

Playmancer– FP7 215839

A European Serious Gaming 3D Environment

Deliverable

D4.3: Final Playmancer multi-modal multi-player gaming platform prototype

Date of delivery: March 31st, 2010
Revised: August 4th, 2011

Authors: TUW, UOP, ST, SGI
Date: 05/09/11 Version: v1.1

Document Control

Title:	D4.3: Final Playmancer multi-modal multi-player gaming platform prototype	
Project:	Playmancer (FP7 215839)	
Type:	Deliverable	Status*: Restricted
Authors:	TUW: Thomas Pintaric, Christian Schönauer	
Origin:	TUW	
Doc ID:	1.1	

Amendment History

Version	Date	Author	Description/Comments
v0.1	16. Feb 10	TUW	Template
V0.5	12. Mar 10	TUW, SGI	Working draft
V0.6	15. Mar 10	TUW, SGI	Improved draft
V0.9	26. Mar 10	TUW, SGI	Final draft (for internal review)
V1.0	31. Mar 10	TUW, SGI	Final deliverable
V1.1	01. Aug 11	TUW	Final improved deliverable (following final review comments)

The information contained in this report is subject to change without notice and should not be construed as a commitment by any members of the **PLAYMANCER** Consortium. The **PLAYMANCER** Consortium assumes no responsibility for the use or inability to use any software or algorithms, which might be described in this report. The information is provided without any warranty of any kind and the **PLAYMANCER** Consortium expressly disclaims all implied warranties, including but not limited to the implied warranties of merchantability and fitness for a particular use.

Table of contents

1	INTRODUCTION.....	7
2	FULL-BODY MOTION-TRACKING FINAL PROTOTYPE	8
2.1	OVERVIEW	8
2.2	SKELETON CALIBRATION	9
2.2.1	Overview	9
2.2.2	Skeleton Calibraton-Assistant	9
2.2.3	Skeleton Calibration Tool	10
2.2.4	Visualization Tool	12
2.3	SKELETON TRACKING MODULE	14
2.3.1	Overview	14
2.3.2	Skeletal Model Acquisition.....	14
2.3.3	Standard Rigid-Body Model-Fitting Algorithm.....	15
2.3.4	Algorithmic Extensions for Kinematic Chains of Rigid-Body Targets.....	17
2.3.5	Skeleton Tracking Algorithm.....	18
2.3.6	Points-of-Interest.....	22
Virtual Marker-Sets	23	
Relevant Positions.....	23	
Specification	23	
Evaluation.....	24	
2.3.7	Evaluation of the Skeleton Tracking Module	26
2.4	INTEGRATION WITH 3 RD -PARTY CHARACTER ANIMATION SOFTWARE	29
2.4.1	Introduction to Autodesk MotionBuilder.....	29
2.4.2	Integration with MotionBuilder	29
2.5	INTEGRATION WITH 3 RD -PARTY MOTION ANALYSIS SOFTWARE	31
2.5.1	Introduction to C-Motion Visual3D.....	31
2.5.2	Integration with Visual3D.....	31
2.5.3	Evaluation of motion-data using Visual3d.....	32
2.6	COMMERCIAL ASPECTS OF THE IOTRACKER MOTION CAPTURE SYSTEM	33
3	MULTI-MODAL DATA-FLOW FRAMEWORK FINAL PROTOTYPE.....	37
3.1	OVERVIEW	37
3.2	OPENTRACKER INPUT DEVICE INTEGRATION.....	37
3.2.1	Motion-Tracking Modules	37
3.2.1.1	Rigid-Body Target-Tracking Module.....	37
3.2.1.2	Full-Body Motion-Tracking Module.....	37
3.2.2	Biosignal Sensor Module.....	38
4	INTEGRATION WITH UNITY3D	39
4.1	OVERVIEW	39
4.2	ARCHITECTURE	39
4.2.1	Input.....	39
4.2.2	Output.....	40
4.3	INPUT DEVICE SPECIFIC IMPLEMENTATION DETAILS	41
4.3.1	Body-state-information	42
4.3.2	Motion-tracking.....	42

D4.3: Final Playmancer multi-modal multi-player gaming platform prototype	4/60
4.3.3 <i>Biosignals</i>	43
4.4 GAME PROTOTYPE	44
4.4.1 <i>Early Test Prototype</i>	44
4.4.2 <i>Final Prototype</i>	46
5 SET-UP AND GAME-WORKFLOW	51
5.1 INSTRUMENTATION OF PATIENTS	51
5.2 CALIBRATION	53
5.2.1 <i>Calibration from Technical Point of View</i>	55
5.3 MINI-GAMES	56
6 CONCLUSION AND NEXT STEPS	57
6.1 FULL-BODY MOTION-CAPTURE (WP 4.1)	57
6.2 MULTI-MODAL DATA-FLOW FRAMEWORK (WP 4.3)	57
6.3 INTEGRATION WITH UNITY3D (WP 4.6)	57
7 APPENDICES	58
7.1 BASELINE ASSESSMENT & GOAL SETTING	58
7.2 OVERVIEW OF THE TECHNICAL EVALUATION OF MOTION TRACKING	60

Table of Figures

Figure 1: Skeleton calibration assistant showing the calibration-movement to the user	11
Figure 2: Screenshot of the Skeleton Calibration tool showing the GUI exposed to the therapist and a visualization of a skeleton.....	11
Figure 3: Kinematic model for skeleton tracking.....	15
Figure 4: Quadratic-time complexity reduction (top). Formulation of the model-fitting problem as maximum-clique search (bottom).	16
Figure 5: Distance limits for markers on adjacent bones.....	17
Figure 6: Pair-wise distances of multiple markers on adjacent bones.....	18
Figure 7: Skeleton Tracking Algorithm.	21
Figure 8: Points-of-interest visualized for two different physical marker-sets and independently calibrated skeletons (left column showing the first marker-set, right column showing the second). Please note that the avatar is just added to better illustrate the locations of the POIs and not perfectly aligned with the real-world-position. Therefore, POIs on the right seem slightly more up and closer to the viewer than on the left.	22
Figure 9: Visualizations of the two different marker-sets. Top row shows a visualization of the point-clouds including labels of the markers as generated by the tracking-software. Lower row visualizes the local bone-coordinate systems of both marker-sets. On the left also marker-labels are displayed, while the right picture shows an error estimate of the joint.....	25
Figure 10: Two MoCap-suits with different marker-sets (passive markers).....	26
Figure 11: Tests of the skeleton tracking module using a MoCap-suit with active markers	27
Figure 12: MotionBuilder integration via the OpenReality SDK.....	29
Figure 13: MoCap data is captured using a MoCap suit and the iotracker (visualized in the lower right corner using the iotracker-client) The data (a labelled point cloud) is then streamed to MotionBuilder, where it can be attached to a character. The pose of the character is then adapted to the tracking data in real time.	30
Figure 14: Screenshot of the iotracker-client and Visual3d showing visualizations of the local bone-coordinate-systems as well as the POIs with their labels. The thigh-bone shows exemplary how a bone is scaled to the POIs and animated live by using POI-positions.....	32
Figure 15: Integration with C-Motion Visual3d (via Real-Time Plug-In).....	33
Figure 16: Receiving input for a game in Unity.....	40
Figure 17: Using an output device from a game in Unity.....	41
Figure 18: Input device hierarchy	41
Figure 19: Prototype while testing skeletal changes on a server and client	43
Figure 20: Example of how input device configuration may be very minigame specific	44
Figure 21: Game prototype running in Unity.....	45
Figure 22: Server prototype interface.....	46
Figure 23. Our MoCap system and two of the mini games during the preliminary testing. „Face of Chronos“ (left), „Three Wind Gods“ (right)	47
Figure 24: Screenshots from the game-world currently under development	49
Figure 25: Screenshot from the game-world currently under development	50

Figure 26: Workflow of a therapy-session 52
Figure 27: Surface-EMG-electrode-placement..... 53

1 Introduction

This report describes the final prototype implementation of the input and output components for the rehabilitation Playmancer game. It outlines the features of the final prototype and work done on the tasks of WP4 since deliverable D4.2.

The remainder of this document is structured according to the tasks of work package WP4. Tasks 4.2, 4.4. and 4.5 were discontinued following the last review¹ and the Revised Technical Annex². Therefore, progress made in these two tasks since the deliverable D4.2 is not reflected in this deliverable. Task 4.6, the integration of the platform components with UNITY3D³ game engine will be covered in more detail, since it has only been superficially discussed in D4.2. For tasks 4.1 and 4.3 previous achievements will be summed up and recent developments documented in more detail.

Consequently, the report is organized into the following major sections:

- **Full-Body Motion-Tracking Final Prototype**
(corresponds to WP4.1).
- **Multi-Modal Data-Flow Framework Final Prototype**
(corresponds to WP4.3).
- **Integration with Unity3D Final Prototype**
(corresponds to WP4.6).

¹ November, 23rd 2009

² Technical Annex version 3.2.

³ <http://unity3d.com>

2 Full-Body Motion-Tracking Final Prototype

2.1 Overview

This section outlines the final product of task T4.1 (*"Development of a Motion tracker and integration with the Playmancer game platform"*). The primary output of this task is a full-body motion capture system that is fully integrated with the Playmancer gaming platform.

The main effort for D4.3 was aimed at extending and improving the functional prototypes of the tools and algorithms proposed in D4.1 and D4.2, which can be categorized into three groups:

1. Rigid-Body Tracking.
2. Skeleton ("Full-Body") Tracking.
3. Integration with 3rd-party Visualization Software.

(NOTE: Rigid-body tracking and skeleton-based tracking are two different, albeit related problems and require distinct methodologies and algorithms.)

All implemented prototypes operate as extensions to the **iotracker** motion-tracking system (see Deliverable D4.1, Section 2.2). Detailed information on these extensions and their current implementation status can be found in the following sub-sections:

- **Skeleton Calibration**
(Implementation status: **fully functional prototype**).
- **Skeleton Tracking Module**
(Implementation status: **fully functional prototype**).
- **Integration with 3rd-party Character Animation Software**
(Implementation status: **fully functional prototype**).
- **Integration with 3rd-party Motion Analysis Software**
(Implementation status: **fully functional prototype**).

2.2 Skeleton Calibration

2.2.1 Overview

This section describes the final prototype of the skeleton calibration tool and associated software. The focus is being placed on work done since the intermediate prototype documented in D4.2 section 2.3. Improvements were mainly concerned with usability and workflow. The major improvements include a graphical user-interface as well as an assistant helping the patient and therapist with the recording of the calibration movements. For detailed descriptions of the basic functionality please refer to deliverables D4.1 section 2.4 and D4.2 section 2.3.

The purpose of this tool is to automatically generate an approximated skeleton-model purely from marker positions during some initial calibration movements in an offline process. This approach enables the game setup to easily and quickly adapt to different patients without the need to take measurements by hand or place markers on precisely predefined positions, which are both time-consuming matters. Furthermore, the skeleton-model can be generated for an arbitrary structure connected by rotational joints. In the context of rehabilitation this means that we can easily create skeleton-models of the body-parts of interest by placing markers on them and running the algorithm. Thus, skeleton-models for generic Motion Capture input for serious games can easily be generated (e.g. for new rehabilitation exercises). The offline nature of the tool results from the use of computationally expensive algorithms like non-linear-optimization or clustering. This, however, has no influence on the MoCap-process itself, which is performed in real-time. The following subsections describe the tools necessary to perform the calibration:

- **Skeleton Calibration Assistant**
(Implementation status: **fully functional prototype**).
- **Skeleton Calibration Tool**
(Implementation status: **fully functional prototype**).
- **Visualization Tool**
(Implementation status: **fully functional prototype**).

2.2.2 Skeleton Calibraton-Assistant

In order to produce a customized skeleton-model for a user, the skeleton calibration tool needs movement recordings of that specific user. It is an important factor of the success of the skeleton calibration that all joints, which are incorporated in the skeleton-model, are exercised in these recordings. Tests, documented in section 4.2.2 of deliverable D8.1, showed that it is rather difficult to communicate the necessary movements only by verbal instruction. We therefore decided to implement an assistant in Unity3D showing the patient how to perform these movements on a projection screen. These, however, are only suggestions, which have been proven to

work well during our evaluation. If the patient for medical reasons is not able to perform a certain movement, the therapist is free to suggest a different one without compromising the success of the calibration.

The assistant loads the animation-data from an xml-file, which uses a format similar to the one specified for the interface between Body-state-fusion and Unity3d as described in deliverable D3.1 section 3.1 (i.e. the XML-packets usually received over the network are saved/loaded from file). The therapist controls the assistant using a graphical user-interface and/or shortcuts, while a patient is performing the required calibration-movements (in Figure 1 a screenshot of the skeleton calibration assistant is depicted). When necessary, the calibration assistant is able to control the iotracker-server using a XML-RPC-call. This is required for starting/stopping the recording of data sets, setting parameters within the server and finally loading the final skeleton-model for further processing.

Tests with multiple users using a motion-suit with passive markers (as depicted in Figure 10) as well as one with active markers (see Figure 11) showed good results. We integrated the calibration assistant in the final workflow as separate tool, but with access to the user profile. Therefore, calibration information can be easily updated in the profile every time the calibration is performed.

2.2.3 Skeleton Calibration Tool

The skeleton calibration tool loads the recorded movement-files and displays the filenames in the graphical user interface (Figure 2 shows a screenshot of the GUI). The tool offers two modes of operation for the calibration:

1. Standard mode, which requires the user to select the movement-recordings and manually initiates calibration of the skeleton.
2. Automatic mode, where the program checks for new recordings in the background and automatically calibrates for each new data set.

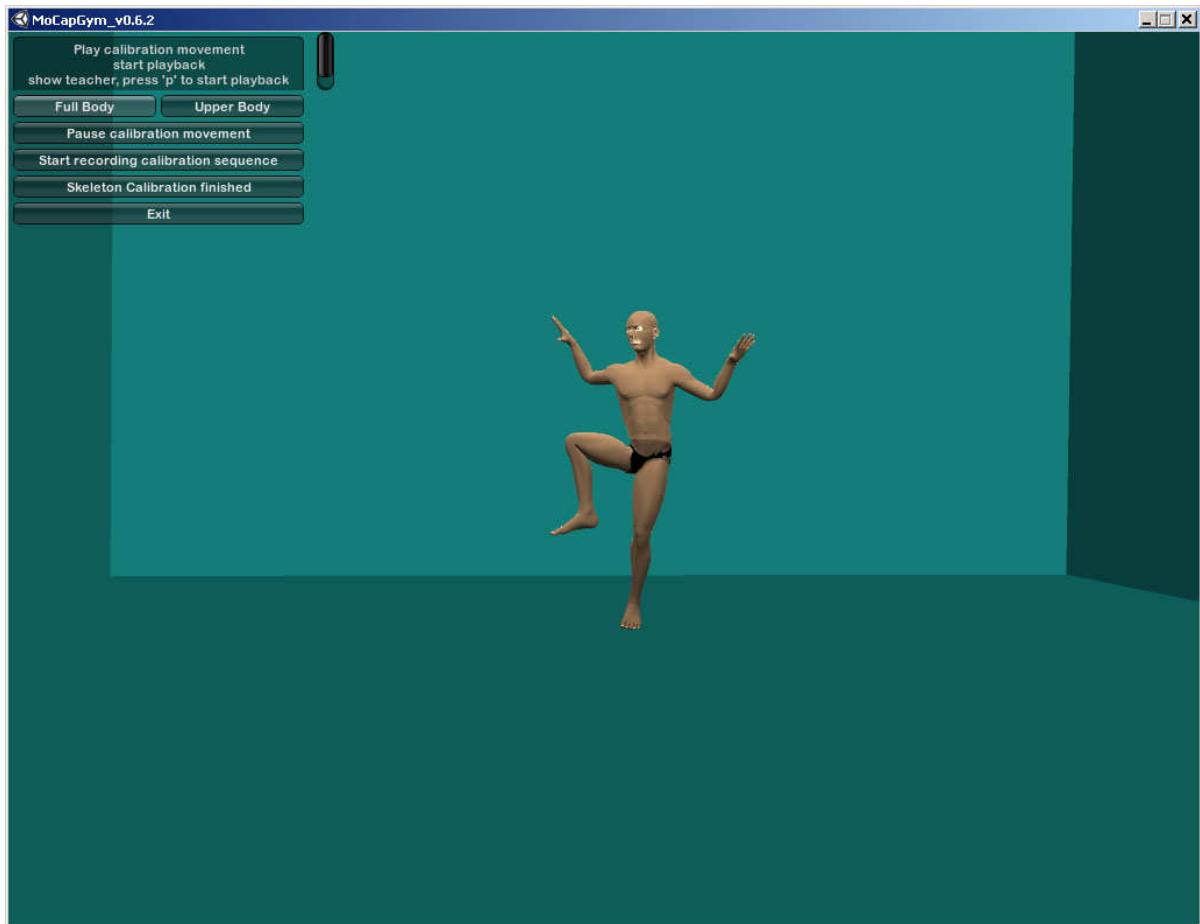


Figure 1: Skeleton calibration assistant showing the calibration-movement to the user

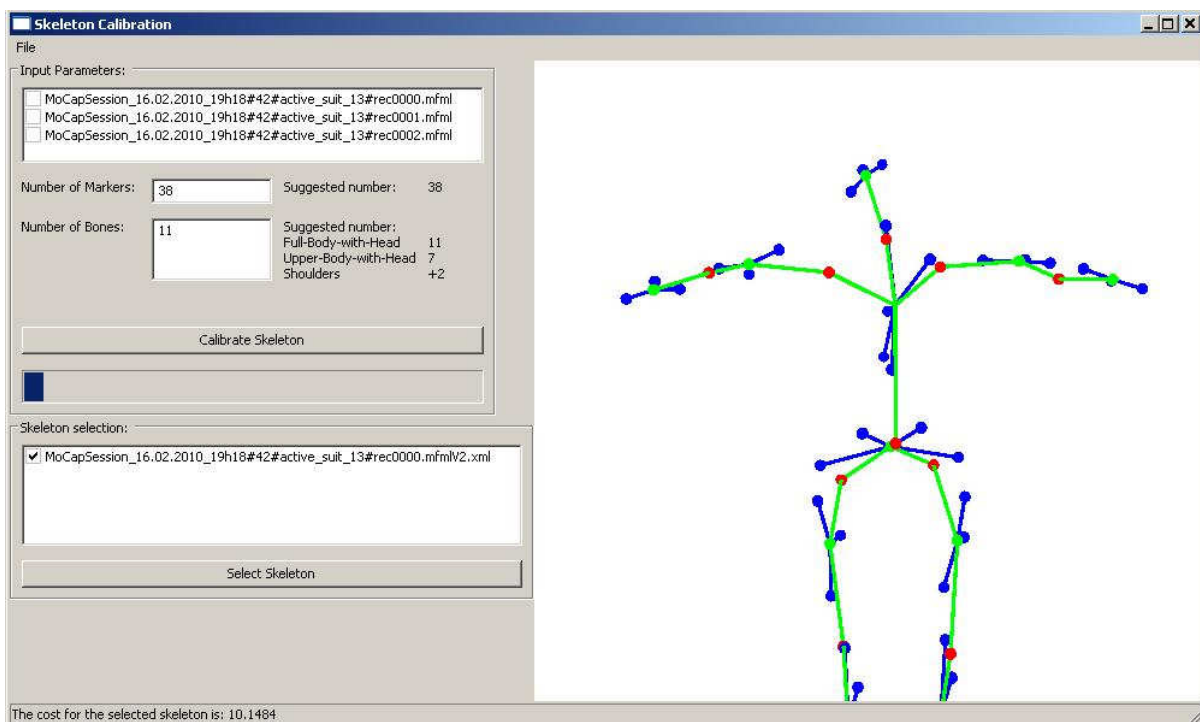


Figure 2: Screenshot of the Skeleton Calibration tool showing the GUI exposed to the therapist and a visualization of a skeleton

Skeleton calibration is an offline-procedure during which skeleton-models are calculated sequentially displaying the result as a filename in the list box as well as graphical representation. Due to the offline-nature of this process, sophisticated algorithms such as non-linear optimization or spectral clustering can be used.

The tool automatically selects the best skeleton-model for further processing; however, the therapist is able to override this selection manually. Automatic selection is based on the joint-optimization-cost of a skeleton. As pointed out in section 2.4 of deliverable D4.1 the calculation of the joint positions (and the skeleton parameters) is based on a non-linear optimization algorithm. The results of this minimization are compared for the skeletons and the skeleton with the smallest cost chosen.

Finally, the choice of a skeleton is acknowledged and is automatically labeled with bone-names by comparison with a predefined template. This is important in order to map the joint angles to the avatar in the game. The labeled skeleton is then handed to the iotracker-server and the calibration tool closed. The MoCap-module of the server loads the skeleton and starts the MoCap process waiting for the games to connect.

Pre-tests described in deliverable 8.1 have shown good usability for the skeleton calibration procedure. It was clear to the participants how to execute the calibration movements and easy to perform them. In almost all cases the automatic selection of the best skeleton has been acknowledged by the therapist. For the others the difference in the quality of the skeleton model was minimal and it can be safely assumed that the deliberate choice by the therapist or advising technician of a certain skeleton model after visual evaluation was of little to no influence to the skeleton tracking quality. Also during the field trial the usability for the skeleton calibration procedure was good. All patients were capable to perform the requested skeleton calibration exercise. Some needed extra instruction and time to learn the exercise. Therefore, for further use of the Playmancer game in rehabilitation we intend to use easier skeleton calibration exercises.

As has been described in deliverables D4.1 and D4.2 the skeleton calibration algorithm is flexible enough to cope with arbitrary articulated structures. Therefore, a skeleton can easily be calibrated for people with a significantly different body model. For wheel chair users for example the skeleton can be calibrated for upper-body only. For amputees the suit can be adjusted with Velcro-strips. For skeleton tracking, usually the torso is required as root of the skeleton model. This can be overridden by an arbitrary bone, might however result in poorer tracking quality. Nevertheless, it has to be stated, that people with such restrictions do not meet the inclusion criteria of the study with chronic pain patients, as described in D8.2. Also it is not listed as requirement in D2.4. Pre-tests also showed, that for subjects suffering from restrictions resulting from age or pain no negative influence on the calibration procedure in terms of calibration quality can be noticed.

2.2.4 Visualization Tool

Parts of the visualization tool have been incorporated in the skeleton calibration tool to provide the therapist with visual feedback (as can be seen in Figure 1). Please refer to the previous subsection for details on that.

2.3 Skeleton Tracking Module

2.3.1 Overview

This section documents the final prototype of the skeleton tracking module.

In order to be able to track human body posture robustly and without the need to attach unique rigid-body targets to every limb, several extensions were made to the standard rigid-body model-fitting algorithm (see Deliverable D4.1, Section 2.5.2 for a detailed discussion, part of which is summarized in Section 2.3.3). Some of the extensions have already been documented in deliverable D4.2 section 2.4.

- **Skeleton Tracking Module**
(Implementation status: **fully functional prototype**).

Tests with the final prototype have shown that the design-goals described in D4.1 section 2.5.1 have been reached.

- The computational complexity of the skeleton-tracking task has been optimized for real-time performance with very little latency.
- The skeletal pose can be estimated with high robustness and accuracy.

2.3.2 Skeletal Model Acquisition

Using the Skeleton Calibration Assistant described in Section 2.2, we are able to reconstruct a subject-specific full-body kinematic constraint model from a pre-recorded sequence of marker positions. This kinematic model is stored as a hierarchical chain of rigid bones, connected by rotational joints. Every bone is associated with one or more optical markers, the position of which relative to the bone remains static.

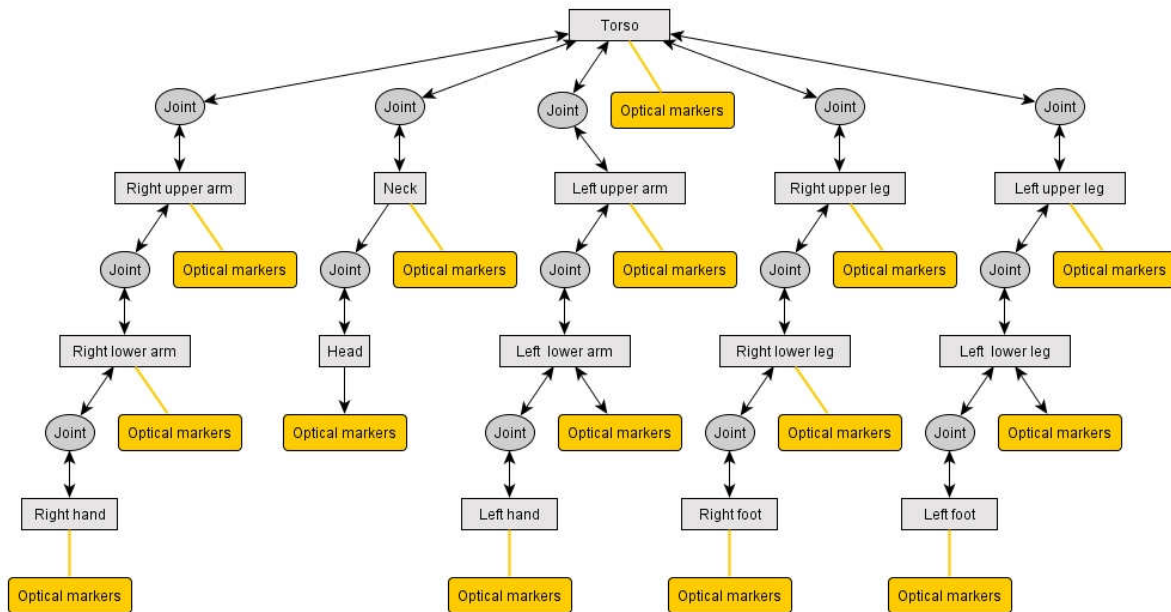


Figure 3: Kinematic model for skeleton tracking.

The full parameterization of the skeletal model is thus given by the set of all joint angles plus the 6-DOF transformation (position + orientation) of the root limb. The torso, as the center of the body, is usually chosen as the root of the kinematic tree. During the calibration procedure, the Skeleton Calibration Assistant will automatically identify the torso (or any other bone of choice) and label it as the root bone.

Effort has been invested mainly into improving the algorithm's handling of skeleton-models and the model-internal errors (i.e. idealized rotational joints and markers changing their position relative to the bone due to muscle- and skin-movement).

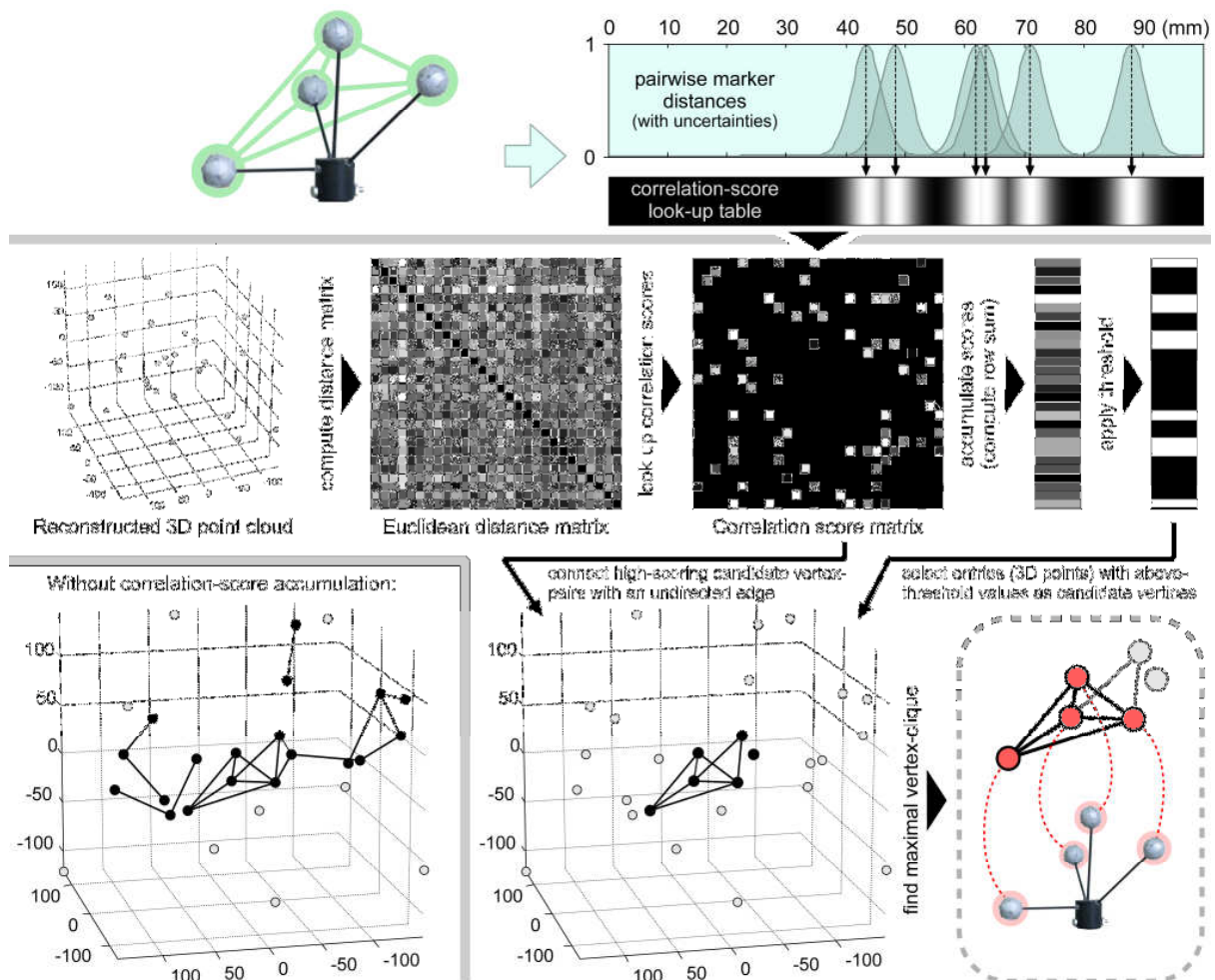
2.3.3 Standard Rigid-Body Model-Fitting Algorithm

Our system attempts to uniquely identify the markers of each rigid-body target ("model") within a larger, unstructured set of observed markers ("observation") by comparing the Euclidean distances between all pairs of observed markers with the known distances between markers of each target. To account for the presence of noise and different types of process errors, two distances \mathbf{d}_i and \mathbf{d}_j are considered equal if $|\mathbf{d}_i - \mathbf{d}_j| < \epsilon$.

We formulate this combinatorial problem in a way that allows for the application of a maximum-clique search. Prior to running the maximum-clique search (which is NP-hard), we apply a polynomial-time complexity-reduction heuristic that takes advantage of an important constraint.

Our complexity-reduction technique is based on a form of geometric hashing: At startup, we compute the Euclidean marker-distance matrix for target and store its entries in a one-dimensional look-up table (see Figure 4, first row). Measurement uncertainties are assumed to follow a Gaussian distribution whose standard deviation is chosen to coincide with the tracker's absolute accuracy. We refer to the

entries stored in this look-up table as *correlation scores*, because they represent the probability with which any given marker-pair (or more precisely, their separating distance) is part of the same target.



**Figure 4: Quadratic-time complexity reduction (top).
Formulation of the model-fitting problem as maximum-clique search (bottom).**

At runtime, we calculate the Euclidean distance between every pair of observed 3D markers for a given frame. For every entry in the Euclidean distance matrix (EDM), we look up its corresponding *correlation score* and store it in a separate Correlation score matrix (CSM). From the CSM, we compute a vector containing the accumulated correlation scores for every observed marker through row- or column-wise summation.

All of these operations can be carried out in a single parallel pass without any inter-thread coordination overhead. A visualization of the procedure is shown in Figure 4 (second row). As can be seen in the image, the CSM will contain mostly near-zero entries, indicating that the majority of marker-pairs do not belong to the same solution-tuple. In other words: the optimal solution to the model-fitting problem is highly unlikely to contain both of the two markers.

Depending on the number n of model-markers, we also expect every member of the solution-set to have an accumulated correlation score of at least $T_n = t * (n-1)$, where $t \in [0.5, 1]$. The easiest way to enforce this constraint is to apply threshold T_n to the accumulation-vector. Afterwards, every marker whose thresholded entry is zero can be eliminated as a candidate.

According to our observations, this elimination procedure will reliably reduce the number of candidate markers by up to 90% (in a real-life scenario, not degenerate test-cases), without being prone to false negatives.

2.3.4 Algorithmic Extensions for Kinematic Chains of Rigid-Body Targets

We extended the complexity reduction algorithm described in Section 2.3.3 from a single rigid-body target to kinematic constraint models by pre-computing additional multi-dimensional marker-to-marker distance lookup tables from the calibrated skeleton. This computation is performed as a separate offline step.

Figure 5 shows a simple two-bone skeleton fragment connected by a spherical joint, where each bone has exactly one rigidly attached marker (“marker1” and “marker2”). In an unstructured point-cloud, only markers that are found to lie at a distance between d_{min} and d_{max} from Marker1 should be considered likely to be labeled Marker2. The value-pair $[d_{min}, d_{max}]$ is computed a-priori for every pair-wise combination of markers on adjacent bones.

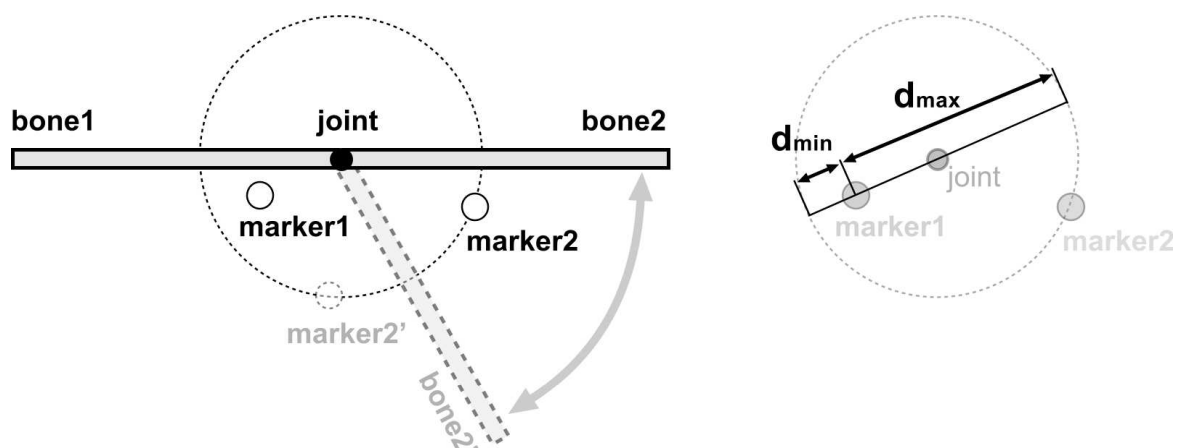


Figure 5: Distance limits for markers on adjacent bones.

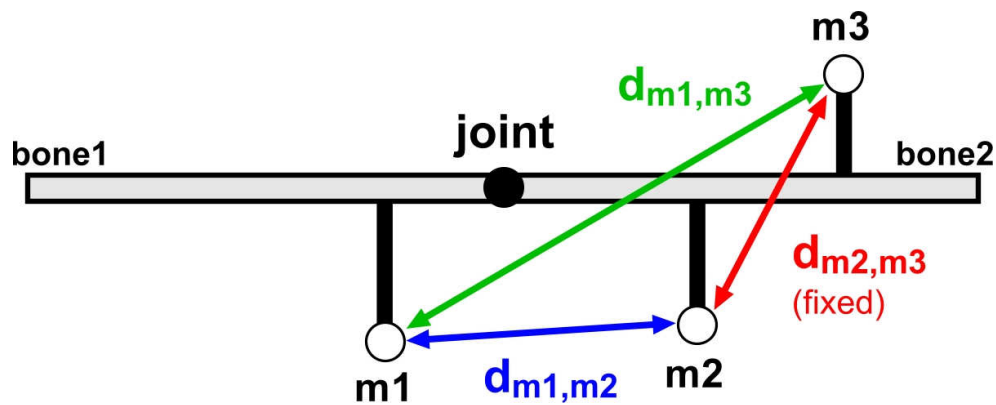


Figure 6: Pair-wise distances of multiple markers on adjacent bones.

Distance lookup-tables can be extended to span multiple markers on adjacent bones.

Example: For any given distance $d_{m1,m2}$ (where $d_{\min} \leq d_{m1,m2} \leq d_{\max}$) between marker **m1** (on bone1) and marker **m2** (on bone2), distance $d_{m1,m3}$ between marker **m1** and marker **m3** (on bone2) can be stored in a lookup table. For reasons of efficiency and memory usage, distance values are discretized with step sizes of multiple millimeters. Figure 6 illustrates this concept.

In an unstructured point-cloud, unique marker identities can only be established by inspecting known pair-wise distances between markers. Often, multiple similar distances between markers can be observed in the entire point-cloud, which leads to labeling ambiguities. Due to the relatively large number of markers in a motion-capture configuration, it is not enough to just use the known pair-wise distances between markers on the same bone in order to establish unique marker labels. Using the distance lookup tables described above, we iteratively inspect the distance relationships between markers on adjacent bones. This allows for significant improvements in marker labeling accuracy, with minimal computational overhead.

2.3.5 Skeleton Tracking Algorithm

Our extended skeleton-model-fitting algorithm is composed of the following steps (see Figure 7):

Step 1: Predict the expected body posture.

In the event that no previous body pose estimates exist, which is the case upon first run or re-initialization after a loss of tracking, we use a generic body posture (a standing pose or the so-called T-pose) as prediction. Otherwise, we use predictive forward kinematics to compute the expected body posture from the last-known pose.

This was implemented in the following manner:

For a kinematic chain of n bones, let θ_i be the joint-angle between bone i and bone $i-1$. Given $\theta_1 \dots \theta_n$, the frame of bone n relative to bone 0 is:

$${}^0T_n = \prod_{i=1}^n {}^{i-1}T_i(\theta_i)$$

Where ${}^{i-1}T_i(\theta_i)$ is the transformation matrix from the frame of bone i to bone $i-1$. In absence of a valid measurement for θ_k ($k = 1..n$), it is not possible to compute ${}^{i-1}T_i(\theta_i)$ for any $i \geq k$.

We solve this problem by employing a double-exponential smoothing-based prediction filter (DESP)⁴ for every joint angle θ_i , which provides us with estimates of the joint angle in absence of actual measurements.

Step 2: Perform rigid-body model-fitting, build a classification graph.

For every limb with two or more rigidly attached optical markers, we perform an independent full rigid-body model-fitting, comprising of the complexity reduction step and maximum-clique graph search, as described in Sections 2.3.3 and 2.3.4.

However, instead of immediately deciding on a unique solution, as we would do when searching for “regular” rigid-body targets, we store all hypotheses whose model-alignment error lies below a certain threshold in a classification graph.

The classification graph has the following structure:

Every vertex in the classification graph represents a unique model-to-observation⁵ relationship. Edges represent geometric transformations and are computed upon insertion of new vertices.

Step 3: Classification graph pruning.

For every edge (i.e. model-to-observation hypothesis) in the classification graph, we first compute the absolute pose of the associated limb in as many degrees of freedom as possible. We then test this pose against the expected posture (determined in Step 1). If the difference between both poses exceeds a certain (experimentally determined) threshold, the vertex and all of its edges are removed from the graph.

We then test the remaining edges of the classification graph against the kinematic constraints of the skeletal model by pair-wise examination of

⁴ Joseph LaViola, “Double exponential smoothing: an alternative to Kalman filter-based predictive tracking”, In Proceedings of EGVE '03: Eurographics Workshop on Virtual Environments, 2003.

⁵ In this context, “*observation*” means a group of markers that was detected by the tracking system and “*model*” means the optical markers attached to a limb of the skeletal model (see Figure 3).

adjacent limbs. When we find an edge that does not satisfy the kinematic constraints (within a certain threshold), we remove both connected vertices from the classification graph. For this purpose, we use the distance lookup-tables described in Section 2.3.4.

Step 4: Fill in missing information.

In the event that occlusions prevent parts of the inner kinematic chain (e.g. upper or lower arm) from being unambiguously reconstructed, but the end-effector pose (hands, feet) is known, we attempt to solve for middle joint pose by an iterative inverse-kinematics approach⁶.

From here, we repeat Step 3 and 4 until all ambiguities are resolved.

Step 5: Posture estimation.

In most cases, all model-to-observation ambiguities are resolved in Step 3. In the rare event that there are multiple hypotheses remaining (i.e. a model is associated with multiple observations), we assign scores to the hypotheses based on the model-alignment error (least-squares distance between observed markers and ideal marker positions).

We then select the highest-scoring hypothesis from the classification graph and remove the remaining hypotheses. We label all observations according to their associated model. If necessary, occluded optical markers are substituted with “virtual markers”.

Based on the uniquely labelled optical marker-set, we iteratively compute the full skeleton parameterization (using forward kinematics) with the goal of minimizing the model-alignment error (least-squares distance between observed markers and ideal marker positions).

⁶ Samuel R. Buss and Jin-Su Kim, "Selectively Damped Least Squares for Inverse Kinematics", In Journal of Graphics Tools, vol. 10, no. 3 (2005) 37-49.

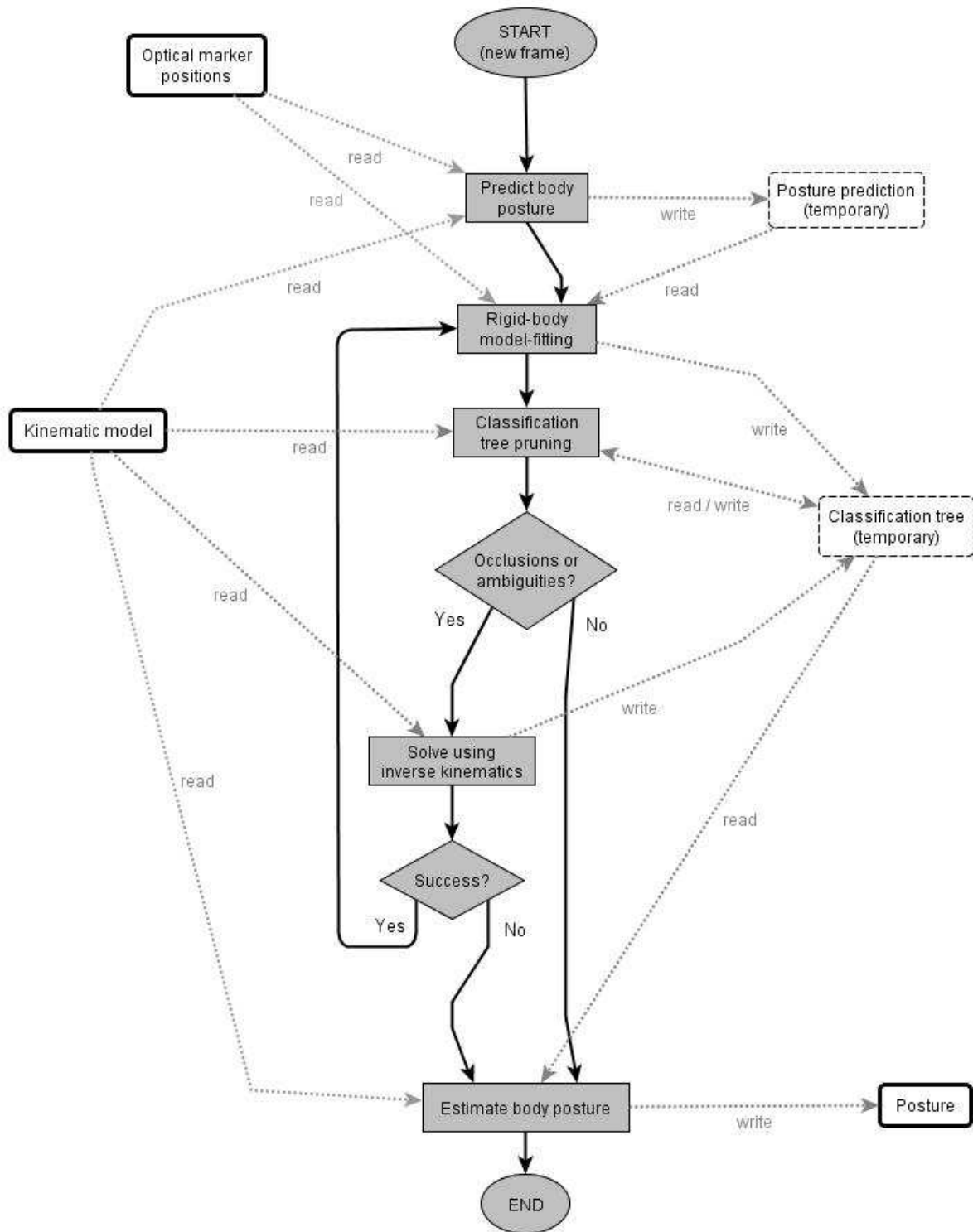


Figure 7: Skeleton Tracking Algorithm.

2.3.6 Points-of-Interest

A Point-of-interest (POI) is a specific position defined within the skeleton-model. They can be defined relative to a joint while being located on a bone. POIs have two main purposes:

1. POIs can be used to define a set of virtual markers
2. POIs can define positions on the skeleton that have a special relevance for (medical) application (e.g. heel position for gait-analysis or a point on the upper position of the shoulder to detect shoulder-elevation)

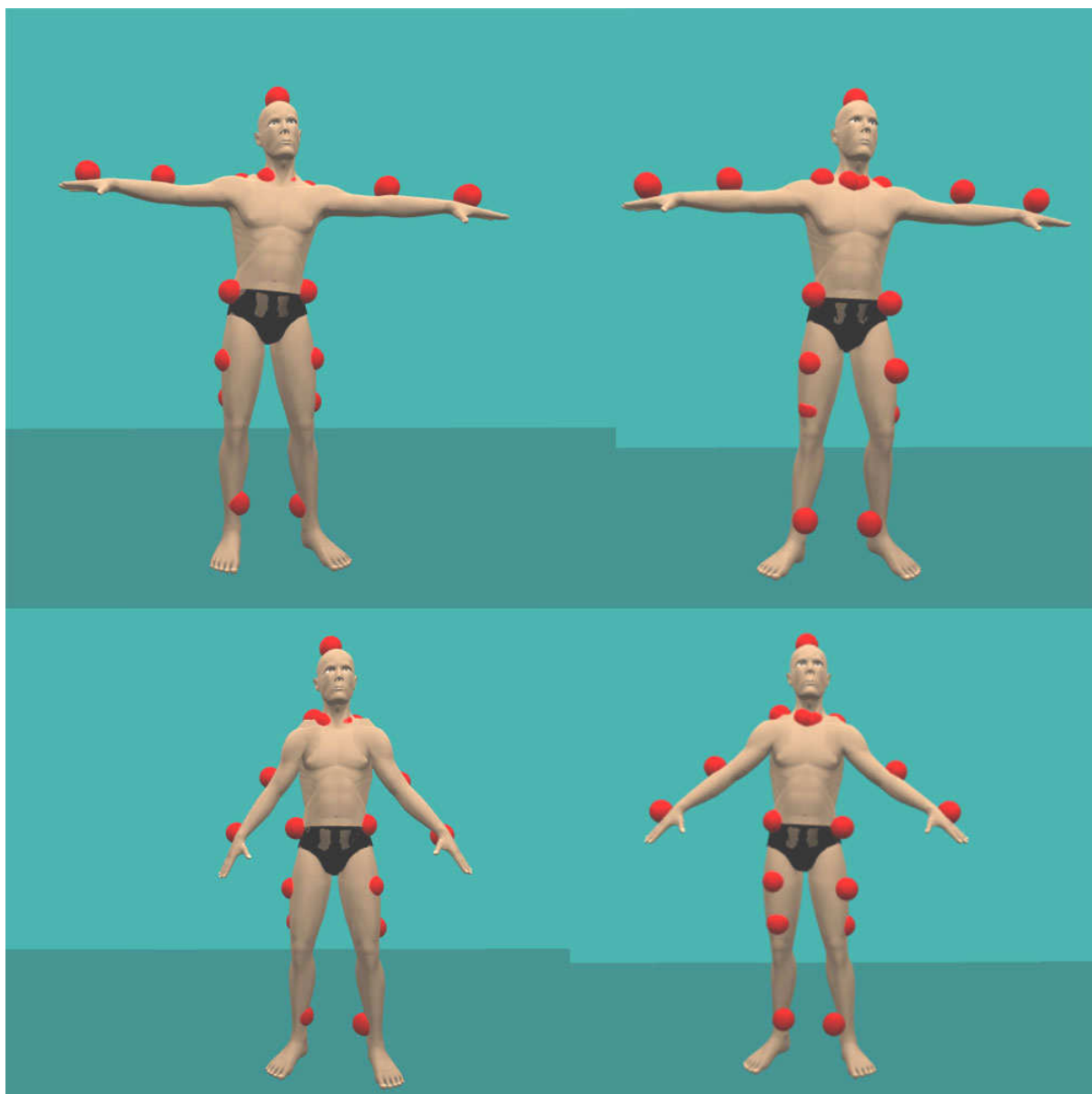


Figure 8: Points-of-interest visualized for two different physical marker-sets and independently calibrated skeletons (left column showing the first marker-set, right column showing the second). Please note that the avatar is just added to better illustrate the locations of the POIs and not perfectly aligned with the real-world-position. Therefore, POIs on the right seem slightly more up and closer to the viewer than on the left.

Virtual Marker-Sets

The flexible marker-arrangement in our MoCap-system offers many advantages, while it has one major disadvantage. When in use with an application that relies solely on the input of 3d-marker-positions, it is almost impossible to automatically put a marker in context (i.e. label it and assign it to a bone of the internal representation). Manual labeling is also not an option, since we want to keep human interaction to a minimum. Therefore, the solution is to specify a set of virtual markers, which always remain in the same relative position on the bone. This position can be easily specified and calculated as soon as the tracking-module has computed the pose of the skeleton from the real-world-markers. Furthermore, the tracking-module labels the virtual markers with a unique label, which enables the application to recognize a certain marker even from one tracking-session to another.

Therefore, virtual marker-sets can be used with character-animation or motion-analysis-software, which have their own skeleton-representation. In these applications the virtual markers are assigned to the skeleton only once by hand and can then be reused with different real-world marker-sets.

It is also possible to emulate other marker-sets with virtual markers (e.g. Helen Hayes marker-set as described by Tabakin et al.⁷). These can then be used on-the-fly with existing configurations and applications.

Finally, virtual markers have the advantage that they can be placed at arbitrary positions even inside the body, where they can be used for example to determine joints, bones or other relevant landmarks.

Relevant Positions

In many cases, a certain point on/in the skeleton/body is needed for processing in a certain application. However, if we want to keep the marker-setup flexible, we cannot rely on placing a marker on each of these positions (for landmarks in the body this also is virtually impossible).

For the *Playmancer*-rehabilitation-games and the exercises (see D2.4 section 2), that are being mapped to game-interactions, the exact position of certain body-parts is of high importance. Thus the foot (heel)-position needs to be tracked precisely during walking, so we can exactly determine the point in time, when the foot touches the ground. This is important for the synchronization of the walking cycle with the EMG-values. For another exercise, we want to evaluate the arm-elevation. Therefore, we can define a point on the wrist as POI and with that measure the elevation-level of the arm.

Specification

The position of a POI is defined in the skeleton-XML-file by its offset relative to a joint-position. In addition it is associated with a bone to define the order in which it is

⁷ Tabakin, D. & Vaughan, C.L. 2000. A comparison of 3D gait models based on the Helen Hayes marker set, Proceedings of the Sixth International Symposium on the 3D Analysis of Human Movement, Cape Town, South Africa, 98-101.

processed within the skeleton hierarchy. From this relative position the absolute position is calculated for every frame by the tracking-software. The XML-representation of a POI looks like the following.

```
<point_of_interest id="0" bone_id="10" joint_id="1" name="optional POI-
name">
  <translation>
    <vector length="3">
      <vec_elem pos="0" val="-3.689926" />
      <vec_elem pos="1" val="-19.364331" />
      <vec_elem pos="2" val="33.250347" />
    </vector>
  </translation>
</point_of_interest>
```

For the computation the hierarchy of the skeleton is traversed and the bone/joint-positions updated according to the current joint-angles. Finally, the offset of the POI is added to the referenced joint/bone-position resulting in the absolute POI-position. The XML-representation of the POI and its translation is then transmitted using TCP-sockets (as described in D3.1 section 3.1) . The XML-packet sent for every frame looks similar to the specification of a POI except that the joint-id and the bone-id are not specified and the position is an absolute one:

```
<point_of_interest id="0" name="optional POI-name">
  <translation>
    <vector length="3">
      <vec_elem pos="0" val="590.3" />
      <vec_elem pos="1" val="190.4" />
      <vec_elem pos="2" val="600.8" />
    </vector>
  </translation>
</point_of_interest>
```

Evaluation

To test the POIs we have used two different suits and equipped them with two different marker-sets. Pictures of the suits can be seen in Figure 10. Then we calibrated a skeleton for both. For the tests 17 different POIs were defined throughout the body, which could for example be used as virtual marker-set. During the calibration procedure the POIs are automatically added to the skeleton-XML-file from a configuration-file. Thus we were using different skeleton-models while employing the same POIs. Figure 9 shows visualizations of the two different marker-sets. The top row shows a visualization of the point-clouds including labels of the markers as generated by the tracking-software. The lower row visualizes the local bone-coordinate systems of both marker-sets. On the left also marker-labels are displayed, while the right picture shows an error estimate of the joint. The latter can be calculated from the skeleton-model for bones that employ three or more markers.

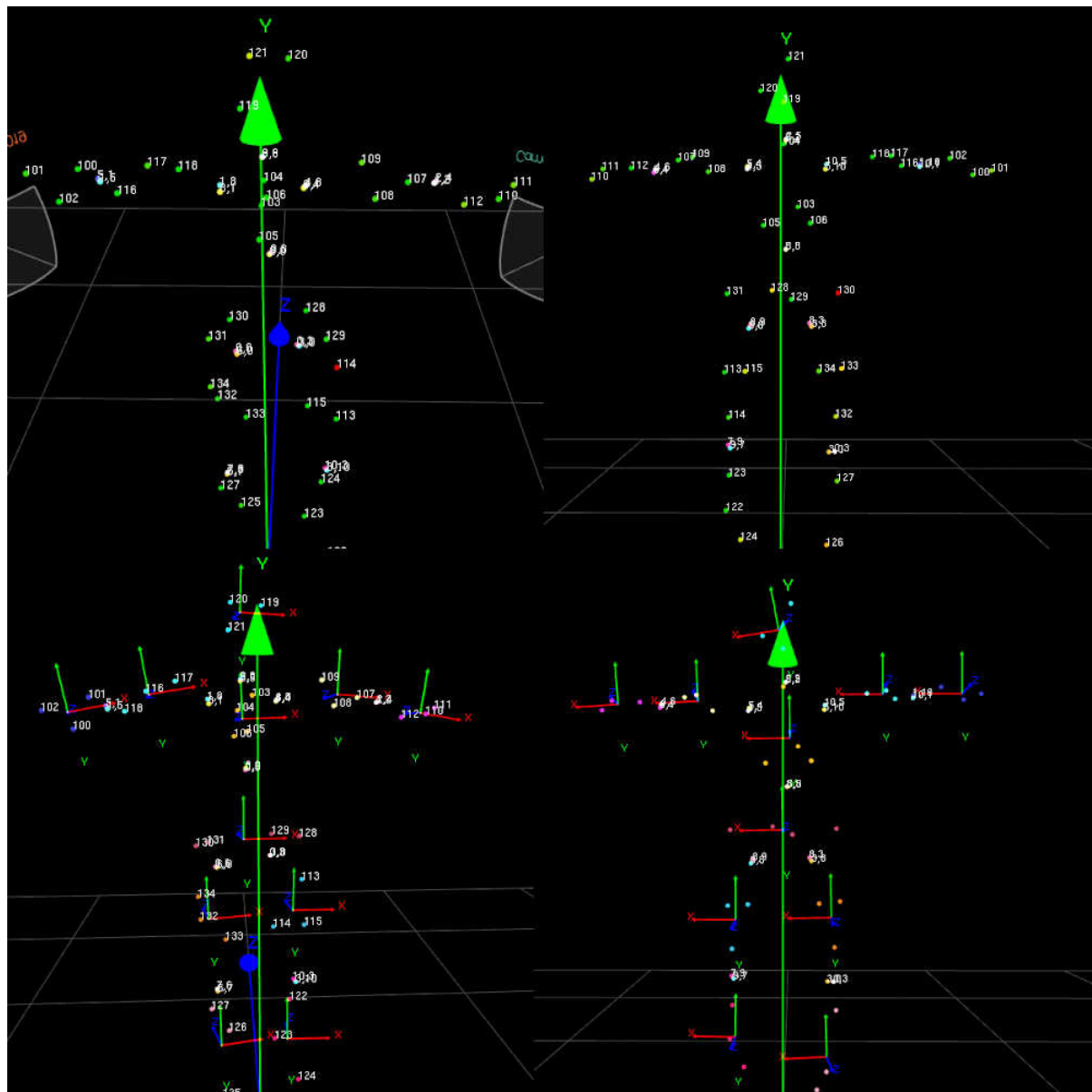


Figure 9: Visualizations of the two different marker-sets. Top row shows a visualization of the point-clouds including labels of the markers as generated by the tracking-software. Lower row visualizes the local bone-coordinate systems of both marker-sets. On the left also marker-labels are displayed, while the right picture shows an error estimate of the joint.

Using the MoCap-integration of Unity3D, we visualized the POIs as red spheres, as can be seen in Figure 8. In contrast to the physical markers the POIs are located at the same positions for both cases.

Please note that the avatar is just added to better illustrate the locations of the POIs and not perfectly aligned with the real-world-position. This comes from the fact that the avatar has a fixed root-position (pivot-point) within the skeleton-structure, while the pivot-point of the skeleton is defined by the center-of-gravity of the markers situated on the root-bone. The POIs, on the other hand, are displayed in real-world-coordinates. Therefore, POIs on the right seem slightly more up and closer to the viewer than on the left. Requirements for the games described in D2.4 do not specify such an absolute alignment as necessary. Instead joint-angles and

measurements derived from POIs (see “Relevant Positions” above) are being used. For a short summary, why exact full-body avatar animation by motion capture data is not needed within the game prototypes please refer to section 4.4.2.

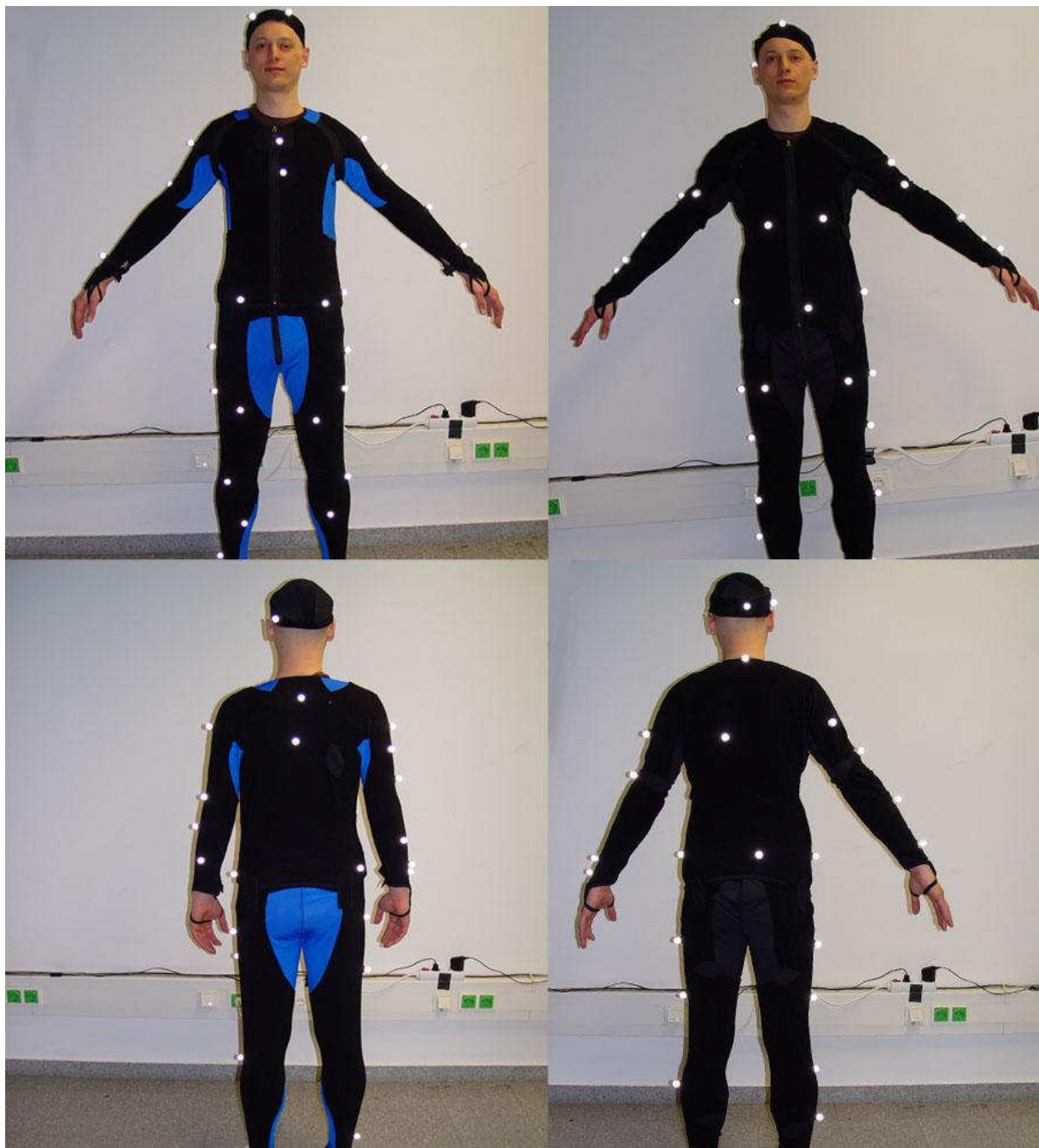


Figure 10: Two MoCap-suits with different marker-sets (passive markers).

2.3.7 Evaluation of the Skeleton Tracking Module

During development the skeleton tracking module has been tested multiple times during different stages with various users and setups.

An evaluation of an early prototype with a group of ten subjects is documented in deliverable D8.1 section 4.2 and already showed acceptable results.

Later a more mature prototype of the module has been assessed using an active motion suit (see Figure 11). For this suit infrared-LEDs were used as markers, instead of spheres coated in retroreflective material. Performance of the system was equally well as with passive markers.

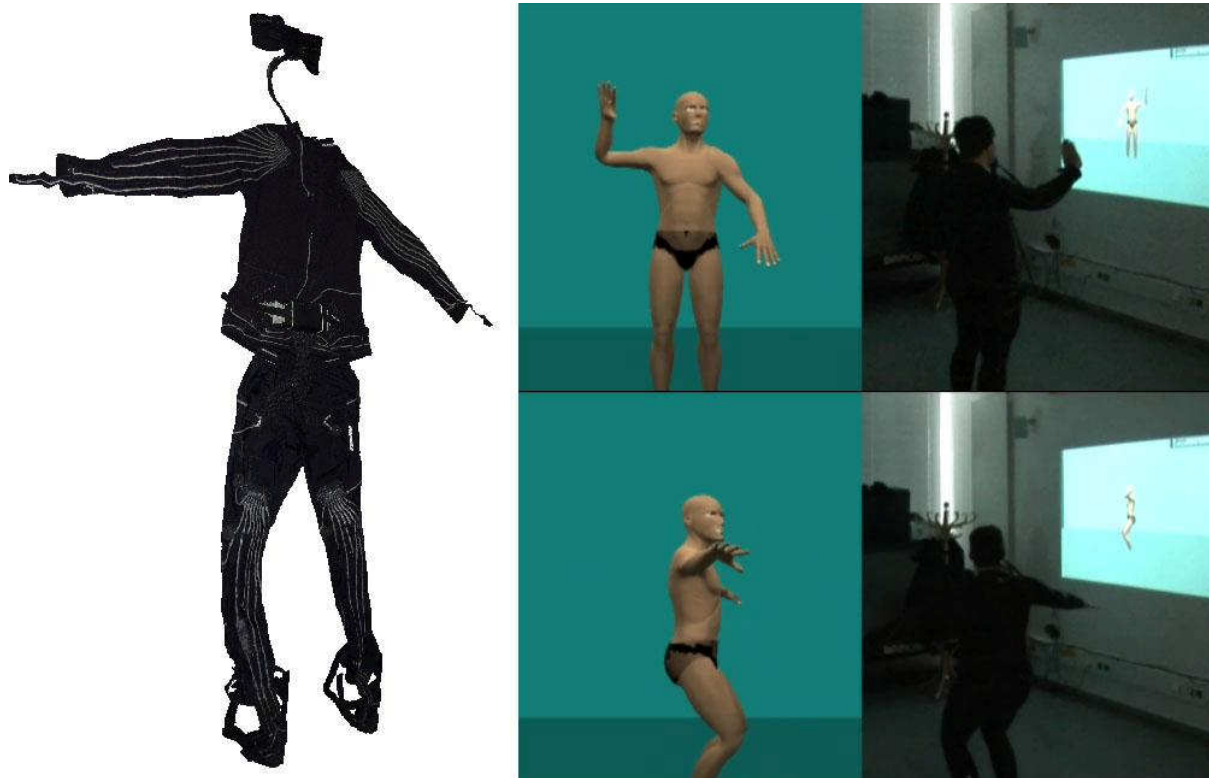


Figure 11: Tests of the skeleton tracking module using a MoCap-suit with active markers

For the skeleton calibration usually two to three sequences of 15 to 20 seconds are recorded, so the algorithm/therapist can choose a good skeleton. Calibration for each takes about 45 seconds on a quad-core-machine. However, calibration can start in the background after the first recording, while the patient is going through the next sequence.

POIs as described in the above subsection are being defined solely via the XML-Protocol specified above. There is no GUI integration implemented and therefore, it is intended mainly for developers and technicians to specify any new POIs, for the described usecases. Due to the fact that on-the-fly changes by therapists and health workers are not intended, this interface is not described in the manual for the games (as is the case for the skeleton calibration and tracking). Documentation of the interface is provided by this deliverable, coding samples and inline-documentation in the available configuration-files.

Accuracy of the specification of the POIs will be largely dependent on the accuracy of the real world measurements, which they are based on (i.e estimates of certain points on the patient's body). Therefore, it is hard to provide a general estimate of

the error introduced by the interface itself. Accuracy of the tracked POIs, in turn, is dependent on the skeleton tracking and the quality of the tracked bone.

A discussion of the iotracker's positional and rotational accuracy and error rates can be found in deliverable 8.1 section subsection 4.1.2 (for a short summary please refer to subsection 7.2). For human motion capture also skin- and tissue-movement have to be taken into account to provide a reliable measurement of the error-rates. Also, inaccuracies of human joints, as compared to perfect rotational joints affect the accuracy. These factors depend on a number of parameters, like marker placement, specific human joint, the suit and how it fits the user, posture and others. The main problem, however, is to determine a ground truth for these measurements. As described in deliverable 4.1 section 4.2.1 measurements obtained by use of a universal goniometer are depending on the tester and the articulation used. The same is true for other measurements taken from subjects by unintrusive means. The effort for measurements by intrusive means, providing information on the systems accuracy on a general level, however, can be considered too high to be done within the scope of the project. Also, to our knowledge, this kind of tests have not been performed on larger scale for other tracking systems, which are well established in medical applications. Subsection 7.2 gives a short overview of some measures, which we have taken a look at with respect to the use cases.

For the application in chronic pain rehabilitation, patients are equipped with a MoCap suits in their size. Suits in different sizes for men and women have been acquired for that purpose. Patients are only wearing underwear/thin clothing beneath the suit so markers can be placed as close to the body as possible. Therefore, clothing is of little to no influence to the tracking process. As stated in deliverable 8.1 section 4.2.1, participants of pre-tests have rated the suit comfortable to wear. Temperature of the suit and fabric have been determined comfortable (even after walking several minutes on treadmill) and there was little to none restriction in movements caused by the suit. Therefore, using the MoCap suits can be seen in accordance with the user requirements described in D2.4.

- The suit doesn't hinder movements and therefore exercises targeting mobility and coordination can be executed.
- The suit is comfortable to wear under training conditions and can therefore be used for physical reconditioning.
- The suit was rated comfortable and will therefore not negatively influence relaxation of the whole body or specific muscle groups, as required by some exercises.

Furthermore, the patients participating in the field trial had no problem with wearing the MoCap suit multiple times. All were willing to wear it and for every patient there was a suit that fitted. The first time the patients needed some help and extra time to put on the suit, but after the first session, every patient was capable to put on the suit himself. Most patients found the suit funny and wanted a picture of themselves in the suit. Quote patient: "I had no problem at all with the suit. Maybe sometimes it was a little bit warm when you were walking. I think it was a funny suit". However, despite these positive results, for further use of the Playmancer game in rehabilitation it might be an idea to separate the MoCap suit into subsystems instead of using one whole-body suit (e.g. a jacket and/or a glove).

2.4 Integration with 3rd-party Character Animation Software

2.4.1 Introduction to Autodesk MotionBuilder

Autodesk MotionBuilder is the industry-leading, real-time 3D character animation software for games, feature film, and television productions. Its core focus is on interactive real-time workflows. MotionBuilder is a package designed for 3D data acquisition, manipulation, and visualization. With its many functionalities and good extensibility it is a good test bed for the implementation of MoCap algorithms and also offers options for further use of the iotracker for animation purposes.

2.4.2 Integration with MotionBuilder

The basic idea of integrating MoCap data from the iotracker into MotionBuilder is shown in Figure 12. The iotracker sends the data to a specified client workstation using its own protocol (see D4.2 section 2.5). Iotracker and client can also be collocated on one PC but in many cases they will be distributed on two different machines. On the client-side the data is received by the Optical Device Plug-In streaming the data into MotionBuilder. There the Marker set is visualized as point cloud and can then be assigned to a character for animation.

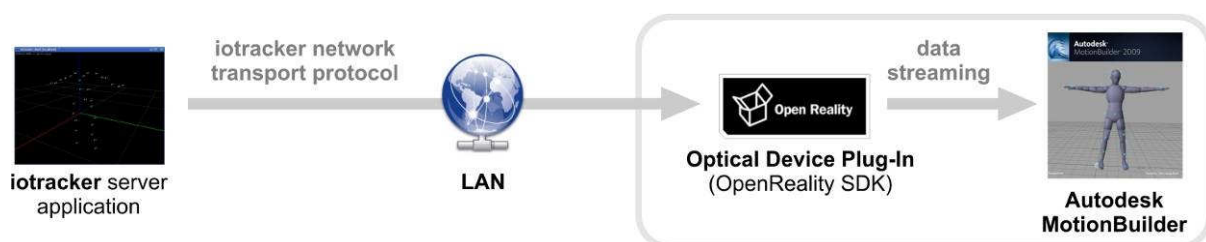


Figure 12: MotionBuilder integration via the OpenReality SDK

We tested the MotionBuilder integration during different tracking sessions using rigid body targets as well as a MoCap suit (see D4.2 section 2.5).

In order for the MotionBuilder Plug-In to work, the marker-labels have to stay constant during the whole session. This has been improved for the final version of the prototype using “Points-of-interest” as has been described in section 2.3. A visualization can be seen in Figure 13. Additionally, this makes handling of the tracking-data much easier, because marker(POI)-positions have to be assigned to the avatar only once. After that MotionBuilder can easily identify the POIs by their labels.

Virtual Marker-sets defined by POIs have also been tested in MotionBuilder. For that reason we equipped two MoCap-suits with different marker-sets and had two users perform the skeleton calibration. This resulted in two different skeleton-models. For

these we defined the same POIs in the skeleton.xml for each user. Finally we had the users perform some movements to test the mapping from real-world marker-set to virtual marker-set.

Now the tracker is able to recover lost marker labels by using the skeleton-model. Therefore, the tracker is ruling out degradation of animation quality and occlusions are not an issue in the final prototype even without the use of rigid-body targets.

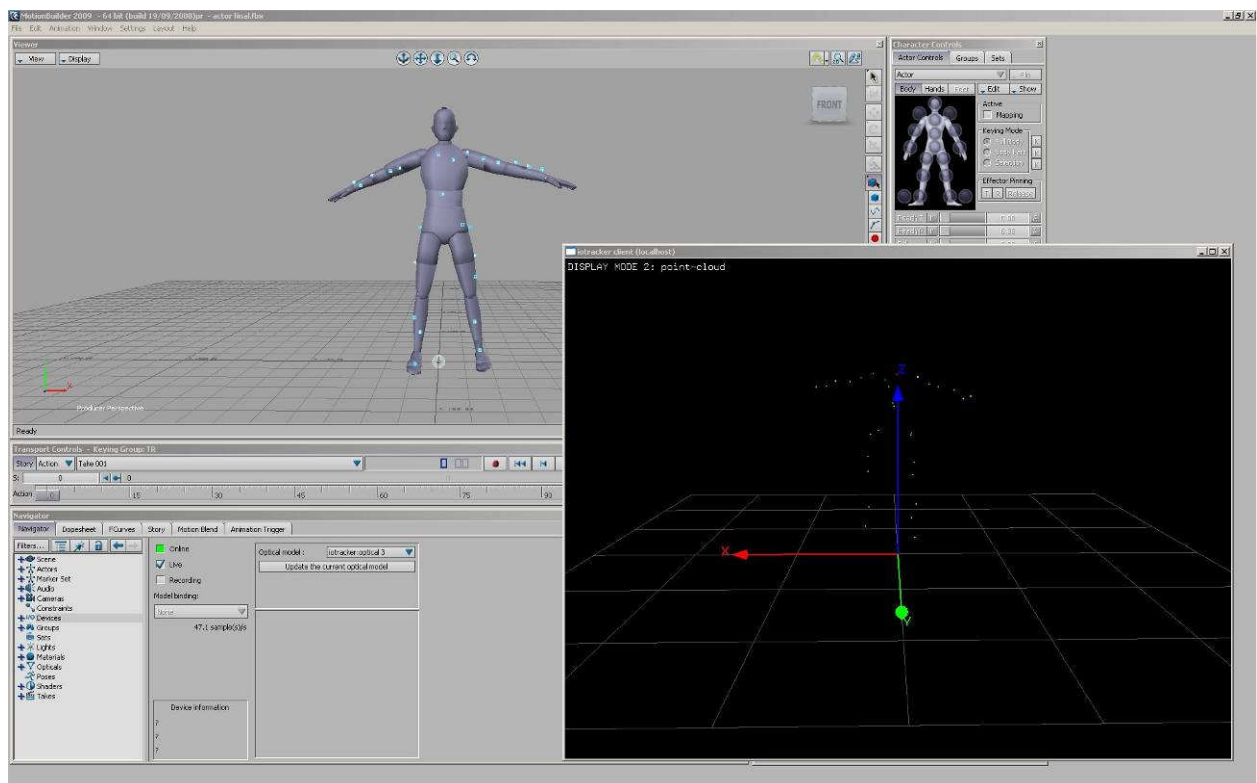


Figure 13: MoCap data is captured using a MoCap suit and the iitracker (visualized in the lower right corner using the iitracker-client) The data (a labelled point cloud) is then streamed to MotionBuilder, where it can be attached to a character. The pose of the character is then adapted to the tracking data in real time.

2.5 Integration with 3rd-party Motion Analysis Software

2.5.1 Introduction to C-Motion Visual3D

Visual3D offers biomechanical analysis for researchers and clinicians. Clinically, the software package is used for performance-analysis and movement-assessments. It has been chosen over other products for use within the project for a variety of reasons described in section 2.6 of deliverable D4.2.

2.5.2 Integration with Visual3D

Integration of the Playmancer-framework with Visual3D has been achieved using Visual3D's real-time Plug-In interface. C-Motion for that purpose offers a C++ API, which implements methods for streaming data into Visual3d. In addition, useful features of the API like XML-file handling and recording to c3d-files have been used in our Plug-In.

As with Motion Builder, points-of-interest (POIs) are being used instead of markers to avoid having to adapt the skeleton-model for every change of the marker-setup.

The basic idea of the integration is illustrated in Figure 15. First a customized Plug-In connects to the iotracker-server (directly or indirectly via Opentracker) using TCP-sockets. The iotracker-software then sends XML-packets to the Plug-In containing labeled POIs using the structure specified in section 2.2.

The id-labels of the POIs are mapped onto marker-names used in Visual3d using an XML-configuration file with the following format.

```
<Marker_name_mapping>
  <point_of_interest id="0" name="left_heel">
  <point_of_interest id="1" name="right_heel">
    ...
  <point_of_interest id="35" name="right_elbow">
</Marker_name_mapping/>
```

The Plug-In then parses the POI-packets and extracts the positional information. From the Plug-In, they are written into a buffer provided by Visual3d. The POI-positions are used within Visual3d as marker- and landmark-positions. Using this data Visual3d is able to update the skeleton-parameters and the pose of the skeleton for the current frame.

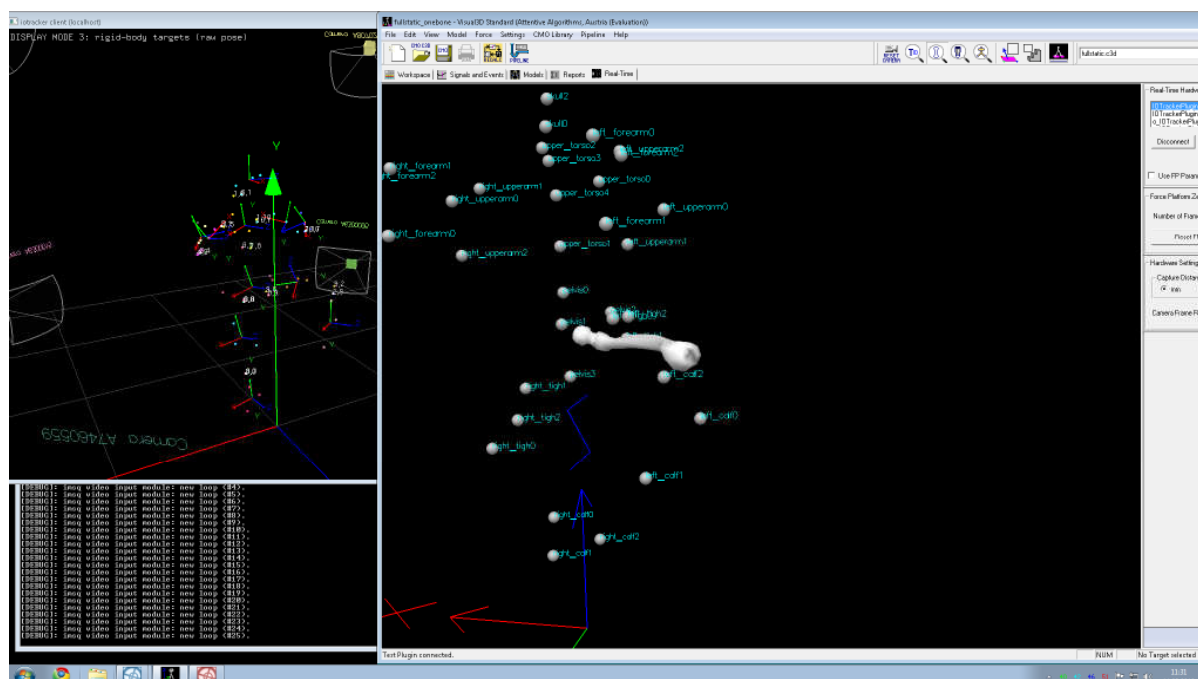


Figure 14: Screenshot of the iotracker-client and Visual3d showing visualizations of the local bone-coordinate-systems as well as the POIs with their labels. The thigh-bone shows exemplarily how a bone is scaled to the POIs and animated live by using POI-positions.

The skeleton-parameters are adapted using the “Model Builder Real-time” feature from Visual3D, which can capture a static trial (a snapshot of the marker-positions in one or multiple frames). From the static trial Visual3D automatically updates the bone-lengths of the skeleton (for example the thighbone can be scaled according to the distance between hip-joint and knee as shown in Figure 14).

The pose for each frame is updated using the inverse kinematic algorithms provided by Visual3d, which use the skeleton and the marker-/POI-positions as input.

In addition, the Plug-In offers the functionality to save a real-time stream of motion-data in .c3d format. C3D⁸ is an open file format created by Vicon Motion Systems⁹ and can be considered a de-facto standard, because many of the major analysis software products can import this file format. Therefore these files can be used as an interface to other tools if needed.

2.5.3 Evaluation of motion-data using Visual3d

Visual3D offers functionalities to create graphs and charts to evaluate the patient’s movements. This includes positional information as-well-as joint angles and calculation of metrics like Root-Mean-Square, median, integration etc.

The therapist starts the iotracker-server as well as Visual3D. In Visual3D, he then loads the skeleton-model predefined for the virtual marker-set being used. Then he

⁸ <http://www.c3d.org>

⁹ <http://www.vicon.com>

initializes the Plug-In (as depicted in Figure 15), which connects to the server and starts streaming the data.

Using the “Model Builder Real-time” feature from Visual3D, he can capture a static trial (a snapshot of the marker-positions in one or multiple frames). From the static trial Visual3D is then able to automatically, update the bone-lengths of the skeleton.

Using the “Signals and Events” functionality of Visual3D, the therapist can then create graphs and charts to evaluate the patient’s movements. This includes positional information as-well-as joint angles and calculation of metrics like Root-Mean-Square, median, integration etc. Depending on his preferences (and the license type of Visual3D) the therapist can either watch this evaluation in real-time and/or save it as .c3d file for further use.

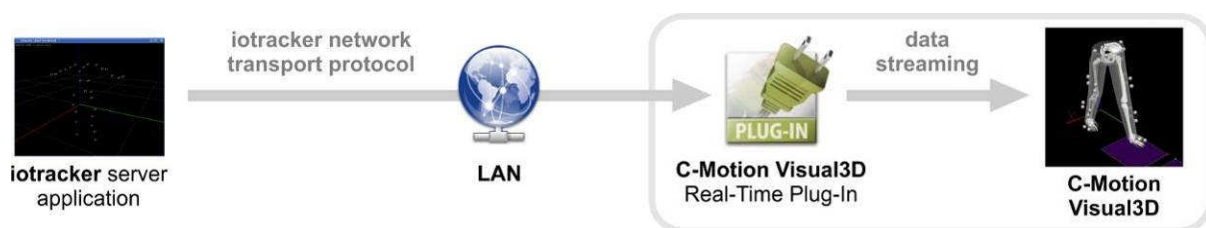


Figure 15: Integration with C-Motion Visual3d (via Real-Time Plug-In).

2.6 Commercial Aspects of the iotracker Motion Capture System

Market situation

Tracking is a critical component of any virtual reality system and typically the most expensive one. Costs of tracking systems, especially of optical tracking systems which provide the highest measurement accuracy have always been high. They are sold by a small number of companies worldwide and due to a small end user market for such systems, prices were kept high for over a decade. This especially applies to Motion Capture (MoCap) systems which are mainly used in movie/animation production and (human) motion analysis.

Four different, basic technologies are commercially used for capturing motion. We distinguish between motion capture systems using optical sensors, magnetic sensors, inertial sensors and mechanical linkages (exoskeleton). Due to the technological principle these systems differ in many aspects such as measurement accuracy, cost, range of motion, scalability and ease of use, to mention only the most relevant parameters.

Before the start of this project, the partners have decided to use a motion capture system with the highest possible quality to enable medical partners to record, analyze, compare and publish high quality data and results. In the medical

community optical motion capture systems are the de-facto standard in rehabilitation. Due to their high costs at the start of the project such systems have mainly been used in rehabilitation research or applied gait rehabilitation for human motion analysis. They provide the highest measurement accuracy and highest capture quality (e.g. smoothness of motion) compared to other MoCap technologies, and they provide good usability. In order to publish results comparable to prior work in the medical domain (e.g. bioengineering, human motion analysis etc.) high quality data are of primary importance. Today, many years after this decision has been made, this is still the case. Although lower cost (and lower quality) systems that capture movement have been introduced to the market recently.

In November 2010 – three years past the project's start and after the final submission of this deliverable – Microsoft released the Kinect (for Xbox 360) based on Primesense's system on a chip which outputs a depth map in addition to an RGB image of the scene. An infrared light pattern is projected onto the scene, a depth map is then calculated from the pattern's deformation. The system uses a standard CMOS sensor to keep hardware costs low. In April 2011 ASUS released the Xtion Pro, a PC version of the Kinect, which also uses the Primesense chip.

A comparison of the Kinect to our iotracker system can be found in our publication¹⁰. In summary, Kinect may be usable for some serious games where tracking accuracy and smoothness of motion are not of primary concern. For Kinect, jitter in positional data is clearly visible in the extreme positions. Jitter also strongly disturbs the velocity, which we calculated on a frame by frame basis. Since velocity is being used to determine smoothness of motion this is considered problematic for our purpose.

Due to the low quality of motion data captured by a device based on the current Primesense sensor, such data cannot be used for a medical evaluation of human motion. Therefore, low cost motion capture devices such as the Kinect or ASUS Xtion Pro cannot be considered competitors to our iotracker high quality MoCap system.

However there is a range of professional optical motion capture systems and companies which can be considered competitors to the iotracker system. They are all established companies and provide high quality systems.

<http://www.vicon.com/products/bonita.html>

<http://www.vicon.com/products/viconmx.html>

<http://www.naturalpoint.com/optitrack/>

<http://www.motionanalysis.com>

<http://www.ptiphoenix.com/>

<http://www.phasespace.com/>

<http://www.ndigital.com/medical/polarisfamily.php>

<http://www.ndigital.com/medical/certus.php>

http://www.atracsys.com/_products/tracking_systems.php

<http://www.worldviz.com/products/ppt/index.html>

<http://www.codamotion.com/products.html>

¹⁰ Schönauer, C. et al. (2011). Chronic Pain Rehabilitation with a Serious Game using Multimodal Input. In Proceedings *International Conference on Virtual Rehabilitation*, Zurich, Switzerland.

<http://www.qualisys.com/>

http://www.btsbioengineering.com/BTSBioengineering/Kinematics/BTSSMARTD/BT_S_SMARTD.html

<http://www.ar-tracking.de/>

In a configuration tailored to a room-sized multi-user environment, all of the existing optical systems have price tags in the range of tens of thousands of Euros. The urging matter of costs originally led to the development of the iotracker low-cost infrared optical tracking system which was further developed within this project into a full motion capture system. The original goal was to reduce costs of high quality optical tracking systems without sacrificing quality i.e. speed or accuracy. Iotracker is commercially available for target tracking (<http://www.iotracker.com>) and has already been a success in this respect. Since the introduction of iotracker which is available at a fraction of the price of other systems, some vendors have already reduced their prices or introduced new products at lower prices.

While corporate entities, movie production companies and well-funded research laboratories will not be deterred by such costs, many smaller corporations, medical or educational institutions operate on tightly constrained budgets that leave little, if any, room for an expense of this magnitude. High quality lower cost tracking technology in return opens up new end user markets and new application areas.

All of the above mentioned optical systems have a price higher than 35.000 USD except the Naturalpoint OptiTrack system. Their lowest cost 8-camera USB version is being sold for 8.000 USD, a 6-camera Ethernet bundle starting at 14.500 USD. It is the only real competitor to iotracker. No statements about OptiTrack's quality and range of motion can be made due to missing data and lack of opportunity to compare both systems. Therefore it is not clear if its quality is comparable to iotracker. Iotracker is licensed by the company Imagination (www.imagination.at) which manufactures and sells an 8-camera bundle starting at around 17.500 EUR.

In addition to optical motion capture systems there are systems based on inertial sensors such as one from the Dutch company Xsense (<http://www.xsens.com/>). Its entry price is about 15.000 EUR.

Mechanical motion capture systems such as the Gypsy 7

<http://www.metamotion.com/gypsy/gypsy-motion-capture-system.htm>
start at 8.000 USD.

Magnetic systems such as the MotionStar from Ascension Technologies start at about 10.000 USD.

<http://www.ascension-tech.com/realtime/RTMotionSTARTethered.php>

We define “software based” optical motion capture systems as systems which use conventional 2D cameras and perform all image processing and motion tracking on PC(s) via software. They either operate offline, are very expensive (e.g. OrganicMotion using at least 12 HD cameras; costs ~80.000 USD) or use low cost cameras which typically result in low capturing quality and high latency (due to high software processing times). If only few cameras are used – or even a single camera – occlusions are the biggest problem since occluded body parts cannot be tracked. Positions can only be estimated in such a case. The quality of such systems for medical purposes is usually not acceptable.

The decision not to display the high quality of iotracker's motion capture data to the user was a game design and therapeutic decision and not a technical one (see also subsection 4.4.2 on that). Abstract visualizations of users' movement are supposed to avoid frustration and discouragement in case of wrongly executed movements. Patients are not supposed to see every tiny fault; they are not supposed to feel unsecure by variations in velocity and smoothness. Therapists, in contrasts, need high quality MoCap data to be able to record, analyze and interpret patient's behavior. Finally, only reliable and robust data can be published in the medical community.

Whereas for patients at home a very low cost system based on the Primesense chip may be advantageous, clinics need high quality motion capture systems for medical analysis and rehabilitation research. This is a commercial opportunity for the iotracker system.

3 Multi-Modal Data-Flow Framework Final Prototype

3.1 Overview

As introduced in previous deliverables (2.1d, 3.1, 4.1 and 4.2), OpenTracker is a generic data-flow network designed to deal with tracking data, but not limited to it. Therefore, OpenTracker has been chosen to serve as an abstraction-layer that will provide a common interface to all input devices and biosignal sensors used throughout the Playmancer multimodal platform.

Integration in OpenTracker has already been described in much detail in deliverable D4.2 section 4.2, thus here we will only give an overview.

3.2 OpenTracker Input Device Integration

3.2.1 Motion-Tracking Modules

For the integration of the *iotracker* motion-capture system into OpenTracker two different scenarios and therefore two source node types were considered. The first scenario is the tracking of a rigid body target, which is the standard tracking scenario, where position and orientation is needed. In the second option, which will be primarily utilized for the Playmancer mini-games, the tracker has to produce information on the skeleton pose during the Motion Capture. Meaning, the tracker has to relay information on the current positions of the skeleton-model's limbs to the game/application for every frame.

3.2.1.1 Rigid-Body Target-Tracking Module

Integration of rigid-body-targets is straight forward, since that is what OpenTracker has originally been designed for. Using specialized nodes OpenTracker establishes a network communication to the *iotracker* server utilizing common protocols (see D4.2 section 4.2.1 for details). Another node then sends events of the OpenTracker standard data type into the tracker tree, where they can be modified and/or forwarded to e.g. Unity3D.

3.2.1.2 Full-Body Motion-Tracking Module

For streaming skeleton-based motion-capture data into OpenTracker a pivot point and the joint angles are propagated into the data flow network. Also the skeleton-model is loaded to reconstruct the pose starting from the pivot point and following the joints angles. This option can be considered minimal in terms of the required data

(which can be important in case the Tracking-PC has to transfer the tracking data over a *network*).

In addition points-of-interest can be transferred as described in section 2.3.6. In Opentracker tracking data can then be modified and evaluated using various nodes and python scripts.

The interface utilized between iotracker and Opentracker is the same as between Opentracker and Unity3D. Therefore, if the game/application needs unmodified pose-data only, Opentracker can be bypassed in order to avoid additional latency. Evaluation has shown that the latter option performs better. Therefore, this option is being used for the current prototype.

3.2.2 Biosignal Sensor Module

For the final prototype of the biosignal sensor module we have implemented many nodes to access, retrieve, filter, display und postprocess data. Most are described in deliverables D4.1 section 5.2.2 and in a more recent version in D4.2 section 4.2.2. Updates since D4.2 include an improved version of the Heartratefilter, a trigger for recording baselines (HR and GSR) as well as an emotionfilter, which determines level of arousal. Furthermore, filters have been implemented in order to process raw EMG signals. These are RMS- (root mean square), SRE- (smooth rectified) and RRT-(relative rest time – of the muscle) filter.

Several classes have been added to OpenTracker to provide the functionality necessary to support gMOBIIlab (see D4.2 section 4.2.2 for details).

4 Integration with Unity3D

4.1 Overview

To ensure easy integration with the Playmancer framework a well-defined interface between emotion/body state fusion modules and the Unity3D [Unity] has been designed. Details on that interface can be found in D3.1¹¹ section 3.1.

Big parts of the functionality are being shared by the Playmancer components. It has therefore been a goal from the beginning to design the framework in a way that makes it easy to add new components without much component-specific integration being necessary. This has been achieved with an object oriented approach with low coupling and high coherence between the objects in the framework. Using XML to describe the data transferred between components has allowed the system to be very flexible. By using TCP/IP for communication between the input device and the game allows the machine executing the game not to spend precious CPU cycles analysing raw input data.

A client-like approach is chosen whereby the game acts as a receiver, absorbing data and acting accordingly. The game components within Unity can subscribe to information from the various input devices using the XML format described in 3.1 and thereafter utilize this data in whatever fashion. Motion capture and posture data can for example be used to position and orient an in-game avatar or various biosensory inputs can be applied to adjust the game world.

Each input device inherits properties from a general InputDevice class which contains the various methods each will need about how to process the incoming data. Each specific device then overwrites parts that are device-specific as well as including the variables independent to the device.

An object residing in the game world is then able to subscribe to these devices by adding them to a general device list of which it then monitors. When new data is received, a data recipient handler fires events which can be handled within the game. How often this data should be monitored/handled can be adjusted according to the requirements of the game itself.

4.2 Architecture

The general architecture of the implementation of the Playmancer framework will now be described below.

4.2.1 Input

¹¹ version 1.2 from 2009-10-31

Figure 16 shows how a game written in Unity3D receives input from an input device by using the Playmancer framework. The components on the right are running in Unity while the components on the left side could be running outside Unity possibly on a different machine.

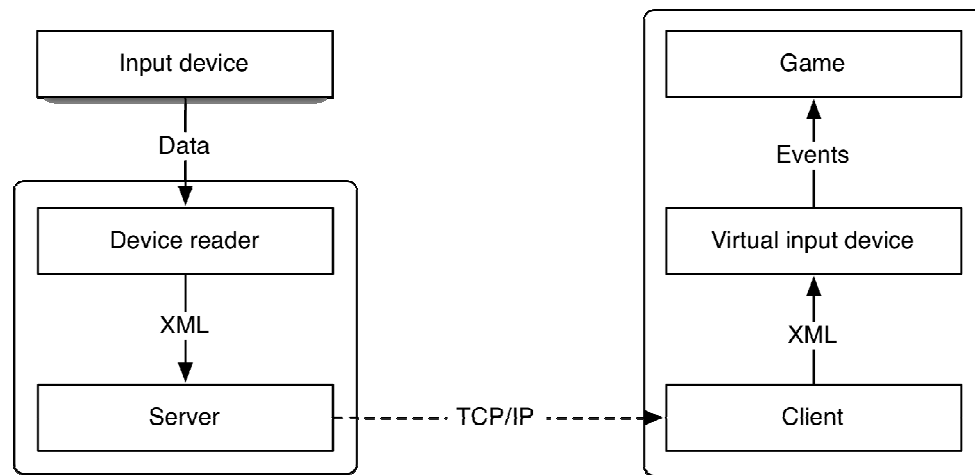


Figure 16: Receiving input for a game in Unity

The game receives input from the input device by subscribing to events exposed by the virtual input device in Unity. The virtual input device will fire an event each time the real input device has generated new data.

The virtual input device extracts the state of the real device from the XML formatted data supplied by the network component. The network client continuously receives new data from the server every time the server has new data ready.

On the server side, the input device is connected to the device reader which is the software responsible for interpretation of the raw input data and handling of the device at hardware level. The data is then output in XML format and sent using a server component to the client.

4.2.2 Output

In addition to receiving input the game can also use the Playmancer framework to control external devices as shown in Figure 17. The right side are components running in Unity while the left side is running outside the engine.

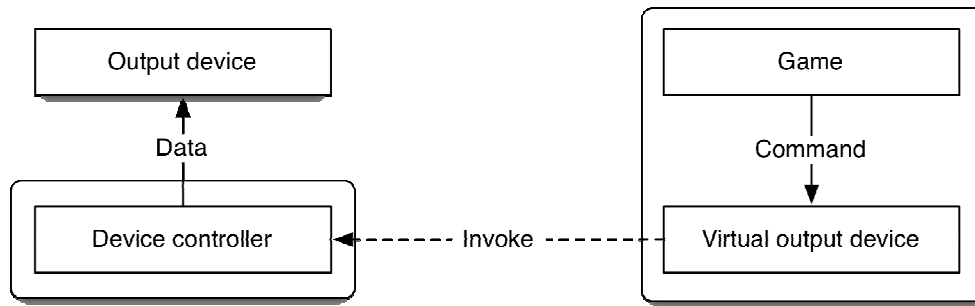


Figure 17: Using an output device from a game in Unity

The game interfaces with the virtual output device which provides access to the functionality of the real device. The output device is in our case controlled by calling a DLL on the same machine as is executing the game.

4.3 Input device specific implementation details

The Unity-side acts as a client and receive data via the TCP network connection in the form of XML messages that are then processed within the Unity environment and exposed for in-game usage within the game world. Since each device generates very difficult forms of data, this section will describe the implementation of the various input devices into Unity and some of the reasoning for the choices made.

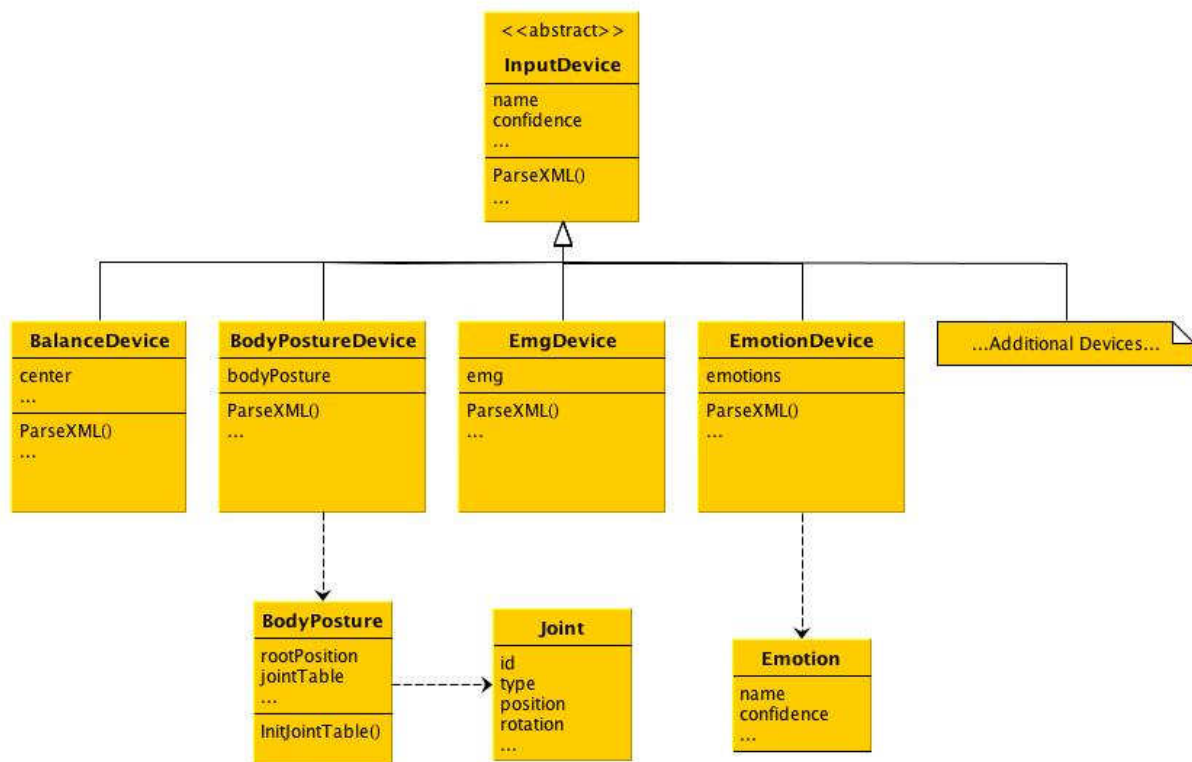


Figure 18: Input device hierarchy

4.3.1 Body-state-information

An OpenTracker context is run in a separate process on the same machine that runs Unity3D or one that is connected via local area network. Network communication (TCP sockets) is used to exchange data between OpenTracker and Unity3D. This solution has been chosen and implemented after thorough investigation of the possible options.

4.3.2 Motion-tracking

The information received from these devices contain data regarding positions and relative rotations of various joints within a body posture. During the initial phase, a structure must be generated to depict the joint composition of the new body we are monitoring. This is done by running through the skeletal form of the body and creating a dictionary from which joints can thereafter be referenced from. After this is complete, the data transferred only needs to be that of the specific joint that has been altered, and the rotations of which have been performed on it.

On the in-game side, the initial body posture of an avatar is created and remembered. The joints that are used during the movement are then mapped directly onto the values placed on the avatar puppet within the game world. This way, assuming that the joint structure remains the same, we can apply the movements to any avatar/3d model within the Playmancer game world.

The motion tracking part of the implementation uses the changes in state received over time but is also very in-game requirement-specific. A specific minigame may for example only be interested in the reach of the patient's arms. In the case that a specific joint(s) needs to be monitored, a minigame may choose to only receive data directly related to these areas.

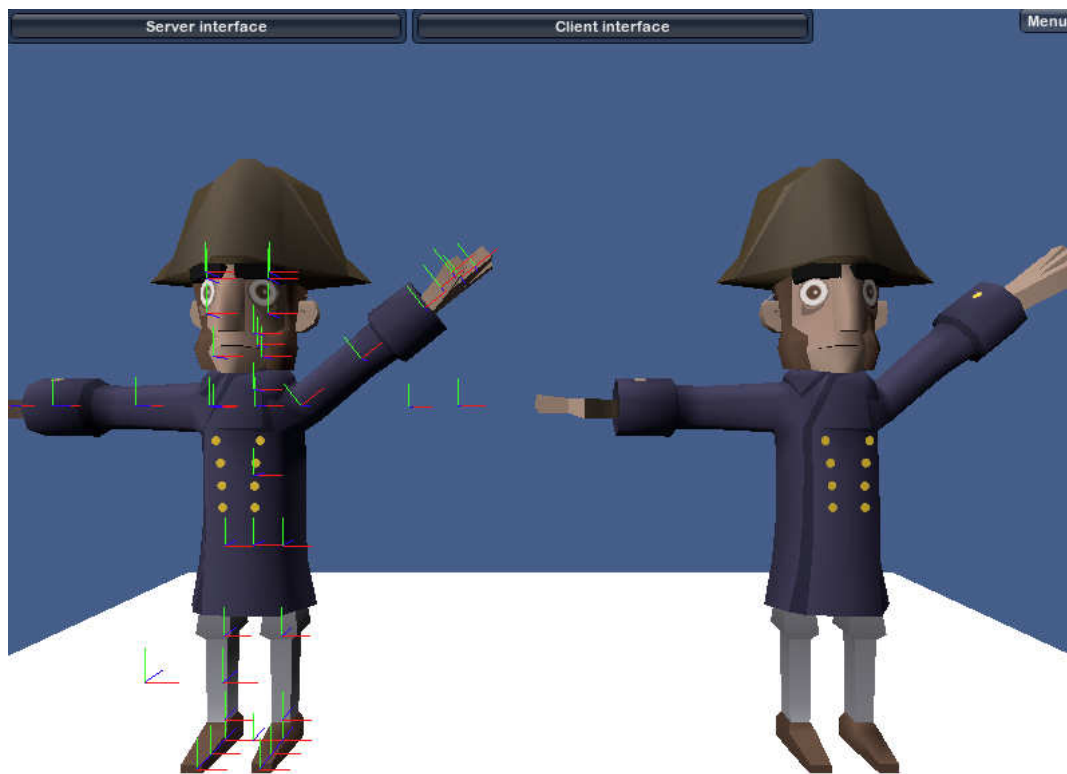


Figure 19: Prototype while testing skeletal changes on a server and client

4.3.3 Biosignals

The data from within biosignal devices are generally in the form of a certain numeric value accompanied by a confidence factor. Currently the supported devices are EMG, Galvanic data, Heart Rate sensing, and emotional recognition based on face and voice. Some of these however are yet to be tested and used directly within the Unity environment.

Since the nature of the data gathered from this kind of device cannot be categorized in certain terms, their use is determined on a case-by-case basis on a game-based-requirement level, in other words, how they are used within a minigame. Thresholds of acceptance are defined and are linked to a calibration process.

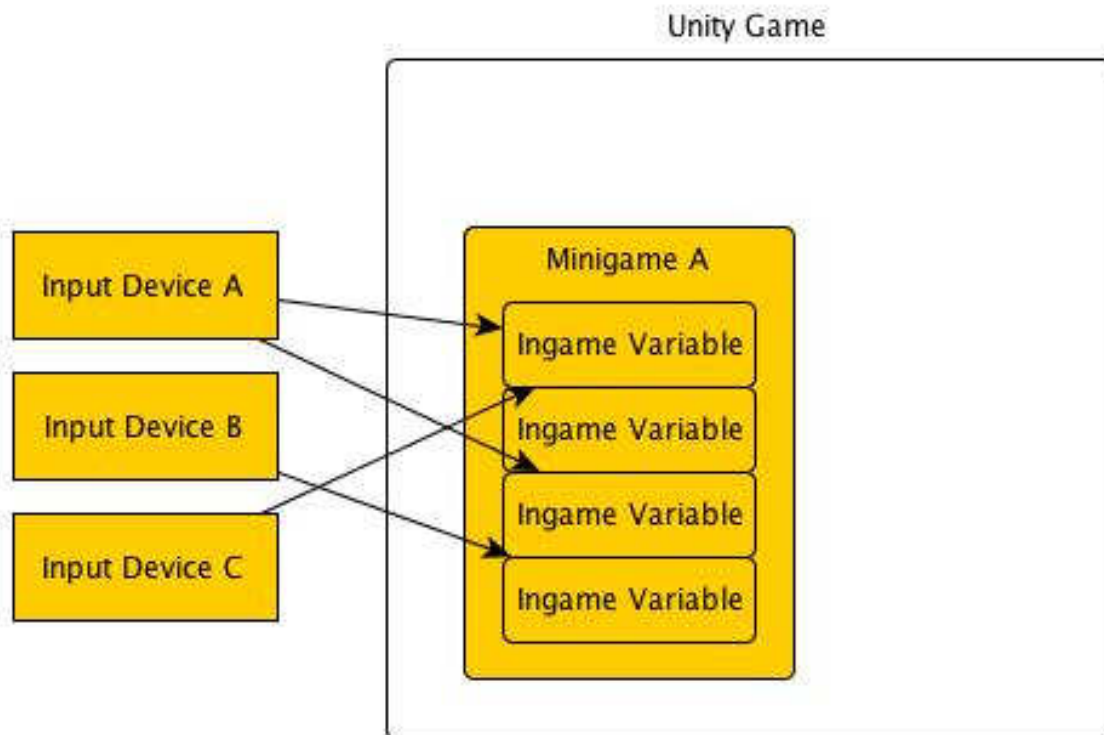


Figure 20: Example of how input device configuration may be very minigame specific

4.4 Game prototype

4.4.1 Early Test Prototype

An early prototype of a game in Unity making use of the Playmancer framework was developed for testing purposes. A screenshot from the game prototype can be seen in Figure 21. This prototype has mainly been used for testing the interface between the input modalities and Unity.



Figure 21: Game prototype running in Unity

In the left side of the screen, the client interface is shown. It contains a list of all input devices supported by the Playmancer framework and shows their current state. It allows the game to connect to a server hosting these modalities. In the middle of the picture is an example of what an in-game avatar may look like. It is in the default pose but as soon as the game has connected to the server it reflect the current state of the body posture.

For testing purposes, a server was also developed, as can be seen in Figure 22.

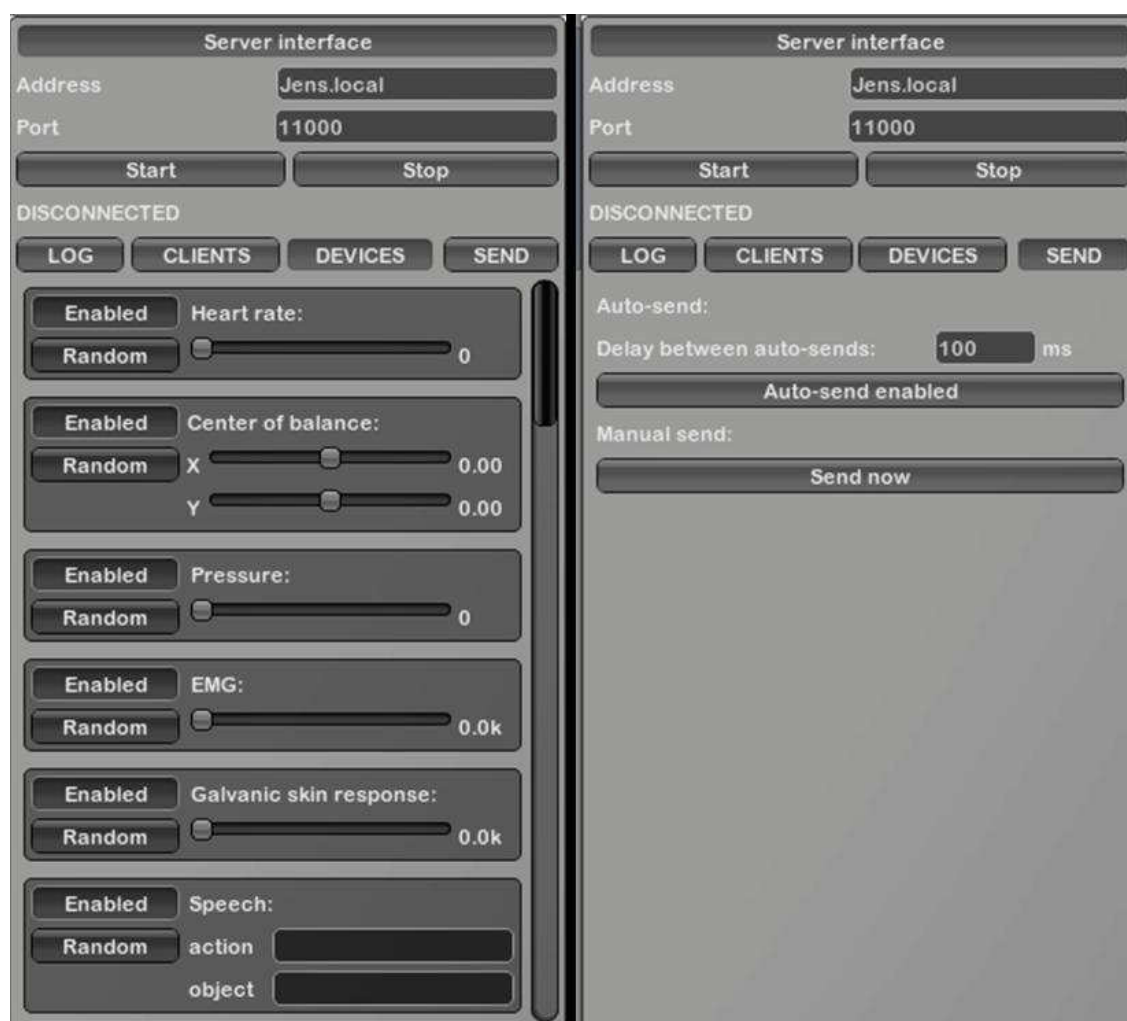


Figure 22: Server prototype interface

With this prototype we are able to mimic a situation where a number of input devices transfers data over to a client. At the moment the body posture is the only data that is used on the client side and positions as well as rotates the joints on a chosen in-game avatar to that of the person in the device.

4.4.2 Final Prototype

The final prototype, which has been used in the field trials is described in deliverable D5.6 and is therefore treated only superficially in this deliverable to avoid redundancy.

When a patient first logs on to the game, a profile is created. The game can be configured individually to each patient's needs by calibrating goals and baselines. This is meant to measure (calibrate) the player's physical vantage point in the various exercises and use this information to set the difficulty of the game.

Then the therapist or patient can select the mini game, which should be played. After he has finished the configured tasks the game returns to an overview scene, where the patient can take a look at his game achievements. If the configured settings don't provide the right amount of challenge (e.g. the patient makes faster progress than expected), the therapist can adjust them easily for the next session. Following each

games session the therapist and patient take a look at the data stored in the patient's profile to determine the progress and issues that have to be worked on.

To achieve the therapy goals three separate mini games have been developed. They are embedded within an adventure setting, linking the games with a common story line. In that story the player arrives with his ship at a deserted island and discovers the remains of an ancient civilization. Pictures of test users playing the mini games can be seen in Figure 23.

The ship is the base of the patient. While on the ship, the game can be configured individually to each patient's needs and abilities by calibrating goals and baselines for the mini games.

During the mini games the player can collect items, which can be seen in an inventory. When the player has collected enough items he is rewarded by unlocking the next part of the story, which should provide additional motivation.

During game play the three mini games provide game feedback (scores, collected items) as well as visual, auditory and textual feedback on the patient's performance and results. Furthermore, after every session, therapist and patient view the patient's progress, by having a look at the patient's profile. In this profile objective data, recorded during gaming, about the reaching ability, cervical range of movement and the level of muscle activation during gaming are presented.



Figure 23. Our MoCap system and two of the mini games during the preliminary testing. „Face of Chronos“ (left), „Three Wind Gods“ (right)

Following the user requirements the visualization of the exact movements of the patients performed during the game is limited. Movements are used as input in the game to monitor, whether a certain goal has been reached, like range of motion of the head or arm and thus whether the patient is being rewarded in the game. In addition, the exact movements are also stored into the log files for monitoring the progress of the patient over time. The exact movements are however not directly presented as feedback in the game to the patients as we want to prevent patients from a too strong focus on his past way of moving. The game should motivate them to perform and reach the desired goal and distract them from the movements that are often associated with pain. For instance, in one game the patient is motivated to

speed up by being given continuous feedback on a colour scale from green to red, representing the patients deviation from the set goal.

Some early screenshots of the island that is part of the game world can be seen in Figure 24 and Figure 25. This is the environment in which the player can navigate between the locations of the mini-games (if he chooses not to use the shortcuts provided by the game interface). Control of the avatar in this case is performed by conventional input devices (mouse/keyboard), since it is of no therapeutic value.



Figure 24: Screenshots from the game-world currently under development



Figure 25: Screenshot from the game-world currently under development

5 Set-Up and Game-Workflow

The intended application-area for the motion-tracking final prototype is in the rehabilitation centre collocated with RRD. There, therapists have to handle the system by themselves after a short introductory course. Therefore, we have made efforts to improve the workflow of the motion-tracking prototype. An overview of the workflow can be seen in Figure 26. Minor updates have been applied to the workflow since the delivery in March 2010. For a description of the final workflow used please see our publication¹².

5.1 Instrumentation of patients

Before the games session can be started the patient has to be equipped with the necessary sensors. For the case of chronic-pain-rehabilitation, this includes the markers for motion-tracking (placed on a motion-suit similar to the one shown in Figure 10) as-well-as electrodes for EMG.

In contrast to the marker-placement EMG-electrode-placement has to be rather precise in order to assure repeatability of the measurements. Furthermore, hair has to be removed and the skin has to be cleaned. Adhesive surface electrodes will be used for the tests.

Depending on the muscle-activity to be assessed with surface-EMG, two different electrode-placements will be used.

1. Activity of the dominant upper trapezius muscle will be recorded using electrodes placed 2 cm laterally to the midpoint between C7 and the lateral end of the acromion with an inter-electrode distance of 2.5cm (see Figure 27a).
2. For the lower back electrodes will be placed on the left and right side of the muscle belly, 30 mm lateral to the first lambar processes spinosus (L1), according to the SENIAM¹³ guidelines, and 30 mm lateral to the fourth lumbar processus spinosus (L4) with an inter-electrode distance of 23mm(see Figure 27b).

¹² Schönauer, C. et al. (2011). Chronic Pain Rehabilitation with a Serious Game using Multimodal Input. In Proceedings International Conference on Virtual Rehabilitation, Zurich, Switzerland.

¹³ Surface electromyography for the Non-Invasive Assessment of Muscles, [<http://www.seniam.org>]

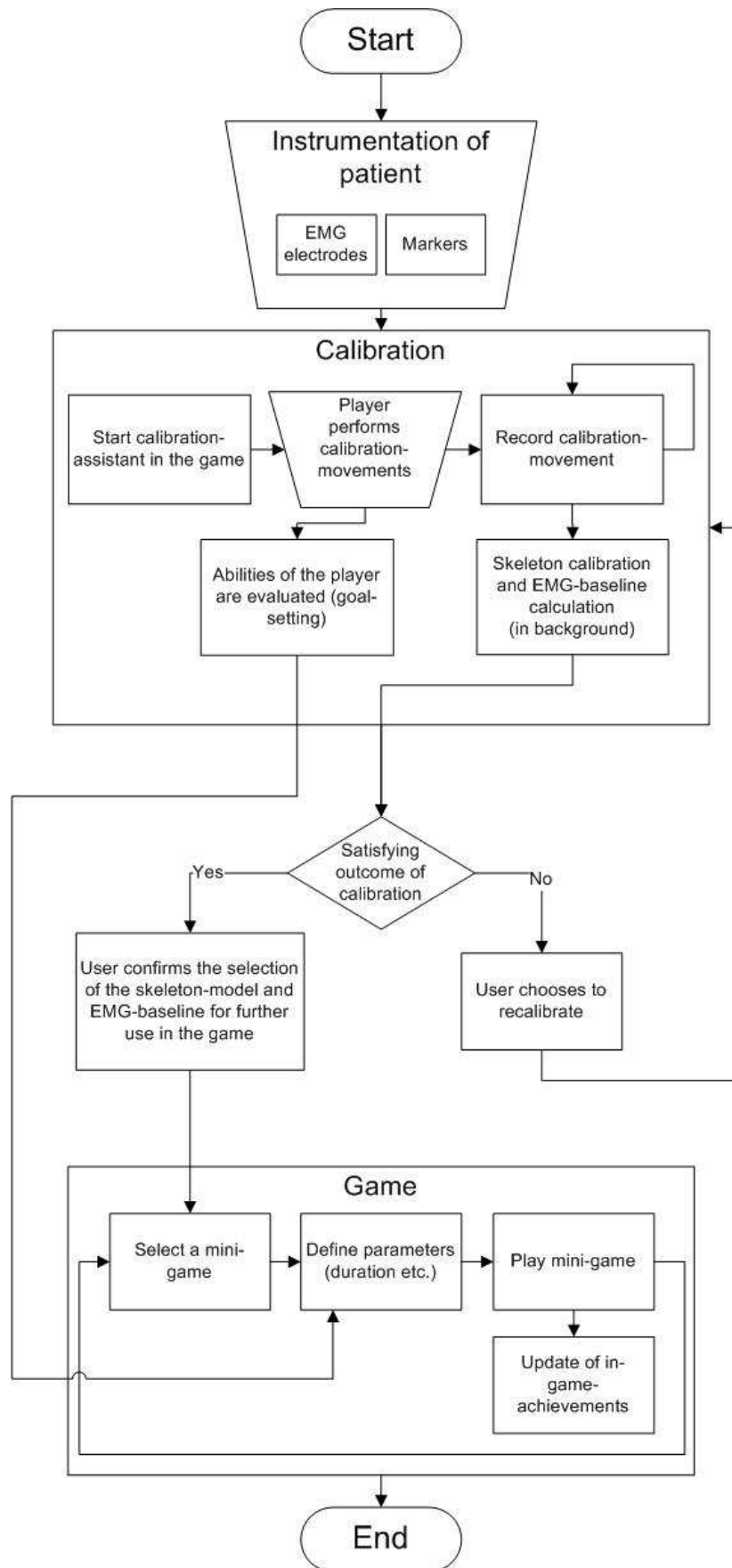


Figure 26: Workflow of a therapy-session

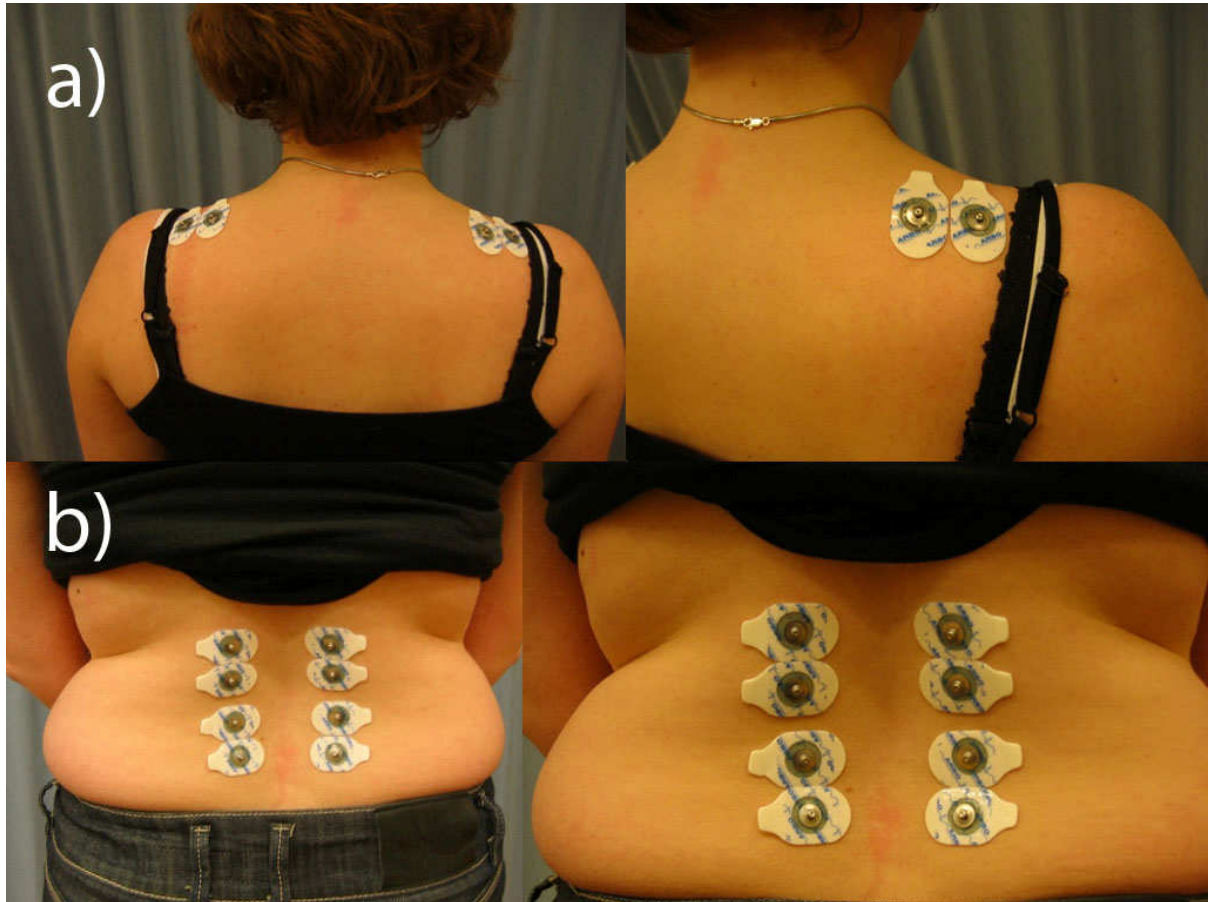


Figure 27: Surface-EMG-electrode-placement

- a) EMG-electrode placement for upper trapezius muscle. b) EMG-electrode placement for the lower back

5.2 Calibration

The game starts with a sequence located on a ship. There calibration takes place and the status of the progress (in-game achievements) is displayed. Furthermore, it can be decided which task/mini-game to start next.

After the game (in which the skeleton calibration assistant is being integrated) has been loaded, the therapist can start the animation sequence by clicking a button or by shortcut. During the animation, the user is encouraged to follow the movements of the animated avatar. When the patient performs the movements correctly, the therapist can start recording the patient. After the patient has gone through the movement-sequence, the therapist stops the recording. The procedure can be repeated an arbitrary number of times in order to improve calibration quality. In most cases, however, one recording is sufficient, as has been documented in deliverable D8.1 subsection 4.2.2.

The skeleton calibration tool loads the recorded movement-files and displays the filenames in the graphical user-interface. For the skeleton calibration, two modes of operation exist, as has been described in sub-section 2.2.3.

In standard-mode the therapist can deselect specific files, if he is under the impression that the patient did not perform well (e.g. missed something in the sequence, stopped to ask a question etc.) For the selected movement-recordings, the therapist can start the calibration by clicking a button.

In automatic-mode, the movement-files are loaded and processed automatically as soon as they are generated by the tracking software. Therefore, it requires less user-interaction and can use resources to perform calibration in the background, while the therapist is recording additional movement-sequences. This option would be preferable for the field-trial at RRD, because it leaves the therapist more time to concentrate on the medical tasks.

The tool automatically selects the best skeleton-model for further processing, however, the therapist is able to override this selection manually.

Finally, the therapist acknowledges the choice of a skeleton. The skeleton is then handed to the iotracker-server and the calibration tool closed. The MoCap-module of the server loads the skeleton and starts the MoCap process waiting for the games to connect.

Once the skeleton is calibrated, it can be used to define baselines for the different exercises within the games. The appendix in section 7.1 contains an internal working-document that describes in more detail, how the baseline is being retrieved for each exercise. Therefore, at this point only an overview is given.

For the baseline-calibration the patient has to go through the different movements, while the game is logging the results of range-of-motion (ROM), step-frequency, velocity etc. This is done in the same calibration-scene within the game which is used for the skeleton calibration. During the mini-games the movements of the patient are evaluated and weighted relative to the recorded baselines.

For the EMG, the root-mean-square values are similarly logged and evaluated to multiple baselines for the different muscle groups. This is done during 60 second to two-minute intervalls, where EMG-values are being recorded. During the game mainly median percent of EMG are used relative to those baselines. Logged, however, will be the absolute values for evaluation, future baseline-calculation and goal-setting.

Due to the calculation of values relative to a baseline the games can easily adapt to different patients and their skills.

After the baseline has been calibrated the therapist sets the goal for the therapy for the different exercises. This is done only during the first use of the Playmancer-game and the patient performs the exercises with help of the therapist. The therapist, for example, moves the head of the patient to a certain position and in that determines how much progress the patient must make for the range of motion.

5.2.1 Calibration from Technical Point of View

From a more technical point-of-view the different steps of the calibration can be listed as follows:

1. Therapist starts iotracker-server and skeleton calibration-tool, which both run in the background at the beginning.
2. Therapist starts game with ship-scene (skeleton calibration assistant is integrated into that scene).
3. In the ship scene the patient is shown how to perform the calibration sequence.
4. The therapist clicks on start recording in the ship-scene/skeleton calibration assistant GUI (iotracker-server starts recording the patients movements).
5. Two things happen:
 - XML-RPC-Call turns off Rigid-Body-Targets in iotracker-server.
 - XML-RPC-Call starts recording of movement-sequence in iotracker-server.
6. Therapist stops recording (and maybe starts a new recording by going back to 4.).
 - XML-RPC-Call turns on Rigid-Body-Targets in iotracker-server.
 - XML-RPC-Call stops recording of movement-sequence in iotracker-server.
7. As soon as there is a recording available the skeleton calibration-tool starts calibrating in the background.
8. The calibration can take a while (for three calibration sequences this will less than 2 minutes) during which we can:
 - Fixate EMG-electrodes.
 - Calibrate EMG-baseline.
9. As soon as the Skeleton calibration-Tool is finished the therapist confirms selection of the best calibrated skeleton (in the tool) or discards skeletons and returns to 4.
10. In the ship-scene/skeleton calibration assistant GUI the therapist confirms that a skeleton has been generated (iotracker-server loads the skeleton and is ready for MoCap):
 - XML-RPC-Call loads the skeleton in iotracker-server

11. The game connects to the iotracker-server MoCap module.
12. The therapist can now proceed in the game with the goal-setting or the selection of mini-games.

5.3 Mini-Games

When the calibration-scene on the ship is completed, it can be decided which task/mini-game to start. For the pilot-trial the therapist will select the appropriate exercises/mini-games, while for future application the patient might be more autonomous in the choice of the games.

Once a mini-game is selected, the player travels to the island where she plays the chosen mini-game. This can either be done in one or several repetitions. When the task is complete, or the player decides to end it, she is transported back to the ship, or moves her avatar to the next location.

Finally, the player is rewarded (if the task was successful). The player can now choose to either do another task or end the game.

For a detailed description of the game-prototypes and workflow within the games please refer to the deliverables of work-package 5 (D5.3 and D5.6).

6 Conclusion and Next Steps

In this document the final Playmancer multi-modal gaming platform prototype has been described. The following paragraphs sum up development status and provide an outlook on the next steps.

6.1 Full-Body Motion-Capture (WP 4.1)

The motion capture system mainly consists of extensions to the *iotracker* software. These include a rigid-body target optimization tool, an application, which generates an approximated skeleton-model and a module/Plug-In to fit the model to the tracking data. All tools/modules have reached fully functional status.

Setup of the Full-Body Motion-Capture-System for the pilot-trials at the Roessingh rehabilitation center in Enschede is scheduled for May 2010. A training-workshop with the therapists will take place immediately after the setup is completed. Finally pre-tests will be executed.

6.2 Multi-Modal Data-Flow Framework (WP 4.3)

During the work on this task Opentracker has been extended to support *iotracker*, various gMOBllab sensors, the SqueezeOrb and the Wii BalanceBoard by implementing and adding modules to the framework. The Unity3D game engine is connected to the OpenTracker data flow network and supporting all input modalities within the games. Supported measurements include MoCap-data of rigid-bodies and full-body MoCap data using a generic skeleton-model. Furthermore various bio-signals like EMG, ECG, pulse rate and GSR are incorporated. In addition, hand pressure strength can be measured as well as the centre of balance. Finally nodes for filtering and evaluation have been added.

6.3 Integration with Unity3D (WP 4.6)

The integration of Unity3D into the Playmancer framework has been achieved by a client-like approach whereby the game acts as a receiver, absorbing data and acting accordingly. The game components within Unity can subscribe to information from the various input devices using an XML format and thereafter utilize this data in whatever fashion. Motion capture and posture data can for example be used to position and orient an in-game avatar or various biosensor-inputs can be applied to adjust the game world.

7 Appendices

7.1 Baseline assessment & goal setting

The first time the patient enters the PLAYMACER game he will perform the baseline assessment and the goal setting exercises in the ship.

Exercise	Baseline assessment	Goal setting
1. Cervical range of motion	<p>In the ship, the patient is asked to perform three times the three sub exercises without help. The patient is asked to bend down (and to the other directions) as far as possible.</p> <ul style="list-style-type: none"> - Flexion / extension - Right/left side bending - Left/right rotation <p>The outcomes (in degrees) of these exercises are the baseline values of the patient.</p>	<p>During the first use of the PLAYMACER game, the patient performs the three sub exercises again with help of the therapist. The therapist moves the head of the patient to the final end and determines how much progress the patient must make. This is the goal. Reaching this goal should be spread over the several training weeks for example 4- 6 weeks of x game sessions meaning steps of 15-25% each week.</p>
2. Walking	<p>In the ship, the patient walks for 2 minutes on a comfortable speed. During this 2 minutes walk the different parameters are measured; step frequency, velocity, distance and muscle tension (RMS). These outcomes are the baseline values of the patient concerning the walking exercise.</p>	<p>The ultimate goal is a normal / healthy walking velocity. These values are known from earlier experimental research. This goal should be reached in 4-6 weeks (or in x game sessions) so steps of 15-25% each week. For this, the system logs the walking velocity for every week/session and uses this data for the goal setting of the next session.</p> <p>The second goal is lowering average RMS values during walking. The goal values are the values of healthy controls that are known from literature/experimental research. Again this goal should be reached in 4-6 weeks (or in x game sessions) meaning steps of 15-25% each week. Therefore, the system logs the RMS value during the walking exercises for every</p>

		week / session and uses this for the goal setting of the next session.
3. Tension & Relaxation	The patient is asked to relax his neck-/shoulder muscles for 60 seconds. The screen shows a relax view and the patient hears relax sounds. The average muscle tension (in RMS) of the last 30 seconds of the neck-/shoulder muscle during relaxation is the baseline value of the patient.	The goal is lowering average RMS values during relaxation after an activity, like climbing. The goal values are the values of healthy controls that are known from literature/experimental research. Again this goal should be reached in 4-6 weeks (or in x game sessions) meaning steps of 15-25% each week. Therefore, the system logs the RMS value during the relaxation exercises for every week / session and uses this for the goal setting of the next session.
4. Reaching overhead	The patient is asked to reach to an object on different (heights) for example small bookshelves overhead. He has to reach as far as possible to one of the bookshelves. The endpoint of the patient reach is the baseline value of the patient.	During the first use of the PLAYMANCER game the patient reach overhead again with help of the therapist. The therapist moves the hand of the patient to the final endpoint and determines how much progress the patient must make. It is especially the further extending the shoulder joint that makes the goal. This progress should be spread over the several training weeks or x gaming sessions meaning steps of 15-25% each week. Therefore, the system logs the endpoint of the patient reach during the reaching overhead exercises for every week / session and uses this for the goal setting of the next session.

7.2 Overview of the Technical Evaluation of Motion Tracking

Measurements of the tracking system indicate minimal jitter (RMS less than 0.05mm), submillimeter location resolution, sub-degree angular resolution and an absolute accuracy of ± 0.5 cm. For a more detailed analysis please refer to deliverable D8.1 subsection 4.1.2. Skeleton tracking shows good repeatability of measurements. For range-of-motion of the neck, as inspected from a session of a healthy subject within one session, variations within a range of two degrees were observed. Also the reaching height showed good repeatability (accuracy of about 1-2 cm), which was also assessed in the field trials as described in D8.2. Furthermore, analysis of the data from head-movements showed fine graduations of rotation angles (in the sub-degree domain) and smooth movements (of a healthy subject) were detected with little jitter.