T r(k) \qquad \text{where } r^T = [r_1, \ldots, r_N] \tag{B.2}$$

The purpose of the G-N algorithm is to find the value of k such that the cost function E is minimal. In the following, for clarity, a notation simplification will be utilized by writing E_k in place of $E(k)$ and similarly for the other functions of k. The first step is to provide an initial guess k_0 that is assumed to be close to the global minimum. Then the cost function is Taylor expanded to the second order, so that:

$$\overline{E}_k = E_0 + \nabla E_0 \cdot (k - k_0) + \frac{1}{2}(k - k_0)^T \cdot \nabla^2 E_0 \cdot (k - k_0) \tag{B.3}$$

where ∇E_0 and $\nabla^2 E_0$ are respectively the gradient and the Hessian matrix of the cost function evaluated in k_0. Let us derive the gradient of the cost function:

$$\nabla E_k = \frac{1}{2}\nabla[r_k^T r_k] = \frac{1}{2}[J_k^T r_k + [r_k^T J_k]^T] = J_k^T r_k \tag{B.4}$$

where J_k is the Jacobian of the residual vector r_k. The Hessian matrix is given by:

$$\nabla^2 E_k = \frac{1}{2}\nabla^2[r_k^T r_k] = \nabla[J_k^T r_k] = J_k^T J_k + \sum_{i=1}^{N} r_{k_i}\nabla^2 r_{k_i} \tag{B.5}$$

and by setting $Q = \sum_{i=1}^{N} r_{k_i}\nabla^2 r_{k_i}$, the more compact expression is obtained:

$$\nabla^2 E_k = J_k^T J_k + Q \tag{B.6}$$

The Hessian matrix of the cost function is given by the sum of two terms, one of which only contains first order derivatives. When k is a critical point, the derivative of the truncated cost function is vanishing, so that:

$$\frac{d\overline{E}_k}{dk} = J_0^T r_0 + J_0^T J_0(k - k_0) + \frac{1}{2}\frac{d}{dk}[(k - k_0)^T Q_0 (k - k_0)] = 0 \tag{B.7}$$

In the approximation $Q = 0$, the update rule of the G-N algorithm is finally obtained:

$$J_0^T r_0 = -J_0^T J_0(k - k_0) \tag{B.8}$$

C Tables

C.1 Results of 30 motion sequences

	Kinect		**AICP**	
Run	**Mean[Δ]**	**Std[Δ]**	**Mean[Δ]**	**Std[Δ]**
1	7.4	8.0	2.6	2.5
2	20.3	20.2	1.5	5.1
3	35.2	18.6	9.6	4.7
4	30.4	22.3	0.8	5.8
5	14.1	9.4	1.4	4.0
6	23.7	14.3	4.3	5.4
7	8.4	11.4	1.3	2.9
8	22.6	17.2	5.3	5.2
9	22.4	15.5	-0.1	4.8
10	47.2	25.3	2.0	4.5
11	27.4	29.0	2.4	5.0
12	30.0	34.3	3.4	3.9
13	28.2	11.1	5.4	5.7
14	38.6	66.8	3.4	4.6
15	21.6	21.5	6.3	6.0
16	28.7	16.8	7.4	6.6
17	27.6	16.3	4.2	7.8
18	32.3	20.6	5.2	7.2
19	20.0	15.6	1.4	4.6
20	35.1	33.2	3.4	3.7
21	14.0	27.1	3.1	3.0
22	42.9	36.8	6.9	4.8
23	17.0	13.0	1.5	5.6
24	19.9	16.6	3.1	3.7
25	20.0	13.6	4.0	4.7
26	8.8	8.5	0.6	4.0
27	15.2	9.9	2.7	3.9
28	14.2	12.9	3.6	3.5
29	31.9	24.3	3.9	5.7
30	31.5	37.2	8.2	5.5

Table C.1: Table of results relative to 30 unconstrained motion sequences lasting one minute each.

C.2 Sub-sampling factor

k	FPS	Point cloud size	NREP[Δ]	Std NREP[Δ]
1	20.2	2318	6.0	7.8
2	25.6	1174	3.7	5.8
3	29.5	795	4.2	6.6
4	29.7	583	4.4	11.0

Table C.2: Mean values of frame rate, NREP, and point cloud size for various sub-sampling factor k. The values refer to a ten minute long unconstrained motion sequence. The program was run on a PC equipped with cpu Intel Core i5-3570K 3.50 GHz.

C.3 Relative residual error threshold

k	FPS	Std	Iterations	Std	Mean NREP[Δ]	Std NREP[Δ]
0.1	29.8	2.4	1.5	0.5	13.0	15.0
0.05	29.7	2.4	1.9	0.5	4.6	11.0
0.01	28.1	3.2	2.9	1.3	4.5	8.0
0.005	26.4	3.7	3.43	1.6	1.2	6.2
0.001	22.5	4.5	5.7	3.0	6.7	11.0

Table C.3: Frame rate, NREP, and iterations of Gauss-Newton optimization for various values of relative residual error thresholds. The values refer to a ten minute long unconstrained motion sequence. The program was run on a PC equipped with cpu Intel Core i5-3570K 3.50 GHz.